LOGIC FOR THE MATHEMATICAL

Course Notes for PMATH 330—Spring/2006

PETER HOFFMAN

CONTENTS

Contents

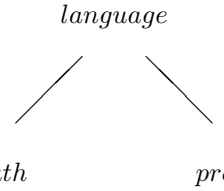
Preface

This book is intended as a fairly gentle, even friendly (but hopefully not patronizing) introduction to mathematical logic, for those with some background in university level mathematics. There are no real prerequisites except being reasonably comfortable working with symbols. So students of computer science or the physical sciences should find it quite accessible. I have tried to emphasize many computational topics, along with traditional topics which arise from foundations.

The first part of the book concerns propositional logic. We find it convenient to base the language solely on the connectives not *and* and. *But the other three 'famous' connectives are soon introduced (as abbreviations) to improve readability of formulae (and simply because ignoring them would gravely miseducate the novice in logic). The biggest part of this first half is Chapter 4, where a few topics connected to computer science are strongly emphasized. For a class which is oriented in that direction, it is possible to bypass much of the material in Chapter 3, which is mainly devoted to proving completeness of a proof system for propositional logic. In Chapter 2 there is also a brief discussion of the use of logical languages in software engineering.*

The proof system in Chapter 3 (and the one for $1^{\underline{\text{st}}}$*order logic in Chapter 7) also use this language. So a few derivations (which would usually be needed) are subsumed by the treating of other connectives as abbreviations defined in terms of* and *and* not. **Hilbert & Ackermann** *(p.10) say: "It is probably most natural to start with the connectives & and — as primitive, as Brentano does . . . ". And it is usually unwise to disagree with Hilbert. (They presumably used* or *and* not *to make it easier for those already familiar with* **Russell & Whitehead** *to read their book.)* **Rosser** *has given a very austere proof system using* not *and* and, *which has only three axiom schema, and modus ponens as the only rule of inference. Here we are slightly more profligate, having three extra 'replacement' rules, in order that: (1) the axioms and rules of inference should look as natural as possible (e.g. the axioms more or less immediately seen to be tautologies), and (2) the path to the proof of adequacy (completeness) be reasonably short, passing mostly through derived rules of inference which are also natural and classical. I am indebted to Stan Burris' text for giving me an appreciation of the utility of using a few replacement rules of inference to make things go smoothly. We only use the replacement rules for double negation, and for the commutativity and associativity of conjunction (plus MP). So the system is easy to remember, and proving completeness is at least a bit of a challenge.*

We have followed the usual triad



in the order of the first three chapters, after a brief introduction. The fourth chapter (still on propositional logic) gives more details on semantics; on the use of resolution, which is of interest for automated theorem proving; and on algorithms and complexity.

Then there are four chapters on 1st order logic, each analogous to the one four earlier on propositional logic. One feature of the proof theory is that we deal with both common approaches to the treatment of non-sentence formulae, giving the appropriate deduction theorem and completeness (and a slightly different proof system) for each. I have not found any text with either type of system which points out the existence and viability of the other (though the discussion in **Kleene**, *pp.103–107, is closely related). So even otherwise knowledgeable PhD students in logic, perhaps weaned on* **Mendelson**, *can be sometimes taken aback (or should I say "freaked out", so as not to show my age too much) when confronted with a deduction theorem which appears to have a hypothesis missing.*

The final chapter relates the 20th century style of logic from earlier chapters to what Aristotle and many followers did, as well as to a few important developments from the romantic age, and to Russell's paradox.

The short Appendix VQ to Chapter 5 gives an introduction to the self-reference 'paradoxes', as seen from the perspective of 1st order logic. It also includes Boolos' proof, modulo 1st order definability of the naming relation, of the Gödel incompleteness theorem.

Near the end are three more appendices. Appendix B to Chapter 8 treats some very dry material which seemed best left out of the main development—questions of the unique readability of terms and formulae, and of procedures for checking strings for being terms and formulae.

Both of the other two appendices can be read by students with a fair amount of undergraduate mathematics experience, independently of the rest of the book and of each other (the appendices, not the students, that is!).

Appendix G is geometry, treating axioms in a more naive, informal way than the main text. It deals with the famous matter of the parallel postulate from Euclid's geometry.

Appendix L treats the incompleteness phenomena in a much more sophisticated and demanding style than the rest of the book. It ends with proofs of the theorems of Gödel (incompleteness), Church (undecidability) and Tarski (undefinability of truth), modulo Gödel's results on expressibility. (So no rigorous definition of algorithm appears in the book.)

If the labelling of appendices seems a bit bizarre, let's just say 'B for boring, G for geometry, and L for lively logic'. See also the last exercise in the book.

For a course with students in mathematical sciences, many of whom are majoring in computer science, I would normally cover much of Chapters 1 to 5, plus a light treatment of Chapter 6, and then Chapters 8 and 9. So Gödel's completeness theorem (Chapter 7) is not done (other than to mention that the existence of a good proof system means that valid arguments from decidable premiss sets can in principle be mechanically verified to be so, despite the validity problem being undecidable). The material in much of Chapter 7 is naturally more demanding than elsewhere in the main body of the book.

Our basic course for advanced (pure) mathematics majors and beginning graduate students would include Gödel completeness and more, definitely using a brisker treatment for much of the course than is emphasized here. On the other side of the coin, there is, generally speaking, a strong contrast between teaching students who are already comfortable with using symbols, and those who are not. This is more a book for the former, but at a very elementary level.

In various places, there are lines, paragraphs and whole sections in smaller print (the

size you are reading here). These can be omitted without loss of continuity. All the regular print gives a complete, but quite elementary, course in logic. The smaller print is usually somewhat more advanced material. Chapter 7 is an exception. There I arbitrarily chose one of the two proof systems to be in regular print, so the reader could work through one complete treatment without continually being interrupted by the analogous topic for the other system. Sometimes the smaller print material is written more with an instructor in mind, rather than a beginning student, as the typical reader.

The title has several possible meanings, such as 'Logic for students in the mathematical sciences' or 'Logic for students with mathematical inclinations' or 'Logic for applications to the mathematical sciences'. It also has the virtue that it will turn up in a search under "mathematical" and "logic", without simply repeating for the nth time the title which originated with Hilbert and Ackermann.

My colleagues Stan Burris, John Lawrence and Ross Willard have been absolutely invaluable in the help they have given me. If even one of them had not been here, the book likely would not have been written. A notable improvement to the axiom system in an early version of Chapter 3 was discovered by John Lawrence, and has been incorporated into the present version. Many students in my classes have helped a great deal with finding obscurities and other mistakes. On the other hand, I insist on claiming priority for any errors in the book that have escaped detection. Lis D'Alessio has helped considerably in producing the text, and I am very grateful for her cheerful and accurate work.

To the beginner in logic :

When they first see relatively sophisticated things such as arguments by contradiction or by induction, possibly in secondary school, students are often somewhat taken aback. Many perhaps wonder whether there isn't something more basic which somehow 'justifies' these methods. Here I mainly want to say that mathematical logic, although it touches on these kinds of things, is NOT done to provide these justifications. Equally, its purpose is not to provide some new kind of language for humans to use in actually doing mathematics.

> *Most likely, logic is capable of justifying mathematics to no greater extent than biology is capable of justifying life.*
>
> Yu. I. Manin

If you study its history, two fairly strong motivations for doing this subject can be discerned. And one of these, arising mainly in the late 1800's and early 1900's, certainly is trying to find basic justification for, or at least clarification of, some crucial mathematical procedures. But these are much deeper and more troublesome matters than the ones above. Examples would be the question of the overall **consistency of mathematics**, the use of **infinite sets**, of the **axiom of choice**, etc., and also the question of whether all of mathematical truth can be discovered by some purely mechanical process. This goes generally under the name **foundations of mathematics**. The main part of this book will give you a good background to begin studying foundations. Much of the material in

Appendix L is regarded as part of that subject. One point that must be made is that we shall quite freely use methods of proof such as contradiction and induction, and the student should not find this troubling. Not having appealed to something like the axiom of choice could of course be important if and when you move on and wish to use this material to study foundations.

The second historically important motivation for logic was the dream of finding some universal, automatic (perhaps 'machine-like') method of logical reasoning. This goes much further back in history, but is also a very modern theme, with the invention of computers (by mathematical logicians, of course!), the rise of the **artificial intelligentsia**, attempts at **automated theorem proving**, etc. A related matter is the fact that difficult problems in computer science, particularly **complexity theory**, are often most clearly isolated as questions about the existence and nature of algorithms for carrying out tasks in logic. This side of things has been emphasized strongly here, particularly in Chapters 4 and 8. We also touch briefly on the use in software engineering/programme verification of languages (if not theorems) from logic.

There is at least one other major reason for wanting to know this subject: instead of using elementary mathematical techniques to study logic, as we do here, people have in recent decades turned this back on itself and used sophisticated results in logic to produce important new results in various fields of mathematics. This aspect is usually known as **model theory**. An example is the construction of a system of numbers which includes infinitesmals, as well as the usual real numbers, and lots of other numbers. This resulted in many suspect mathematical methods from before 1850 suddenly becoming reputable by standards of the 21st century. This book provides a good foundation for getting into model theory, but does not cover that subject at all.

A fourth area, where a basic understanding of mathematical logic is helpful, relates to the physical sciences. The most basic of these sciences, a foundation for the others, is physics. And the foundations of physics, particularly quantum theory, is a thorny subject of much interest. A knowledge of classical logic is prerequisite to **quantum logic**. The latter has considerable bearing on discussions of things like **hidden variable theories**, the apparent incompatibility of quantum and relativistic physics, etc. Due to the need for a good knowledge of physics (by both the reader and the writer!), none of this is directly considered in this book.

The electron is infinite, capricious, and free,
and does not at all share our love of algorithms.
Yu. I. Manin

INTRODUCTION

I can promise not to spell "promise" as "promiss".
But I cannot promise to spell "premise" as "premiss".
Anon.

A major aim of logic is to analyze the validity of arguments. To a logician, an argument is not a shouting match. An *argument* consists of statements: premisses and a conclusion. Being *valid* means that one can be sure of the truth of the conclusion as long as one is sure of the truth of all the premisses. So far, we are being quite inexact.

In this book, the first exact version concerning the validity of arguments comes up in the middle of the second chapter. Before that can be done, we need to be as unambiguous as possible in the language to be used for expressing these arguments. This is achieved, starting in the first chapter, by using symbols. There was a time when the subject was popularly known as *symbolic logic*. A more complicated language ($1^{\underline{\text{st}}}$*order*) is introduced in Chapter 5. In Chapter 1, we begin with the language of *propositional logic*.

Before introducing that language, here are some examples of arguments. Using the logical languages studied in this book, these (and much more complicated arguments) can be analyzed, as done for some of these in Chapter 9. Mathematical and physical sciences consist largely of such arguments.

Some smokers run marathons.
All marathon runners are healthy.
Therefore, smoking isn't bad for your health.

This argument is (outrageously!) invalid from the point of view of simple logic, as we'll see later. The two premisses are possibly true; the conclusion isn't.

A good motto for *invalid* arguments would be the title of the Gershwin song "It ain't necessarily so." See also the end of Exercise 2.22 iii).

Here's another one:

Bushworth is an honest man.
You should vote for an honest man.
Therefore, you should vote for Bushworth.

This argument also is not logically valid. The two premisses are probably not true; but the conclusion does not follow from them anyway.

Actually, in that argument, the word "should" is probably better left out. Mostly, we want to deal with statements which simply state some kind of claimed fact, statements which are clearly either true or false, though which of the two might not be easy to determine. Such statements are often called *assertive sentences* or *declarative sentences*. As written with the word "should" above, we have something closer to commands, or so-called *imperative sentences*. But logic can be used there too, as you may learn if you study computer science. (See also pp. 40-41.)

Here is another argument. After removing the "should", it is close to, but different than, the one above.

Bushworth is the only honest candidate in this election.
You vote for an honest candidate.
Therefore, you vote for Bushworth.

This argument *is* logically valid. The conclusion does follow from the two premisses. However, the first premiss is probably not true. You might have suspected this from a feeling that the conclusion is false and that the second premiss is true.

As you will learn, being logically valid is something which depends only on the *form* of the argument in a certain sense, and not really on the *content* of the individual statements. So far, again I am being vague and inexact. To see the precise versions of what this dependence 'only on form, not on content', it is necessary to study the symbolic versions and precise definitions in the main part of the book. However, here is an example of the 'same' argument, that is, one with the same form, but with the statements having different content.

Woodsworth is the only baboon in this zoo.
You see a baboon at this zoo.
Therefore, you see Woodsworth.

The earlier version above of this valid argument (with both the conclusion and one premiss false) is analogous to a type of thinking which is one of the main ways in which empirical sciences (such as physics and biology) approach the truth by eliminating error. For example, a first, lengthy, premiss might be Sir Isaac Heisenstein's speculative theory of quantum gravity, and a second, some experiment description. The conclusion might then be the result of that experiment, as predicted by the theory, which turned out **not** to be what happened when the experiment was actually performed. If the argument from theory and experimental setup to conclusion was logically valid, this would then eliminate Sir Isaac's theory from contention as an accurate theory.

I look at the world and I notice it's turnin',
While my guitar gently weeps.
With every mistake we must surely be learnin',
Still my guitar gently weeps.

G. Harrison

Next is an example due to Smullyan of three statements which appear to be an argument, which seems to be valid in some purely formal sense, yet is clearly nonsense. For this example, the very attempt to translate it into a more symbolic form is one way to isolate what is wrong.

Some cars rattle.
My car is some car.
Therefore, my car rattles.

Finally, here is an obviously valid argument related to the question of whether the most extreme members of the artificial intelligentsia are even close to the truth when they claim that the brain is simply a computer made out of meat, and that thermostats exhibit a form of consciousness:

The external manifestations of consciousness are a form of physical action.
Consciousness cannot be simulated computationally.
Therefore, not all physical action can be simulated computationally.

Strong arguments for the truth of the second premiss are given in **Penrose**, partly based on the deepest theorem which comes up in this book, namely Gödel's incompleteness theorem. The conclusion, if true, then leads to belief that an adequate scientific theory of consciousness will depend partly

on some completely new theory in physics, since present-day basic theories all lie within the realm of the computable.

In the first section of Chapter 9 near the end of the book, we shall return to the earlier three arguments, and analyse them more closely, using some very simple versions of what has been learned. Also, note that, despite all these examples, the number of premisses in an argument is often different than two! (It can even be zero or infinite.)

1. THE LANGUAGE OF PROPOSITIONAL LOGIC

We wish to build complicated statements out of 'simple' statements. The first two examples would be: "both P and Q"; and simply: "not P"; where P and Q are these simple statements. The first example will be written as $(P \wedge Q)$, and the second, as $\neg P$. So the degree of complication can be small. But, by compounding (i.e. iterating) these constructions, more-or-less arbitrary complexity can be achieved. A more complicated example is

$$(R \wedge \neg(\neg(P \wedge \neg Q) \wedge P)) \ .$$

This is a typical *formula*, as defined below. But soon we'll learn some abbreviations which make these easier to read.

Now it is not necessary to be more precise about what a 'simple' statement is (and, in applications, there is no need to put any restrictions at all on the size of statements for which P and Q etc. are names). Also, we won't say very much at this point about these intended interpretations ("not", "and", etc.) of the symbols. It's good not to lose sight of the fact that our analysis of this subject (in the narrowest sense) does not depend on these meanings. As indicated beginning with the first section below, we are, strictly speaking, simply going to study strings of symbols and other arrays.

Another point here on interpretation: the symbol "$\rightarrow$" occurring several paragraphs below has "implies" as its intended interpretation. But we shall *define* this in terms of "not" and "and" by saying (symbolically) that "P implies Q" is an abbreviation for "not (both P and not Q)". Later, we shall also have the symbols "$\vee$" (meaning "or") and "$\leftrightarrow$" (meaning "if and only if"). They also will be defined in terms of "$\neg$" and "$\wedge$" ; these latter two symbols we have chosen to regard as being more basic than the other three.

You should probably read fairly quickly through this chapter and the first section of the next. The important thing is to get comfortable working with examples of formulae built using the *connectives* $\neg$, $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$, and then with their truth tables. Much less important are the distinctions between abbreviated formulae and actual formulae (which only use the first two of those symbols, use lots of brackets, and fuss about names given to the P's and Q's). It does turn out to be convenient, when proving a theoretical fact about these, that you don't have to consider the latter three symbols. It's also interesting that, in principle, you can reduce everything to merely combinations of: (i) the first two connectives; (ii) the simple statement symbols,

called *propositional variables*; and (iii) brackets for punctuation. Also the rules given for abbreviating are not a really central point. You'll pick them up easily by working on enough examples. For some, reading the contents of the boxes and doing the exercises may suffice on first reading.

Here is a quick illustration of the formal language of propositional logic introduced in the next two chapters.

Let P be a name for the statement "I am driving."

Let Q be a name for "I am asleep."

Then $\neg P$ is a name for "I am not driving ."

And the string $P \wedge Q$ is a name for "I am driving and I am asleep." But the translation for $P \wedge Q$ which says "I'm asleep driving" is perfectly good; a translation is **not** unique, just as with translating between natural languages.

The string $P \rightarrow \neg Q$ is a name for "If I am driving, then I'm not asleep"; or, for example, "For me, driving implies not being asleep."

The actual 'strict' formula which $P \rightarrow \neg Q$ abbreviates in our setup (as indicated several paragraphs above) is $\neg(P \wedge \neg\neg Q)$. A fairly literal translation of that is "It is not the case that I am both driving and not not asleep." But the triple negative sounds silly in English, so you'd more likely say "It is not the case that I am both driving and sleeping," or simply, "I am not both driving and sleeping." These clearly mean the same as the two translations in the previous paragraph. So defining $F \rightarrow G$ to be $\neg(F \wedge \neg G)$ seems to be quite a reasonable thing to do. Furthermore, if asked to translate the last quoted statement into propositional logic, you'd probably write $\neg(P \wedge Q)$, rather than $P \rightarrow \neg Q \;=\; \neg(P \wedge \neg\neg Q)$. This illustrates the fact that a translation in the other direction has no single 'correct' answer either.

As for the last two (of the five famous connectives), not introduced till the next chapter, here are the corresponding illustrations.

The string $P \vee Q$ is a name for "Either I am driving, or I am sleeping (or both);" or more snappily, "Either I'm driving or asleep, or possibly both." The actual 'strict' formula for which $P \vee Q$ is an abbreviation is $\neg(\neg P \wedge \neg Q)$. A literal translation of that is "It is not the case that I am both not driving and not asleep." A moment's thought leads one to see that this means the same as the other two quoted statements in this paragraph.

The string $P \leftrightarrow \neg Q$ is a name for "I am driving if and only if I am not sleeping." The abbreviated formula which $P \leftrightarrow \neg Q$ further abbreviates is $(P \rightarrow \neg Q) \wedge (\neg Q \rightarrow P)$. A translation of that is "If driving, then I am not asleep; and if not sleeping, then I am driving" (a motto for the long-distance

truck driver, perhaps). The two quoted statements in this paragraph clearly have the same meaning.

1.1 The Language

By a *symbol*, we shall mean anything from the following list:

$$\neg \qquad \wedge \qquad) \qquad (\qquad P_{|} \qquad P_{||} \qquad P_{|||} \qquad P_{||||} \qquad \cdots$$

The list continues to the right indefinitely. But we're not assuming any mathematical facts about either 'infinity' or anything else. We're not even using $1, 2, 3, \ldots$ as subscripts on the P's to begin with. We want to emphasize the point that the subject-matter being studied consists of simple-minded (and almost physical) manipulations with strings of symbols, arrays of such strings, etc. (The study itself is not supposed to be simple-minded !) Having said that, from now on the symbols from the 5th onwards will be named P_1, P_2, $\cdots$. This is the first of many abbreviations. But to make the point again, nothing at all about the mathematics of the positive integers is being used, so inexperienced readers who are learning logic in order to delve into the foundations of mathematics don't need to worry that we're going round in circles. (The subscripts on the P's are like automobile licence numbers; the mathematics that can be done with those particular notations is totally irrelevant for their use here and on licence plates.) Actually, in most discussions, we won't need to refer explicitly to more than the first four or five of those "P symbols", which we'll call *propositional variables*. The first few will also be abbreviated as P, Q, R, S and T. Sometimes we'll also use these latter symbols as names for the 'general' propositional variable, as in: "Let P and Q be any propositional variables, not necessarily distinct ...".

The word *formula* (in propositional logic) will mean any finite string, that is, finite horizontal array, of these symbols with the property in the third paragraph below, following the display and exercise.

One way (just slightly defective, it seems to me) to define the word "formula" is as follows.

(i) Each propositional variable is a formula.
(ii) If F is a formula, then so is $\neg F$.
(iii) If F and G are formulae (not necessarily distinct), then so is $(F \wedge G)$.
(iv) Every formula is obtained by applying the preceding rules (finitely often).

That's a good way to think about building formulae. But to exclude, for example, the unending string $\neg\neg\neg\cdots$ does then sometimes require a bit of explaining, since it can be obtained by applying rule (ii) to itself (but isn't finite). So the wording just after the exercise below is possibly better as a definition.

To precede the official definition with some examples, here are a couple of ways to systematically build (going down each column) the formula $(\neg P \wedge Q)$, and one way to similarly build the (different!) formula $\neg(P \wedge Q)$.

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

Exercise 1.1 Produce a similar column for the example

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

from the introductory paragraph of the chapter, using the rules below.

The official definition of *formula* which we'll give is the following. For a finite horizontal array of symbols to be a formula, there must exist some finite vertical array as follows, each of whose lines is a finite horizontal array of symbols. (These symbols are the ones listed at the start of this section—you can't just introduce new symbols as you please—the rules of the 'game' must be given clearly beforehand!) The array to be verified as being a formula is the bottom line of the vertical array. Furthermore, each array line is one of:
(i) a propositional variable; or
(ii) obtainable from a line higher up by placing a $\neg$ at the left end of that line; or, finally,
(iii) obtainable by placing a $\wedge$ between a pair of lines (not necessarily distinct) which are both higher up, and by enclosing the whole in a pair of brackets.

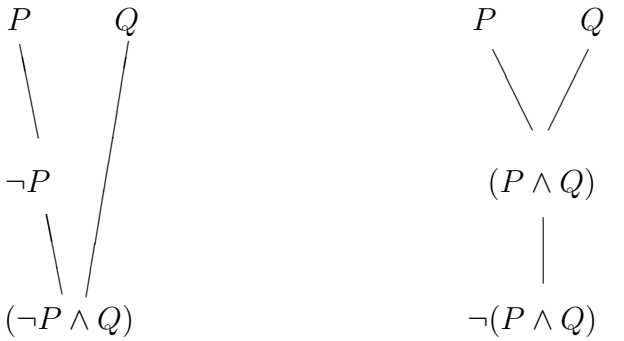
Notice how (i), (ii) and (iii) here are just reformulations of (i), (ii) and (iii) in the box above.

The vertical array referred to in the above definition will be called a *formula verification column* for the formula being defined. It isn't unique to the formula, as the left and middle columns in the display before 1.1 show. Note that the top line in such a column is always a propositional variable. Also, if you delete any number of consecutive lines, working up from the bottom, you still have a formula verification column, so *every* line in the column is in fact a formula.

Exercises. 1.2 Convince yourself that there are exactly 6 formulae, among the 720 strings obtained by using each symbol in $\neg(P \wedge Q)$ exactly once.

1.3 Find examples of strings of symbols F, G, H, K with $F \neq H$ and $G \neq K$, but such that the formula $(F \wedge G)$ is the same as $(H \wedge K)$. Can this happen if F, G, H, K are all formulae? (Just guess on this last one—a proof is more than can be expected at this stage—the material in **Appendix B** to Chapter 8 provides the tools for a proof.)

You may have realized that the difference is rather superficial between the left and middle columns, just before Exercise 1.1. It can be quite revealing to instead use arrays such as the two below, where the left one corresponds to the left and middle columns above, and the right one to the right column.



But the columns will do perfectly well for us, so we'll not use these *trees*.

One use of the columns will be in proving general properties of formulae. The idea will be to work your way down the 'typical' column, checking the property one formula at a time. A good way to phrase such a proof efficiently is to begin by assuming, for a contradiction, that the property fails for at

least one formula. Then consider a shortest verification column for which the formula at the bottom fails to have the property—it must be shortest among all verification columns for **all** formulae for which the property fails). Looking at that column of formulae, somehow use the fact that all the formulae *above* the bottom one do have the property to get a contradiction, so that the bottom one *does* have the property. And so the property does hold in general for formulae.

The above type of proof is known as *induction on formulae*. An alternative way to set up such a proof is to first verify the property for propositional variables (start of induction), then verify it for $\neg F$ and for $(F \wedge G)$, assuming that it does hold for F and for G (inductive step). This is probably the best way to remember how to do induction on formulae. It is much like the more common mathematical induction. The initial case ($n = 1$) there is replaced here by the initial case ($F = P$, a propositional variable). The inductive step (deduce it for $n = k + 1$ by assuming it for $n = k$) there is replaced here by a pair of inductive steps, as indicated above. (The method in the previous paragraph really amounts to the same thing, and does explain more directly why induction on formulae is an acceptable method of proof.)

An example might be to prove that, *in any formula, the number of occurrences of the symbol " (" equals the number for the symbol "∧"*. (Here we are dealing with 'strict' formulae which use only the connectives $\neg$ and $\wedge$, and not the extra ones $\rightarrow, \vee$ and $\leftrightarrow$ mentioned in the introduction.)

To first prove this in the earlier style, proceed as follows. In a verification column where all but possibly the bottom formula have this property, you'd observe the following. If the bottom one was a propositional variable, then these numbers are both zero. If the bottom one is obtained by placing a "¬" in front of a line further up, then the numbers of "(" and of "∧"do not change from those for the higher formula, so again they agree. Finally, considering the only other justification for the bottom formula, we see that both numbers are obtained by adding 1 to the sum of the corresponding numbers for the two higher up formulas, since here we insert a "∧" between these two, and add a "(" on the left [and nothing else, except the ")" added on the right]. Since the numbers agree in both the higher up formulae, they agree in the bottom one, as required.

Here is the same proof written in the other, standard, "induction on formulae" style: For the initial case, if $F = P$, a propositional variable, then both numbers are zero, so F has the property. For the inductive steps, first assume that $F = \neg G$, where G has the property. But the number of "(" in F

is the same as that in G. That's also true for the numbers of "$\wedge$". Since they agree for G, they agree for F. Finally, assume that $F = (G \wedge H)$, where G and H both have the property. Obviously, for both "(" and "$\wedge$", the formula $(G \wedge H)$ has one more copy than the total number in G and H combined. Thus the numbers agree, as required, finishing the inductive steps and the proof.

If you find the above a rather tedious pile of detail to establish something which is obvious, then you are in complete agreement with me. Later however, induction on formulae will be used often to prove more important facts about formulae. (See, for example, **Theorem 2.1/2.1*** near the end of the next chapter.)

Similar boring arguments can be given to establish that the numbers of both kinds of bracket agree in any formula, and that, reading left to right, the number of "(" is never less than the number of ")" .

Exercises 1.4 Prove

(i) the two facts just stated, and also

(ii) that the number of occurrences of propositional variables in a formula is always one more than the number of occurrences of the symbol "$\wedge$".

(When we start using the other connectives [$\rightarrow, \vee$ and $\leftrightarrow$], this exercise must be modified by adding in the number of their occurrences as well.)

(iii) Show that a finite sequence of symbols can be re-arranged in at least one way to be a formula if and only if

$$\#\text{ “(” } = \#\text{ “}\wedge\text{” } = \#\text{ “)” } = (\#\text{ occurrences of prop. variables}) - 1 \ .$$

To summarize the main points in this subsection:

MAIN POINTS

For dealing with general properties of formulae, we may assume that every formula is built using just the two connectives $\neg$ and $\wedge$. This is done inductively, starting with propositional variables, and building, step-by-step, from 'smaller' formulae F and G up to 'bigger' formulae $\neg F$ and $(F \wedge G)$.

To prove a general property of formulae, usually you use **induction on formulae**, first proving it for propositional variables P, then deducing the property for 'bigger' formulae $\neg F$ and $(F \wedge G)$ by assuming it for 'smaller' formulae F and G.

A formula verification column is simply a listing of all the intermediate steps in building up some formula.

1.2 Abbreviations.

The following will be a very frequently occurring abbreviation:

$(F \to G)$ abbreviates $\neg(F \wedge \neg G)$.

For the moment, there's no need to talk much about the *meaning* of "$\to$". But recall that the word "implies" is normally what you would vocalize, when reading "$\to$" aloud. (For more on why the symbol array on the right in the box above ought to be translated as "implies", see the discussion just before the previous section, and also the one after the truth table for "$\to$", p. 25)

The basic idea concerning extra connectives in the language:

For producing readable formulae, we may also use additional connectives $\to$ (and, introduced in the next chapter, also $\vee$ and $\leftrightarrow$). This is done inductively as before. So you start with propositional variables, and build from 'smaller' (abbreviated) formulae F and G up to 'bigger' (abbreviated) formulae $\neg F$, $(F \wedge G)$, $(F \vee G)$, $(F \to G)$ and $(F \leftrightarrow G)$.

Next, we shall give some more rules to abbreviate formulae, by removing brackets, making them more readable without causing ambiguity.

First note that we didn't use any brackets when introducing "$\neg$" into formulae, but did use them for "$\wedge$". They are definitely needed there, so that we can, for example, distinguish between $\neg(P \wedge Q)$ and $(\neg P \wedge Q)$. But the need for the outside brackets only comes into play when that formula is to be used in building up a bigger formula. So :

> (I) The outside brackets (if any) on a formula (or on an abbreviated formula) will usually be omitted.

You must remember to re-introduce the outside brackets when that formula is used to build a bigger formula. Abbreviation rule (I) also will be applied to an abbreviated formula built from two smaller formulae using "$\rightarrow$". We deliberately put the brackets in when defining the arrow above, but again only because they must be re-introduced when using $F \rightarrow G$ as a component in building a larger formula.

The reason, in our original definition of "formula", for not using $\neg(F)$ or $(\neg F)$, but rather just $\neg F$, is to avoid unnecessary clutter. It can be expressed as saying that "$\neg$" *takes precedence over* "$\wedge$". For example, the negation sign in $\neg P \wedge Q$, whose real name is $(\neg P \wedge Q)$, 'applies only to P, not to all of $P \wedge Q$'.

Now I want to introduce an abbreviation to the "$\rightarrow$" abbreviation, by having "$\wedge$" take precedence over "$\rightarrow$" :

> (II) If an abbreviated formula $F \rightarrow G$ is built where either or both of F and G has the form $(H \wedge K)$, then the outside brackets on the latter formula may be omitted, producing a shorter abbreviation.

For example, the bracket-free abbreviation $P \wedge \neg Q \rightarrow \neg R \wedge S$ has a well-defined meaning : We have

$$P \wedge \neg Q \rightarrow \neg R \wedge S \;=\; (P \wedge \neg Q) \rightarrow (\neg R \wedge S) \;=\; \neg((P \wedge \neg Q) \wedge \neg(\neg R \wedge S))$$

The right-hand component of the display is a formula; the other two, separated by "=" signs, are abbreviations of that formula. Putting brackets around the outside of either of the abbreviations gives two more abbreviations.

(We are being a bit sloppy when saying that the right-hand component of the display is a formula, in that P, Q, etc. should be replaced by $P_{|}$, $P_{||}$, etc., to get a formula rather than an abbreviated formula. But there's no real need to continue with that particular distinction. So from now on, we won't make any fuss about the names and nicknames of propositional variables.)

Exercises

1.5 Work out the actual formula of which each of the following is an abbreviation.

(i) $P \wedge \neg Q \to \neg(R \wedge S)$ (ii) $P \wedge (\neg Q \to \neg R \wedge S)$

(iii) $P \wedge (\neg Q \to \neg(R \wedge S))$ (iv) $P \wedge ((\neg Q \to \neg R) \wedge S)$

(v) $(P \wedge (\neg Q \to \neg R)) \wedge S$

Note that these five are all different from each other and from the one just before the exercise.

1.6 Find any other formulae, besides these five above, which have abbreviations with exactly the same symbols in the same order as above, except for altering the position and existence of the brackets.

1.7 Find the actual formula whose abbreviation is

$$(T \to S) \to (Q \to R) \wedge P \ .$$

1.8 Translate the following math/English statements into formulae in propositional logic. In each case, choose substatements as small as possible and label each with the name of a propositional variable.

(i) *If every man is a primate, and every primate is a mammal, then every man is a mammal.*

(iia) *If $x = y$, and $x = z$ implies that $y = z$, then $y = y$.*

(iib) *If $y = z$ is implied by the combination of $x = y$ and $x = z$, then $y = y$.*

(So commas are effective!)

(iii) *It is not the case that not all men are males.*

(iv) *The French word "maison" has a gender but not a sex, and the person "Ann" has a sex but not a gender, and one of the nouns just used is also used*

as an abbreviated name for an act whose name some people are embarrassed to utter. Taken together, these imply that the English language has been (irretrievably, as always) corrupted by prudes who design questionnaires and by some social scientists who invent university programmes.
Write those two statements as a single formula. (The use of the word "implies" here is questionable, but maybe I was having difficulties finding interesting examples. It is quite ironic that, besides its use as a noun, you will find a little-used definition of "gender" as a verb in the OED. And that meaning of the word is exactly the one which the prudes are trying to avoid bringing to mind!—"please bring your bull to gender my cow"!)

We can make more interesting exercises once a symbol for "or" is introduced as an abbreviation in the next chapter. But try to translate the following in a reasonable way without reading ahead, using only the symbols and abbreviations so far introduced. There are at least two good ways to do it.

(v) *If Joe is on third or Mary is pitching, then we win the game.*

1.9 Eliminate brackets in each of the following to produce the shortest possible corresponding abbreviated formula.

(i) $((P \wedge (Q \wedge \neg P)) \rightarrow (\neg Q \rightarrow (R \wedge P)))$

(ii) $(((P \wedge Q) \wedge \neg P) \rightarrow (\neg Q \rightarrow (R \wedge P)))$

(iii) $(P \wedge ((Q \wedge \neg P) \rightarrow (\neg Q \rightarrow (R \wedge P))))$

(iv) $((P \wedge ((Q \wedge \neg P) \rightarrow \neg Q)) \rightarrow (R \wedge P))$

(v) $(P \wedge (((Q \wedge \neg P) \rightarrow \neg Q) \rightarrow (R \wedge P)))$

You should need no more than two pairs of brackets for any of these. Note how your answer formulae are far more readable than the question formulae. Except for the first two, they are four very different formulae. The first two are, of course, different formulae, but we'll see in the next chapter that they are *truth equivalent*.

In 2.0 you'll need to write out a verification column for each of these, but one where lines may be written as an abbreviated formula using the connective $\rightarrow$.

1.10 Restore brackets in each of the following abbreviated formulae, reproducing the actual formula of which it is an abbreviation.

(i) $P \wedge (Q \to R) \to (S \wedge T \to P \wedge Q)$

(ii) $P \wedge ((Q \to R) \to S \wedge T) \to P \wedge Q$

1.11 Show that a pair of brackets must be added to the following to make it into an abbreviated formula; that this can be done in two different ways; and find the actual formula for both of these abbreviated formulae.

$$P \wedge (Q \to (R \to S \wedge T) \to P \wedge Q)$$

1.12 Decide which of the following is a formula or an abbreviated formula, or just a string of symbols which is neither.

(i) $((P \wedge Q \to R) \to S \wedge T) \to P \wedge Q$

(ii) $(P \wedge Q) \to (R \to (S \wedge T) \to P \wedge Q)$

(iii) $P \wedge Q \to ((R \to S) \wedge T \to P \wedge Q)$

(iv) $P \wedge (Q \to R) \to (S \wedge T \to P \wedge Q))$

1.13 Give a detailed argument as to why neither $P \to \wedge Q$ nor $P \wedge \to QR$ is a formula nor an abbreviated formula.

1.14 A complete alteration of the language, called *Polish notation*, in which no brackets are needed, can be achieved with the following change.

Replace $(F \wedge G)$ by $\wedge FG$ everywhere.
And, for abbreviations,
replace $(F \to G)$ by $\to FG$ everywhere.
Similarly, after we introduce them in the next chapter,
replace $(F \vee G)$ by $\vee FG$, and replace $(F \leftrightarrow G)$ by $\leftrightarrow FG$.

Convert each of the following abbreviated formulae into the language *Polish propositional logic.*

(i) $P \wedge (Q \to R) \to (S \wedge T \to P \wedge Q)$

(ii) $P \wedge ((Q \to R) \to S \wedge T) \to P \wedge Q$

Is POPISH logic always nowadays (2002) expressed in the language POLISH logic?

1.15 A sign at the cashier of my local hardware store says :

(i) We are happy to accept checques drawn on local banks.

I suspect that what they really mean is different, or at least that they wish to imply more than that, without actually saying it. Possible candidates are the following.

(ii) We are not happy to accept checques drawn on non-local banks.

(iii) We are happy to not accept checques drawn on non-local banks.

(iv) We are not happy to not accept checques drawn on non-local banks.

Just this once, I'll use four non-standard letters to denote propositional variables, namely the following:

C = you pay by checque;
L = your bank is local;
H = we are happy;
A = we accept your checque.

Use these to express each of (i) to (iv) as a formula in propositional logic.

As a final example, note that

$$\neg(\neg R \wedge S) \qquad \neg\neg(R \wedge S) \qquad (\neg\neg R \wedge S)$$

are all formulae (and so definitely *different* formulae!) The last one has $\neg\neg R \wedge S$ as an abbreviation. One is actually tempted to rewrite this last one as $(\neg\neg R) \wedge S$, to make it more readable (for me, anyway). But I won't, since that would actually be incorrect according to the conventions introduced in this chapter.

2. RELATIVE TRUTH

We'll begin with an example. If you think about the 'meaning' of them, the two formulae, $\neg\neg(R \wedge S)$ and $\neg\neg R \wedge S$, at the end of the last chapter, are really saying the same thing. Two simpler ways of saying that same thing are just $R \wedge S$ and $S \wedge R$. In the previous chapter, 'meaning' played no role, though we introduced it briefly in the introductory paragraphs. In this chapter, it will begin to play a role, but a very restricted one. This is because we'll be just analyzing compound sentences in terms of their 'simple' components. It will be possible to make a precise definition of when two formulae are equivalent. All four of the formulae just above will turn out to be equivalent to each other.

2.1 Truth assignments and tables.

We want to keep the abstract 'meaningless' propositional variables. What will be done is to allow all possible combinations that designate which of them are 'true', and which 'false', and then to analyze how bigger formulae become either true or false, depending on the combination as above with which one is dealing.

A *truth assignment* e can be defined in two ways.

The simplest is to say that it is a function, or assignment, which associates, to each propositional variable, one of the *truth values*, namely Tr or Fs; that is, True or False. Once a truth assignment has been fixed, each formula, by extension, obtains its own truth value via the following *basic definitional truth tables* :

F	$\neg F$
Tr	Fs
Fs	Tr

F	G	$F \wedge G$
Tr	Tr	Tr
Tr	Fs	Fs
Fs	Tr	Fs
Fs	Fs	Fs

To be complete, one has to proceed by induction on formulae to show that each formula then gets a unique truth value. (But this seems pretty obvious. For actually calculating truth values, see the considerable material on truth tables a few paragraphs below. Strictly speaking, to show that there couldn't be two calculations which produce different answers for a given formula, we need results left to **Appendix B** to Chapter 8. These are a bit tedious, but not difficult.)

The basic tables above can be summarized by saying that they require the extension of the function e from being defined only for propositional variables to being defined for all formulae so that e satisfies the following for all formulae F and G :

$$\mathsf{e}(\neg F) = \mathsf{Tr} \iff \mathsf{e}(F) = \mathsf{Fs} \quad ; \quad \mathsf{e}(F \wedge G) = \mathsf{Tr} \iff \mathsf{e}(F) = \mathsf{Tr} = \mathsf{e}(G)\,.$$

Since $\neg F$ and $F \wedge G$ (and all other formulae) get the value Fs wherever they don't get the value Tr, this display 'says it all'. There is surely no doubt that assigning truth values this way coincides with how the words "not" and "and" are used in English.

We moderns don't worry too much about the distinction between "Jane got married *and* had a baby" and "Jane had a baby *and* got married"! In any case, the sense of a time sequence which "and" sometimes conveys in ordinary English is not part of its logical meaning here. Really this extra 'timelike' meaning for "and" in English comes from shortening the phrase "and then" to just the word "and". Another amusing example of this is the apparent difference in meaning between "Joe visited the hospital and got sick" and "Joe got sick and visited the hospital."

When calculating the truth table of a specific formula, you use the basic truth tables for $\neg$ and $\wedge$. But later, in theoretical discussions, to check that a given function e from the set of all formulae to $\{\mathsf{Tr}, \mathsf{Fs}\}$ is a truth assignment, we shall check the two conditions in the display above (which are equivalent to those basic truth tables).

In the display above we're using the symbol " $\iff$ " as an abbreviation for the phrase "if and only if". *This is definitely not part of the* **formal language** *of propositional logic* (though soon we'll introduce a connective "$\leftrightarrow$" within that language which has a role analogous to " $\iff$ "). The notation " $\iff$ " is a part of our English/math jargon, the **metalanguage**, for talking *about* the propositional logic (and later 1$^{\underline{\text{st}}}$order logic), but not for 'talking' *within* those formal languages. The symbol "=" is also not part of

the formal language. In this book, it always means "is the same as", or, if you prefer, "$x = y$" means "x and y are names for the same thing". A more thorough discussion of these points is given in Section **2.5** ahead.

The second possible definition of a *truth assignment* (which some might prefer to give a different name, say, *truth map*) is to say that it is any function e defined (from the beginning) on **all** formulae (not just on propositional variables), and having the two properties, with respect to negation and conjunction, given in the display a few lines above the box. But one cannot get away without having to prove something—namely that there is exactly one truth map for which the propositional variables take any given preassigned set of values, some variables (or none, in one case) true, and the rest false.

It is clear that the two possible definitions above essentially coincide with one another, once the above-mentioned facts to be proved have been established. The only difference is whether you think of e as a truth *map* (a function given on all formulae) or as a truth *assignment* (given initially just on the set of propositional variables). From now on, we'll call e a truth *assignment* in both situations, since that has become a fairly standard terminology in logic. (Other common names for the same thing are *truth evaluation* and *boolean evaluation*.)

At least for the time being, we'll take what should have been proved above as known, and proceed to examples of actually calculating $\mathsf{e}(F)$, given that we have specified $\mathsf{e}(P)$ for all the propositional variables P which occur in the formula F. Although another method is given in Section **4.2**, the most common method is to use a *'calculational' truth table*, in which the answer is obtained for all possible combinations of truth values for the relevant propositional variables. It is calculated systematically, getting the truth values one-by-one for all the formulae occurring as lines in a formula verification column for F .

For example, $\mathsf{e}(P \to Q)$ is worked out below for all possible truth assignments e.

P	Q	$\neg Q$	$P \wedge \neg Q$	$\neg(P \wedge \neg Q) = P \to Q$
Tr	Tr	Fs	Fs	Tr
Tr	Fs	Tr	Tr	Fs
Fs	Tr	Fs	Fs	Tr
Fs	Fs	Tr	Fs	Tr

We used the basic definitional truth table for "$\neg$" to calculate the 3rd and 5th columns; halfway through that, we used the basic definitional truth table for "$\wedge$" to work out the 4th column. The top line is a verification column written sideways. Each line below that corresponds to a choice of truth assignment. The first two columns have been filled in so as to include every possible combination of truth values for the two relevant propositional variables P and Q.

In general, you may have many more than just two relevant propositional variables to deal with. Filling in the initial several columns corresponding to them is done in some consistent way, which varies from day-to-day according to which side of the bed you got out of. My usual way is, for the leftmost propositional variable, to first do all possibilities where it is true, then below that, the bottom half is all those where it is false. Each of these halves is itself divided in half, according to the truth values of the second variable, again with "true" occupying the top half; and so on, moving to the right till all the variables are considered. Note that in this scheme (you may, along with the computer scientists, prefer a different scheme), the rightmost variable has a column of alternating Tr and Fs under it. It is not hard to see that the number of lines needed to cover all possibilities, when constructing a truth table for a formula that uses "n" different propositional variables, is $2^n = 2 \cdot 2 \cdot 2 \cdots$ ("n" times). For an example with $n = 3$, a truth table for the formula $F = R \wedge \neg(\neg(P \wedge \neg Q) \wedge P)$, from the introductory paragraphs of this chapter, is as follows (so the top row gives one answer to Exercise 1.1).

P	Q	R	$\neg Q$	$P \wedge \neg Q$	$\neg(P \wedge \neg Q)$	$\neg(P \wedge \neg Q) \wedge P$	$\neg(\neg(P \wedge \neg Q) \wedge P)$	F
Tr	Tr	Tr	Fs	Fs	Tr	Tr	Fs	Fs
Tr	Tr	Fs	Fs	Fs	Tr	Tr	Fs	Fs
Tr	Fs	Tr	Tr	Tr	Fs	Fs	Tr	Tr
Tr	Fs	Fs	Tr	Tr	Fs	Fs	Tr	Fs
Fs	Tr	Tr	Fs	Fs	Tr	Fs	Tr	Tr
Fs	Tr	Fs	Fs	Fs	Tr	Fs	Tr	Fs
Fs	Fs	Tr	Tr	Fs	Tr	Fs	Tr	Tr
Fs	Fs	Fs	Tr	Fs	Tr	Fs	Tr	Fs

Now truth tables can be done in a sense more generally, where instead of having a formula written out explicitly in terms of propositional variables, we have it written in terms of smaller unspecified formulae. A simple example is $F \to G$. Such a string is really a name for infinitely many formulae, one for each particular choice of F and G. In the example, to work out

$\mathsf{e}(F \to G)$ for arbitrary formulae F and G, you use the first two columns to fill in all the possibilities for the truth values of F and G. (A minor remark here is that, for certain specifications of F and/or G, not all four of these possibilities occur—see, for example, *tautologies* below, or consider the case when $F = G$—but no harm is done and nothing incorrect is stated by giving the entire table and asserting that it applies for every choice of F and G.) The truth table for $F \to G$ is the same as the second previous table, for $P \to Q$, except for changing P to F, and Q to G, everywhere on the top line. So I won't write it all out, but the final answer is worth recording below for two reasons.

F	G	$F \to G$
Tr	Tr	Tr
Tr	Fs	Fs
Fs	Tr	Tr
Fs	Fs	Tr

The first reason is that you should remember this basic table, and use it as a shortcut (rather than only using the two definitional tables), when working out the truth table of a formula given in abbreviated form involving one or more "→"s.

Exercise 2.0**.** Work out the truth table of each of the following from Exercises 1.5, 1.6 : (This is pretty brutal, so you might want to only do a few—one point I'm trying to illustrate is that increasing the number of propositional variables really blows up the amount of work needed! However, the Marquis de Masoche would doubtless enjoy doing this exercise; he might even find it arousing.)

(i) $P \wedge \neg Q \to \neg(R \wedge S)$ (ii) $P \wedge (\neg Q \to \neg R \wedge S)$

(iii) $P \wedge (\neg Q \to \neg(R \wedge S))$ (iv) $P \wedge ((\neg Q \to \neg R) \wedge S)$

(v) $(P \wedge (\neg Q \to \neg R)) \wedge S$ (vi) $(P \wedge \neg Q \to \neg R) \wedge S$

The second reason for writing down the basic truth table for the connective → is to point out the following. Many treatments of this whole subject will include the "→" as a basic part of the language—that is, as one of the

original symbols. In these treatments, $(P_| \to P_{||})$ would be an actual formula, not an abbreviation for one. (I have temporarily gone back to the real names of the first two propositional variables here, hopefully avoiding confusion and not creating it!) So, in these treatments, the table just above is a basic definitional one. It seems better to have "$\to$" as an abbreviation (as we're doing) for reasons of economy, especially later in the proof system.

But in addition, you'll often find a lot of baffle-gab at this point, in a treatment from the other point of view, about why it should be that "false implies true"—look at the second last line of the table above. From our point of view, there's no need to fuss around with silly examples such as: "If $2 + 3 = 7$, then the moon is not made of green cheese." These come from a misunderstanding of the role of "$\to$" in logic. It is certainly true that the word "implies" and the combination "If ..., then $\cdots$" are used in several different ways in the English language, especially in the subjunctive. But when used to translate "$\to$" in classical logic, they mean exactly the same as saying "Not, both... and not $\cdots$". So if Joe wants to know whether this particular (very central) bit of logic is truly applicable to some English sentence involving the word "implies" or the combination "If ..., then $\cdots$", then he need only translate it into the "Not, both... and not $\cdots$" form and decide for himself whether that's what he really is saying in the original sentence. For example, the silly sentence above becomes "It is not the case that both $2+3 = 7$ and the moon is not not made of green cheese." (The "not not" there is **not** a typo!) So in the case of a "false implies true" statement, you've got a *double* assurance that the sentence is true in the propositional logic sense, whereas for the other two cases where such a sentence is true, it reduces to 'not both F and $\neg G$' such that only *one* of F and $\neg G$ is false.

I don't mean to imply that a discussion of the other word usages related to "'if—then" is a waste of time. But it is not really part of the subject which we are studying here. It is part of a subject which might have the name "Remedial Thinking 101".

Now is a good time to introduce two more *binary connectives* which play a big role in other treatments, and are of considerable interest in our discussions too. The following are very frequently occurring abbreviations:

$(F \vee G)$	abbreviates	$\neg(\neg F \wedge \neg G)$;
$(F \leftrightarrow G)$	abbreviates	$(F \to G) \wedge (G \to F)$.

As before, we'll drop the outermost brackets in such an abbreviated formula on its own, but restore them when it is used to build a bigger formula. Usually we use the following rules to get rid of some inner brackets ('precedence'):

> (III) For bracket abbreviations, treat $\vee$ and $\wedge$ on an equal footing, and treat $\leftrightarrow$ and $\rightarrow$ on an equal footing.
>
> For example, $P \vee Q \leftrightarrow R \wedge (S \rightarrow T)$ is a more readable version of $((P \vee Q) \leftrightarrow (R \wedge (S \rightarrow T)))$.
>
> Whereas, $(P \vee Q \leftrightarrow R \wedge S) \rightarrow T$ is a more readable version of $(((P \vee Q) \leftrightarrow (R \wedge S)) \rightarrow T)$.

Exercises. 2.1 (a) Show that the unabbreviated form of $F \leftrightarrow G$ is

$$(\neg(F \wedge \neg G) \wedge \neg(G \wedge \neg F)) \quad ,$$

and

(b) that the following are the basic truth tables for these connectives:

F	G	$F \vee G$
Tr	Tr	Tr
Tr	Fs	Tr
Fs	Tr	Tr
Fs	Fs	Fs

F	G	$F \leftrightarrow G$
Tr	Tr	Tr
Tr	Fs	Fs
Fs	Tr	Fs
Fs	Fs	Tr

(c) Because of the ambiguity of English, computer science books on logic can sometimes get into quite a tangle when dealing with "unless", in the course of giving a zoology of additional connectives. We can simply say

"P, unless Q" means the formula $\neg Q \rightarrow P$,

since that really is how the word is *normally* used in English. For example,

It rains unless I bring an umbrella

means the same as (in view of the definition above)

If I don't bring an umbrella, then it rains,

which itself means (in view of the definition of $\rightarrow$)

It never happens that I don't bring an umbrella and it doesn't rain.

Give the truth table for "P, unless Q", and compare to part (b).

Here is a brief summary of where we are now:

> At this point we have finally introduced in detail all the five connectives, ($\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$), which are commonly used in building formulae. Later, connectives in general will be discussed, but these five are the only ones that have standard notation and are used very commonly to build (abbreviated) formulae.
>
> Each formula has a truth table which tells you which truth value, Tr or Fs, it is assigned by every possible truth assignment. This is usually calculated by doing it (inductively) for each intermediate step used in building up the formula; that is, successively for each formula in a verification column. To do this, you apply the basic tables for the five connectives (basic tables which you should memorize!).

Now take a look at the table for $F \vee G$ in 2.1(b). That formula has truth value Tr as long as at least one of the two component formulae F and G is 'true'. So it seems that the meaning of "$\vee$" is "or". That is certainly how you should translate it, or at least for now how you should vocalize it. In any case, it is evident that saying "not (both not F and not G)" [which I got from the definition of $F \vee G$ just above] really does mean the same as saying "F or G". But it means "or" only when this is the so-called *inclusive or*, in which one is including the possibility of both F and G being the case. This brings up another illustration of multiple meanings in English. In fact, "or" is much more often used in English as the *exclusive or*—for example, "Sarah is either in Saskatchewan Province or in Szechwan Province." No one who knows any geography will entertain the possibility that she is in both places

simultaneously, since Canada and China are a good distance apart. So the fact that the exclusive "or" is intended becomes obvious from knowledge of the world.

We reserve "$\vee$" for the inclusive "or", that is, for the connective with the truth table displayed on the left in 2.1 (b), because that's what everybody else does in logic, and because it is the common use of "or" in mathematics. For example, saying "either $x > 2$ or $x^2 < 25$" means the same as saying that the number x is bigger than -5; no one would take it to exclude the numbers between 2 and 5, which satisfy both conditions. (By 2.1 (c), the word *unless* is normally used with the same meaning as the "inclusive or".)

We have no particular name for the binary connective that would be interpreted as corresponding to the "exclusive or". But such a connective does exist (and a computist—my abbreviation for "computer scientist"—will sometimes call it "xor"):

Exercise 2.2. Write down a truth table for such a connective, find a formula (strictly in terms of "$\neg$" and "$\wedge$") for that connective, and verify that your formula does have the appropriate truth table.

This leads nicely into the subject of **adequate connectives**. We shall leave this and several further topics on the so-called *semantics* of propositional logic to Chapter 4, where some other questions of particular interest to computists are also treated.

The introduction of truth assignments now permits us to present some extremely important definitions and results :

2.2 Truth Equivalence of Formulae.

We define two formulae to be *truth equivalent* when they are true for exactly the same truth assignments. (And so, false for the same ones as well, of course!) Let us be more formal about this important definition:

Definition. For any F and G, we write $F \overset{\text{treq}}{} G$ to mean $\mathsf{e}(F) = \mathsf{e}(G)$ for all truth assignments e

In effect, and as a means to actually decide truth equivalence for particular examples, this means that, if you make a 'giant' truth table which includes a column for each of F and G, those two are truth equivalent exactly

in the case when, on every line, you get the same entry in those two columns.

Exercise 2.3. (i) By making a truth table containing all four, show that the four formulae in the first paragraph of this chapter are indeed truth equivalent, as claimed there; in fact, replace the propositional variables by arbitrary formulae—i.e. show that $\neg\neg(F \wedge G)$, $\neg\neg F \wedge G$, $F \wedge G$ and $G \wedge F$ are all equivalent.

(ii) Show that, for any formula F, we have $F \overset{\text{treq}}{} (\neg F \to F) \overset{\text{treq}}{} \neg\neg F$.

(iii) Show that, for any formulae F and G, we have $F \vee G \overset{\text{treq}}{} G \vee F$.

This idea of truth equivalence chops up the set of all formulae, and tells us to look at it as a union of equivalence classes, each class containing all formulae which are equivalent to any one formula in that class. Actually, for this, one should first check that truth equivalence has the properties of what mathematicians generally call an equivalence relation. These are

$$F \overset{\text{treq}}{} F \;\; ; \;\; F \overset{\text{treq}}{} G \Longrightarrow G \overset{\text{treq}}{} F \;\; ; \;\; (F \overset{\text{treq}}{} G \;\; \text{and} \;\; G \overset{\text{treq}}{} H) \Longrightarrow F \overset{\text{treq}}{} H \;\; .$$

It goes without saying that we are claiming the above to be true for *every* choice of formulae F, G and H. The proofs of all three are very easy, since the relation $\overset{\text{treq}}{}$ is defined by equality after applying e, and since equality ('sameness') clearly has these three properties of reflexivity, symmetry and transitivity.

A minor point about checking truth equivalence can be made by the following example. Check that $(P \to Q) \wedge ((R \wedge S) \vee \neg(R \wedge S)) \overset{\text{treq}}{} P \to Q$. You need a 16-line truth table to do it, even though $P \to Q$ only involves two variables, because the other formula involves four. So, in general, to check that a bunch of formulae are all equivalent, you need a truth table which involves all the propositional variables in all the formulae. But if they turn out to be equivalent, then, in some sense, you will have shown that each of these formulae 'only depends on the propositional variables which they all have in common (at least up to equivalence or as far as truth or meaning is concerned)'. In the example, the formula $(P \to Q) \wedge ((R \wedge S) \vee \neg(R \wedge S))$ is independent of R and S, as far as 'truth' or 'meaning' is concerned.

There are even formulae whose truth values are completely independent of the truth values of any of their propositional variables :

Definition. A formula F is a *tautology* if and only if $\mathsf{e}(F) = \mathsf{Tr}$ for all truth assignments e.

For example, $P \vee \neg P$ is a tautology for each propositional variable P. In fact, $H \vee \neg H$ is a tautology for any formula H, and that partially explains what is going on in the example above, since the subformula $(R \wedge S) \vee \neg(R \wedge S)$ is a tautology.

A formula which only takes the truth value Fs is often called a *contradiction* (example : $H \wedge \neg H$). We won't emphasize this definition here nearly as much as that of 'tautology'. The word 'contradiction' is used in other slightly different ways in logic and mathematics, in addition to this usage. Another word used for a formula which is never true is **unsatisfiable**. In Chapters 4 and 8, this will play a central role in the search for efficient algorithms for 'doing logic'.

The set of all tautologies is a truth equivalence class of formulae (that is, any two tautologies are equivalent to each other, and no other formula is equivalent to any of them). That set sits at one end of the spectrum, and the equivalence class of all unsatisfiable formulae at the other end. 'Most' formulae are somewhere in the middle, sometimes true and sometimes false (where "*sometimes*" really means "*for some but not all truth assignments*").

Caution. Occasionally, a beginner reads more into the definition of truth equivalence than is really there. For example, let P, Q, R and S be four distinct propositional variables. In some vague sense, it can be said that the two formulae $P \to Q$ and $R \to S$ are 'made use of' for the same purposes. So in some very vague sense, which we have NOT been using, they 'mean the same thing'. But these two are definitely NOT truth equivalent, as can be seen, for example, by considering a truth assignment which makes all of P, Q, R true, but makes S false. (Check this out carefully!) It is quite easy to define a new kind of equivalence which is broader, as above, but that's not truth equivalence, and it's not nearly as useful.

Glass-house dwellers and stonethrowers.

A common saying is :

People who live in glass houses shouldn't throw stones.

A less common related expression is:

People who throw stones shouldn't live in glass houses.

Do these really mean the same thing? They certainly seem to. Of course

the intended meaning is more 'poetic' in some sense, being a warning about not being critical of someone if you are likely to be subject to exactly the same criticism. Also the word *shouldn't* can have many shades of meaning. Getting quite literal and away from the poetical, these sentences aren't even in the *assertive* or *declarative* form of sentences which propositional logic is supposed to help us analyze. They are more like *commands*, or so-called *imperative* sentences. For example,

If thou livest in a glass house, thou shalt not throw stones.

(And similarly for the other one.)

So let's replace them by declarative sentences as follows:

If x lives in a glass house, then x doesn't throw stones.

And :

If x throws stones, then x doesn't live in a glass house.

Here x is some fixed individual. So, until x is made explicit, the truth of either is unclear. We are beginning to get towards $1^{\underline{\text{st}}}$*order logic*, the subject of the second half of this book, where *variables* like x play a major role. But thinking of x as having been fixed, it's pretty clear that these two sentences are either both true or both false. In fact, we can get right back to propositional logic, and a very important truth equivalence here, as follows.

Replace "x lives in a glass house" by the propositional variable P, and replace "x throws stones" by the propositional variable Q. Then our first sentence above becomes

$$P \to \neg Q \quad ,$$

and our second sentence above becomes

$$Q \to \neg P \quad .$$

It is very easy to check that these two are truth equivalent, without any reference to the meanings which we assigned to P and Q. Do it!

Actually, converting both of these to actual, as opposed to abbreviated, formulae (in the conventions of this book), we get

$$\neg(P \wedge \neg\neg Q) \qquad \text{and} \qquad \neg(Q \wedge \neg\neg P) \quad .$$

In this form, perhaps you can see that the truth equivalence above follows from the facts that $\neg\neg H$ is truth equivalent to H, that $F \wedge G$ and $G \wedge F$ are always truth equivalent, and that $\neg K$ and $\neg J$ will necessarily be truth equivalent as long as K and J are. See Section **2.7** .

Exercises.

2.4 (a) Show that $F \to G$ is truth equivalent to $\neg F \vee G$. (This truth equivalence will be used very often later in Chapter 4, much more often than simply replacing $F \to G$ by its definition in terms of $\neg$ and $\wedge$.)

(b) Show that $H \vee \neg H$ and $\neg H \to \neg H$ are actually (abbreviations for) the *same* formula, and verify that it is a tautology via truth tables three times, using both abbreviations and the unabbreviated formula itself.

(c) Show that the formula $F \to \neg G$ discussed in the digression is truth equivalent to $\neg F \vee \neg G$. (This is yet another way to explain the symmetry from the digression which allows us to interchange F and G up to truth equivalence.)

2.5 Show that each of the following is a tautology:

$$F \to F \wedge F \quad ; \quad F \wedge G \to F \quad ; \quad (F \to G) \to (F \wedge H \to G \wedge H) \ .$$

2.6 Show that F is a tautology if and only if $\neg F$ is unsatisfiable, and also the other way round.

2.7 Show that $F \overset{\text{treq}}{} G$ if and only if $F \leftrightarrow G$ is a tautology. Deduce that if $F \leftrightarrow G$ and $G \leftrightarrow H$ are both tautologies, then so is $F \leftrightarrow H$

2.8 Show that, if $F \to G$ is a tautology, then, for all e with $\mathsf{e}(F) = \mathsf{Tr}$, we also have $\mathsf{e}(G) = \mathsf{Tr}$.

2.9 Show that, if $F_1 \wedge F_2 \to G$ is a tautology, then, for all e for which $\mathsf{e}(F_1) = \mathsf{Tr} = \mathsf{e}(F_2)$, we also have $\mathsf{e}(G) = \mathsf{Tr}$.

2.10 Prove that $(F \wedge (F \to G)) \to G$ is a tautology. [And so, by the previous question, for all e with $\mathsf{e}(F) = \mathsf{Tr} = \mathsf{e}(F \to G)$, we also have $\mathsf{e}(G) = \mathsf{Tr}$.]

2.11 Show that $F \wedge (G \wedge H) \overset{\text{treq}}{} (F \wedge G) \wedge H$.

2.12 (i) Show that $(F \to G \vee H) \leftrightarrow (F \wedge \neg G \to H)$ is a tautology.

(ii) Show that $\neg(F \vee G) \overset{\text{treq}}{} \neg F \wedge \neg G$.

(iii) Show that $F \to G \overset{\text{treq}}{} \neg G \to \neg F$.

(iv) Show that $(F \vee G \to H) \leftrightarrow (F \to H) \wedge (G \to H)$ is a tautology.

(v) Show that the following three formulae are truth equivalent:

$$F \vee G \to H \;\; ; \;\; (F \to H) \wedge (G \to H) \;\; ; \;\; \neg H \to \neg F \wedge \neg G \;\; .$$

How is this related to Exercise 1.8(v)?

2.3 Validity of Arguments.

Finally, we come to the propositional logic version of what this subject is supposed to be all about, namely, understanding when an argument is *logically sound* or *valid*. An argument in general takes the form

$$F_1 \; , \; F_2 \; , \; \cdots \; , \; F_n \; ; \quad \text{therefore } F \; .$$

It is asserting that the formula F (the conclusion) is true 'whenever' (or 'because') all the formulae $F_1 \; , \; F_2 \; , \; \cdots \; , \; F_n$ (the premisses) are true. Although it can be misleading, some might even say that the truth of F is *caused* by the truth of $F_1 \; , \; F_2 \; , \; \cdots$.

As you know, sometimes people (most of the time for politicians and PR men) make invalid arguments, deliberately or inadvertently. In the North America, the gun lobby, the medical establishment (not the researchers usually) and the tobacco companies provide many examples of this. Of course, what I've just done is give a kind of ad hominem argument, which is not a logical argument at all. But I'll leave it to you to find examples. Just watch TV news for five minutes.

We say that an argument as above is *valid* if and only if the *conclusion* F is indeed true at all truth assignments for which all of the *premisses* $F_1 \; , \; F_2 \; , \; \cdots \; , \; F_n$ are true. A technical notation for this is given in the following definition.

Definition. We write

$$\boxed{\{F_1 \; , \; F_2 \; , \; \cdots \; , \; F_n\} \;\models\; F}$$

to mean :

$$\boxed{\text{for all } \mathsf{e}, \;\; \text{if } \;\; \mathsf{e}(F_1) = \mathsf{e}(F_2) = \cdots = \mathsf{e}(F_n) = \mathsf{Tr} \; , \;\; \text{then } \;\; \mathsf{e}(F) = \mathsf{Tr} \; .}$$

So this definition is just another way of saying that the argument discussed just above is valid. It is important to note that (as with $\Longleftrightarrow$ and $=$), the verb "$\models$" is **not** part of the formal language of propositional logic, but part of our informal language for talking about propositional logic.

(The symbol $\models$ is often called the *double-turnstile*—NOT, as heard more than once, the *double-turned style*!

I grow old... I grow old...
I shall wear the bottoms of my trousers rolled.
T. S. Eliot

See also the end of this section.)

Just as with checking equivalence of formulae, checking validity of an argument is easy (intellectually) by using a truth table. (It is not easy, but often very slow, in the computational sense, as is explained in detail in Chapter 4.) Just make a giant truth table which has a column for the conclusion, and one for every one of the premisses. Then inspect each line which has a Tr under every premiss, and check that there's also one under the conclusion in that line. If this fails on even one such line, then the argument is invalid.

Examples.

(i) $\{P \to Q\ ,\ Q \to R\ ,\ \neg R\ \} \models \neg P$

To see this, just inspect the following truth table:

P	Q	R	$P \to Q$	$Q \to R$	$\neg R$	$\neg P$
Tr	Tr	Tr	Tr	Tr	Fs	Fs
Tr	Tr	Fs	Tr	Fs	Tr	Fs
Tr	Fs	Tr	Fs	Tr	Fs	Fs
Tr	Fs	Fs	Fs	Tr	Tr	Fs
Fs	Tr	Tr	Tr	Tr	Fs	Tr
Fs	Tr	Fs	Tr	Fs	Tr	Tr
Fs	Fs	Tr	Tr	Tr	Fs	Tr
Fs	Fs	Fs	Tr	Tr	Tr	Tr

Only the last row has Tr under all three premisses. But it also has Tr under the conclusion. And so the argument is valid, as claimed. (Note that we really didn't need to write down anything in the last column except the bottom entry.)

A concrete version of this valid argument might be to let :
P be a name for "Peter is a man" ;
Q be a name for "Peter is a mammal" ; and
R be a name for "Peter is furry" .
The argument becomes

If Peter is a man, then Peter is a mammal.
If Peter is a mammal, then Peter is furry .
Peter is not furry.
Therefore, Peter is not a man.

Probably, at least Peter would prefer to think that the conclusion is not true. If so, and without looking at the premisses, from the validity of the argument, you would know that at least one of the premisses is false. Undoubtedly it's the second one. (Homo sapiens isn't the only non-furry mammal—the underground dwelling *naked mole-rat* is not only virtually hairless; its 'skin' almost appears transparent, so you can imagine observing the workings of its digestive system, from the outside, if you have the stomach for it!)

(ii) $\{\ P \wedge Q \rightarrow R\ ,\ P\ ,\ R\ \} \not\models Q$, for three *distinct* propositional variables P, Q and R.

To see this, inspect :

P	Q	R	$P \wedge Q$	$P \wedge Q \rightarrow R$
Tr	Tr	Tr	Tr	Tr
Tr	Tr	Fs	Tr	Fs
Tr	Fs	Tr	Fs	Tr
Tr	Fs	Fs	Fs	Tr
Fs	Tr	Tr	Fs	Tr
Fs	Tr	Fs	Fs	Tr
Fs	Fs	Tr	Fs	Tr
Fs	Fs	Fs	Fs	Tr

The first and third rows are the only ones having Tr under all three premisses. The first also has Tr under the conclusion, *but the third doesn't*. And so the argument is invalid, as claimed.

A concrete version of this is as follows: Let
P be a name for "Becky had good technique on skis" ;
Q be a name for "Becky's skiis were expertly waxed" ; and
R be a name for "Becky won the ski race" .
The argument becomes

If Becky had good technique on skis and her skis were expertly waxed, then she won the ski race.

Becky had good technique on skis .

Becky won the ski race.

Therefore, Becky's skis were expertly waxed.

It's not hard to see why this argument is not valid. Despite possibly badly waxed skis, Becky may have had such good technique that even competitors with much better skis couldn't ski as fast. (They never can when she races USians!)

Just below is a challenge for the reader, which illustrates how arguments can be quite complicated, using many propositional variables. This also illustrates the fundamental inefficiency of truth tables, a major theme of the second half of Chapter 4. The argument is adapted from a puzzle of Lewis Carroll, via **Burris**, page 16, who treats it as a 'different kind' of logic from propositional ("universal supersyllogisms" as in Section **9.4**). Perhaps I should add that, although doing puzzles like this is plenty of fun and good for learning, doing logic is worthwhile for reasons which considerably supersede the reason for the crossword puzzle page or any of the other entertainment pages of your newspaper. It is possible to get considerable entertainment from studying problems which have an importance which extends beyond the attention span of individual humans.

Later in the book, we give a solution to this, but the location will remain a secret for now, to help motivate doing it yourself and reading assiduously.

Your challenge is to give a convincing proof of the following, using only enough writing or printing as would fill this page at most, if printed in this size of printing—it actually can be done in far less than the full page. (So an 8,192-line truth table would NOT be acceptable!)

Challenge. Prove, where the 13 capital letters denote 13 distinct propositional variables, that

$$\{ \quad P \wedge Q \wedge R \to S \quad , \quad \neg P \wedge A \to Z \quad , \quad U \wedge T \to \neg S \ ,$$

$$C \wedge A \to R \quad , \quad \neg Q \wedge U \to \neg V \quad , \quad \neg S \wedge Q \wedge T \to C \ ,$$

$$A \wedge \neg W \to X \quad , \quad V \wedge A \to Y \quad , \quad Y \wedge X \wedge \neg Z \to T \ ,$$

$$\neg V \wedge U \to Z \ , \ A \wedge Z \to \neg U \ , \ Y \wedge W \wedge P \to T \quad \} \quad \models \quad A \to \neg U$$

Although we have only talked about having a finite set of premisses, and this would almost always be the case in any application, for theoretical reasons, it is useful to extend the above definition to an arbitrary set of premisses. (Although it's not an argument from propositional logic, here's an example of a valid argument from mathematics which has an infinite set of premisses : x is a real number, $x > \frac{1}{2},\ x > \frac{2}{3},\ x > \frac{3}{4},\ x > \frac{4}{5}, \cdots\cdots$; therefore $x \geq 1$.)

Definition. Let Γ be any set of formulae, and let F be a formula. We write

$$\Gamma \models F$$

to mean :

$$\text{for all } \mathsf{e}, \quad \text{if } \mathsf{e}(G) = \mathsf{Tr} \ \text{ for all } \ G \in \Gamma \ \text{ then } \ \mathsf{e}(F) = \mathsf{Tr} \ .$$

This just repeats the earlier definition, except the notation is now chosen so as not to preclude the possibility of the set of premisses being infinite.

(The question of verbalizing this is often ignored in texts. But you have to say *something* ! I would suggest saying, when reading "$\Gamma \models F$", one of "Gamma double-turnstiles F", or "The formula F is semantically inferrable from the set, Gamma, of formulae", or just " [Gamma, therefore F] is a logically valid argument".)

Program Verification/Software Engineering

Programming languages are strictly defined languages, rather like logical languages. But their statements usually tend to resemble *imperative* sentences (commands) from natural language, rather than resembling *assertive* sentences (statements of purported fact) which our formulae in logic resemble. (We ignore functional programming languages and logic programming here.) However, notation and ideas from logic can be useful in CS, often in quite elementary ways (which can quickly become much less elementary in the general study of programme correctness). See **[CM]**, Ch. VIII.

Somewhat simplistically, a programming *command* :

If F, do D ,

can be refomulated as a *statement* :

If F, then the procedure does D ,

and then as a *formula from logic* :

$F \to G$, where G denotes "the procedure does D".

One use of the languages of logic (but perhaps closer to what is termed *remedial thinking* than logic itself), is for programmers to learn skills of translating sentences from

ordinary language into the language of propositional logic, and then of being able to detect vagueness and ambiguity when this translating is hard to do or when the 'correct' answer doesn't seem to be unique (and also to learn the diplomatic skills of forcing clients to specify their requirements with no vagueness nor ambiguity). There are languages, such as Z, which 'sit between' (1) the ordinary English (or math/English) in which a specification for a procedure is initially given and (2) the actual programming languages. Many aspects of Z resemble logical languages very closely. More specifically, these aspects resemble the $1^{\underline{\text{st}}}$order languages from Chapter 5 onwards which generalize (in a sense) our present propositional language.

At a somewhat higher level, one can use the idea of a logically valid argument from this section to make progress in verifying that a procedure or programme actually does what the specifications say it is supposed to do. Since this book is not intended as a text in computer science, let's just illustrate this with a particularly simple example, as follows.

A procedure to control the environment in a large store might use as input both the air temperature (denoted T) and the number of customers present (denoted N), being connected to a thermometer and to some device that counts bodies entering and leaving the building. As output, it controls whether the fan is blowing hot air from the furnace, cool air from the air conditioner, or just plain unaltered air. Let's denote these three commands as D_{heat}, D_{cond} and D_{plain}, respectively. A correct procedure should always (i.e. for any input) 'ask the fan' to do one and only one of these three tasks. Below we'll illustrate, for a certain set of specifications, how formulae from propositional logic might arise and be used to formally check the condition in the previous sentence.

The specification which the store owner, or his consulting heating/AC engineer, gave to our programmer is as follows:

Give us heat when the temperature is less than 18, unless there are more than 50 people and the temperature is more than 15, in which case just blow air. If the temperature is 18 or more, give us air conditioning unless there are less than 10 people in the store and the temperature is less than 24, in which case just blow air.

After doing some sort of translation of these informal specifications, then converting 'commands' to 'assertions' in the style indicated a few paragraphs above, you might find yourself with the following four formulae, where H_{++} is a formula (probably just a propositional variable) which translates "$T \geq 24$", H_+ translates "$T \geq 18$", H_- translates "$T \leq 15$", F_+ translates "$N > 50$", and F_- translates "$N < 10$"; and finally G_h, G_c and G_p are the conversions of D_{heat}, D_{cond} and D_{plain}, respectively, from 'commands' to 'assertions' as suggested several paragraphs above:

$$\begin{aligned}
\neg H_+ \wedge \neg(\neg H_- \wedge F_+) &\rightarrow G_h \wedge \neg G_p \wedge \neg G_c \\
\neg H_+ \wedge (\neg H_- \wedge F_+) &\rightarrow \neg G_h \wedge G_p \wedge \neg G_c \\
H_+ \wedge \neg(\neg H_{++} \wedge F_-) &\rightarrow \neg G_h \wedge \neg G_p \wedge G_c \\
H_+ \wedge (\neg H_{++} \wedge F_-) &\rightarrow \neg G_h \wedge G_p \wedge \neg G_c
\end{aligned}$$

Let Γ denote the set containing exactly these four formulae. Now the expectation that a correct procedure would, for any input, always turn on exactly one of the heat, the air conditioning or just the fan, gets translated into the fact that we have $\Gamma \models (G_h \text{ xor } G_c \text{ xor } G_p)$. (Recall xor, the 'exclusive or' connective, from Exercise 2.2).

Note that checking the validity of this last argument using a truth table would cause major pain (256 lines!). A much better argument is to assume you have a truth assignment with the conclusion is false, and deduce that at least one of the four premisses is false.

Exercise Prove the validity of the argument.

There is something profoundly unprofound about all this. It seems that the need for logical notation here in more complex specifications and procedures has more to do with 'logical bookkeeping', than logic itself.

2.4 Satisfiability.

Here is a definition which we really won't use until the latter part of the next chapter, but we put it here, because this is where it belongs, in a discussion of semantics.

Definition. Let Γ be any set of formulae. We say that Γ is *satisfiable* when there is at least one truth assignment e with $\mathsf{e}(G) = \mathsf{Tr}$ for all $G \in \Gamma$.

An example of an unsatisfiable set is easy to find: just take Γ to be $\{F, \neg F\}$ for any formula F, or to be any singleton set whose one element is an unsatisfiable formula, such as $\{F \wedge \neg F\}$.

One slightly curious consequence of our definition of *validity of an argument* is that, for an unsatisfiable set of premisses, any conclusion whatsoever will give a valid argument. So everything follows from a contradiction (unsatisfiable premiss). This is because there are no truth assignments at all where we have to check the truth of the conclusion. So this is another situation where you have to watch out for the politicians—they might make a very convincing argument for something, but from a set of premisses which could never be all simultaneously true. Can you think of an example?

> There is a perfume marketed under the name 'Contradiction' (for men). It might advertise: "Use this, and everything follows (you home)."

(This joke was discovered on the office door of Dave DeVidi.)

For a finite set, checking satisfiability is easy intellectually (but slow computationally) by just writing out a 'giant' truth table which has a column for every formula in the set. You simply check whether there is at least one line giving Tr in each of these columns.

In Chapter 4 we'll explain a different method for this, called *resolution*,

which is sometimes much faster. This often gives a good way to check validity of an argument, because of **Theorem 4.3** from the chapter after next :

$\{F_1\ ,\ F_2\ ,\ \cdots\ ,\ F_n\} \models F \iff \{F_1\ ,\ F_2\ ,\ \cdots\ ,\ F_n\ ,\ \neg F\}$ *is unsatisfiable.*

The proof is a direct application of the definitions, so you should try to do it as an exercise.

As for examples, first look at the invalid argument in example (ii) of the previous section. If you believe **4.3** just stated, you expect the following set to be *satisfiable* :

$$\{\ P \wedge Q\ \rightarrow\ R\ ,\ P\ ,\ R\ ,\ \neg Q\ \}$$

We simply took the set of premisses from before, and added to it the negation of the conclusion, as **4.3** says to do. (Note that **in**validity corresponds to satisfiability, since, in **4.3** , validity corresponds to **un**satisfiability.

To check directly that the set above is satisfiable, in the following truth table, you must find at least one line which has Tr under all four of the formulae in the set :

P	Q	R	$P \wedge Q$	$P \wedge Q \rightarrow R$	$\neg Q$
Tr	Tr	Tr	Tr	Tr	Fs
Tr	Tr	Fs	Tr	Fs	Fs
Tr	Fs	Tr	Fs	Tr	Tr
Tr	Fs	Fs	Fs	Tr	Tr
Fs	Tr	Tr	Fs	Tr	Fs
Fs	Tr	Fs	Fs	Tr	Fs
Fs	Fs	Tr	Fs	Tr	Tr
Fs	Fs	Fs	Fs	Tr	Tr

The third line is such a line (and is the only one in this case).

If we go back to the first example of the previous section, we expect the following set to be *unsatisfiable* :

$$\{P \rightarrow Q\ ,\ Q \rightarrow R\ ,\ \neg R\ ,\ P\ \}$$

Note that, negating the conclusion, I should have $\neg\neg P$ as the fourth element in that set. I replaced it by just P, because the two are truth equivalent, so it won't make any difference. (Almost anything of real interest in propositional logic won't be affected by replacing a formula by any other formula which is truth equivalent to it.)

To see unsatisfiability directly, inspect the same truth table as in that example:

P	Q	R	$P \to Q$	$Q \to R$	$\neg R$
Tr	Tr	Tr	Tr	Tr	Fs
Tr	Tr	Fs	Tr	Fs	Tr
Tr	Fs	Tr	Fs	Tr	Fs
Tr	Fs	Fs	Fs	Tr	Tr
Fs	Tr	Tr	Tr	Tr	Fs
Fs	Tr	Fs	Tr	Fs	Tr
Fs	Fs	Tr	Tr	Tr	Fs
Fs	Fs	Fs	Tr	Tr	Tr

Notice that none of the eight lines has Tr under every one of the four formulae in the set. So the set is indeed unsatisfiable, as expected.

To summarize the main points in the last few sections:

MAIN POINTS

There are six major definitions which play a big role in the semantics of propositional logic : **truth assignment, truth equivalence, tautology**, contradiction or **unsatisfiable formula**, **validity** of an argument, **satisfiability** of a set of formulae.

For the latter five definitions, a (slow but sure!) algorithm, for checking whether or not an example has the property, is to calculate and then inspect a suitable truth table.

For the latter four definitions, exchanging a formula for a truth equivalent formula will not change the result.

Exercises.

2.13 Prove $\{F_1, \cdots, F_n\} \models F \iff \{F_1 \wedge \cdots \wedge F_n\} \models F$.

2.14 Prove $\{G\} \models F \iff \models G \to F$, where $\models H$ means $\emptyset \models H$ (which is equivalent to H being a tautology, as can be easily seen).

2.15 Deduce that the argument $[F_1, \cdots, F_n \quad \text{therefore } F]$ is valid if and only if $F_1 \wedge \cdots \wedge F_n \to F$ is a tautology.

2.16 Show that the following six statements are all equivalent to each other; that is, each holds if and only if they all hold.

(i) $F \overset{\text{treq}}{} G$.

(ii) $F \leftrightarrow G$ is a tautology.

(iii) Both $F \to G$ and $G \to F$ are tautologies.

(iv) $\{F\} \models G$ and $\{G\} \models F$.

(v) For all Γ, $\Gamma \models F \iff \Gamma \models G$.

(vi) For all Γ and H, we have $\Gamma \cup \{F\} \models H \iff \Gamma \cup \{G\} \models H$.

2.17 (i) Show that a finite set $\{G_1, \cdots, G_k\}$ is satisfiable if and only if the formula $G_1 \wedge G_2 \wedge \cdots \wedge G_k$ is not a contradiction, i.e. not unsatisfiable in the previous sense, which is basically a special case of the new sense.

(ii) Show that $\{G_1, \cdots, G_k\}$ is unsatisfiable if and only if the formula $\neg G_1 \vee \neg G_2 \vee \cdots \vee \neg G_k$ is a tautology.

2.5 Remarks on metalanguage vs. object language.

This continues some very brief comments made earlier on this topic. I'll use both single and double boxes in a systematic way, to help make the point here.

One can speak in [English] about the [[French language]] .

The single boxes below will contain statements in the *metalanguage* (which is English here), and the double box will contain a statement in the *object language* (which is French here).

[“Maison” is a French word meaning “house”.]

[The string “la maison est ici” is a sentence in French .]

[[La maison est ici.]]

Of course, reversing the roles of the two languages makes sense here. Below it doesn't, because the language of propositional logic is very limited in what it can talk about.

Similarly, one can speak in [math/English] about the [[language of propositional logic]] .

Here the metalanguage is math/English, whereas the object language is not a *natural* language, but rather the *formal* language of propositional logic. Later in the book we'll have many more formal languages, the various $1^{\underline{\text{st}}}$order languages.

The formula $P \vee \neg P$ is a tautology.

Good style would probably require quotation marks around $P \vee \neg P$ in this example of a math/English statement about propositional logic. But this whole book consists of almost nothing but statements in math/English about the formal languages, so adding all those quotations would produce a huge mess!

Here are three more such statements. Particularly the second and third might sometimes be mistaken for formulae in propositional logic. It is very important to be aware that they are not. The 'marks' "=" and " treq " are definitely **not** symbols used **within** the language of propositional logic. (I've been careful here to make sure that the following three are true; but actually, that's irrelevant to this discussion.)

$(P \leftrightarrow Q)$ is a tautology $\iff P = Q$.

$(P \vee \neg P)$ treq $(P \to P)$.

$P \to P \;=\; \neg(P \wedge \neg P)$.

Finally here, of course, **is** a formula in the object language, the formal language:

$(P \wedge \neg P)$

One way to detect errors of this form (where you have something which you at first mistakenly believe to be a formula, but it isn't) is this: *If any symbol at all occurs which is not one of the symbols given as part of the language, then the string being considered is definitely not a formula.* The rules must be given completely at the beginning, rather like the restrictions on what you can feed into a computer, when writing software. For propositional logic, the only allowable symbols are the brackets, the propositional variables, and the connectives. (Normally there are the five famous connectives, all acceptable here, even though we started out with only the first two, $\neg$ and $\wedge$.)

Of course *most* strings which do only use the acceptable symbols still are not formulae—recall Exercise 1.2 where 714 out of 720 of the possible strings using a certain sequence of symbols turned out to not be formulae. There is a conflict of terminology with some other logic books of which you should be aware. Once in ancient times, someone had the really dumb idea of giving the name 'formula' to an arbitrary string of symbols—somehow the 1-syllable word 'string' wasn't adequately profound for this. Then what we call a formula was named a 'well-formed formula', getting us up to 3 words and 5 syllables, quite a tongue-twister for something to be repeated innumerable times. So then that was abbreviated to wff, which at least has the mysteriousness about it to impress the naively impressionable! So our [*string—formula*] becomes their [*formula—well-formed formula*] or [*formula—wff*]. There is even one CS text used at my university which first introduces

this unfortunate notation, and soon switches without any warning to the reader to use "formula" only for what was going to be called a "wff". To add to the content of the book, perhaps something confusing was needed to convince readers that logic is a profound and difficult subject!

Here again are the main examples of symbols that we use in the 'math' part of math/English, and which **must not be confused for symbols in the language of propositional logic** :

$$= \quad , \quad \overset{\text{treq}}{} \quad , \quad \Longrightarrow \quad , \quad \Longleftrightarrow \quad , \quad \models \quad , \quad \vdash \text{ (see Ch. 3)}$$

Coming from the other direction, possibly a more common error is to use a formula from propositional logic directly within the math/English metalanguage as though that formula were making a statement in math/English. Such a formula must always be regarded just as a single object being referred to (as in the single boxes above). For example, the following statement is really **meaningless**:

Once we know $F \to G$, then we get $\{F\} \models G$,

since the languages are being mixed when $F \to G$ is being used as though it were a phrase in math/English. What the speaker was trying to say was probably one of the following four statements, all of which are not only meaningful, but correct (and all are really saying the same thing):

Once we know that, for some given formulae F and G, the formula " $F \to G$ " is a tautology, it then follows that $\{F\} \models G$,

or

If $F \to G$ is a tautology, then $\{F\} \models G$,

or

$(\models F \to G)$ implies that $\{F\} \models G$,

or

$(\models F \to G) \implies \{F\} \models G$.

In each case, $F \to G$ is treated as a single object, and the quotation marks in the first one are probably the best style to make this clear, but become tedious, so are usually dropped.

Even by those who disagree with what is called the **formalist philosophy of mathematics**, there ought to be general appreciation of how that school, founded by Hilbert, brought a considerable degree of clarity to discussions of logic and the foundations of mathematics by emphasizing the **metalanguage versus object language** viewpoint. Although Frege, in the 19th century, had initiated this clarification process, one can read surprisingly confused paragraphs by justifiably famous philosopher/mathematicians from the period just before Hilbert's influence was felt, the confusion apparently caused by failing to emphasize this distinction between languages. Even discussion of a question as simple as: *What's the difference between " $\neg F$ " and " F is false "?* become lengthy and seemingly inconclusive without this dichotomy of languages. Our answer would simply

be that "$\neg F$" belongs to the object language, whereas "F is false" is a statement in the metalanguage (albeit one which is ambiguous without mentioning which truth assignment is referred to).

> $\cdots$ *one finds listed among the "axioms":* a proposition implied by a true premise is true. *Frege would never have permitted such a confusion between the formal language and* $\cdots$ *natural language* $\cdots$
>
> Martin Davis

Exercise. Discuss the similarities and differences between the following six sentences:
(i) The dog is black. (ii) It is true that the dog is black.
(iii) It is true that it is true that the dog is black.
(iv) It is not true that the dog is not black.
(v) It is true that it is not true that the dog is not black.
(vi) The dog is not not black.

2.6 Associativity and non-associativity.

The truth equivalences

$$(F \wedge G) \wedge H \ \overset{\text{treq}}{} \ F \wedge (G \wedge H) \qquad \text{and} \qquad (F \vee G) \vee H \ \overset{\text{treq}}{} \ F \vee (G \vee H)$$

are very basic, intuitively obvious, and easy to prove. In Chapter 4 we shall often avoid many pairs of brackets by writing just $F \vee G \vee H$, and also $F \wedge G \wedge H$. Strictly speaking, this is ambiguous. But almost everything we say in that chapter remains true when a formula is replaced by another which is truth equivalent to the first, so no real confusion is caused by this avoidance of 'associativity brackets'.

When more than three ingredients are involved, say with $\vee$, we shall use expressions such as $F_1 \vee F_2 \vee \cdots \vee F_k$. The basic associativity law for $\vee$ above is easily iterated to show that all the formulae obtainable from $F_1 \vee F_2 \vee \cdots \vee F_k$ by inserting enough pairs of brackets are truth equivalent to each other. The same is true for $\wedge$.

On the other hand,

the binary connective $\rightarrow$ *is* **not** *associative up to truth equivalence.*

In particular,

$$(P \rightarrow Q) \rightarrow R \quad \overset{\text{treq}}{\not} \quad P \rightarrow (Q \rightarrow R)$$

for three distinct propositional variables P, Q and R. Prove this!

On the other hand, though it is a bad idea to remove the brackets from either of $F \leftrightarrow (G \leftrightarrow H)$ or $(F \leftrightarrow G) \leftrightarrow H$, they turn out, oddly enough, to

be truth equivalent. Prove this also. These two formulae appear to have quite different meanings. Their truth equivalence is one of a number of examples of the limitations of propositional logic, and of the dangers of arguing intuitively, rather than with careful definitions and calculations. I speak from personal experience, having at one time made exactly that silly mistake of thinking $\leftrightarrow$ to be non-associative up to equivalence!

Now sometimes in mathematics a statement of the form $\mathcal{S} \iff \mathcal{T}$ is proved in the form

$$\mathcal{S} \iff \mathcal{R}_1 \iff \mathcal{R}_2 \iff \mathcal{R}_3 \iff \cdots \iff \mathcal{R}_k \iff \mathcal{T} \ .$$

This makes perfectly good sense—it is shorthand for the sequence of statements in math/English as follows :

$$\mathcal{S} \iff \mathcal{R}_1 \quad \text{and} \quad \mathcal{R}_1 \iff \mathcal{R}_2 \quad \text{and} \quad \cdots \quad \text{and} \quad \mathcal{R}_k \iff \mathcal{T} \ .$$

An example from a linear algebra text, referring to a square matrix A, might be :

$$A \text{ has an inverse} \iff A \text{ is row equivalent to } I \iff \det(A) \neq 0 \iff$$

$$\text{the linear system } (x_1, \cdots, x_n)A = (0, \cdots, 0) \text{ has only the trivial solution.}$$

But note carefully that a string of symbols such as $P \leftrightarrow Q \leftrightarrow R$, and corresponding ones with four or more ingredients, definitely are not formulae in propositional logic. Even when enough brackets are inserted to get a formula, such a formula really bears no relationship to the math/English jargon just above.

Facts from propositional logic related to, for example,

$$\mathcal{S} \iff \mathcal{R} \iff \mathcal{T} \ .$$

would be that

$$\{ \ S \leftrightarrow R \ , \ R \leftrightarrow T \ \} \models (S \leftrightarrow T) \ ,$$

and that the formula

$$(S \leftrightarrow R) \wedge (R \leftrightarrow T) \rightarrow (S \leftrightarrow T)$$

is a tautology. Here S, R and T could be any formulae, not merely propositional variables.

2.7 The Replacement Theorem and subformulae.

To begin with a couple of examples, firstly we can ask the following question. Since we know that $F \overset{\text{treq}}{} \neg\neg F$, can we immediately conclude that both of the following hold?

$$G \wedge F \ \rightarrow \ H \vee G \quad \overset{\text{treq}}{} \quad G \wedge \neg\neg F \ \rightarrow \ H \vee G$$
$$F \vee H \ \rightarrow \ G \wedge F \quad \overset{\text{treq}}{} \quad \neg\neg F \vee H \ \rightarrow \ G \wedge F$$

Notice that, in each case, the right-hand side is simply obtained by replacing one copy of F on the left by $\neg\neg F$. (The second one has another occurrence of F which is left unaltered.)

The second question is similar. From the fact that $F \wedge G \quad \overset{\text{treq}}{} \quad G \wedge F$, which is pretty obvious, can we immediately conclude that

$$(F\wedge(H \rightarrow J))\vee(F\wedge G) \ \rightarrow \ G\wedge H \quad \overset{\text{treq}}{} \quad (F\wedge(H \rightarrow J))\vee(G\wedge F) \ \rightarrow \ G\wedge H \ ?$$

Here, the only occurrence of $F \wedge G$ has been replaced by $G \wedge F$.

The answer to all three questions is yes. These are cases of the theorem of this section, which tells us, among other things, that $XFY \overset{\text{treq}}{} X\neg\neg FY$ and $X(F\wedge G)Y \overset{\text{treq}}{} X(G\wedge F)Y$, where X and Y are as described just below in the most general case.

If F is a formula, and X and Y are strings of symbols ("finite horizontal arrays"), we are interested in the situation where the array XFY, obtained by juxtaposing these three strings, is actually a formula. In fact, we'd usually think of this the other way round, starting with a 'big' formula, and looking inside for a consecutive string which is a formula F, and then denote by X everything to the left of F in the big formula, and by Y, everything to the right of F.

Definition. In this case we say that F is a *subformula* of the formula XFY. In fact, we have even specified a particular occurrence of F as a subformula. It may occur more than once.

For example, $(G \wedge H)$ is a subformula of

$$(((\neg(F \wedge \neg(G \wedge H))) \wedge (F \wedge G)) \wedge H) \ ,$$

and it only occurs once. In this case

$$X \ = \ (((\neg(F \wedge \neg \qquad \text{and} \qquad Y \ = \ )) \wedge (F \wedge G)) \wedge H) \ .$$

Note that neither X nor Y is a formula; in fact, this is always the way it happens. Furthermore, we cannot take the 1st, 2nd, 3rd, 6th and 9th symbols from the right-hand end, and claim that they form another occurrence of the subformula $(G \wedge H)$. Subformulae are consecutive strings which give a formula.

The formula above has many other subformulae, including itself as the biggest subformula. Can you find all these subformulae? See further down (and also **Appendix B** to Chapter 8) for systematic ways to do this which are better than just considering every possible consecutive string.

The main point of this section is to deal with the following important theorem.

Theorem 2.1 *If F, XFY and G are formulae, then the formula XGY (obtained from XFY by replacement of that occurrence of F by G) is truth equivalent to the original formula XFY, as long as G is equivalent to F.*

Symbolically,

$$F \ \stackrel{\text{treq}}{} \ G \implies XFY \ \stackrel{\text{treq}}{} \ XGY \ .$$

Examples.

$$X\neg\neg FY \ \stackrel{\text{treq}}{} \ XFY \quad ;$$
$$X(F \wedge G)Y \ \stackrel{\text{treq}}{} \ X(G \wedge F)Y \quad ;$$
$$X((F \wedge G) \wedge H)Y \ \stackrel{\text{treq}}{} \ X(F \wedge (G \wedge H))Y \quad .$$

'Proof'. We proceed by induction on formulae as follows. Suppose that the theorem is false. Among all situations with $F \ \stackrel{\text{treq}}{} \ G$, but XFY not truth equivalent to XGY, choose one in which a formula verification column for XFY is as short as possible. By replacing F by G at the relevant occurrence all the way up and down that column, you get a formula verification column for XGY. Now because of choice of a shortest column where 'things go wrong', all corresponding pairs of lines in these two columns, except the last pair, are equivalent. But then the last pair must also be equivalent (completing the proof), because those last lines are both obtained either by negating a pair of equivalent lines further up, or by conjuncting two pairs of equivalent lines. The last sentence depends on the easily checked fact that,

$$[\, F_1 \ \stackrel{\text{treq}}{} \ G_1 \text{ and } F_2 \ \stackrel{\text{treq}}{} \ G_2 \,] \implies [\, \neg F_1 \ \stackrel{\text{treq}}{} \ \neg G_1 \text{ and } F_1 \wedge F_2 \ \stackrel{\text{treq}}{} \ G_1 \wedge G_2 \,] \, .$$

(If we'd had the other three famous connectives as part of our basic set of symbols, there'd have been a bunch more fooling around checking here !)

Exercise 2.18 Prove the assertion in the display just above.

What's wrong with this so-called 'proof'? Firstly, how do we know that XGY is even a formula? And if it is, how do we know that the column mentioned in the proof is in fact a formula verification column for XGY ? Subformulae have been introduced as above in order to be straightforward with the definition and the notation. Everything above is correct, but somewhat incomplete. Furthermore, there is an alternative way of dealing with subformulae which is a lot more convenient for theoretical purposes.

Here is this less direct approach to subformulae. This is an example of *defining* (as opposed to proving) by induction on formulae:

> (i) The only subformula occurrence in P is P itself, where P is any propositional variable.
> (ii) The subformula occurrences in $\neg F$, besides $\neg F$ itself, are all the subformula occurrences within F.
> (iii) The subformula occurrences in $(F \wedge G)$, besides $(F \wedge G)$ itself, are all subformula occurrences within F, and all subformula occurrences within G.

And then we say that a *subformula* of a given formula H is defined to be any formula which is the string of at least one subformula occurrence in H. One needs to prove (rather easy) that the strings which are subformula occurrences as defined inductively above *are* actually formulae. Not quite as easy is to prove that our original straightforward definition of a subformula occurrence amounts to the same thing as this inductive definition. These proofs are given in the third section of **Appendix B** to Chapter 8. We have been referring to actual (unabbreviated) formulae above, but analogous facts hold for abbreviated formulae as well. So does the proof that, if F, XFY and G are all formulae, then so is XGY (whether or not G is equivalent to F).

A more mathematically stylish statement and inductive proof can now be given to the previous theorem as follows.

Theorem 2.1* *If* $F \overset{treq}{} G$ *, then* $H \overset{treq}{} J$ *, where* J *is obtained by replacing any single occurrence of* F *in* H *by* G *.*

Proof. Proceed by induction on H. For the initial case, if H is a propositional variable, then the only possibility for F is H, and so $J = G$. Thus the conclusion $H \ ^{\text{treq}} \ J$ just amounts to repeating the assumption $F \ ^{\text{treq}} \ G$.

For the two possibilities for the inductive step, first assume that $H = \neg H_1$, and the theorem holds with H_1 in place of H. If F is all of H, the argument is identical to that in the initial case. If not, then F is occurring as a subformula of H_1 (by the inductive definition). But then $J = \neg J_1$, where J_1 is obtained from H_1 by replacing that occurrence of F by G . By our inductive assumption, $H_1 \ ^{\text{treq}} \ J_1$. Therefore $\neg H_1 \ ^{\text{treq}} \ \neg J_1$, that is, $H \ ^{\text{treq}} \ J$, as required.

The other possibility for the inductive step is that $H = (H_1 \wedge H_2)$, and the theorem holds with either H_1 or H_2 in place of H. If F is all of H, once again the argument is identical to that in the initial case. If not, then F is occurring as a subformula of H_1 or of H_2 (by the inductive definition). In the former case, $J = (J_1 \wedge H_2)$, where J_1 is obtained from H_1 by replacing that occurrence of F by G . By our inductive assumption, $H_1 \ ^{\text{treq}} \ J_1$. Therefore $(H_1 \wedge H_2) \ ^{\text{treq}} \ (J_1 \wedge H_2)$, that is, $H \ ^{\text{treq}} \ J$, as required. The other case, where F is occurring as a subformula of H_2, has almost the same argument, which you should write down as an exercise.

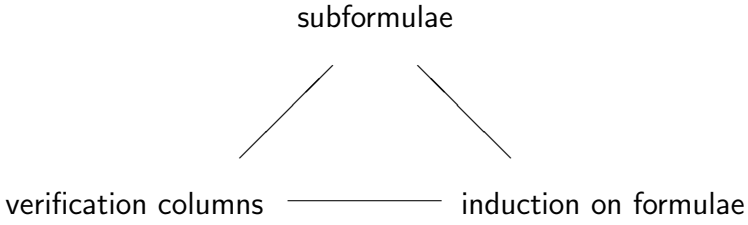
This completes the proof.

Exercise 2.19 Write out the argument for the other case just above.

Again it is very convenient to not need to deal with the other three famous connectives in that last proof, by making them subordinate to our basic ones, $\neg$ and $\wedge$, right from the beginning.

If you have constructed formula verification columns for several formulae, much of this new approach must seem like dejá vu. The reason is that every one of the subformulae of a formula must occur in a verification column for that formula. We really have defined *formula* in a way that says that it is an object built up from its subformulae. So the process of checking that a finite horizontal array of symbols *is* a formula, and the process of finding all subformulae of that formula, are virtually identical. See the 2nd section of **Appendix B** to Chapter 8 for complete details on this. Furthermore, in dealing with abbreviations which use the other three famous connectives, you can determine all the subformulae (also in abbreviated form) by the same process of gradually breaking down the formula into its components.

There is a triad here of closely connected topics:



The reader is invited to stare at this for awhile, and ruminate about the connections between any two of these.

Exercises

2.20 In Exercise 1.9, you found more readable abbreviated versions of the following formulae. Now make a truth table for each, and decide which are equivalent to which.

(i) $((P \wedge (Q \wedge \neg P)) \rightarrow (\neg Q \rightarrow (R \wedge P)))$

(ii) $(((P \wedge Q) \wedge \neg P) \rightarrow (\neg Q \rightarrow (R \wedge P)))$

(iii) $(P \wedge ((Q \wedge \neg P) \rightarrow (\neg Q \rightarrow (R \wedge P))))$

(iv) $((P \wedge ((Q \wedge \neg P) \rightarrow \neg Q)) \rightarrow (R \wedge P))$

(v) $(P \wedge (((Q \wedge \neg P) \rightarrow \neg Q) \rightarrow (R \wedge P)))$

2.21 (i) The formula $\neg Q \rightarrow P$ could be translated as 'P, unless Q'. Determine its truth table. Find an abbreviated formula with only three symbols which is equivalent to it, and a third one with four symbols, equivalent but unequal to it.

(ii) Find a formula translatable as 'not both P and Q'— often called "nand" by computists. Find two abbreviated formulae translatable as 'neither P nor Q'—called "nor", one with 5 symbols and one with 6, counting brackets. Work out the truth tables for all three.

(iii) For each of the 16 possible truth tables for a formula using at most two propositional variables, P and Q, (i.e. for all choices of Tr and Fs for the four slots in the last column), find such a formula with that truth table.

(iv) Find a formula translatable as '(if P then Q) else R'— similar to ones often used by computists. Work out its truth table.

2.22 Each of the following paragraphs is from some older literature. You will find in each an argument. Convert it into an argument in the form of several premisses, then a conclusion, in English. Try to add any needed extra premisses which aren't stated by the author (perhaps because they are obvious facts about the world), but which would make the argument closer to being 'purely logical'. These arguments are examples where it is hard, even perhaps impossible, to convert what you have into an argument in propositional logic. But 1st order logic later is more expressive, so we'll get you to translate these into arguments in a 1st order language, at the appropriate point. See Exercise 9.12.

(i) *In the beginning man was born from creatures of a different kind; because other creatures are soon self-supporting, but man alone needs prolonged nursing. For this reason he would not have survived if this had been his original form.*

—Anaximenes

(ii) *It is by no means established that the brain of a woman is smaller than that of a man. If it is inferred because a woman's bodily frame generally is of less dimensions than a man's, this criterion would lead to strange consequences. A tall and large-boned man must on this showing be wonderfully superior in intelligence to a small man, and an elephant or a whale must prodigiously excel mankind.*

—John Stuart Mill

(iii) *Some whales have been captured far north in the Pacific, in whose bodies have been found the barbs of harpoons darted in the Greenland seas. Nor is it to be gainsaid, that in some of these instances it has been declared that the interval of time between the two assaults could not have exceeded very many days. Hence, by inference, it has been believed by some whalemen, that the Nor'-West Passage, so long a problem to man, was never a problem to the whale.*

—Herman Melville

Oh Jonah, he lived in the whale.
Oh Jonah, he lived in the whale.
For he made his home in
that mammal's abdomen.

⋮

•

It ain't necessarily so.

—Ira Gershwin, with minor 'corrections'.

(iv) *Let us reflect in another way, and we shall see that there is great reason to hope that death is good; for one of two things—either death is a state of nothingness and utter unconsciousness, or as men say, there is a change and migration of the soul from this world to another. Now if you suppose that there is no consciousness, but a sleep like the sleep of him who is undisturbed even by dreams, death will be an unspeakable gain . . . for eternity is then only a single night. But if death is the journey to another place, and there, as men say, all the dead abide, what good, O my friends and judges, can be greater than this?. . . Above all, I shall then be able to continue my search into true and false knowledge; as in this world, so also in the next; and I shall find out who is wise, and who pretends to be wise, and is not.*

—Socrates

2.23 Many of the following are analogous to propositions proved in the next chapter. Those propositions, together with something called soundness, will imply the correctness of these assertions about argument validity. **But you give direct proofs of them.** (Some of these appear with different wording as an exercise in the next chapter.)

(i) $\models (Q \wedge R) \wedge P \rightarrow P \wedge R$.
(Recall that this means $\emptyset \models (Q \wedge R) \wedge P \rightarrow P \wedge R$, where $\emptyset$ is the empty set, so really it's just saying that the formula is a tautology.)

(ii) $\models (F \rightarrow G) \rightarrow (\neg G \rightarrow \neg F)$

(iii) $\{F \rightarrow G\} \models \neg G \rightarrow \neg F$

(iv) $\{F \rightarrow G\ ,\ \neg G\} \models \neg F$

(v) $\{F \rightarrow G\ ,\ G \rightarrow H\} \models F \rightarrow H$.

(vi) $\{F \rightarrow G\} \models F \rightarrow F \wedge G$

(vii) $\{F \rightarrow (G \rightarrow H)\} \models F \wedge G \rightarrow H$.

(viii) $\{F \wedge G \to H\} \models F \to (G \to H)$.

(ix) $\{F\ ,\ G\} \models F \wedge G$

(x) $\{G\} \models F \to G$.

(xi) $\{F \to G\ ,\ F \to (G \to H)\} \models F \to H$.

(xii) $\models F \wedge \neg F \to G$.

(xiii) $\{F\ ,\ \neg F\}\ \models\ G$.

2.24 Later we shall make great use of the following principle: *To check the validity of an argument, it suffices to check that the set consisting of the premisses together with the negation of the conclusion is* **unsatisfiable**. (See **4.3**.) Actually the converse is also true. Because of this, the following exercise is really 'the same' as the previous one. But you directly check that each of the following sets is unsatisfiable.

(i) $\{\neg((Q \wedge R) \wedge P \to P \wedge R)\}$

(ii) $\{\neg((F \to G) \to (\neg G \to \neg F))\}$

(iii) $\{F \to G\ ,\ \neg(\neg G \to \neg F)\}$

(iv) $\{F \to G\ ,\ \neg G\ ,\ \neg\neg F\}$

(v) $\{F \to G\ ,\ G \to H\ ,\ \neg(F \to H)\}$.

(vi) $\{F \to G\ ,\ \neg(F \to F \wedge G)\}$

(vii) $\{F \to (G \to H)\ ,\ \neg(F \wedge G \to H)\}$.

(viii) $\{F \wedge G \to H\ ,\ \neg(F \to (G \to H))\}$.

(ix) $\{F\ ,\ G\ ,\ \neg(F \wedge G)\}$

(x) $\{G\ ,\ \neg(F \to G)\}$.

(xi) $\{F \to G\ ,\ F \to (G \to H)\ ,\ \neg(F \to H)\}$.

(xii) $\{\neg(F \wedge \neg F \to G)\}$.

(xiii) $\{F\ ,\ \neg F\ ,\ \neg G\}$.

2.25 Prove: $\Gamma \cup \{F\} \models G \iff \Gamma \models F \to G$.

2.26 Prove: *For a set of formulae* Γ*, we have* $\Gamma \models G$ *for all* G *if and only if, for some formula* F*, both* $\Gamma \models F$ *and* $\Gamma \models \neg F$.
Find a single word which characterizes such a set Γ.

2.27 Prove that the following formula is a tautology.

$$(F \wedge G \to H) \ \to\ (F \to H) \vee (G \to H)\ .$$

(This is a slightly off-putting example to some, since it could be misinterpreted as saying that if H follows from $F \wedge G$, then it follows from one or the other of F or G.)

Show that

$$\{F \wedge G \to H\} \ \models\ (F \to H) \vee (G \to H)$$

is correct, but the following is NOT correct :

$$\{F \wedge G \to H\} \ \models\ F \to H \qquad \text{or} \qquad \{F \wedge G \to H\} \ \models\ G \to H\ .$$

2.28 Using the formula

$$(F \to G \vee H) \ \to\ (F \to G) \vee (F \to H)\ ,$$

invent a question analogous to the previous one, and answer it.

2.29 Decide which if any of the following are correct, and justify your answers.

(i) $\{F\} \ \models\ G \vee H \qquad \Longrightarrow \qquad \{F\} \models G \quad \text{or} \quad \{F\} \models H$;

(ii) $\{F\} \ \models\ G \wedge H \qquad \Longrightarrow \qquad \{F\} \models\ G \quad \text{and} \quad \{F\} \models H$;

(iii) $\{F \vee G\} \ \models\ H \qquad \Longrightarrow \qquad \{F\} \models\ H \quad \text{and} \quad \{G\} \models H$.

(iv) Is the answer to either (i) or (ii) different with $\{F\}$ changed to some other set Γ ?

2.30 (i) Show that $\{P, Q, \neg(P \wedge Q)\}$ is an unsatisfiable 3-element set, each of whose 2-element subsets is satisfiable.

(ii) For every $n \geq 3$, find an example of an unsatisfiable n-element set, each of whose $(n-1)$-element subsets is satisfiable.

2.30.5 After doing this exercise, you might want to look up **Christensen**, to see that even as late as 25 years ago, scholars could actually publish in reputable places the kind of nonsense arising from misunderstanding the distinctions between colloquial uses of the verb 'to imply', and our symbols '$\rightarrow$' and '$\models$'. In this case, the author has confused (i) and (ii) below with (iii).

Prove the following:

(i) ' $\neg(F \rightarrow G) \ \rightarrow \ F \wedge \neg G$ ' is a tautology.

(ii) $\{\neg(F \rightarrow G)\} \models F \wedge \neg G$.

(iii) "If $\{F\} \not\models G$, then $\models F \wedge \neg G$ " is a false statement (in general—shouldn't need to say that!).

2.8 Maximal satisfiable sets and the compactness theorem.

The following involve **infinite** sets of formulae. Some of them are more difficult than earlier exercises.

2.31 Given a truth assignment e, define $\Delta_{\mathsf{e}} := \{ F : \mathsf{e}(F) = \mathsf{Tr} \}$, the set of all formulae which are true at e. (Answers to the following are brief.)

(i) Show that, for all formulae G, exactly one of G or $\neg G$ is in Δ_{e} .

(ii) Show that $G \wedge H \in \Delta_{\mathsf{e}}$ if and only if both G and H are in Δ_{e} .

(iii) Show that Δ_{e} is satisfiable. (Very brief!)

(iv) Show that $\Delta_{\mathsf{e}} \models F \Longrightarrow F \in \Delta_{\mathsf{e}}$. (Converse totally trivial!)

A set Δ of formulae which satisfies (i) and (ii) in the previous problem we shall temporarily call a **maxsat** set. That is, Δ is maxsat iff it satisfies

(i) for all G, exactly one of G or $\neg G$ is in Δ ;

and

(ii) $G \wedge H \in \Delta \iff (G \in \Delta \text{ and } H \in \Delta)$.

2.32 (i) Show that **a maxsat set is satisfiable** (so it satisfies 2.31(iii)).

Hint: Think about what a possible e could be such that $\Delta = \Delta_{\mathsf{e}}$. Then go right back to the second definition of *truth assignment*.
(The proof of 2.31(iii) is very easy, but this shows that there is another, more abstract, proof. See also 2.36)

(ii) Combining this with 2.31, show how there is a natural 1-1 correspondence between the set of all truth assignments and the set of all maxsat sets.

(iii) Show that a set of formulae is satisfiable if and only if it is a subset of some maxsat set.

(iv) Show that no maxsat set is a subset of any other satisfiable set.

2.33 Prove that both conditions in defining *maxsat set* are necessary, perhaps as follows.

(i) Show that the set of all tautologies satisfies condition (ii), but not (i).

(ii) Show that the following set, Δ, satisfies (i) but not (ii). Let $\mathcal{P}$ be the set of all formulae—(in the strict sense—only $\neg$ and $\wedge$ as connectives—no abbreviations)—whose leftmost symbol is not $\neg$. Then define

$$\Delta \;:=\; \{J : J \in \mathcal{P}\} \cup \{\neg\neg J : J \in \mathcal{P}\} \cup \{\neg\neg\neg\neg J : J \in \mathcal{P}\} \cup \{\neg\neg\neg\neg\neg\neg J : J \in \mathcal{P}\} \cup \cdots ,$$

That is, Δ is defined to be the set of all formulae beginning with an even number of $\neg$'s. (We are definitely counting brackets as being symbols, so, for example, $(\neg P \wedge Q) \in \Delta$. On the other hand, $P \to Q$ is not in Δ, since, as an actual formula, it begins with one $\neg$ followed by a bracket, and 1 is odd, according to the latest calculations!)
Alternatively, you could do this by proving that the complement of a maxsat set, within the set of all formulae, satisfies (i) but not (ii).

Call a set Γ of formulae **finitely satisfiable** if and only if every finite subset of Γ is satisfiable.

2.34 Show that Γ cannot be finitely satisfiable if neither $\Gamma \cup \{F\}$ nor $\Gamma \cup \{\neg F\}$ are. Hint: Could a finite subset $\Omega \cup \Lambda$ of Γ be satisfiable if neither $\Omega \cup \{F\}$ nor $\Lambda \cup \{\neg F\}$ are?

Given that Γ is finitely satisfiable, and given a listing $F_0, F_1, F_2, \cdots$ of the set of all formulae, define a list, of larger and larger sets of formulae, $\Gamma_0, \Gamma_1, \Gamma_2, \cdots$, as follows : $\Gamma_0 := \Gamma$ and, given Γ_n, let

$$\Gamma_{n+1} := \begin{cases} \Gamma_n \cup \{F_n\} & \text{if the latter is finitely satisfiable ;} \\ \Gamma_n \cup \{\neg F_n\} & \text{if not .} \end{cases}$$

Expressed in words: To form Γ_{n+1}, add to Γ_n one formula (if it's not already in Γ_n). The formula to be added is either F_n or $\neg F_n$, according to whether adding F_n to Γ_n produces a finitely satisfiable set, or not, respectively.

2.35 (i) Prove by induction on n that every Γ_n is finitely satisfiable. (You are given that Γ_0 is. Don't forget 2.34)

(ii) Deduce that the set $\Delta := \cup_{n=0}^{\infty} \Gamma_n$ is finitely satisfiable. In particular, conclude that for no G are both G and $\neg G$ in Δ.

(iii) Show that Δ in (ii) is maxsat.

(iv) Combine all this to conclude that you have a proof of the famous **Compactness Theorem** : *Any finitely satisfiable set is satisfiable.*

The above proof of the Compactness Theorem for propositional logic depends on doing 2.32(i), 2.34 and 2.35. Compactness also follows very easily from the main result of the next chapter. The proof we give of that main result involves a construction very similar to the way Δ above is constructed from Γ. See **3.13.**

Here are two examples of non-obvious mathematical facts which can be proved using compactness.

Matching. Given infinite sets X and Y, let Z be the set of all finite non-empty subsets of Y, and let $f : X \to Z$ be any function. Suppose that for any finite subset X_1 of X there is an injective function $g_1 : X_1 \to Y$ such that $g_1(x) \in f(x)$ for all $x \in X_1$. Then there is an injective function $g : X \to Y$ such that $g(x) \in f(x)$ for all $x \in X$.

Graph colouring. If every finite subgraph of some infinite graph can be coloured with 17 colours, then so can the entire graph. Of course, 17 could be any positive integer.

With what we've done here, "infinite" in both examples above should be changed to "countably infinite". We have always had a countable set of propositional variables; that is, they are given 'listed'. But everything can be generalized to the language where we start with an arbitrary infinite set of propositional variables.

Parts of this next one travel over some of the same ground as problems just above.

2.36 Suppose that Δ satisfies (i) in 2.31; that is, for all formulae G, exactly one of G or $\neg G$ is in Δ . Show that the following four conditions are equivalent for Δ : being *finitely satisfiable*, and (ii), (iii), (iv) of 2.31.

Suggestions.

Show (ii) $\Longrightarrow$ (iii) $\Longrightarrow$ *finite satisfiability* $\Longrightarrow$ (ii).

Then show (iv) $\Longleftrightarrow$ (iii).

Thus there are four equivalent definitions of **maxsat set**. But proving that includes proving compactness. So the definition we gave originally is the appropriate one for proving compactness, which is the main point here.

Since 2.31 and 2.32 showed that the maxsat sets are exactly the sets of the form Δ_{e} (for some truth assignment e), being of that form is yet a fifth condition which is equivalent to each of the other four above [combined with 2.31(i)].

2.37 Show that 2.31(iv) $\Longrightarrow$ 2.31(ii) for any Δ, independently of whether it satisfies 2.31(i).

3. A PROOF SYSTEM

In this chapter, we discuss what it means in propositional logic to *formally prove* (or, to say the same thing, *to derive*, or *give a derivation of*, or *formally deduce*) a formula (the conclusion) from a set of premisses. A very satisfying result is established, with quite a bit of work : namely, that the conclusions for which there is a derivation from a given premiss set are exactly those conclusions for which the argument from those premisses to that conclusion is a valid argument (as defined, basically by truth tables, in the previous chapter). This important result is called the **Completeness Theorem** for the particular proof system which we are about to describe.

Advice and editorial

This chapter is the most demanding and technical of the four on propositional logic. Most of Chapter 4 is more elementary. That chapter gives many important techniques, particularly for the computationally-minded reader. Once you have studied the first section below, it's possible to skip ahead to Chapter 4. However, one good way to really familiarize yourself with the proof system of the first section is to do the first few exercises after the second section, for example, 3.11 (i) to (v).

Each book on mathematical logic has its own proof system, almost always differing from all the others. But truth assignments and valid arguments are the same for everybody, so all these different proof systems end up effectively doing the same thing (since no one is particularly interested in proof systems which don't satisfy the completeness theorem— except for exercises and counterexamples.) The proof system below has been chosen to be as natural and brief as possible, to seem 'correct' intuitively and easy to remember, and yet to make it not too long and tortuous a path to proving completeness. These criteria do conflict with each other, and the compromises made will depend on the authors' aesthetic and pedagogical sensibilities. That's why two books on logic seldom agree in their proof systems.

It is quite common in texts on logic to spend much more time than we do below in trying to train students to produce simple derivations— "another day, another rule of inference", as one of my friends is wont to say. Our viewpoint is that the reason for actually producing examples of derivations is to solidly embed into the memory what sort of thing a derivation is. Mathematicians don't spend their time producing derivations. They prove theorems and disprove conjectures. And then occasionally they remind themselves that the "very satisfying result" above means that a derivation does or doesn't exist (usually in a more complicated version of the subject than just propositional logic, such as 1$^{\text{st}}$order logic, studied from Chapter 5 onwards.)

For becoming more proficient at actually producing logically correct arguments in the informal sense, taking a course in formal logic ranks far behind taking one on a subject, such as mathematics (done as mathematics rather than as computational recipes) and perhaps such as law, where one is producing arguments all the time about the particular kind of math or law being studied. The kind of mickeymouse-formalizing of breathtakingly trivial arguments occasionally seen in abundance in logic courses is far more akin

to the rule-following "education" one gets in the recipe courses going under the name mathematics, than it is to sharpening the logical abilities of students. What one ought to expect a logic course to do (among other things) is to sharpen their ability to detect and explain faulty logic (as exhibited in abundance on TV news, commercials and 'religious' infomercials, for example).

3.1 The proof system : Axioms, rules of inference, derivations, and soundness.

Here are three families of formulae which we shall call the axioms, or sometimes the logical axioms, of our proof system:

$$F \to F \wedge F \quad ; \quad F \wedge G \to F \quad ; \quad (F \to G) \to (F \wedge H \to G \wedge H)$$

They are families (not single formulae), because each of F, G and H can be *any* formula. Every one of them is a tautology, as you checked in Exercise 2.5 from the last chapter. (If not, then do it now!)

Here is a list of four statements which we shall call the rules of inference of our proof system:

(MP) (*modus ponens*) : From $\{F\ ,\ F \to G\}$, one can immediately infer G.

Recall that Exercise 2.10 from the last chapter showed in a slightly roundabout way that, for all e with $\mathsf{e}(F) = \mathsf{Tr} = \mathsf{e}(F \to G)$, we also have $\mathsf{e}(G) = \mathsf{Tr}$; in other words, 'truth is preserved by this rule of inference'. It's also very easy to see this directly.

($\neg\neg$) (*double negative replacement*) : From a formula in which one of $\neg\neg F$ or F occurs as a subformula, one can immediately infer the formula obtained by replacing that occurrence of that subformula by the other, F or $\neg\neg F$, respectively. Expressed more technically, from a formula XFY, infer $X\neg\neg FY$; and from $X\neg\neg FY$, infer XFY. (So X and Y are finite horizontal arrays such that you get a formula by putting the formula F between them. As discussed in the **Section 2.7** , we are singling out a particular occurrence of the subformula F, or of $\neg\neg F$, in a bigger formula, and replacing one with the other.)

(COM) (*commutative replacement*) : From a formula in which $F \wedge G$ occurs as a subformula, one can immediately infer the formula obtained by replacing that occurrence by $G \wedge F$. Technically, from $X(F \wedge G)Y$, infer $X(G \wedge F)Y$.

(ASS) (*associative replacement*) : From a formula in which one of the formulae $(F \wedge G) \wedge H$ or $F \wedge (G \wedge H)$ occurs as a subformula, one immediately can infer the formula obtained by replacing that occurrence of that subformula by the other of these two subformulae (with differing brackets, but otherwise identical). Technically, from $X((F \wedge G) \wedge H)Y$, infer $X(F \wedge (G \wedge H))Y$; and the reverse as well.

In Section 2.7 (see especially Theorem **2.1** and examples) we showed that *truth is preserved by all these replacement rules of inference.* In other words, for any truth assignment e, we have

$$\mathsf{e}(X \sqcup \sqcup \sqcup Y) = \mathsf{Tr} \quad \Longleftrightarrow \quad \mathsf{e}(X \sqcap \sqcap \sqcap Y) = \mathsf{Tr}$$

for each pair ($\sqcup\sqcup\sqcup$, $\sqcap\sqcap\sqcap$) of subformulae which are 'allowed' to replace each other in the above three rules.

Logicians like to get super-technical with notation, and might summarize the six rules of inference as follows:

$$\frac{F \ , \ F \to G}{G} \quad ; \quad \frac{XFY}{X\neg\neg FY} \quad ; \quad \frac{X\neg\neg FY}{XFY} \quad ; \quad \frac{X(F \wedge G)Y}{X(G \wedge F)Y}$$

$$\frac{X((F \wedge G) \wedge H)Y}{X(F \wedge (G \wedge H))Y} \quad ; \quad \frac{X(F \wedge (G \wedge H))Y}{X((F \wedge G) \wedge H)Y}$$

Now let Γ be any set of formulae, referred to as the *premisses* .

A *derivation* from Γ (also called a *formal proof* or *formal deduction*) is a finite vertical array of formulae, with one formula on each line, and having the following property: each of these formulae is

(i) an axiom, or

(ii) in Γ (that is, is a premiss), or

(iii) immediately inferrable using one of the rules of inference from a formula (or two, if (MP) is used) which occurs higher up in the array.
(So the top formula in a derivation will always be an axiom or a premiss.)
The bottom formula in a derivation is the *conclusion* of (or what 'has been proved by') the derivation. Any formula F which so occurs is said to be *derivable from* Γ, and we write $\Gamma \vdash F$.

This last definition is important enough to deserve prominent display:

We write

$$\Gamma \vdash F$$

to mean that there is at least one derivation (using the proof system just presented) in which the last line is the formula F, and in which any premisses used are in the set Γ of formulae.

(The question of verbalizing this is similar to the double-turnstile: I would suggest saying, when reading "$\Gamma \vdash F$", one of "Gamma turnstiles F" or "The formula F is syntactically derivable from the premiss set Gamma" or "The formula F is inferrable from the set Gamma using the proof system".)

As a ridiculously simple example, the following is a derivation of P_2 from $\Gamma = \{P_1\ ,\ P_1 \to P_2\}$:

P_1	(premiss)
$P_1 \to P_2$	(premiss)
P_2	(MP)

Thus $\{P_1\ ,\ P_1 \to P_2\} \vdash P_2$.

The justification on the right on each line is not part of the derivation, just an aid for the reader to figure out what was going on in the writer's mind. Some lines in a derivation may have more than one correct justification. Sometimes a justification using a rule of inference should include some line numbers, if it's not the line (or two lines, in the case of (MP) being used) that is immediately above which contains the appropriate formula(e) to complete the justification.

Below is another, slightly more interesting, derivation. It is expressed in terms of general formulae F, G, H and K, rather than just propositional

variables. So it is really many derivations, one for each specific choice of F, G, H and K. Also this example makes the points that a rule may be used several times in a derivation, and that premisses and axioms don't only occur at the beginning. We show that

$$\{\ F\ ,\ F \to (G \wedge H) \wedge K\ \} \ \vdash\ K \wedge H$$

This example also illustrates that a derivation can be quite lengthy for deriving something which 'obviously must be derivable'—(actually, here this is only 'obvious' once you are confident that your system is *complete*, as discussed in the next section).

$F \to (G \wedge H) \wedge K$	(premiss)
$F \to G \wedge (H \wedge K)$	(ASS)
F	(premiss)
$G \wedge (H \wedge K)$	(MP)
$(H \wedge K) \wedge G$	(COM)
$(H \wedge K) \wedge G \to H \wedge K$	(axiom)
$H \wedge K$	(MP)
$K \wedge H$	(COM)

Of course, chopping off any bottom segment from a derivation leaves an array which is still a derivation, so $\Gamma \vdash G$ holds for every formula G which occurs as a line in a derivation from Γ. So, from the above example, we also have

$$\{\ F\ ,\ F \to (G \wedge H) \wedge K\ \} \ \vdash\ (H \wedge K) \wedge G$$

and

$$\{\ F\ ,\ F \to (G \wedge H) \wedge K\ \} \ \vdash\ H \wedge K$$

etc.

The case when Γ is empty is important. In this case, when a formula F can be derived using only the logical axioms and rules of inference, we shorten the assertion $\emptyset \vdash F$ to just $\vdash F$. (Recall that $\emptyset$ is the notation for the empty set.)

As an example, to show that $\vdash (Q \wedge R) \wedge P \to P \wedge R$, here is a derivation:

$P \wedge Q \to P$	(axiom)
$(P \wedge Q \to P) \to ((P \wedge Q) \wedge R \to P \wedge R)$	(axiom)
$(P \wedge Q) \wedge R \to P \wedge R$	(MP)
$P \wedge (Q \wedge R) \to P \wedge R$	(ASS)
$(Q \wedge R) \wedge P \to P \wedge R$	(COM)

And so $\vdash F$ holds for every line F in the above derivation, not just the last line.

The set Γ of premiss formulae is usually finite, but for theoretical purposes we won't insist on this. But clearly any single derivation from Γ will use only finitely many of the formulae in Γ. So, if $\Gamma \vdash F$, then $\Omega \vdash F$ for some **finite** subset Ω of Γ.

The first thing that should be said about this (or any other) proof system is that the axioms must be tautologies. Equally important is that (fixing some truth assignment) you can immediately infer, using any rule, only a *true* formula from true formula(e). Checking these properties for our axioms and rules of inference is quite easy, as we noted just after stating them. It follows immediately that the conclusion of any derivation is true at all those truth assignments for which all the premisses are true. What I've just said is called the **Soundness Theorem**. Expressed succinctly it says :

$$\Gamma \vdash F \implies \Gamma \models F \ .$$

That's the easy half of the **Completeness Theorem**. In the particular case where Γ is empty, it amounts to saying that a formula which can be derived using only the axioms and rules of inference is necessarily a tautology. So we'll take the soundness theorem as established, and get to work on its converse, which, in the 'zero premisses case', tells us that *every* tautology can be derived from the axioms and rules of inference. The statement in the general case is **3.16** ahead; that is, just change $\implies$ above to $\impliedby$ (or interchange $\vdash$ and $\models$).

Exercise 3.0. Consider the following modified system: Remove the first family, $F \to F \wedge F$, of axioms, but add a new replacement rule of inference which says that subformulae F and $F \wedge F$ may be substituted for each other in any formula.

(i) Show that the new system is sound.

(ii) Show that the new system is just as strong as the old because the formulae $F \to F \wedge F$ can be derived in the new system. Hint: First derive $F \to F$. (Once adequacy of the old system has been proved, it will follow that the two are exactly the same strength, and that the new system is also complete.)

Exercise 3.1. Consider a new proof system obtained from the 'old' one in this chapter simply by replacing (MP) by the new rule :

From $\{F\ ,\ \neg(F \wedge G)\}$, *one can immediately infer* $\neg G$.

Determine whether this new system will derive exactly the same formulae as the old one.

3.2 The proof of completeness.

Prop 3.1 $\{F \to G\} \vdash (\neg G \to \neg F)$

Proof. Consider the derivation

$\neg(F \wedge \neg G) \ = \ F \to G$	(premiss)
$\neg(\neg G \wedge F)$	(COM)
$\neg(\neg G \wedge \neg\neg F) \ = \ \neg G \to \neg F$	$(\neg\neg)$

Remarks. (1) It goes without saying that what is being asserted in the proposition is that, for any choice of formulae F and G, the formula $\neg G \to \neg F$ can be derived from just the axioms, the rules of inference and the one premiss. And the left side in the box constitutes not just a single derivation, but one derivation for every choice of F and G.

(2) The formula being proved here and the statement in the next proposition look fairly natural to a mathematician. They are related to the so-called 'proof by contradiction' and the 'contrapositive'. Loosely expressed : 'If you can get G from F, then you can get $notF$ from $notG$.'

(3) The derivation itself in the proof possibly looks at first like it has been pulled out of a hat. To discover such a derivation in the first place, you usually work backwards, first writing down the conclusion, and manipulating it back to something 'you already know', all the time repeating the mantra "if only I could ...". The same remark applies globally to the series of propositions we are beginning here—you try to see what you need in order to prove the final, desired, theorem, and gradually work backwards to something you *can* do.

(4) One way to think of **3.1** (and many propositions below) is as being *derived* rules of inference. It tells us that, from $F \to G$, one can 'immediately' infer $\neg G \to \neg F$.

Prop 3.2 $\{F \to G\ ,\ \neg G\} \vdash \neg F$. *That is, from* $F \to G$ *and* $\neg G$*, one can infer* $\neg F$. (This derived rule of inference is classically known as modus tollens.)

Proof.

$F \to G$	(premiss)
$\neg G \to \neg F$	**(3.1)**
$\neg G$	(premiss)
$\neg F$	(MP)

Remark. The same remark about the proof as in Remark (1) previous applies here. More importantly, we don't really have a derivation at all, because of the use of **3.1** as a justification. But of course we'll use this type of 'derivation building' all the time below. To convert the proof above into an actual derivation, you'd simply insert the derivation in the proof of **3.1** at the appropriate place. We'll sometimes refer to boxes as written above by the name shortcut derivations.

Prop 3.3 $\{F \to G\ ,\ G \to H\} \vdash (F \to H)$. *That is, from* $F \to G$ *and* $G \to H$*, one can infer* $F \to H$. (This derived rule of inference is classically known as hypothetical syllogism.)

Proof.

$F \to G$	(premiss)
$(F \to G) \to (F \wedge \neg H \to G \wedge \neg H)$	(axiom)
$F \wedge \neg H \to G \wedge \neg H$	(MP)
$\neg(G \wedge \neg H) \ = \ G \to H$	(premiss)
$\neg(F \wedge \neg H)$	**(3.2)**

But the last formula is just $F \to H$, as required.

Exercise 3.2 Show that $\{F \to G\ ,\ G \to H\ ,\ H \to K\} \vdash (F \to K)$.

Prop 3.4 $\vdash (F \to F)$.

Proof.

$F \to F \wedge F$	(axiom)
$F \wedge F \to F$	(axiom)
$F \to F$	**(3.3)**

Aside. We'll take a short break for an example which illustrates two points, the major one being the previous remark about working "backwards" to create a derivation.

Consider $J = (F \to G) \vee (G \to H)$ for any formulae F, G and H. The minor point is that J is a tautology, perhaps surprisingly to those who think that tautologies, or at least ones with few symbols, should always be some kind of 'obvious universal truths'. The formula J certainly doesn't look like one to me! A couple of other examples like this may be found in Exercises 2.27 and 2.28.

Thus, if the completeness theorem is true, we must have $\vdash (F \to G) \vee (G \to H)$. Despite this, the reader must definitely not get sucked into believing the false statement that "$\vdash (F \to G)$ or $\vdash (G \to H)$". Is it possible that the more naive of the intuitionists (see the beginning of the next section, **3.3**), when rejecting the law of the excluded middle, are on some level confusing the true statement $\vdash (F \vee \neg F)$ with the false statement "$\vdash F$ or $\vdash \neg F$"? (That rhetorical question is probably a very unfair slam at a group of serious thinkers!)

To find a shortcut derivation of J in our system, my guesswork/backward thinking might go as follows:

$$(F \to G) \vee (G \to H) \;=\; \neg(F \wedge \neg G) \vee \neg(G \wedge \neg H) \;=\; \neg(\neg\neg(F \wedge \neg G) \wedge \neg\neg(G \wedge \neg H)) \; .$$

So a couple of $(\neg\neg)$'s gets us down to deriving $\neg((F \wedge \neg G) \wedge (G \wedge \neg H))$. But it's really the string $\neg G) \wedge (G$ in the middle which is the key to both getting the derivation and to 'why' J is a tautology. By some (ASS) and (COM), we need to derive $\neg((G \wedge \neg G) \wedge (F \wedge \neg H))$. But $\neg(G \wedge \neg G) \;=\; G \to G$, we can derive that, and

$$(G \wedge \neg G) \wedge (F \wedge \neg H) \to G \wedge \neg G$$

is an axiom.

Exercise 3.3 (i) Prove that J is a tautology in a few sentences, without a big truth table production.

(ii) Produce a shortcut derivation of J, with justifications, perhaps using the backward hints above.

Prop 3.5 $\{F \to G\} \vdash F \to F \wedge G$. *That is, from* $F \to G$ *, one can infer* $F \to F \wedge G$.

Remark. This derived rule of inference is related to the situation where one wants to get some distant goal, say H, from F. But, so far, one has only managed to get G. After trying for awhile to get H from G, you realize that

maybe you need to use F in some other way, along with G, to reach your goal. So you now carry around in your mind that you are able to use both F and G.

Proof.

$(F \to G) \to (F \wedge F \to G \wedge F)$	(axiom)
$(F \to G) \to (F \wedge F \to F \wedge G)$	(COM)
$F \to G$	(premiss)
$F \wedge F \to F \wedge G$	(MP)
$F \to F \wedge F$	(axiom)
$F \to F \wedge G$	**(3.3)**

Prop 3.6 (i) $\{F \to (G \to H)\} \vdash F \wedge G \to H$.
(ii) $\{F \wedge G \to H\} \vdash F \to (G \to H)$.

Remark. These derived rules of inference are reverses of each other. Very crudely, they assert that '[being able to get H from knowing both F and G] is the same as [getting from knowledge of F the fact that you can get H from G].'

Proof. For (ii), the proof is just writing down the following derivation of (i) in reverse order—do that and fill in the justifications as an exercise.

$\neg(F \wedge \neg\neg(G \wedge \neg H)) = \neg(F \wedge \neg(G \to H)) = F \to (G \to H)$	(premiss)
$\neg(F \wedge (G \wedge \neg H))$	$(\neg\neg)$
$\neg((F \wedge G) \wedge \neg H)$	(ASS)

But the last formula is just $F \wedge G \to H$.

Prop 3.7 $\{F\ ,\ G\} \vdash F \wedge G$.

Remark. (1) This derived rule of inference sounds silly in plain English, where we don't distinguish the metalanguage from the formal language : "You can get F and G from F and G !!"

(2) It is easy to see that the converse of this proposition holds—namely, both F and G can be inferred from $\{F \wedge G\}$. The axiom $F \wedge G \to F$ together with (MP) does one; to do the other, use (COM) once as well.

Proof.

$F \wedge G \to F \wedge G$	**(3.4)**
$F \to (G \to F \wedge G)$	**(3.6(ii))**
F	(premiss)
$G \to F \wedge G$	(MP)
G	(premiss)
$F \wedge G$	(MP)

Aside: Using rules of inference to argue that a derivation exists.

Although I don't regard as particularly profitable the spending of a lot of time trying to learn how to produce derivations, here is another example along those lines. The idea is to make use of derived rules of inference as well as the ones in the system itself. A book with the objective of training readers to produce derivations will often use a 'fat' system with lots of axioms and rules of inference from the start. An author trying to be as mathematically lean and elegant as possible will do the opposite. I like to think that the system you are studying here is a happy compromise between the two.

Suppose at this point you were asked to prove

$$\{\ A \to B\ ,\ C \to D\ ,\ B \vee D \to E\ ,\ \neg E\ \} \vdash\ \neg A \wedge \neg C\ \ .$$

This might have arisen from the following argument in English:

Eating carefully gives you a healthy colon. Exercising well makes you fit. If your colon is healthy or you are fit, you live long. You didn't live long. *(To whom is this guy talking, anyway?!)* Therefore, you didn't eat carefully nor did you exercise well.

Rather than actually writing out either a complete or a shortcut derivation, I might argue as follows.

By modus tollens (which is **3.2**), the last two premisses yield $\neg(B \vee D)$, which is just an abbreviation of $\neg\neg(\neg B \wedge \neg D)$. So it is now easy to get to $\neg B \wedge \neg D$. But Remark (2) after **3.7** gives derived rules of inference which will now allow us to get both $\neg B$ and $\neg D$. Pairing these respectively with the first two premisses, and using two more applications of modus tollens now produces both $\neg A$ and $\neg C$. Now just apply the rule of inference given by **3.7** to get the desired result.

Exercise. Write this out as a shortcut derivation. Then maybe write it out in all detail as a derivation. Then prove the corresponding result with $\vdash$ changed to $\models$ (but please, don't appeal to soundness, and don't use a 32-line truth table).

Prop 3.8 (i) $\{G\} \vdash F \to G$.
(ii) $\{F \to G\ ,\ F \to (G \to H)\} \vdash F \to H$.

Remark. These two derived rules of inference are, perhaps for the first time, obscure-looking ones. They are placed here simply because of being exactly what's needed in the induction step in the next proof. This is an example of "working backwards", mentioned earlier. Once I had decided

to try to prove the next theorem by induction (a decision to mimic other authors with other systems), the need for this proposition soon appeared.

Proof. (i)

$G \wedge F \to G$	(axiom)
$G \to (F \to G)$	**(3.6(ii))**
G	(premiss)
$F \to G$	(MP)

(ii) From $F \to G$, we use **3.5** to infer $F \to F \wedge G$. From $F \to (G \to H)$, we use **3.6(i)** to infer $F \wedge G \to H$. From the two formulae ending the last two sentences, using **3.3**, we infer $F \to H$.

Theorem 3.9 *If* $\Gamma \cup \{F\} \vdash G$, *then* $\Gamma \vdash F \to G$.

Remark. This important fact is known as **Herbrand's Deduction Lemma** in propositional logic. Its converse is very easy to prove.

Proof. Suppose it fails for a given Γ and F, and that G is the conclusion of a *shortest* derivation from $\Gamma \cup \{F\}$ for which there is no derivation from Γ of $F \to G$. Let the lines in such a shortest derivation be the formulae $H_1, H_2, \ldots, H_n = G$. Because of it being shortest, we do have $\Gamma \vdash F \to H_i$ for all $i < n$. We'll get a contradiction by showing that, in fact, $\Gamma \vdash F \to G$. (So this is really more a proof by induction on n, than one by contradiction, but the present wording avoids some repetitiveness in considering the case when $n = 1$.)

Now consider the possibilities for the justification of the last line, $H_n = G$, in the above derivation. We have four cases: $G = F$; or G is an axiom or is in Γ ; or G is justified by a replacement rule of inference from some earlier H_i ; or by (MP) using a pair of earlier lines.

In the first case, $F \to G$ is just $F \to F$, so we have $\vdash F \to G$ by **(3.4)**. Thus we'd certainly have $\Gamma \vdash F \to G$, as required.

In the second case, we also get $\Gamma \vdash F \to G$, since $\{G\} \vdash F \to G$ by the first part of **3.8**, and $G \in \Gamma$ or is an axiom.

In the third case, G is obtainable from an H_i for $i < n$ by one of the replacements in those five rules of inference. But then $F \to G$ is obtainable from $F \to H_i$ by the 'same' replacement, so again we get $\Gamma \vdash F \to G$, by adding just one line to a derivation of $F \to H_i$ from Γ.

Finally we have the case where G arises by using (MP). Thus we have two lines: H_i say, and H_j which is $H_i \to G$, where both i and j are less than n. Now both $F \to H_i$ and $F \to (H_i \to G)$ can be derived from Γ. (This is just

from the business of "shortest", as noted above.) But, by **3.8(ii)**, $F \to G$ can be derived from the two formulae in the previous sentence. (Just change G in **3.8(ii)** to H_i, and change H to G.) Thus $F \to G$ *can* be derived from Γ, completing the proof.

Prop 3.10 (i) $\vdash F \wedge \neg F \to G$.
(ii) $\{F\ ,\ \neg F\} \ \vdash \ G$.

Remark. Note that G is *any* formula—'nothing to do with F' !

Proof. Part (ii) is immediate from part (i) and (MP), since, by **3.7**, $\{F\ ,\ \neg F\} \vdash F \wedge \neg F$. To prove part (i),

$\neg((F \wedge \neg G) \wedge \neg F) = (F \wedge \neg G) \to F$	(axiom)
$\neg((F \wedge (\neg G \wedge \neg F))$	(ASS)
$\neg((F \wedge (\neg F \wedge \neg G))$	(COMM)
$\neg((F \wedge \neg F) \wedge \neg G)$	(ASS)

But the last formula is $F \wedge \neg F \to G$, as required.

Exercises.

3.4 Fill in justifications for the following alternative proof of **3.10(i)**—actually, there should be one more line—find it :

$F \wedge \neg G \to F$
$\neg G \wedge F \to F$
$\neg G \to (F \to F)$
$\neg\neg(F \wedge \neg F) \to \neg\neg G$
$F \wedge \neg F \to G$

3.5 With either proof of **3.10(i)**, it is okay to place it before **3.8**. There is then an alternative proof of **3.8(i)** as follows. Write it out as an abbreviated derivation, and fill in the justifications: Write out $G \wedge \neg G \ \to \ \neg F$ as an actual formula, 'manipulate' it so that it becomes $G \to (F \to G)$ written in unabbreviated form, and then complete the proof of **3.8(i)**.

3.6 In each of **3.1** to **3.8** and **3.10**, change $\vdash$ to $\models$, and prove each one directly, by using truth tables. (These *semantic versions* also follow as examples of the Soundness Theorem.)

3.7 Prove **3.9** with $\vdash$ changed to $\models$.

3.8 Prove the converse to **3.9**, and also its semantic version.

3.9 Show that $\{ F \to G \ , \ \neg F \to G \} \ \vdash \ G$.

3.10 State and prove the semantic version of Exercise 3.7 .

3.11(i) Use Exercise 3.7 to prove that, if $\Gamma \cup \{F\} \vdash G$ and $\Gamma \cup \{\neg F\} \vdash G$, then $\Gamma \vdash G$ ("proof by cases").

(ii) State and prove the semantic version of (i) .

(iii) Show that

$$[\ \{F\} \vdash G \implies \{\neg G\} \vdash \neg F\ ] \text{ and } [\ \vdash F \implies \text{for all } G,\ \vdash (\neg F \to G)\ ] .$$

3.12 Generalize the proof of **3.10(i)** to a proof that
$\{F \wedge \neg G \to H\} \ \vdash \ F \wedge \neg H \to G$.

By the adequacy theorem below, namely **3.16**, exercises 3.9(i) and 3.7 can be done using exercises 3.9(ii) and 3.8 respectively. And the reverse is of course true, using the soundness theorem. But we cannot use the adequacy theorem to get **3.1** to **3.8** by first proving their semantic versions as in exercise 3.4, since **3.1** to **3.8** will be used in the proof of the adequacy theorem.

The tautology $G \to (F \to G)$, in Exercise 3.3 and the proof of **3.8(i)**, is made much use of in many treatments of this subject, but it always strikes me as a bit of an unnatural thing.

Theorem 3.11 *For a set of formulae* Γ*, we have* $\Gamma \vdash G$ *for all* G *if and only if, for some formula* F*, both* F *and* $\neg F$ *can be inferred from* Γ .

Definition. We say that a set of formulae Γ is *inconsistent* if it satisfies either (and so both) of the conditions in this theorem. Of course, *consistent* means not inconsistent.

Proof. It is beyond the pale of triviality that the first condition implies the second. The other way round is immediate from **3.10(ii)**.

Prop 3.12 (i) *If* $\Gamma \vdash F$ *and* Γ *is consistent, then* $\Gamma \cup \{F\}$ *is consistent.*
(ii) *If* $\Gamma \nvdash F$ *(so* Γ **is** *consistent!), then* $\Gamma \cup \{\neg F\}$ *is consistent.*

Proof. (i) Clearly, if $\Gamma \vdash F$ and $\Gamma \cup \{F\} \vdash G$, then $\Gamma \vdash G$. This is true for all G, which amounts to saying that if $\Gamma \vdash F$ and $\Gamma \cup \{F\}$ is inconsistent, then Γ is inconsistent.

(ii) If $\Gamma \cup \{\neg F\}$ is inconsistent, then anything, and in particular, F itself, can be inferred from it. By the deduction lemma (**3.9**), the formula $\neg F \to F$ can be inferred from Γ. That last formula is $\neg(\neg F \wedge \neg F)$, so from the axiom $\neg F \to \neg F \wedge \neg F$ and **3.2**, we see that $\neg\neg F$ can be inferred from Γ. One application of $(\neg\neg)$, and we see that F can be inferred from Γ, as required.

Theorem 3.13 *If Γ is consistent, then there is a larger consistent set, Δ, of formulae such that, for every formula F, at least one of F or $\neg F$ is in Δ* (and so, by consistency, **exactly** one of them is in this *extension*, Δ, of Γ).

Remark. This important fact is **Lindenbaum's Theorem** in propositional logic. It is clear from the statement that $\Delta \vdash F$ if and only if $F \in \Delta$. (Remember that not both F and $\neg F$ can be inferred from a consistent set.) Of course Δ is always an infinite set of formulae, so here is one place where we can't just be thinking about finite sets of formulae. This business of dividing the set of all formulae 'exactly in half', in the sense that exactly one of F and $\neg F$ is in the chosen half, looks very much like what a choice of a truth assignment would do—'half' become true, and 'half' false—and what we're going to do in the next lemma is turn this upside down. We'll show how a 'division in half', **as long as you have consistency**, will produce a truth assignment for which the chosen half are the 'true ones'.

Proof. List all the formulae : $F_0, F_1, F_2, \cdots$. (It doesn't even matter if some formula appears more than once in the list. We just want to be sure that every formula is there in the list. Mathematicians will see how this is done, since the set of *all* finite sequences from a countable set of symbols is countable, and the set of all formulae is one of its subsets. Others perhaps won't even consider the possibility that it can't be done.)

Now define a list, of larger and larger sets of formulae, $\Gamma_0, \Gamma_1, \Gamma_2, \cdots$ as follows :

$$\Gamma_0 := \Gamma \quad \text{and, given } \Gamma_n, \quad \text{let } \Gamma_{n+1} := \begin{cases} \Gamma_n \cup \{F_n\} & \text{if } \Gamma_n \vdash F_n \ ; \\ \Gamma_n \cup \{\neg F_n\} & \text{if } \Gamma_n \nvdash F_n \ . \end{cases}$$

Expressed in words: To form Γ_{n+1}, add to Γ_n one formula (if it's not already in Γ_n). The formula to be added is either F_n or $\neg F_n$, according to whether F_n can be inferred from Γ_n, or not, respectively.

It is immediate from **3.12** that, if Γ_n is consistent, then Γ_{n+1} is also. Since $\Gamma_0 = \Gamma$ is assumed consistent, it follows that all the Γ_n are consistent.

Now let Δ be the union of all the Γ_n, i.e. the set of all formulae which are in Γ_n for at least one n. Any finite subset of Δ is certainly a subset of Γ_n for some n, since the Γ_n are an increasing sequence of sets. Thus, if there were derivations of both some F, and its negation $\neg F$, from Δ, these derivations would contain only finitely many formulae, so we'd get derivations of both F and $\neg F$ from some Γ_n. This is impossible by the consistency of Γ_n. So Δ is consistent, as required.

The set Δ certainly contains Γ, so it remains only to show that it contains at least one of F or $\neg F$, for each F. But F occurs in the list, say $F = F_n$. From the definition of Γ_{n+1}, either F_n or $\neg F_n$ is in Γ_{n+1}, and therefore in Δ. This completes the proof.

Lemma 3.14 *For Δ as above, there is a truth assignment* e *such that* $\mathsf{e}(F) = \mathsf{Tr}$ if and only if $F \in \Delta$.

Proof. There certainly is such a function e; just define

$$\mathsf{e}(F) \;:=\; \begin{cases} \mathsf{Tr} & \text{if } \; F \in \Delta \; ; \\ \mathsf{Fs} & \text{if } \; F \notin \Delta \; . \end{cases}$$

But is e a truth assignment? By the remarks in Chapter 2 after the definition of *truth assignment* and the basic definitional truth tables for "$\neg$" and "$\wedge$", we must show that

$$\mathsf{e}(\neg F) = \mathsf{Tr} \iff \mathsf{e}(F) = \mathsf{Fs} \quad \text{and} \quad \mathsf{e}(F \wedge G) = \mathsf{Tr} \iff \mathsf{e}(F) = \mathsf{Tr} = \mathsf{e}(G) \; .$$

For the first,

$$\mathsf{e}(\neg F) = \mathsf{Tr} \iff \neg F \in \Delta \iff F \notin \Delta \iff \mathsf{e}(F) = \mathsf{Fs} \; .$$

For the second,

$$\mathsf{e}(F \wedge G) = \mathsf{Tr} \iff F \wedge G \in \Delta \iff \Delta \vdash F \wedge G$$

$$\iff \Delta \vdash F \text{ and } \Delta \vdash G \iff F \in \Delta \text{ and } G \in \Delta \iff \mathsf{e}(F) = \mathsf{Tr} = \mathsf{e}(G) \; .$$

Only the leftmost $\iff$ on the last line needs comment—the direction $\Longleftarrow$ is immediate from **3.7**, and the other direction from Remark (2) after that proposition.

Theorem 3.15 *If a set* Γ *is consistent, then it is satisfiable.*

Proof. Construct Δ as in **3.13**, and let e be the truth assignment in **3.14**. Than $\mathsf{e}(F) = \mathsf{Tr}$ for all F in Γ, since that's even the case for all F in Δ. By definition, Γ is satisfiable, since we've found a truth assignment which makes all its formulae true.

Theorem 3.16 (Adequacy of the system) *If* $\Gamma \models F$, *then* $\Gamma \vdash F$.

Proof. If $\Gamma \nvdash F$, then, by **3.12(ii)**, $\Gamma \cup \{\neg F\}$ is consistent, and therefore is satisfiable by **3.15**. But that means we have a truth assignment e making $\mathsf{e}(G) = \mathsf{Tr}$ for all $G \in \Gamma$, and $\mathsf{e}(\neg F) = \mathsf{Tr}$ as well. But then $\mathsf{e}(F) = \mathsf{Fs}$. This shows that $\Gamma \not\models F$, as required.

(The last three sentences of this proof gives the proof of half of **4.3**, a result mentioned earlier which is very easy, but also very fundamental to motivation for the algorithms studied in the latter part of the next chapter.)

Combining **3.16** with the soundness theorem from the end of the first section, we have now reached the objective, having established that our proof system for propositional logic is **complete** :

For all formulae F *and all sets* Γ *of formulae,*

$$\Gamma \models F \iff \Gamma \vdash F \quad .$$

Exercises.

3.13 The box in each of (i), (ii) and (iii) is a (short-cut) derivation of the $\vdash$-assertion above the box. Fill in a justification for each line—either basic justifications quoting "premiss", "axiom", or one of the rules of inference; or else the short-cut, quoting one of the propositions of this section, or a previous part of this exercise. There is also an instance where an abbreviated formula has been rewritten without the abbreviation. HINT: Except once in (ii), the zero, one, or two lines used in the justification are immediately above the line being justified.

(i) $\{F_1 \to F_2\} \vdash F_1 \wedge G \to F_2 \wedge G$.

$F_1 \wedge G \to F_1$
$F_1 \to F_2$
$F_1 \wedge G \to F_2$
$F_1 \wedge G \to (F_1 \wedge G) \wedge F_2$
$F_1 \wedge G \to F_2 \wedge (F_1 \wedge G)$
$F_1 \wedge G \to F_2 \wedge (G \wedge F_1)$
$F_1 \wedge G \to (F_2 \wedge G) \wedge F_1$
$(F_2 \wedge G) \wedge F_1 \to F_2 \wedge G$
$F_1 \wedge G \to F_2 \wedge G$

(ib) Find a much, much shorter derivation which does the same job here. (In fact, three lines will do.)

(ii) $\{F_1 \to F_2\ ,\ G_1 \to G_2\} \vdash F_1 \wedge G_1 \to F_2 \wedge G_2$.

$F_1 \to F_2$
$F_1 \wedge G_1 \to F_2 \wedge G_1$
$G_1 \to G_2$
$G_1 \wedge F_2 \to G_2 \wedge F_2$
$F_2 \wedge G_1 \to G_2 \wedge F_2$
$F_1 \wedge G_1 \to G_2 \wedge F_2$
$F_1 \wedge G_1 \to F_2 \wedge G_2$

(iii) $\{F \to (G \to H)\} \vdash F \to (\neg H \to \neg G)$.

$\neg(F \wedge \neg\neg(G \wedge \neg H))$
$\neg(F \wedge \neg\neg(\neg H \wedge G))$
$\neg(F \wedge \neg\neg(\neg H \wedge \neg\neg G))$

And observe that the last line is the conclusion—explain.

(iv) Prove the 'reverse' of (iii)—that is,

$$\{F \to (\neg H \to \neg G)\} \vdash F \to (G \to H) \quad .$$

(v) Here do the same as in (i), (ii) and (iii), except also fill in several lines

in the middle, all of which are justified by (ASS) or (COM).

$$\vdash F \wedge (G \vee H) \;\rightarrow\; (F \wedge G) \vee (F \wedge H)$$

$F \wedge G \rightarrow F \wedge G$
$F \rightarrow (G \rightarrow F \wedge G)$
$F \rightarrow (\neg(F \wedge G) \rightarrow \neg G)$
$F \wedge \neg(F \wedge G) \rightarrow \neg G$
$F \wedge H \rightarrow F \wedge H$
$F \rightarrow (H \rightarrow F \wedge H)$
$F \rightarrow (\neg(F \wedge H) \rightarrow \neg H)$
$F \wedge \neg(F \wedge H) \rightarrow \neg H$
$(F \wedge \neg(F \wedge G)) \wedge (F \wedge \neg(F \wedge H)) \rightarrow \neg G \wedge \neg H$

←fill in

$(F \wedge F) \wedge (\neg(F \wedge G) \wedge \neg(F \wedge H)) \rightarrow \neg G \wedge \neg H$
$F \rightarrow F \wedge F$
$F \wedge (\neg(F \wedge G) \wedge \neg(F \wedge H)) \rightarrow (F \wedge F) \wedge (\neg(F \wedge G) \wedge \neg(F \wedge H))$
$F \wedge (\neg(F \wedge G) \wedge \neg(F \wedge H)) \rightarrow \neg G \wedge \neg H$
$F \rightarrow (\neg(F \wedge G) \wedge \neg(F \wedge H) \rightarrow \neg G \wedge \neg H)$
$F \rightarrow (\neg(\neg G \wedge \neg H) \rightarrow \neg(\neg(F \wedge G) \wedge \neg(F \wedge H)))$
$F \wedge \neg(\neg G \wedge \neg H) \rightarrow \neg(\neg(F \wedge G) \wedge \neg(F \wedge H))$

And observe that the last line is the conclusion—explain.

(vi) Show directly (i.e. not using completeness and a truth table—but use any of **3.1** to **3.8**) that

$$\{F \rightarrow G \;,\; G \rightarrow F\;\} \;\vdash\; F \leftrightarrow G$$

(vii) Show that the $\vdash$-assertion in (v) holds with $\rightarrow$ changed to $\leftrightarrow$.

(viii) Prove the semantic versions of all the earlier parts of this question (which versions, when combined with the completeness theorem, prove the correctness of these $\vdash$-assertions without any need to produce the derivations in those earlier parts).

3.14 Decide which, if either, of soundness or adequacy holds for the following 'proposed proof systems'.⇐(It's better to use the unadorned phrase

proof system only for one which is complete; i.e. which does have these two good properties, with respect to whatever kind of logic you are doing.) Thus show that none of these proposed proof systems is complete (so they don't pass the test for membership in Club Pruf).

(i) **Axioms:** All formulae.
Rules of inference: None.

(ii) **Axioms:** None.
Rules of inference: From nothing at all, infer any formula you want.

(iii) **Axioms:** All formulae of the form $F \to F$.
Rules of inference: Hypothetical syllogism. (See **3.3**.)

(iv) **Axioms:** All formulae of the form $F \to G$.
Rules of inference: (MP).

(v) **Axioms:** All formulae of the form $F \to G$.
Rules of inference: Hypothetical syllogism.

(vi) **Axioms:** All tautologies.
Rules of inference: From any G which is not a tautology, infer any formula at all.

3.15 It would be very elegant if (MP) could be replaced by $\frac{F,G}{F \wedge G}$ in the system of this chapter, but unfortunately it's very false—that is, adequacy fails. To see this, work with actual formulae below, not abbreviations, when counting occurrences.

(i) Show that any F for which $\{P_1\ ,\ P_1 \to P_2\} \vdash_{\mathbf{C}} F$ (where $\vdash_{\mathbf{C}}$ means 'a derivation exists using this proposed proof system') must satisfy the inequality

$$\#\text{occurrences } P_2 \ < \ \#\text{occurrences all prop. variables} \ + \#\text{occurrences } \wedge \ .$$

This just says that if $\wedge$ doesn't occur in F, then some propositional variable other than P_2 must also occur in F. Proceed by induction on derivation length, noting that it holds for any F in which "$\wedge$" does occur.

(ii) Deduce that $\{P_1\ ,\ P_1 \to P_2\} \nvdash_{\mathbf{C}} P_2$, and hence the proposed system is not adequate.

The next four exercises are not unrelated to each other.

3.16 Suppose that $\Gamma \models F$. Prove that we have $\Omega \models F$ for some *finite* subset $\Omega \subset \Gamma$. Hint: Use completeness. (This is one version of something called the **Compactness Theorem** for propositional logic. See also Section **2.8** for another version of this theorem. The two versions are equivalent, as is easily shown.)

3.17 Prove the following easy converse of the previous exercise: Suppose that, for some finite $\Omega \subset \Gamma$, we have $\Omega \models F$. Deduce that $\Gamma \models F$.

3.18 Suppose given a proposed proof system, and we use $\vdash_{\mathbf{P}}$ to mean 'a derivation exists using this proposed proof system'. Assume that for all *finite* Ω, we have

$$\Omega \vdash_{\mathbf{P}} F \iff \Omega \models F \ .$$

Prove that the proposed proof system is complete.

3.19 Suppose given a proposed proof system which has (MP) as a (possibly derived) rule of inference. Assume that the Deduction Lemma holds, and that 'completeness for empty premiss sets' holds : i.e.

$$\vdash_{\mathbf{P}} F \iff \models F \ .$$

Show that the system is complete. Hint: A finite $\Omega = \{F_1, \cdots, F_n\}$ gives rise to formulae $F_1 \to (F_2 \to (\cdots \to (F_n \to G)) \cdots)$.

3.20 Show that both soundness and adequacy do hold for the following 'proposed proof systems' (so they do pass the test for membership in Club Pruf; but all except (i) and (ii) are rather silly systems for other reasons).

The relationship of (i) to our system in this chapter also illustrates how you can, if (MP) is one of your rules of inference, replace other rules of inference $\frac{H}{J}$ by axioms $H \to J$. The previous sentence, plus doing (i) before (ii) or (iv), are hints here.

(i) **Axioms:** Same as our system in this chapter, plus five more schema:

$$XFY \to X\neg\neg FY \quad ; \quad X\neg\neg FY \to XFY$$

$$X(F \wedge G)Y \to X(G \wedge F)Y \quad ;$$

$X((F\wedge G)\wedge H)Y \to X(F\wedge(G\wedge H))Y$; $X(F\wedge(G\wedge H))Y \to X((F\wedge G)\wedge H)Y$.

Rule of inference: (MP).

(ii) **Axioms:** All tautologies.
Rule of inference: (MP).

(iii) **Axioms:** None.
Rules of inference: (MP), and also, from nothing at all, infer any tautology that you want.

(iv) **Axioms:** All tautologies of the form $F \to G$.
Rule of inference: (MP).

(v) **Axioms:** None.
Rules of inference: (MP), and also, from nothing at all, infer any tautology of the form $F \to G$ that you want.

3.3 Some additional remarks, questions and opinions.

A very instructive summary of the history of propositional logic may be found in **Burris**, pages 119 to 131.

None of this final section of the chapter is crucial to the rest of the book. However, the exercises below should help to reinforce much of the preceding section.

Other 'logics'. There are at least four serious variations on classical propositional logic as presented in these first three chapters. These variations are not central to our main interests here, so the following three paragraphs will be brief. More information can be found in several of the references.

The intuitionist philosophy of mathematics would, among other things, quarrel with the whole approach in Chapter 2, because that approach assumes that one can deal with statements which are assumed to be definitely either true or false a priori, without any specific verification one way or the other. For example, the following argument, that *there is a pair of irrational numbers a and b for which a^b is rational*, is unacceptable to an intuitionist, because it doesn't definitely decide which of the following two possibilities is the correct one. The argument just says that it must be the case either for $(a,b) = (\sqrt{2},\sqrt{2})$ or for $(a,b) = (\sqrt{2}^{\sqrt{2}},\sqrt{2})$. Clearly, if a^b for the first of those doesn't give a rational, then the second is a pair of irrationals, and a^b in that case is just 2, certainly a rational! Perhaps created more for us ordinary mathematicians to understand how far you can get using their principles, than for intuitionists themselves, there is a topic known as **intuitionistic propositional logic**. It is also used in some other specialized contexts in both mathematical logic and logics studied by computer scientists.

A second form of 'deviant' propositional logic, known as **quantum logic**, has been studied with a view to the foundations of micro-physics. This is where physicists theorize about elementary particles, such as electrons, photons and quarks, using the empirically very well established (but still theoretically mysterious) quantum principles. In this form of logic one has, for example, many different, and actually inequivalent for them, formulae built from $\neg, \vee$ and $\wedge$, formulae which for us are all equivalent to $P \to Q$ (and therefore to each other).

Finally, there are well established disciplines called **modal logic** and **many-valued logics**. The latter is designed to be closer to natural language, with some 'vagueness' or 'uncertainty' built in. The "many values" refer to truth values in addition to Tr and Fs. Modal logic uses some extra symbols in its formulae which correspond to words like 'possibility' and 'necessity'. It has various uses, including the study of parts of ordinary logic such as we deal with here. For example, see [**Boolos**], where the extra symbols of modal logic are reinterpreted and used to throw light on the famous incompleteness theorems of Gödel. As Quine may or may not have said, "If modern modal logic was conceived in sin, then it has been redeemed through Gödliness."

Other proof systems using this language. The book of **Rosser** was perhaps the first to provide a proof system using $\wedge$ and $\neg$ as the basic connectives. He uses (MP), *no other rule of inference*, our first two axioms, plus an 'obscurification' of our last axiom, i.e. $(F \to G) \to (\neg(G \wedge H) \to \neg(H \wedge F))$. So his system is very austere, perhaps inspired by a system for $\neg$ and $\vee$ by Götlind and Rasiowa which compressed the **Russell-Whitehead** system. It is more economical in itself than our system. On the other hand, I suspect that the shortest path to adequacy for Rosser's system would need more than 200 derivation steps in total, as opposed to the approximately 50 which we need. (. . . pays your money, takes your pick . . .)

What does a proof system in this 'Hilbert style' really need? Look at (ii) vs. (iii), and (iv) vs. (v), in the last exercise, 3.20. It is clear more generally that, within a proposed proof system, there is no real difference between an axiom F, and a rule of inference of the form "From nothing at all, infer F". Since soundness is a sine qua non, F ought to be a tautology. Let us call anything like that (in which F is a tautology) by the name **axiom/nothing rule**.

Now it is clear that if a proposed proof system has none at all of these axiom/nothing rules, then there can be no derivations at all from the empty set of premisses, and therefore completeness fails. In fact, thinking about actual single axioms rather than axiom schemes, it is clear that there must be infinitely many such axiom/nothing rules (because there are infinitely many propositional variables).

But as long as $\{F \wedge G\} \vdash_P F$ and $\{G \wedge F\} \vdash_P F$ are (possibly derived) rules of inference, one can always conjunct together any finite set of axiom schemes, so a single axiom/nothing rule scheme can be enough, with other rule(s).

On the other hand, a proposed proof system which has no rules other than nothing rules will clearly have derivations all of whose lines are either tautologies or premisses. So it won't be complete. Thus it needs at least one such rule which is a 'something rule'. Let

us call such a rule **pregnant** if it is sound, and the system consisting of it, as the only rule of inference, together with all tautologies as the axioms, is complete.

By Exercise 3.20(ii), the rule (MP) is pregnant. This is perhaps one reason why that rule plays such a big role in many systems. In Chapter 4 we mention briefly the decision problem concerning whether a given set of tautologies is enough, along with (MP), to give a complete system. Such a system is sometimes called a **Hilbert proof system**.

Fiddling around with constructing proof systems which don't have that form seems to be a bit sterile, except possibly for automated theorem proving. But it's fun.

Exercise 3.21 (i) Suppose given a proposed proof system whose rules of inference include (MP), and whose axiom schemes include $F \to \alpha(F)$. Use $\vdash_1$ to denote 'existence of derivation' for this system. Now delete that axiom scheme, and add the rule of inference $\frac{F}{\alpha(F)}$. Use $\vdash_2$ to denote 'existence of derivation' for this second system, and assume that the deduction theorem holds for it. Show that $\vdash_1 \;=\; \vdash_2$. (In particular, either both proposed systems are complete, or neither is.)

(ii) Discuss the questions of adequacy, soundness, and whether $\vdash_1 \;=\; \vdash_2$ for the pair of systems as above, except that $F \to \alpha(F)$ is changed to $F \wedge G \to \beta(F,G)$, and $\frac{F}{\alpha(F)}$ is changed to $\frac{F\ ,\ G}{\beta(F,G)}$.

(iii) Generalize any results you got in (ii) to the case of $F_1, F_2, \cdots, F_n$ (in place of F, G) .

Pondering on Exercise 3.15, one should be able to prove that a pregnant rule has to have some sort of shortening effect (on the length of the inferred formula as compared to the premiss formulae). On the other hand, although it would have been nice if otherwise, it is not hard to see that the rule $\frac{F\wedge G \text{ or } G\wedge F}{F}$ above is not pregnant. (Argue as in 3.15.) One wonders whether any pregnant rule must use at least as many symbols as (MP).

Exercise 3.22 Is the rule *modus tollens*, namely $\frac{F\to G\ ,\ \neg G}{\neg F}$, pregnant?

Exercise 3.23 Is *hypothetical syllogism*, namely $\frac{F\to G\ ,\ G\to H}{F\to H}$, pregnant?

See also the last subsection of this section.

Other styles of proof system are useful as well. There is a different approach to proof systems, sometimes called Gentzen systems or sequent-calculi. These are very useful for those who either wish to spend a lot of time constructing derivations themselves, or are interested in automated theorem proving. This is closely related to the material on **resolution and the Davis-Putnam procedure** in the next chapter. Section 3.6 of **Hodel** explains it well. Once you have studied Chapter 7 of this book, see also Chapter VI of **Ebbinghaus-Flum-Thomas** for a 'sequent-calculus' system, but for $1^{\underline{\text{st}}}$order logic.

Is there an automatic method for finding derivations, using the proof system of this chapter? The proof we gave of completeness has the virtue that it can

serve as a template for the more difficult proof of completeness for 1st order logic given in Chapter 7. But it is an *indirect* proof (i.e. a *proof by contradiction*), and can be criticized for not evidently containing a method which produces a derivation showing $\Gamma \vdash F$ from the knowledge that $\Gamma \models F$.

The next chapter is where we deal more thoroughly with questions about 'automatic methods' or 'algorithms'. As explained in more detail there, for *arbitrary* premiss sets Γ, we cannot expect an algorithm for finding derivations. There are plenty of very weird infinite sets of formulae for which there isn't even an algorithm for deciding whether a given formula is in the set.

But for 'decidable' Γ one can simply argue that there is an automatic method for listing all derivations from Γ. And so, given F with $\Gamma \models F$, by inspecting the list, you eventually come to the desired derivation. Thus there is a (very impractical!) algorithm for finding a derivation, when you know beforehand from the completeness theorem together with a truth table calculation that a derivation must exist.

As for a 'practical' method, it's interesting to see what can be done when Γ and F are small in some sense. Note that our particular system quickly gives a derivation showing $\{G_1, \cdots, G_k\} \vdash F$ from a derivation which shows $\vdash G_1 \wedge \cdots \wedge G_k \to F$.

So we're interested in an algorithm for producing derivations of tautologies from the empty set of premisses. We might hope for one which is reasonably practical, at least when the tautology is fairly small. (For the proof system in 3.20(ii), which has every tautology as an axiom, this would be rather easy, to say the least!) It can be done for the proof system of this chapter using the reduction of the tautology to its 'disjunctive normal form', which is explained in the 1st and 3rd sections of the next chapter. Another method, due to Kalmar, works well for our system and many others.

These methods give alternative proofs of the completeness theorem for finite Γ. Some discussion of getting from finite Γ to general Γ is given in the exercises leading up to 3.17. Details of this 'Kalmar approach', to proving adequacy of our system by giving a direct algorithm for finding a derivation of a given tautology, are given below the following exercises. The first exercise relates to the above remark about disjunctive normal form. The rest are used for the Kalmar approach.

Exercises

3.24 Find derivations showing $\{J\} \vdash K$ (using the system of this chapter), where J and K are, in either order :

(i) $\neg(F \vee G)$ and $\neg F \wedge \neg G$;

(ii) $\neg(F \wedge G)$ and $\neg F \vee \neg G$;

(iii) $F \vee (G \wedge H)$ and $(F \vee G) \wedge (F \vee H)$;

(iv) $F \wedge (G \vee H)$ and $(F \wedge G) \vee (F \wedge H)$.

3.25 Produce an explicit derivation showing $\vdash G_1 \wedge G_2$, using given derivations showing $\vdash G_1$ and $\vdash G_2$.

3.26 Produce a derivation showing $\{F_1 \wedge F_2\} \vdash G_1 \wedge G_2$, using given derivations showing $\{F_1\} \vdash G_1$ and $\{F_2\} \vdash G_2$.

3.27 Write down an explicit 'derivation scheme' showing

$$\{F \wedge G \to H\} \vdash G \to (F \to H) \ ,$$

by following the proof of **3.6**.

3.28 Write down a 'derivation scheme' showing

$$\{F \to G\} \vdash \neg G \to \neg F \ ,$$

by following the proof of **3.1**.

3.29 Write down a 'derivation scheme' showing

$$\{F \to G \ , \ G \to H\} \vdash F \to H \ ,$$

by following the proofs of **3.1, 3.2, 3.3**.

3.30 Write down an explicit 'derivation scheme' showing

$$\{(F \to G) \wedge (F \to H)\} \vdash F \to G \wedge H \ ,$$

perhaps using the following suggestions:

First get $F \to G$ and $F \to H$ separately. Then, from these, get $F \wedge F \to F \wedge G$ and $F \wedge G \to G \wedge H$, respectively. Now, applying 3.27 to the latter two, and then applying it again, you're done.

3.31 Write down a 'derivation scheme' showing

$$\{(F \to G) \wedge (\neg F \to G)\} \vdash G \ ,$$

perhaps using the following suggestions.

Use 3.28 and 3.25 to get $(\neg G \to \neg F) \wedge (\neg G \to F)$. Then apply 3.30, then 3.28, and finally notice that $\neg(F \wedge \neg F) = F \to F$ is an axiom.

Now we sketch how to explicitly write down a derivation for an arbitrary tautology, using the system of this chapter.

The main step is the following. Assume that the formula F involves no P_i for $i > n$. The formula $L_1 \wedge \cdots \wedge L_n \to F$ is a tautology, if F is true whenever $L_1 \wedge \cdots \wedge L_n$ is true. Here we'll take each L_i to be either P_i or $\neg P_i$.

(Note that the truth assignment sending P_i to Tr if $L_i = P_i$, and to Fs if $L_i = \neg P_i$ is really the only one where F need be true in the paragraph above. That truth assignment is unique in its behaviour on the first "n" variables, which are the only ones that matter here. So "whenever" in the above paragraph really only refers to one truth assignment, in the sense above. That is, there will be a separate case of the tautology above for each line in the truth table of F which gives Tr for F. [I've repeated myself twice just now!] The L_i are called literals in the next chapter.)

One can write down a specific derivation of that tautology, $L_1 \wedge \cdots \wedge L_n \to F$, as follows, by induction on the length of F, for a fixed $L_1 \wedge \cdots \wedge L_n$:

If the length is 1, then $F = P_i$ for some $i \leq n$. Thus we must have $L_i = P_i$. So write down the line

$$L_i \wedge (L_1 \wedge \cdots \wedge L_{i-1} \wedge L_{i+1} \wedge \cdots \wedge L_n) \to F \ ,$$

which is an instance of the axiom $F \wedge G \to F$. Now just apply the rules (COM) and (ASS) as many times as needed, depending on how things are bracketed, to arrive at $L_1 \wedge \cdots \wedge L_n \to F$, as required.

If the length is 2, then $F = \neg P_i$ for some $i \leq n$. Thus we must have $L_i = \neg P_i$. Now proceed exactly as written in the previous paragraph.

For the inductive step, assume that F has length greater than 2.

If $F = G \wedge H$, then the inductive assumption applies to both G and H, so we have derivations of both $L_1 \wedge \cdots \wedge L_n \to G$ and $L_1 \wedge \cdots \wedge L_n \to H$. Then Exercise 3.28 immediately gives a derivation of $L_1 \wedge \cdots \wedge L_n \to G \wedge H$, as required.

If $F = \neg G$, then we must go one level deeper into the induction, and ask where G came from.

If $G = \neg H$, then $F = \neg\neg H$, and the inductive assumption applies to H, so we have a derivation of $L_1 \wedge \cdots \wedge L_n \to H$. One application of $(\neg\neg)$ then gives a derivation of $L_1 \wedge \cdots \wedge L_n \to \neg\neg H$, as required.

Finally, if $G = G_1 \wedge G_2$, then either G_1 or G_2 is false when $L_1 \wedge \cdots \wedge L_n$ is true. If G_2 is false, then, by the inductive assumption, $L_1 \wedge \cdots \wedge L_n \to \neg G_2$ has a specific derivation. Use Exercise 3.28 to continue this to a derivation of

$$\neg\neg G_2 \to \neg(L_1 \wedge \cdots \wedge L_n) \ .$$

An application of $(\neg\neg)$ yields

$$G_2 \to \neg(L_1 \wedge \cdots \wedge L_n) \qquad (*) \ .$$

Write down the axiom $G_2 \wedge G_1 \to G_2$, and apply (COM) to get

$$G_1 \wedge G_2 \to G_2 \qquad (**) \ .$$

Use Exercise 3.29, applied to $(*)$ and $(**)$, to continue this to a derivation of

$$G_1 \wedge G_2 \to \neg(L_1 \wedge \cdots \wedge L_n) \ .$$

Use Exercise 3.28 to extend the derivation to one for

$$\neg\neg(L_1 \wedge \cdots \wedge L_n) \to \neg(G_1 \wedge G_2) \ .$$

One more application of $(\neg\neg)$ does it, since $F = \neg(G_1 \wedge G_2)$. On the other hand, when it is only G_1 which is false, the derivation can be constructed almost exactly as above, the only changes being to interchange G_1 and G_2, and omit the application of (COM) at $(**)$.

Now suppose that F itself is a tautology (and still involves no P_i for $i > n$). Then the above "main step" gives "2^n" derivations, showing that $\vdash L_1 \wedge \cdots \wedge L_n \to F$, for every one of the choices of the sequence of 'literals' $L_1, \cdots, L_n$.

I'll write down the case $n = 2$ of the process for getting a derivation of F itself—generalizing to arbitrary n involves no new ideas, only messier notation. Here is a sequence of steps to produce a derivation of a tautology F which involves at most P_1 and P_2.

First use all $2^n = 4$ instances of the derivation above to produce a derivation of

$$(P_1 \wedge P_2 \to F) \ \wedge \ (P_1 \wedge \neg P_2 \to F) \ \wedge \ (\neg P_1 \wedge P_2 \to F) \ \wedge \ (\neg P_1 \wedge \neg P_2 \to F) \ .$$

This also uses the extension of 3.25 from two to four ingredients.

Next use four instances of 3.27 to continue the derivation to arrive at

$$(P_2 \to (P_1 \to F) \ \wedge \ (\neg P_2 \to (P_1 \to F) \ \wedge \ (P_2 \to (\neg P_1 \to F) \ \wedge \ (\neg P_2 \to (\neg P_1 \to F) \ .$$

This also uses the extension of 3.26 from two to four ingredients.

Now use 3.31 to continue the derivation to arrive at

$$(P_1 \to F) \ \wedge \ (\neg P_1 \to F) \ .$$

This also uses 3.26.

Finally use 3.31 again to arrive at F, as required.

It is worth pointing out that, when the tautology to be derived involves many propositional variables, the process above is very inefficient. There is a 'combinatorial explosion' at the point where we decided to wimp out and just write down the case $n = 2$. The first formula written there would be a conjunction of "2^n" subformulae, so would become practically impossible, before n got very large at all. This is the kind of thing which we discuss at length in the latter part of the next chapter. If you could figure out a way to do this problem efficiently (or more accurately, the problem of seeing whether a formula is a tautology), you could become rich and famous. See Section **4.7** on **Efficiency of algorithms** in the next chapter.

A more fundamental completeness theorem? There is a whiff of arbitrariness about completeness, because of the wide possibilities for choice of proof systems. A statement which seems to be more fundamental is just Exercise 3.20(ii); namely, that *the proof system which takes all tautologies as axioms, and* (MP) *as the only rule of inference, is complete.*

Proving soundness is extremely easy.

How would we prove adequacy from scratch [that is, without relying on the expected method for doing 3.20(ii)—namely, already knowing that there is a complete proof system with only (MP) as a rule of inference]?

Firstly, it can be reduced to dealing with sets of premisses which are *finite*. This was the gist of 3.18. That exercise however was placed in the context of employing a version of the Compactness Theorem, given in 3.16, which was supposed to be proved **using** the fact that there is at least one proof system which is complete (so that would be circular here). But we gave exercises in the previous chapter, 2.34, for proving compactness purely semantically. You should find an elementary argument to demonstrate the equivalence of the two different statements of compactness:

(I) *Every finitely satisfiable set is satisfiable;* and

(II) $\Gamma \models F \implies \Omega \models F$ *for some* **finite** $\Omega \subset \Gamma$.

Now, to prove adequacy for finite sets of premisses (which we obviously can assume to be non-empty for this system): First write down a quick derivation of the derived rule of inference $\{F, G\} \vdash F \wedge G$, starting with the tautology $F \to (G \to F \wedge G)$, followed by two applications of (MP). We easily get by induction that $\{F_1, F_2, \cdots, F_m\} \vdash F_1 \wedge F_2 \wedge \cdots \wedge F_m$, where some particular choice of bracketing is made for the latter formula. Now given $\{F_1, F_2, \cdots, F_m\} \models G$, it follows that $F_1 \wedge F_2 \wedge \cdots \wedge F_m \to G$ is a tautology, so combining the previous sentence with (MP), we quickly get $\{F_1, F_2, \cdots, F_m\} \vdash G$, as required.

So the proof of the more "fundamental" completeness theorem is very brief if you start from compactness (or simply limit yourself to finite premiss sets). Fooling around with cutting down your axioms from all tautologies to some nice 'little' set (which is what we essentially did in this chapter) is interesting for answering the question 'How little do we need to base proofs on?', is lots of fun, and is useful in automated theorem proving. But any particular choice of a 'little' set of axioms and proof of completeness for it [plus (MP)] is hard to regard as being particularly fundamental. On the other hand, cutting the axioms down to a finite list of axiom schemes at least guarantees that there is an *efficient* algorithm for deciding whether a formula is an axiom. See also the final few sentences of this chapter just below.

It should be pointed out that this complete proof system with *all* tautologies as axioms is considered somewhat 'acceptable' only because the set of axioms is decidable—a given formula can be automatically checked as to whether or not it is an axiom. When we come to 1$^{\text{st}}$order logic in the second half of the book, there will turn out to be no proof system analogous to this particular one. There, the analogue of a tautology is called a *logically valid formula*, and the set of those is generally not decidable. In 1$^{\text{st}}$order logic, picking your logical axioms and proving completeness is a rather fundamental process. See Chapter 7.

It should further be pointed out that, for a system to be acceptable, its rules of inference also should be decidable, in a suitable sense. But again because of truth tables, this tends to be easy to satisfy in propositional logic (as opposed to 1$^{\text{st}}$order). One could even say that the ridiculous proof system as follows is 'acceptable': take all tautologies as axioms, and, for every F and finite set Γ for which $\Gamma \models F$, take $\frac{\Gamma}{F}$ as a rule of inference. So anything that won't violate soundness is thrown in as part of the system, except that we avoid infinite Γ, since many of those sets are not 'membership decidable'. Adequacy for finite premiss sets here is a **zero line proof**! (And no derivation needs to be lengthier than [1+ the size of the premiss set]; and actually, the axioms are all repeated as 'nothing rules'.) For general adequacy, you just need the compactness theorem, which must be proved semantically. But again, as we'll see later, in 1$^{\text{st}}$order logic, the process of picking your rules of inference is much more subtle and fundamental. The very existence of a complete proof system with decidable axiom and rule of inference sets is a fairly deep result about 1$^{\text{st}}$order logic, and is due to Gödel.

A general criterion for a language is that there should be an algorithm to check whether or not a given string of symbols is a formula. (See **Appendix B** to Chapter 8.) And, in addition to soundness and completeness, a general criterion for a proof system, given a [language, plus general notion of interpretation for it], is that there should be an algorithm to check whether or not a given sequence of formulae is a derivation. That's where the decidability referred to above of the sets of axioms and rules of inference is needed. Most mathematicians would still regard the more-or-less ridiculous systems for propositional logic in the previous few paragraphs as unacceptable. Perhaps one reason for that is the fact that all the known algorithms for showing decidability of the set of all tautologies are inefficient—not 'polynomial-time'. In the next chapter, this important idea will be explained in more detail. Semantic versions of the fact above will be major theme in the latter part of that chapter. So perhaps the general criterion above should be strengthened by requiring the algorithms for deciding axiomhood and rule-of-inference-hood to be *efficient* algorithms.

Thus, despite giving definitions of neither the noun **algorithm** nor the adjective **efficient**, here is the basic result of this chapter as a theorem:

Theorem. *There exists a proof system for propositional logic which is complete, and such that there is an efficient algorithm for deciding whether a given sequence of formulae is a derivation, as long as the premiss set comes equipped with an efficient algorithm for deciding whether a given formula is a premiss. Furthermore this theorem can, if desired, be proved constructively by giving a (not necessarily efficient) algorithm for producing a derivation when one exists i.e. when the argument is valid.*

(At present (2003), no efficient algorithm is known for first deciding whether a derivation exists for a given (finite premiss set, conclusion), that being one form of the main problem studied in the second half of the next chapter.)

And now (briefly) for something completely different, of interest to computists. One has proof systems for formally proving the correctness of computer programs. Sometimes the subject goes by the name *Floyd-Hoare logic*. A subject called *dynamic logic* has a large overlap with it, "dynamic" referring to the possibility of a program changing some truth values as 'time' progresses. See Ch VIII of **[CM]**. An interesting quote from **Gordon**, p.33:

> There are various subtle technical problems in formulating precisely what it means for a Floyd-Hoare logic to be complete ... The completeness of a Floyd-Hoare logic must thus be defined independently of that of its assertion language ... Clarke's paper also contains discussion of his important results showing the impossibility of giving complete inference systems for certain combinations of programming language constructs. For example, he proves that it is impossible to give a sound and complete system for any language combining procedures as parameters of procedure calls, recursion, static scopes, global variables and internal procedures as parameters for procedure calls. These features are found in Algol 60, which thus cannot have a sound and complete Floyd-Hoare logic.

4. CALCULATIONS AND ALGORITHMS IN PROPOSITIONAL LOGIC

Remarkably, (despite his internet connections) **Al Gore** didn't invent any of the **algor**ithms in this chapter; the word "algorithm" comes rather from the name of an Islamic mathematician from the 9th century. His name is sometimes written **al-Khuwarizmi**; a more complete version, with slightly different Roman alphabet spelling is **Abu Ja'far Mohammed ibn Mûsâ al-Khowârizm**. An influential book of his dating from about 825 AD is entitled Kitab al-jabr wa'l-muqabala. The word *algebra* is derived from the obvious one of those Arabic words.

Here is a brief preview of a main topic from the initial five sections below. Suppose we came upon what looked like a truth table, but for some reason we couldn't read the formula whose truth values the table was supposed to have calculated. From the values in the table, it's not possible to *exactly* determine the formula. But one can at least determine a few formulae which are *truth equivalent* to it.

For example, here is such a table.

P	Q	R	$F =?$
Tr	Tr	Tr	Fs
Tr	Tr	Fs	Fs
Tr	Fs	Tr	Tr
Tr	Fs	Fs	Fs
Fs	Tr	Tr	Fs
Fs	Tr	Fs	Fs
Fs	Fs	Tr	Tr
Fs	Fs	Fs	Tr

For this table, one such formula is

$$F \;=\; (P \wedge \neg Q \wedge R) \vee (\neg P \wedge \neg Q \wedge R) \vee (\neg P \wedge \neg Q \wedge \neg R)$$

Another is

$$F \;=\; (\neg P \vee \neg Q \vee \neg R) \wedge (\neg P \vee \neg Q \vee R) \wedge (\neg P \vee Q \vee R) \wedge (P \vee \neg Q \vee \neg R) \wedge (P \vee \neg Q \vee R)$$

Having written these down, you can directly check that both do have that truth table above. But how would you find them in the first place? I'll just

say that the first came from looking only at the lines above giving Tr for F. And the second came from looking only at the lines above giving Fs for F. You might enjoy trying to puzzle out the method, before reading all about it in the following sections. The first is a DNF; the second, a CNF.

4.1 Disjunctive Normal Form

Definitions. A *DNF-constituent from* $P_1, \cdots, P_n$ is a string of the form $L_1 \wedge L_2 \wedge \cdots \wedge L_n$, where each L_i is either the literal P_i or the literal $\neg P_i$. A *literal*, to give a cute definition, is just a formula of length less than 3. But let's not be cute—the literals are simply the following :

$$P_1 \ , \ \neg P_1 \ , \ P_2 \ , \ \neg P_2 \ , \ P_3 \ , \ \neg P_3 \ , \ P_4 \ , \ \cdots\cdots \quad .$$

So, for example, $\neg P_1 \wedge P_2 \wedge \neg P_3 \wedge \neg P_4$ is one of the "$2^4 = 16$" DNF-constituents from P_1, P_2, P_3, P_4. The complete list of DNF-constituents from P_1, P_2 is

$$P_1 \wedge P_2 \quad , \quad \neg P_1 \wedge P_2 \quad , \quad P_1 \wedge \neg P_2 \quad , \quad \neg P_1 \wedge \neg P_2 \quad .$$

Note. For $n \geq 3$, such a constituent is not quite a formula, or even an abbreviated formula, since some pair(s) of 'associativity' brackets would need to be added to make it into a formula. But all the different formulae you'd get by adding brackets in different ways are equivalent, so we speak (loosely) of the 'formula' $L_1 \wedge L_2 \wedge \cdots \wedge L_n$.

Definition. A *disjunctive normal form* (or just *DNF*) from $P_1, \cdots, P_n$ is a string $(C_1) \vee (C_2) \vee \cdots \vee (C_r)$, where each C_j is some DNF-constituent from $P_1, \cdots, P_n$, and all the C_j are different from each other.

For example, $(P_1 \wedge \neg P_2 \wedge P_3) \vee (\neg P_1 \wedge \neg P_2 \wedge \neg P_3)$ is a DNF from P_1, P_2, P_3 in which the number, "r", of constituents is 2. There are "$2^{2^3} - 1 = 255$" other DNF's from P_1, P_2, P_3, if we ignore the order in which the constituents appear. (See the next paragraph.)

Note. Again, to get (an abbreviation for) a formula, one needs to insert more brackets. But all formulae you get by inserting them, in different possible ways, are equivalent. Also, changing the order of appearance of the constituents, C_j, will not alter the truth equivalence class of that formula. We refer (loosely) to a DNF as a "*formula in DNF-form*", sometimes adding the phrase "*from* $P_1, \cdots, P_n$". Since there are "$8{=}2^3$" DNF constituents from P_1, P_2, P_3, there are (ignoring bracketing and order of constituents) a

total of "$2^8 = 256$" different DNF's from P_1, P_2, P_3 ; in general , "2^{2^n}" from $P_1, \cdots, P_n$. Note also that the convention we have given is, within each constituent, to keep the literals in their natural order with increasing subscripts.

Notice that *a DNF is true at some particular truth assignment* e *if and only if it contains a constituent all of whose literals are true at* e , partly since it's a *disjunction* of its constituents. But such a constituent clearly determines all the values of e at the relevant propositional variables (the first "n"), since that constituent is a *conjunction* of literals, and there's one literal corresponding to every relevant variable. And every other constituent (whether appearing in the given DNF or not) is false at that particular e.

For example, 'the' truth assignment e which makes $\neg P \wedge Q \wedge R \wedge \neg S$ true is given by

$$\mathsf{e}(P) = \mathsf{Fs} \quad , \quad \mathsf{e}(Q) = \mathsf{Tr} \quad , \quad \mathsf{e}(R) = \mathsf{Tr} \quad , \quad \mathsf{e}(S) = \mathsf{Fs} \quad .$$

The Fs's correspond to where a "$\neg$" occurs, of course. And every other DNF-constituent in P, Q, R and S is necessarily false at the e just above. And finally, e is only non-unique to the extent that it can do what it wants on variables other than P, Q, R and S.

Another way of saying this important general fact is as follows: In the truth table of a DNF-constituent from $P_1, \cdots, P_n$, that constituent has Tr in only one line, and Fs in all the "$2^n - 1$" other lines. The 'true line' is different for different constituents. And so, the number of lines where a DNF formula is true is the same as the number of constituents in that formula.

Just so that the next theorem can be stated elegantly, we shall allow the **empty DNF** to *be* a DNF. This is the one where $r = 0$ in the previous definition; that is, there are no constituents at all. Looking at the criterion for a DNF being true just above, we must say that *the empty DNF is false at* **every** e (because we cannot find a constituent all of whose literals are true at e, since there are no constituents at all!) So a formula corresponding to the empty DNF is any contradiction; that is, any unsatisfiable formula. (The empty DNF was included above when we did the counting.)

At the opposite end of the spectrum is the DNF which contains **every** constituent from the given relevant set of the first "n" propositional variables. It is true at every truth assignment, so corresponds to (and belongs to) the equivalence class consisting of tautologies.

Theorem 4.1 *If F is a formula not involving any P_i for any $i > n$, then F is truth equivalent to a DNF-formula from $P_1, \cdots, P_n$. Furthermore, that DNF is unique, up to rearranging the order of the constituents*
(and up to choice of bracketing, if you insist on writing it as an actual abbreviation for a formula)

We'll give a second proof and algorithm for this further down, one which doesn't use truth tables.

Proof and algorithm. By the remarks in the four paragraphs prior to the theorem, a pair of DNF's give a pair of truth equivalent formulae if and only if those DNF's have the same constituents. This establishes the uniqueness claimed in the second sentence of the theorem.

But those paragraphs also tell us how to find a DNF for a formula F. Write out a basic truth table for F, that is, one which has just "n" columns for the relevant propositional variables, plus a last column for F. Look down that last column for lines with "Tr" under F. For each such line (if any), write down one constituent, determined by looking at the other entries in that line. If "Tr" occurs under P_i, just use P_i as the literal L_i for this constituent. If "Fs" occurs under P_i, use $\neg P_i$ as the literal. This DNF (obtained by '$\vee$-ing' all those constituents, one for each line with a Tr under F) has exactly the same truth table as the given formula F, and is therefore truth equivalent to it by definition, as required.

Here is an example of the algorithm described in the above proof. I'll use the example $F = R \wedge \neg(\neg(P \wedge \neg Q) \wedge P)$ from the introduction (page 7), whose truth table was determined in the second chapter:

P	Q	R	F
Tr	Tr	Tr	Fs
Tr	Tr	Fs	Fs
Tr	Fs	Tr	Tr
Tr	Fs	Fs	Fs
Fs	Tr	Tr	Tr
Fs	Tr	Fs	Fs
Fs	Fs	Tr	Tr
Fs	Fs	Fs	Fs

Only the third, fifth and seventh lines in the truth table above have "Tr" under F, so there will be three constituents. The DNF is given below, with

the constituents, from left to right obtained from those three lines in that order:

$$F \quad \text{treq} \quad (P \wedge \neg Q \wedge R) \vee (\neg P \wedge Q \wedge R) \vee (\neg P \wedge \neg Q \wedge R) \quad .$$

This DNF formula clearly has a much greater degree of symmetry than the formula, $F = R \wedge \neg(\neg(P \wedge \neg Q) \wedge P)$, with which we started. Every process of real interest in the semantic part of logic essentially doesn't change if we replace a formula by an equivalent formula. (Of course, you can't just replace a formula on one line in a derivation by another one to which it is truth equivalent!) For many purposes, the DNF form of a formula, and/or the CNF form four sections below, are very convenient. For example, after calculating a few of these, you'll see how you more or less have the truth table built right into the DNF—you can just read off which are the e's where it's true (and the rest of the e's are where it's false). That comes from the example and the paragraph preceding it, before the statement of **4.1**.

We have treated P, Q and R as 'abbreviations' for the relevant variables P_1, P_2 and P_3, respectively, which occurred in the general treatment. In anything up to 5 or 6 variables, it's cleaner to get away from all those subscripts. One needs to use at least all the variables occurring in the formula. The only time you want to include some other variables, each of which will double the number of constituents, is for the purpose of comparing the DNF's of several formulae, which may involve slightly different variables. Then you write a DNF for each of them using all the variables occurring in all of them. By renaming things, these can always be taken to be the *first* "n" variables for some n. Alternatively, in the general treatment, we clearly could have been dealing with any sequence of "n" variables, not just the first "n" in their natural order. If it's necessary to add one or more extra variables which don't occur in the formula being dealt with, the algebraic manipulation procedure in the third section of this chapter (rather than use of truth tables) is the best way to 'pump up' a DNF, throwing in an extra variable which actually didn't occur in the formula itself. Effectively you just add both the variable and its negation, producing two constituents out of each constituent which occurred before adding the 'irrelevant' variable.

This chapter's introductory paragraphs give a second example of a DNF calculated by the truth table method.

Corollary *The number of truth equivalence classes of formulae in* $P_1, \cdots, P_n$ *is* 2^{2^n} . This includes formulae which don't actually involve **all** those propositional variables.

This is clear, since that's the number of DNF's. (The set of constituents has 2^n elements, and each DNF is essentially a *subset* of that set, of which there are 2^{2^n} .)

Exercises.

4.1 Use the truth table method to find the DNF of the following formulae. Here just use $P, Q, \cdots$ instead of their other nicknames $P_1, P_2, \cdots$; and only use the variables actually occurring in the formula (not any extra ones) in the DNF.

(i) $R \leftrightarrow P \wedge (Q \rightarrow R)$
(ii) $P \rightarrow (Q \rightarrow (R \rightarrow S))$
(iii) $(P \rightarrow Q) \wedge (P \wedge \neg Q)$

(Which of the above is designed to convince you that a better method needs to be found?)

4.2 (One way to deal with both parts of this question is to think about Theorem **4.1**, and to regard the questions as partly a calculation of DNF's.)

(i) Find a list of formulae which
(a) use no variables other than P, Q and R; and
(b) have the following 'partial' truth table (two values missing); and
(c) are such that any formula for which (a) and (b) hold is truth equivalent to exactly one from your list:

P	Q	R	$F =?$
Tr	Tr	Tr	Fs
Tr	Tr	Fs	?
Tr	Fs	Tr	Tr
Tr	Fs	Fs	Tr
Fs	Tr	Tr	Fs
Fs	Tr	Fs	?
Fs	Fs	Tr	Tr
Fs	Fs	Fs	Fs

(ii) Using DNFs, write down a list of formulae which use no variables other than P and Q, and such that every formula in those two variables is equivalent to exactly one in your list. (This is like (i), except that the question marks predominate.)

(iii) Now do the same as in (ii), except don't necessarily use DNFs, but rather make each abbreviated formula in your list is *as brief as possible* within its equivalence class. (Don't count brackets in checking the length of a formula, and allow yourself all five 'famous' connectives.)

4.2 Hilbert's Truth Calculating Morphism.

This is a good place to describe a procedure which can replace truth tables wherever they are used. It is less popular (but pretty cool). It was the procedure emphasized by Hilbert and Ackermann in the first elementary book on this material.

The idea is, for a given e, to substitute the truth values of the variables directly into a formula and do a kind of algebraic calculation, where, rather than the basic definitional truth tables, one uses the 'rules' :

$$\neg\mathsf{Tr} = \mathsf{Fs} \quad , \quad \neg\mathsf{Fs} = \mathsf{Tr} \ ;$$

and

$$\mathsf{Tr} \wedge \mathsf{Tr} = \mathsf{Tr} \quad , \quad \mathsf{Tr} \wedge \mathsf{Fs} = \mathsf{Fs} \quad , \quad \mathsf{Fs} \wedge \mathsf{Tr} = \mathsf{Fs} \quad , \quad \mathsf{Fs} \wedge \mathsf{Fs} = \mathsf{Fs} \ .$$

(In many books, the notation for the numbers 0 and 1 is used instead of Fs and Tr, respectively. Then the reader is encouraged to think of $\wedge$ as being just like multiplication, because of the rules just above. Also, $\neg$ acts like the function $x \mapsto 1 - x$ in this context, and $\vee$ acts somewhat like addition. Sometimes this analogy is carried a bit too far, and becomes counterproductive.)

Applying this substitution method to the formula $F \ = \ R \wedge \neg(\neg(P \wedge \neg Q) \wedge P)$ of the previous example, the calculation for the "e" corresponding to the fifth line of the truth table is

$$\begin{aligned} &\mathsf{Tr} \wedge \neg(\neg(\mathsf{Fs} \wedge \neg\mathsf{Tr}) \wedge \mathsf{Fs}) \ = \ \mathsf{Tr} \wedge \neg(\neg(\mathsf{Fs} \wedge \mathsf{Fs}) \wedge \mathsf{Fs}) \\ &\quad = \ \mathsf{Tr} \wedge \neg(\neg\mathsf{Fs} \wedge \mathsf{Fs}) \ = \ \mathsf{Tr} \wedge \neg(\mathsf{Tr} \wedge \mathsf{Fs}) \ = \ \mathsf{Tr} \wedge \neg\mathsf{Fs} \ = \ \mathsf{Tr} \wedge \mathsf{Tr} \ = \ \mathsf{Tr} \ . \end{aligned}$$

This, of course, is the correct answer, since the fifth row did give "true".

To do a similar calculation starting from a formula abbreviation which uses some of our other famous connectives, we need the following rules. They are obtained from, and take the place of, the basic truth tables for these connectives :

$$\mathsf{Tr} \vee \mathsf{Tr} = \mathsf{Tr} \quad , \quad \mathsf{Tr} \vee \mathsf{Fs} = \mathsf{Tr} \quad , \quad \mathsf{Fs} \vee \mathsf{Tr} = \mathsf{Tr} \quad , \quad \mathsf{Fs} \vee \mathsf{Fs} = \mathsf{Fs} \ ;$$

$$\mathsf{Tr} \to \mathsf{Tr} = \mathsf{Tr} \quad , \quad \mathsf{Tr} \to \mathsf{Fs} = \mathsf{Fs} \quad , \quad \mathsf{Fs} \to \mathsf{Tr} = \mathsf{Tr} \quad , \quad \mathsf{Fs} \to \mathsf{Fs} = \mathsf{Tr} \ ;$$

$$\mathsf{Tr} \leftrightarrow \mathsf{Tr} = \mathsf{Tr} \quad , \quad \mathsf{Tr} \leftrightarrow \mathsf{Fs} = \mathsf{Fs} \quad , \quad \mathsf{Fs} \leftrightarrow \mathsf{Tr} = \mathsf{Fs} \quad , \quad \mathsf{Fs} \leftrightarrow \mathsf{Fs} = \mathsf{Tr} \ .$$

We'll use the first of the three lines, plus an 'iteration' of the rules for "$\wedge$" and "$\vee$" to see how quickly and uniformly this calculational trick works, when applied instead to the DNF form of the previous formula, rather than to the formula itself. Again we are using the "e" which gives the fifth line of the previous truth table :

$$\begin{aligned} &(\mathsf{Fs} \wedge \neg\mathsf{Tr} \wedge \mathsf{Tr}) \vee (\neg\mathsf{Fs} \wedge \mathsf{Tr} \wedge \mathsf{Tr}) \vee (\neg\mathsf{Fs} \wedge \neg\mathsf{Tr} \wedge \mathsf{Tr}) \\ &\quad = \ (\mathsf{Fs} \wedge \mathsf{Fs} \wedge \mathsf{Tr}) \vee (\mathsf{Tr} \wedge \mathsf{Tr} \wedge \mathsf{Tr}) \vee (\mathsf{Tr} \wedge \mathsf{Fs} \wedge \mathsf{Tr}) \ = \ \mathsf{Fs} \vee \mathsf{Tr} \vee \mathsf{Fs} \ = \ \mathsf{Tr} \ , \end{aligned}$$

again giving the right answer.

In a calculation like this with a DNF formula, at most one constituent will yield Tr, of course. Notice how the 'triple true' occurred as the middle one. This is because the fifth line in the truth table was the middle of the three lines which yielded "true".

Exercise 4.3. Do the Hilbert truth morphism calculation for the seven other truth table lines, for both F above, and for its DNF.

4.3 DNF by Manipulation with Elementary Equivalences.

Here we describe a method, for getting the DNF, which resembles algebraic manipulations usually first learned in secondary school.

This method has been sometimes called 'transformational proof', it being quite obvious that if you can manipulate from F and eventually obtain G, then $\{F\} \models G$ (since then, $F \overset{\text{treq}}{} G$). But that name for the process is unfortunate, since certainly it is very far from being complete as a 'proof system'. The problem isn't that one can only have a single premiss, since any finite set of premisses can always be conjuncted together as a kind of hidden rule of inference, to reduce to a single premiss. But there are plenty of examples where F is *not* truth equivalent to G, and yet $\{F\} \models G$. For example, $F = P \wedge (P \to Q)$ and $G = Q$ gives an example (closely related to modus ponens). Prove that it's an example! Then you can't get from F to G by manipulation as described below, so this appellation 'transformational proof', (due to the computists I believe) is a pretty bad idea pedagogically. It is true however that the method below is very syntactical, despite this chapter being really about semantics. However, the notion of truth equivalence is definitely at the centre of semantical considerations.

A DNF only involves $\neg$, $\wedge$ and $\vee$. So if one starts with an abbreviated formula which may use $\to$ and/or $\leftrightarrow$ as well, it is best to eliminate $\leftrightarrow$ using its definition, and then replace each $\to$ by either its definition, or, usually better, use the equivalence below:

$$\neg(F \wedge \neg G) \quad = \quad (F \to G) \quad \overset{\text{treq}}{} \quad \neg F \vee G \quad .$$

At this point, you have a formula involving only $\neg$, $\wedge$ and $\vee$. As explained below the list, using the following list of 'elementary' equivalences, it is always possible to produce the DNF of the formula by successive use of the 'identities' in the list.

(1) $(IDEM)_{\vee}$	.....	$F \vee F$	treq	F
(2) $(IDEM)_{\wedge}$	.....	$F \wedge F$	treq	F
(3) $(CANC)_{\neg\neg}$	.....	$\neg\neg F$	treq	F
(4) $(DIST)_{\neg/\wedge}$	.....	$\neg(F \wedge G)$	treq	$\neg F \vee \neg G$
(5) $(DIST)_{\neg/\vee}$	.....	$\neg(F \vee G)$	treq	$\neg F \wedge \neg G$
(6) $(DIST)_{\vee/\wedge}$	.....	$(F \wedge G) \vee H$	treq	$(F \vee H) \wedge (G \vee H)$
(7) $(DIST)_{\wedge/\vee}$	.....	$(F \vee G) \wedge H$	treq	$(F \wedge H) \vee (G \wedge H)$
(8) $(CANC/ADD)_{\text{taut}}$	.....	$(F \vee \neg F \vee H) \wedge G$	treq	G
(9) $(CANC/ADD)_{\text{unsat}}$	.....	$(F \wedge \neg F \wedge H) \vee G$	treq	G
(10) $(COMM)_{\wedge}$	.....	$F \wedge G$	treq	$G \wedge F$
(11) $(COMM)_{\vee}$	.....	$F \vee G$	treq	$G \vee F$

The theorem (**2.1** or **2.1***), which tells us that *replacement preserves equivalence*, is used many times without comment in such a manipulation. We also use 'associativity' of "$\wedge$" and "$\vee$" up to equivalence implicitly, by just dropping the pairs of brackets which are used in parsing them, as we've done in the DNF itself. Without getting too formal, the algorithm to use goes as follows. Laws (1) and (2) are used whenever an opportunity arises, to simplify by 'getting rid of extra copies'.

(I) Alternately move all $\neg$'s inside any brackets [using (4) and (5)], and delete double negations [using (3)]. At this point, you have something resembling a polynomial, except that "$\wedge$" and "$\vee$" are the operations (rather than multiplication and addition), and they are operating on literals (rather than on the variables in the polynomial).

(II) Using the distributive laws, (6) and (7), and deleting terms by using (8) and (9), get eventually a *disjunctive form*, not necessarily normal. This looks like a DNF except that what would be constituents may be missing some of the relevant variables, and the order of the variables is not 'right'. This form of a formula is sometimes *more* useful than the DNF.

(III) Now use (8) and (9) ['in reverse' compared to before] to add missing variables to make full-fledged constituents, using (10) to get the variables into the correct order.

Let's find the DNF form of $P \leftrightarrow Q$ by this method.
Firstly, the definition gives $(P \to Q) \wedge (Q \to P)$.
Using the definition of "$\to$" would then produce an a formula with no arrows, but it's probably better now to write down the equivalent formula

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

This almost looks like a DNF formula, except that the $\vee$'s and the $\wedge$'s are reversed. (It's actually a CNF formula, as studied two sections below.)
Now using the list of elementary equivalences, we get the following. As an exercise, the reader should find which elementary equivalences justify each step. (Sometimes we do two steps in one.)

$$(\neg P \vee Q) \wedge (\neg Q \vee P) \quad \overset{\text{treq}}{} \quad ((\neg P \vee Q) \wedge \neg Q) \vee ((\neg P \vee Q) \wedge P)$$

$$\overset{\text{treq}}{} \quad (\neg P \wedge \neg Q) \vee (Q \wedge \neg Q) \vee (\neg P \wedge P) \vee (Q \wedge P) \quad \overset{\text{treq}}{} \quad (\neg P \wedge \neg Q) \vee (P \wedge Q) \ .$$

One can virtually read off the truth table for $P \leftrightarrow Q$ from the last formula, which is its DNF.

Here is an example which we make about as complicated as it can get, illustrating all the points above. We'll do our old friend again.

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

$$\overset{\text{treq}}{} \quad R \wedge \neg((\neg P \vee \neg\neg Q) \wedge P) \qquad \text{using} \qquad (4)$$

$$\overset{\text{treq}}{} \quad R \wedge \neg((\neg P \vee \ \ Q) \wedge P) \qquad (3)$$

$$\overset{\text{treq}}{} \quad R \wedge (\neg(\neg P \vee Q) \vee \neg P) \qquad (4)$$

$$\overset{\text{treq}}{} \quad R \wedge ((\neg\neg P \wedge \neg Q) \vee \neg P) \qquad (5)$$

$$\overset{\text{treq}}{} \quad R \wedge ((P \wedge \neg Q) \vee \neg P) \qquad (3)$$

$$\overset{\text{treq}}{} \quad R \wedge ((P \vee \neg P) \wedge (\neg Q \vee \neg P)) \qquad (6)$$

$$\overset{\text{treq}}{} \quad R \wedge (\neg Q \vee \neg P) \qquad (8)$$

$$\overset{\text{treq}}{} \quad (R \wedge \neg Q) \vee (R \wedge \neg P) \qquad \text{(a disjunctive form)} \qquad (7)$$

$$\overset{\text{treq}}{} \quad ((P \vee \neg P) \wedge R \wedge \neg Q) \vee (\neg P \wedge (Q \vee \neg Q) \wedge R) \qquad (8) \text{ twice}$$

$$\overset{\text{treq}}{} \quad (P \wedge \neg Q \wedge R) \vee (\neg P \wedge \neg Q \wedge R) \vee (\neg P \wedge Q \wedge R) \vee (\neg P \wedge \neg Q \wedge R) \qquad (7)\&(10)$$

$$\overset{\text{treq}}{} \quad (P \wedge \neg Q \wedge R) \vee (\neg P \wedge Q \wedge R) \vee (\neg P \wedge \neg Q \wedge R) \qquad (1)\&(11)$$

Notice that this does, as it must, produce the *same* DNF constituents for our formula which the truth table method had produced two sections earlier. (With enough practice, you soon learn to jump directly from any disjunctive form to the DNF formula, doing the last four lines above in one step—basically, you just put in all the missing variables in every way possible.)

You may have noticed that the previous example is done inefficiently in at least two places: We could get the 6th line in two steps by moving the leftmost "¬". Then from the 6th line, we could have proceeded as follows:

. . . . . .

$$\begin{array}{lll} \overset{\text{treq}}{} & (R \wedge (P \wedge \neg Q)) \vee (R \wedge \neg P) & (7)\&(10) \\ \overset{\text{treq}}{} & (R \wedge P \wedge \neg Q) \vee (\neg P \wedge R) & (10) \\ \overset{\text{treq}}{} & (R \wedge P \wedge \neg Q) \vee (\neg P \wedge (Q \vee \neg Q) \wedge R) & (8) \\ \overset{\text{treq}}{} & (P \wedge \neg Q \wedge R) \vee (\neg P \wedge Q \wedge R) \vee (\neg P \wedge \neg Q \wedge R) & (7)\&(10) \end{array}$$

So in the end, we could have done it in seven lines, rather than twelve; or even in six, jumping directly from the disjunctive form of the second line to the DNF in the fourth line just above. However, the over-long way we first used does give plenty of illustrations of typical steps!

This method gives a different proof and algorithm for **4.1**.

It might have occurred to you to ask about what happens with this method when you start with an unsatisfiable formula. The answer is that you always end up with $P \wedge \neg P$ for one of the variables P, and then you're stuck there (unless you want to erase it and leave a blank space, which is more-or-less correct for the DNF of an unsatisfiable formula!). You'd like to use rule (9), but there's no G there to write down. Here's a simple example of this :

$$P \leftrightarrow \neg P \quad \overset{\text{treq}}{} \quad (P \rightarrow \neg P) \wedge (\neg P \rightarrow P)$$

$$\overset{\text{treq}}{} \quad (\neg P \vee \neg P) \wedge (\neg\neg P \vee P) \quad \overset{\text{treq}}{} \quad (\neg P \vee \neg P) \wedge (P \vee P)$$

$$\overset{\text{treq}}{} \quad (\neg P \wedge P) \vee (\neg P \wedge P) \vee (\neg P \wedge P) \vee (\neg P \wedge P) \quad \overset{\text{treq}}{} \quad P \wedge \neg P \,.$$

The last step used (9) three times [as well as (10)], but I guess we can't get rid of the last one, since it's hard to distinguish between a blank space which says something, and one which doesn't!

Alternatively, we could have just skipped the second last step, applying (1) twice to the third last formula.

Yet a third way to do this example is to substitute $\neg P$ for Q in the previous DNF for $P \leftrightarrow Q$. That gives $(\neg P \wedge \neg\neg P) \vee (P \wedge \neg P) \overset{\text{treq}}{} P \wedge \neg P$, as required.

Exercises. 4.4 Use the manipulation method to find the DNF of the following formulae. Only use the variables actually occurring in the formula, not any extra ones, in the DNF.

(i) $R \leftrightarrow P \wedge (Q \rightarrow R)$

(ii) $P \rightarrow (Q \rightarrow (R \rightarrow S))$

(iii) $(P \rightarrow Q) \wedge (P \wedge \neg Q)$

(iv) $P \rightarrow \neg(Q \rightarrow \neg(R \rightarrow S))$

(Which of the above is designed to convince you that a better method still needs to be found?)

4.5 Find 'the' DNF for the example in 4.4(i), but the DNF which uses the variable S, as well as P, Q and R.

4.4 (A more than adequate discussion of) ADEQUATE CONNECTIVES

One question which this section will settle is the following. How do we know that we're not missing something by only using the connectives "$\neg$" and "$\wedge$" when we set up the language of propositional logic? The same question could be asked no matter which connectives were used from the beginning.

By the very definition of *formula* which we have given, it is clear that every formula "can be expressed in terms of $\neg$ and $\wedge$".

But in a different treatment where, for example, all of $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ are considered to be *basic* symbols, the corresponding fact has to be restated to say: *Every formula is equivalent to some formula in which the only connectives used are* $\{\neg, \wedge\}$ *(as well as brackets and propositional variables).* This then must be proved. Doing so amounts to observing two things:

(i) that the other connectives can be 'expressed' in terms of these two (so our three definitions, of $\{\vee, \rightarrow, \leftrightarrow\}$, become truth equivalences in this treatment); and

(ii) that replacing (in a 'large' formula) an occurrence of a subformula, by

any formula equivalent to that subformula, does not alter the large formula up to equivalence. (This was **2.1/2.1***.)
To re-iterate, **we** don't need to do this, because of our definition of what a formula is, and of $\vee, \rightarrow$ and $\leftrightarrow$ as abbreviations. The fact we have been discussing is sometimes stated as :

The set $\{\neg, \wedge\}$ *is an* **adequate** *set of connectives, in the weak sense that every formula is equivalent to at least one formula which uses only these two connectives.*

But for us, and for other treatments, there is a real question about other sets of connectives. For example, both of the sets $\{\neg, \vee\}$ and $\{\neg, \rightarrow\}$ are adequate in the sense above. In each case, it suffices to express the other famous connectives in terms of the given ones as follows :

For $\{\neg, \vee\}$

$$F \wedge G \quad \text{treq} \quad \neg(\neg F \vee \neg G)$$

$$F \rightarrow G \quad \text{treq} \quad \neg F \vee G$$

$$F \leftrightarrow G \quad \text{treq} \quad (F \rightarrow G) \wedge (G \rightarrow F) \quad \text{treq} \quad \ldots\ldots$$

Exercise 4.6 (i) Complete the last one in a couple more steps, getting a formula that uses only "$\neg$" and "$\vee$".

(ii) Find the equivalences needed to prove the adequacy of $\{\neg, \rightarrow\}$.

Note that we did more than necessary above: because of the way we set things up, for us to show that some set of connectives is adequate, we need only 'express' both $\neg F$ and $F \wedge G$ in terms of the connectives in the set. Since "$\neg$" is actually *in* all the sets above, expressing $\neg F$ was not exactly a gargantuan task.

It might be suspected that any adequate set of connectives *must* contain "$\neg$". This turns out to be wrong; in fact, one can find an adequate set with only one connective in it !

Theorem 4.2 *If* $F * G$ *is defined to be* $\neg F \wedge \neg G$*, then the singleton set* $\{*\}$ *is adequate.*

Proof. Expressing our two basic connectives can be done as follows :

$$\neg F \quad \text{treq} \quad \neg F \wedge \neg F \;=\; F * F$$

$$F \wedge G \quad \text{treq} \quad \neg\neg F \wedge \neg\neg G \;=\; \neg F * \neg G \quad \text{treq} \quad (F * F) * (G * G)$$

The connective $*$ is called a *Sheffer connective*, though it turns out that Schröder had discovered it earlier. It occurred in Exercise 2.21 with the computists' name "nor". Actually, the one which Sheffer independently discovered was $\neg F \vee \neg G$, equivalent to the one they call "nand". These two are the only possibilities for a single adequate connective expressible using just two formulae F and G, i.e. a *binary* connective. There are many more using three or more i.e. single *trinary, quaternary, 5-ary*, etc. connectives which turn out to be adequate by themselves. Until recently, the business of single adequate connectives was just a 'mathematical circus trick' since none of them corresponds to a very natural linguistic construction (although **Russell and Whitehead** were excited about them). So they would be unnatural to use as a fundamental object in studying logic. But studying this sort of thing now turns out to be useful in the manufacture of logic gates and circuits for computers.

There is a somewhat deeper alternative definition of the notion of an adequate set of connectives. To introduce it, consider a '*nude* $n||1$ *truth table*' as follows:

P_1	P_2	$\cdots$	$\cdots$	P_{n-1}	P_n	
Tr	Tr	$\cdots$	$\cdots$	Tr	Tr	$\diamond$
Tr	Tr	$\cdots$	$\cdots$	Tr	Fs	$\diamond$
Tr	Tr	$\cdots$	$\cdots$	Fs	Tr	$\diamond$
$\cdot$	$\cdot$	$\cdots$	$\cdots$	$\cdot$	$\cdot$	$\diamond$
$\cdot$	$\cdot$	$\cdots$	$\cdots$	$\cdot$	$\cdot$	$\diamond$

The only nude part is the top right corner. This is a potential truth table in search of a formula whose truth table it wants to be. So the $\diamond$'s in the right-hand column are a collection of Tr's and Fs's filled in in any way you wish. All the other entries to the left of that are Tr's and Fs's filled in in the standard way you do that, so below the top line there are "2^n" lines, as usual. The question is : Is there a formula involving at most $P_1, \cdots, P_n$ (i.e. not involving P_i for $i > n$) whose truth table is exactly the given one? The answer is **yes**. We have really already proved that with DNF-theory. To find a formula (in fact a DNF formula), just look in the right-most column for all the Tr's. Then use the entries in each such line to write down one DNF-

constituent. Disjuncting all these constituents together into a DNF formula is all we need to do, by the theory of the first section of this chapter. [See also Exercise 2.21(iii)/4.2(ii).] If there are no occurrences of Tr in the right-hand column, of course any unsatisfiable formula will do, since the so-called empty DNF isn't really a formula.

So we conclude that *any adequate set of connectives in the previous sense is also adequate in this stronger sense. That is, any possible combination of truth values can be expressed using the connectives we already have.*

This fact, as applied to the set $\{\neg, \wedge\}$, is the main technical justification for using the restricted language which we did, right from the beginning of the book.

Exercise 4.7 (i) Show that the Sheffer connective nand is adequate by itself.

(ii) Is "$*$" associative (up to equivalence)? That is, do we have

$$F * (G * H) \quad \text{treq} \quad (F * G) * H$$

for all formulae F, G and H?

(iii) Is nand associative up to equivalence?

(iv) Which (if either) of $*$ and nand is commutative up to equivalence?

I have heard a rumour that there are 56 inequivalent 3-ary connectives, each adequate by itself. An example is the if...then...else connective. See Exercise 2.21. More generally, it is rumoured that the number of inequivalent singly adequate n-ary connectives is $2^k(2^k - 1)$, where $k = 2^{n-1} - 1$. So for large n, this is approximately $1/4$ of all the 2^{2^n} possibilities.

What about the question of showing that some set of connectives is **not** adequate? This tends to be a bit harder.

For example, the set $\{\neg, \leftrightarrow\}$ is inadequate. We need to prove that not *all* formulae can be obtained (up to equivalence) by using an inductive definition as in the original definition of "formula", except that $\neg$ and $\leftrightarrow$ are used, instead of $\neg$ and $\wedge$. To see this, define a *halfable-truth* to be any formula which takes both of the two truth values an even number of times, in any truth table which uses more than one propositional variable. Then F *is* a halfable truth, for any F built using only $\{\neg, \leftrightarrow\}$. This follows easily by induction. But $P \to Q$ is not a halfable-truth, for any distinct propositional variables P and Q. Thus no formula equivalent to $P \to Q$ can be formed just using $\neg$ and $\leftrightarrow$, completing the proof.

Another inadequate set is the singleton $\{\wedge\}$. See 4.9 and 4.11 below, which are more difficult than the earlier exercises—adequate arguments for inadequacy are harder than for

adequacy, partly because a clear definition of "connective" is needed—see a few paragraphs below.

Exercises.

4.8 (i) Give the details of the induction two paragraphs above.

(ii) By expressing it in terms of $\{\neg, \leftrightarrow\}$, show that the "exclusive or" connective (sometimes called xor by computists), together with negation, does not give an adequate set of connectives. (That's another reason for preferring the "inclusive or", namely "$\vee$".)

4.9 Show that any binary connective, which is adequate by itself, has the same truth table as either the Schröder or the Sheffer connective. (If worse comes to worst, you can just one-by-one eliminate the other 14, which come up in the answer to 2.21(iii)/4.2(ii). Perhaps prove first that if $F \oslash G$ defines an adequate binary connective, then $F \oslash F$ is equivalent to $\neg F$.)

4.10 Show that no unary connective is adequate by itself.

4.11 Show that the set $\{\wedge, \vee, \rightarrow, \leftrightarrow\}$ is inadequate.

Digression : Exercises for the fussy.

In more advanced books, this stuff is sometimes expressed in terms of functions from '*n-space over the 2-element field* $\{\mathsf{Tr}, \mathsf{Fs}\}$ *to that field.*' The point is that one could regard the nude truth table above, with its top line removed, as the 'n-fold multiplication table' explicitly giving an n-ary multiplication function $\{\mathsf{Tr}, \mathsf{Fs}\}^n \rightarrow \{\mathsf{Tr}, \mathsf{Fs}\}$. But for us that's a bit of notational overkill. If you feel the need for more mathematical rigour, have a look below—otherwise, go on to the next section.

To be mathematically rigorous about connectives, we should have a definition of "connective". It was initially used just as a single word referring to the two symbols, $\neg$ and $\wedge$, in the basic language. But it clearly means something more comprehensive now. A bit later we used it to refer also to the three symbols, $\rightarrow, \vee$ and $\leftrightarrow$, regarded as abbreviations here, but as basic symbols in some treatments. To say that a connective is either a basic or defined symbol of a certain kind is also not mathematically kosher, because what is a connective would then change in time, depending on how many definitions you happen to have made up to that time.

If you asked a computist for a definition of *connective*, it might not be exactly a mathematical definition. Looking hard, you may be able to find a computer science book which gives such a definition.

The simplest mathematical definition seems to be the following.

*An n-***ary connective** *is a function Γ which maps the set of n-tuples of (possibly abbreviated) formulae to the set of (possibly abbreviated) formulae (that is, $\Gamma(F_1, \cdots, F_n)$ is an abbreviated formula where all F_i are as well) which has the following property:*

for any truth assignment e*, and formulae $G_1, \cdots, G_n$ and $F_1, \cdots, F_n$ we have* $\mathsf{e}[\Gamma(G_1, \cdots, G_n)] = \mathsf{e}[\Gamma(F_1, \cdots, F_n)]$ *if* $\mathsf{e}(G_i) = \mathsf{e}(F_i)$ *for all i.*

Another way of saying this is that *if F_i and G_i are truth equivalent for all i, then $\Gamma(F_1, \cdots, F_n)$ is truth equivalent to $\Gamma(G_1, \cdots, G_n)$*

For example, if $\Gamma(F_1, F_2) = \ F_1 \to F_2$, then Γ is the connective called "$\to$" before.

The results of this section can now be formulated and proved a bit more rigorously, though probably less clearly. Some readers might like to go through this and write out proofs of some of the results as exercises (and some might not!).

The earlier, less subtle, sense of the phrase "*The set X of connectives is adequate*" should now evidently be defined to mean the following.

If a set of formulae
(i) contains all the propositional variables; and
(ii) contains $\Gamma(F_1, \cdots, F_n)$ for all $\Gamma \in X$ whenever it contains each F_i;
then every formula is equivalent to at least one formula in that set.

Now it is a one-line proof from the definition of *formula* that the two basic connectives do form an adequate set. For the other adequate sets referred to in this section, the proofs are only a bit longer.

As for the later sense of the phrase *adequate set of connectives* (referring to each possible combination of truth values occurring for some formula), this is just a property of any proper choice for setting up the language of propositional logic in the first place.

Firstly, *no set of more than 2^n truth assignments has the property that no two agree on all of the first n variables.*

Then it says that, *for any sequence of exactly 2^n truth assignments with the property that no two agree on all of the first n variables, and any sequence of values* Tr *and* Fs *of length 2^n, there is a formula involving the first n variables at most, which, for all $i \leq 2^n$, takes the ith value when the ith truth assignment is applied to it.*

The corresponding connective maps $(F_1, \cdots, F_n)$ to the formula obtained by substituting F_i for the ith propositional variable in some formula whose existence is guaranteed by the previous paragraph. That formula isn't unique—for example, $\neg P_1 \vee \neg P_2$ and $\neg(P_1 \wedge P_2)$ give technically different connectives $(F_1, F_2) \mapsto \neg F_1 \vee \neg F_2$ and $(F_1, F_2) \mapsto \neg(F_1 \wedge F_2)$. But then we can talk about *equivalent* connectives.

You might wonder whether every connective can be obtained by this substitution process. Up to equivalence, yes; but in general, no; as the example

$(F_1, F_2) \mapsto \neg F_1 \vee \neg F_2$ if F_1 has length < 1000; and

$(F_1, F_2) \mapsto \neg(F_1 \wedge F_2)$ if F_1 has length > 999;

shows. (That rather silly one would be equivalent to both of those above.) The connectives of that substitution kind are the 'natural' ones: those for which $\Gamma(F_1, \cdots, F_n)$ is obtained by substitution into $\Gamma(P_1, \cdots, P_n)$. The latter formula completely determines the connective, so that natural connectives are in 1–1 correspondence with formulae.

And equivalence classes of connectives are in 1–1 correspondence with equivalence classes of formulae.

It is quite possible to spend many pages describing various species of the genus *connective*. This is done for natural languages in books on remedial thinking. For example,

in English, we have the unary connective sending each assertive sentence S (for example, S="Gods exist") to the sentence "The Pope believes S". This (hopefully) satisfies the definition of connective above. (There is, however, no truth to the rumour that the phrase "diagnostic services" was first used for ceremonies involving those who religiously maintain their uncertainty about the existence of two gods.)

Similar lengthy discussions describing many different connectives are sometimes appropriate in computer science. But this is more like zoological logic than mathematical logic. So let's not get carried away.

4.5 Conjunctive Normal Form

There is a symmetry between "$\wedge$" and "$\vee$" which you may have noticed. It can be made mathematically precise and exploited, but that is more than we want to do here.

If you had a formula in DNF and simply interchanged the two connectives above, you'd get a formula (*almost never* equivalent to the one you started with). However, it's very believable that every formula is truth equivalent to an (essentially unique) formula in this form, a so-called **CNF** formula as in the section title. To give the definitions of *CNF-constituent*, etc. is merely a matter of doing that interchange (of $\wedge$ and $\vee$) in the definitions given in the section on DNF. See the examples just ahead.

Conjunctive form is not being introduced merely so you can see the symmetry between it and disjunctive form. In the final several sections of this chapter, it will play an essential role in studying algorithms for deciding validity of arguments and satisfiability of sets. And *disjunctive* form cannot be used that way, so the symmetry is broken in that respect.

There are three methods for producing the CNF of a given formula:

(I) Write down a truth table for the formula. Look at each line where it's **false**, and produce one constituent using that line. Use P_i as the literal if Fs appears under P_i in that line, and use the literal $\neg P_i$ if instead $\mathsf{e}(P_i) = \mathsf{Tr}$ for the e corresponding to that line. Notice how this is almost exactly the opposite to the procedure for DNF.

(II) If you happen to know the DNF of the formula, first just write down another DNF which contains precisely all the constituents which are *not* in the first one. Now reverse the $\neg$'s on all the literals, and finally interchange $\vee$ and $\wedge$ everywhere. That this works is immediate by comparing the truth table methods for getting DNF's and CNF's. Note that this method also

works to get the DNF from the CNF.

(III) Use manipulations with the same set of basic equivalences from before. This is analogous to the manipulations to get DNF's, but the two connectives $\vee$ and $\wedge$ are treated in reverse fashion for getting CNF's.

Both (III), and the fact that students are often surprised at how quickly they can learn to do these manipulations, need explanation, as follows. In secondary school, you learn to manipulate an algebraic expression involving a bunch of variables and just addition and multiplication. The objective is to get the expression into the form where it's a sum of 'constituents', each of which is a product of some of the variables. Well, the main (middle) section in the manipulations for a DNF—(after the $\neg$'s have been moved 'inside'; that is, the steps getting the formula to a disjunctive form, not necessarily normal)—is exactly the same, with $\vee$ and $\wedge$ playing the roles of $+$ and $\times$, respectively, and the literals playing the role of the variables. So that's why it's easy, if you did high school algebra. And also, what you must do if trying to get a CNF, is to reverse the roles—$\wedge$ and $+$ are now the 'same', and $\vee$ and $\times$ have analogous roles. You can play both games because the distributive law works both ways here—(6) and (7) in our list of basic truth equivalences. But in high school algebra and ring theory in abstract algebra, addition does *not* distribute over multiplication. And, of course, not every high school expression as above can be written as a product of 'constituents', each of which is a sum of individual variables; for example, $x_1x_2 + x_3$ cannot be written that way.

Let's illustrate these methods with our old friend $R \wedge \neg(\neg(P \wedge \neg Q) \wedge P)$.

From the truth table (see the 1st section of this chapter), we have five lines where it is false, and following the recipe in (I), we get the CNF to be

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

The constituents have been written from left to right in the downwards order of occurrence of the truth table lines used to get them.

As for method (II), we first write down the DNF constituents which are not in the DNF of our formula, and they are

$$\neg P \wedge \neg Q \wedge \neg R \ ; \ \neg P \wedge Q \wedge \neg R \ ; \ P \wedge Q \wedge R \ ; \ P \wedge Q \wedge \neg R \ ; \ P \wedge \neg Q \wedge \neg R \ .$$

After reversing negation signs and switching connectives, we get the following as the CNF constituents for our given formula, written in the same order as the things they came from immediately above :

$$P \vee Q \vee R \ ; \ P \vee \neg Q \vee R \ ; \ \neg P \vee \neg Q \vee \neg R \ ; \ \neg P \vee \neg Q \vee R \ ; \ \neg P \vee Q \vee R \ .$$

As must be the case, these *are* the constituents we got by method (I).

Notice that, from either of the above methods, it is clear that, for a given formula, the number of DNF-constituents plus the number of CNF-constituents is always 2^n, where n is the number of variables being used. That's because the total number of possible constituents is 2^n, whether dealing with disjunctive or conjunctive constituents. Or from the truth table method, it's because the number of lines in each table is 2^n. Thus, when using many variables, a short DNF implies a long CNF (and the reverse). For the example we're now working on, this arithmetic amounts to the fact that $3 + 5 = 2^3$.

Finally, the manipulation method (III) also gives these constituents for our example. Starting from $R \wedge (\neg Q \vee \neg P)$, which is one of the lines in the manipulation for the DNF, we take the R half above, and rewrite it as $R \vee (Q \wedge \neg Q) \vee (P \wedge \neg P)$. Expanding this out with two applications of the distributive law (6) and several uses of commutative laws will produce four of the five CNF constituents, all except $\neg P \vee \neg Q \vee \neg R$. Taking the other half, $(\neg Q \vee \neg P)$, and '$\vee$-ing' it into $(R \wedge \neg R)$ will produce two of those constituents, including the one missing above. The one which is produced both times is $\neg P \vee \neg Q \vee R$. The reader should check this calculation.

This chapter's introductory paragraphs give a second example of getting a CNF by the truth table method. A suggestion would be for you to derive both the CNF and the DNF for that example, getting each from the other by the manipulation method. Also note how they do relate to each other correctly with respect to the method of directly getting the CNF/DNF from knowing the DNF/CNF.

For the material a few sections ahead, it will be important to get formulae into a conjunctive form which has as few literals as possible, and so is seldom conjunctive *normal* form. For that, you can either go part way through the manipulations; or possibly write down the CNF using the truth table, and manipulate backwards to a more economical (but less symmetrical) conjunctive form. But three sections ahead, we give a couple of much more efficient algorithms for producing a conjunctive-form formula which is associated to, but not equivalent to, a given formula (and which use lots of literals).

Exercises.

4.12 Use all three methods to find the CNF of the following formulae (some from 4.4). Only use the variables actually occurring in the formula,

not any extra ones, in the CNF.

(i) $R \leftrightarrow P \wedge (Q \rightarrow R)$

(ii) $P \rightarrow (Q \rightarrow (R \rightarrow S))$

(iii) $(P \rightarrow Q) \wedge (P \wedge \neg Q)$

(iv) $\neg((P \rightarrow Q) \wedge (P \wedge \neg Q))$

(v) $P \rightarrow \neg(Q \rightarrow \neg(R \rightarrow S))$

(vi) Do (i) again, but this time get the CNF which uses the variable S, as well as P, Q and R.

(Which of the above is designed to convince you that a better method still needs to be found?)

4.13 Find a list of formulae in conjunctive form which
(a) use no variables other than P, Q and R; and
(b) have the following 'partial' truth table (two values missing); and
(c) are such that any formula for which (a) and (b) hold is truth equivalent to exactly one from your list:

P	Q	R	$F =?$
Tr	Tr	Tr	Fs
Tr	Tr	Fs	?
Tr	Fs	Tr	Tr
Tr	Fs	Fs	Tr
Fs	Tr	Tr	Fs
Fs	Tr	Fs	?
Fs	Fs	Tr	Tr
Fs	Fs	Fs	Fs

Which statement from this section are you using to justify (c)?

4.6 Existence of Algorithms.

We'll not be precise about the definition of *algorithm*, though this can be done (and is important in theoretical computer science and more advanced logic). For us, an algorithm may be described roughly as a finite set of **'mechanical'** instructions

it is the meaning of "mechanical" which needed much careful thought

> before various satisfactory definitions of algorithm were found in the 1930's—for example, your finite set of instructions can't simply be: *ask God for the answer, then write it down*—there are examples of functions f from the set of natural numbers to itself which are NOT computable, i.e. for which no algorithm exists with input n and output $f(n)$—asking God what $f(n)$ is won't do

which takes any example from an explicitly described set of possible inputs, and either

Type D algorithm : after finitely many of these mechanical steps, produces one from an explicitly described set of possible outputs; or

Type L algorithm : runs 'forever' (never stops 'taking steps'), producing a piece of output every so often, effectively generating an infinite sequence of output.

("D" is for "decide", though that's an accurate description only when the set of possible outputs contains only two elements; "L" is for "list". This is a convenient, somewhat naive, dichotomy. See [**CM**] for much detail.)

We must distinguish two questions that can be asked about any given computational problem :

(A) Is there any algorithm at all for solving it?

(B) If the answer to (A) is yes, is there an *efficient (—'feasible'—'fast')* algorithm?

Let's first concentrate on (A). By considering the material of these first four chapters, the answer to (A) can be seen to be **yes**, an algorithm does exist to solve all twelve of the following problems.

(1) (Type D) Input—a finite string, $\mathcal{S}$, of symbols (from the language of propositional logic);
Output—the word "yes", plus a formula verification column for $\mathcal{S}$, if it is a formula;
—the word "no", if it is not a formula.

(2) (Type L) Input—nothing;
Output—a list containing every formula exactly once.

(3) (Type D) Input—a formula;
Output—the truth table of the formula.

(4) (Type D) Input—a pair of formulae, F and G;
Output—the word "yes", if $F \overset{\text{treq}}{} G$;
—the word "no", if not.

(5) (Type D) Input—a formula;
Output—the constituents in the DNF of that formula.

(There is a similar algorithm for the CNF.)

(6) (Type L) Input—nothing;
Output—a list (of formulae) which includes exactly one formula from each equivalence class of formulae—that is, every formula will be equivalent to exactly one formula which appears in the list.

(7) (Type D) Input—a formula F ;
Output—the word "yes", if F is a tautology ;
—the word "no", if not.

(8) (Type D) Input— formulae F and $F_1, F_2, \cdots F_n$;
Output—the word "yes", if $\{F_1, F_2, \cdots F_n\} \models F$, (or $\cdots \vdash F$, by the completeness theorem);
—the word "no", if not.

(9) (Type L) Input— a set of formulae $\{F_1, F_2, \cdots F_n\}$;
Output—a list of all derivations from $\{F_1, F_2, \cdots F_n\}$ (using the proof system of Chapter 3—or any system in which the axioms and rules form decidable sets).

(10) (Type D) Input— formulae F and $F_1, F_2, \cdots F_n$;
Output—a derivation (using the Chapter 3 system) of F with premisses $\{F_1, F_2, \cdots F_n\}$, if $\{F_1, F_2, \cdots F_n\} \vdash F$;
—the word "no", if $\{F_1, F_2, \cdots F_n\} \nvdash F$.

(11) (Type D) Input— a set of formulae $\{F_1, F_2, \cdots F_n\}$;
Output—the word "yes", plus a suitable truth assignment, if $\{F_1, F_2, \cdots F_n\}$ is satisfiable.
—the word "no", if not.

(12) (Type D) Input— a formula F;
Output—all the steps in a 'reduction' of F to its DNF using the list of elementary equivalences.

Is there an algorithm which does the following?

(13) (Type D) Input— a set of connectives.
Output—the word "yes", plus truth equivalences expressing $\wedge$ and $\neg$ in terms of the input, if the input is an adequate set;
—the word "no", if not.

I'll leave (3) and (12) for you to think about how to produce an algorithm. See **Appendix B** to Chapter 8 for (1) and (2).

To do (4), (5), (7), (8) and (11) [but very inefficiently], write out and inspect a suitable truth table. For (8) and (11), getting a much better algorithm than that is important. Most of the rest of this chapter is concerned with that question.

To do (6), successively for each $n > 0$, list all subsets of the set of all DNF (or CNF) constituents from $P_1, \cdots, P_n$, and eliminate those which are equivalent to ones occurring for smaller n.

To do (9), generate a sequence whose terms are finite lists of formulae, giving all such lists once; then check each list to see if it's naughty or nice—no, sorry, to see whether that list *is* a derivation. If yes, spit it out; if no, go on to check the next list in the sequence. The details of the algorithm will depend on which proof system is used. This can be done for any proof system for which there is an algorithm for deciding whether a given formula is an axiom of the system, and also an algorithm for deciding whether a given combination of formulae, premisses plus a conclusion, is indeed immediately inferrable in one step using one of the rules of inference. Our system and all the systems used in other texts certainly have this property. A system which didn't would be of dubious value. (See also two paragraphs down.)

Doing (10) is almost the same as (8), by completeness, but also produces

a derivation. Just run (8), and if the answer is yes, run (9) until you come to the first derivation whose conclusion (last line) is F.

On the opposite side of the coin, there are examples of problems for which it has been proven that *no algorithm can exist*. For such a proof, one certainly needs a precise definition of the word *algorithm*.

Our first example is very simple, but rather abstract, and dependent on knowledge of Cantor's theory of cardinal numbers, so some readers may want to skip this one. Given a fixed set consisting of formulae, the computational problem here is whether there is an algorithm whose input is any single formula, and whose output is "yes" or "no", according to whether the input formula is, or is not, in the given set. The fact is that, *for 'most' sets of formulae, there can be no algorithm.* The reason is that the set of all algorithms, once that concept is given an exact definition, can be seen to be a so-called **countably infinite set**. Therefore the set, consisting of all sets of formulae for which the membership problem is decidable, must also be countable. But the set, consisting of *all* sets of formulae, is a bigger infinity, a so-called **uncountable set**. Therefore, in a certain sense, there are far more choices for our initial fixed set of formulae with undecidable membership problem, than there are choices of that set where there is such an algorithm.

The other two undecidable problems which we'll mention both relate to proof systems for propositional logic. It is simpler to consider only *proof systems à la Hilbert*. This just means that (MP) is the only rule of inference. In **Hodel** and **Burris**. you'll find many examples (though the system actually used in **Burris** is almost the opposite, with many rules of inference). See also the final remarks at the end of the last chapter.

Our system from Chapter 3 is not 'à la Hilbert'. But it is easily converted into one of that form as follows. Remove all the replacement rules of inference, and add the following five new axiom schema, with the same notation as in those replacement rules of inference.

$$XFY \to X\neg\neg FY \quad ; \quad X\neg\neg FY \to XFY$$

$$X(F \wedge G)Y \to X(G \wedge F)Y \quad ;$$

$$X((F \wedge G) \wedge H)Y \to X(F \wedge (G \wedge H))Y \quad ; \quad X(F \wedge (G \wedge H))Y \to X((F \wedge G) \wedge H)Y \quad .$$

The first undecidable problem is the *completeness problem* :

(14) (Type D) Input— a proposed proof system à la Hilbert, with a decidable set of tautologies as axioms;

Output—the word "yes", if the system satisfies the completeness theorem, (and so, can properly be termed a proof system for propositional logic);

—the word "no", if not.

The second undecidable problem is the *independence problem* :

(15) (Type D) Input— a proposed proof system à la Hilbert, with a decidable set of tautologies as axioms, plus one extra tautology;

Output—the word "yes", if the tautology can be derived within the system;

—the word "no", if not.

4.7 Efficiency of Algorithms.

Now we come to question (B) from the beginning of the previous section. It needs to be formulated more precisely. This touches on one of the most important unsolved problems in mathematics, sometimes phrased as : "Is $\mathcal{P} = \mathcal{NP}$?" There is a large area in the theory of computing, called *complexity theory*, which deals with this type of question.

Consider problems (8) and (11) above, which are almost identical, because of **4.3**, stated at the end of Chapter 2 , and treated in detail in the next section. There are several different algorithms for solving these problems, some as yet undiscovered by humans. Using truth tables (the method we had suggested earlier) is extremely slow when the input is large in the sense of involving many propositional variables. After all, each extra such variable doubles the size of the truth table needed.

For any algorithm to do (8) or (11), the input is a set of formulae. Let's say that the *size* of any particular input is

$I =:$ 2+the total number of symbols (with repeats) in all the input formulae

$= \ 2 +$ sum of the lengths of the input formulae .

(The "2+" is there just to make sure you never get 1.)

If you run one such algorithm on a given input, it will use a definite **number of steps; call this number** S, which depends on both the input and which algorithm is used. **The big question is :** *Can you find an algorithm $\mathcal{A}$, and a fixed positive integer N, such that $S < I^N$ for every possible input using this algorithm.* This is so-called *polynomial-time*, and such an algorithm is regarded as being *efficient* (or *fast* or *feasible*).

If you can be first to either
(i) prove that no such $\mathcal{A}$ exists; or
(ii) find such an algorithm $\mathcal{A}$ and integer N, or at least prove that they do exist (perhaps using a proof by contradiction which doesn't actually find $\mathcal{A}$ and N);
then you'll become rather famous, perhaps even wealthy.

It was shown several decades ago that many notorious computational problems are efficiently solvable if and only if the above satisfiability (or va-

lidity) problem is efficiently solvable. These are the so-called *NP-complete* problems. Examples of these notorious problems are the graph isomorphism problem, the travelling salesman problem, $\cdots$. Another famous problem is the integer factorization problem used in encryption. It might be feasibly solvable even if, as expected by many, the ones just above are not. But if it isn't, then neither are the earlier ones. As you can imagine, this subject contains many interesting and subtle theoretical chapters. And yet, at the same time, it is of immediate practical importance. A related (but easier) problem from number theory, namely that of deciding whether a given positive integer is a prime or not, has only very recently (2002) been shown to be solvable in polynomial time.

The next section will be devoted to describing an algorithm for the validity/satisfiability problem, using **resolution**, which is known to be **not efficient in general** (see the penultimate subsection of this chapter for an example). But resolution can be fast for certain special types of input. It has been employed by people attempting to use computers for automated theorem proving.

The section after that explains in more detail the situation about algorithms for satisfiability.

The final section of this chapter exhibits an important class of formulae for which there **is** an efficient algorithm for deciding satisfiability.

4.8 Resolution and the Davis-Putnam Procedure.

To begin, here is the theorem connecting unsatisfiability with validity of an argument. For a finite set Γ, it was stated in the penultimate section of Chapter 2, where the notion of satisfiability was introduced. Exercises 2.23 and 2.24 give thirteen examples of this. Furthermore, the proof of one direction in this already occurred in **3.16**, but we'll do it again, since it's so fundamental.

Theorem 4.3 *If F is a formula and Γ is any set of formulae, then $\Gamma \models F$ if and only if $\Gamma \cup \{\neg F\}$ is unsatisfiable.*

Proof. Suppose that $\Gamma \cup \{\neg F\}$ is unsatisfiable, and that e is any truth assignment for which $\mathsf{e}(G) = \mathsf{Tr}$ for all $G \in \Gamma$. Then $\mathsf{e}(\neg F) = \mathsf{Tr}$ is impossible, by unsatisfiability. Thus $\mathsf{e}(F) = \mathsf{Tr}$. This shows $\Gamma \models F$, giving half the proof.

Conversely, suppose that $\Gamma \models F$. Then, if there are any e for which $\mathsf{e}(G) = \mathsf{Tr}$ for all $G \in \Gamma$, we have $\mathsf{e}(F) = \mathsf{Tr}$, so $\mathsf{e}(\neg F) = \mathsf{Fs}$. Thus there are certainly no truth assignments at which all formulae in $\Gamma \cup \{\neg F\}$ are true, proving unsatisfiability, as required.

A more brutally succinct write-up of this proof, which you may come to prefer, is as follows. We use the upsidedown A to mean "for all", and the backwards E to mean "there exists".

$$\Gamma \models F \iff \forall \mathsf{e}\,[\mathsf{e}(G) = \mathsf{Tr}\ \forall G \in \Gamma \implies \mathsf{e}(F) = \mathsf{Tr}]$$

$$\iff \forall \mathsf{e}\,[\mathsf{e}(G) = \mathsf{Tr}\ \forall G \in \Gamma \ \Rightarrow\ \mathsf{e}(\neg F) \neq \mathsf{Tr}]$$

$$\iff \nexists \mathsf{e} \text{ with } \mathsf{e}(\neg F) = \mathsf{Tr} \text{ and } \mathsf{e}(G) = \mathsf{Tr}\ \forall G \in \Gamma$$

$$\iff \Gamma \cup \{\neg F\} \text{ is unsatisfiable .}$$

The symbols $\forall$ and $\exists$ will occur as part of the formal languages (1^{st}order) in the next four chapters, so we won't use them very much in our math/English metalanguage, and not at all after this chapter, in order to avoid confusion between the two usages.

Actually, when converted to its contrapositive form, "$\Gamma \not\models F$ *if and only if* $\Gamma \cup \{\neg F\}$ *is satisfiable*", the proof of this important result is even easier to formulate:

$$\Gamma \not\models F \iff \exists \mathsf{e} \text{ with } \mathsf{e}(F) = \mathsf{Fs} \text{ and } \mathsf{e}(G) = \mathsf{Tr}\ \forall G \in \Gamma$$

$$\iff \exists \mathsf{e} \text{ with } \mathsf{e}(\neg F) = \mathsf{Tr} \text{ and } \mathsf{e}(G) = \mathsf{Tr}\ \forall G \in \Gamma$$

$$\iff \Gamma \cup \{\neg F\} \text{ is satisfiable .}$$

To re-iterate for a finite premiss set: the process, of converting a question about the validity of an argument $[F_1, \cdots, F_n\ ,\ \text{therefore}\ F]$ into a question about unsatisfiability, is to negate the conclusion and throw it in with the premisses—then check whether this set, $\{F_1, \cdots, F_n, \neg F\}$, is UNsatisfiable.

To get started on the new algorithm for deciding satisfiability, here's an example of a set of three formulae in conjunctive (non-normal) form:

$$\{\ (P \vee \neg Q) \wedge (\neg P \vee R)\ ,\ (\neg P \vee Q \vee \neg R) \wedge (P \vee \neg R)\ ,\ \neg Q \vee R\ \}\ .$$

Clearly this set is satisfiable if and only if we can find a truth assignment e at which all of the following hold :

$$\left\{\begin{array}{l}\text{at least one of } P \text{ or } \neg Q \text{ is true,}\\ \text{at least one of } \neg P \text{ or } R \text{ is true,}\\ \text{at least one of } \neg P \text{ or } Q \text{ or } \neg R \text{ is true}\\ \text{at least one of } P \text{ or } \neg R \text{ is true,}\\ \text{at least one of } \neg Q \text{ or } R \text{ is true,}\end{array}\right\}, \text{because} \left\{\begin{array}{l} P \vee \neg Q\\ \neg P \vee R\\ \neg P \vee Q \vee \neg R\\ P \vee \neg R\\ \neg Q \vee R\end{array}\right\} \begin{array}{l}\text{must all be true}\\ \text{(at some } \mathsf{e}\text{).}\end{array}$$

Actually, this particular example happens to be satisfiable, as we can see by a **not** *generally recommended* trial-and-error method : Try P true to make the 1st line work; that forces R to be true for the 2nd line, and Q to be true for the 3rd line, and by pure good luck, the 4th and 5th lines also work with these choices.

Before going on to describe the recommended **resolution** method, we can illustrate the theorem using this example. Notice that $\neg Q \vee R$ is equivalent to the negation of $Q \wedge \neg R$. So the satisfiability of the above set establishes that the argument

$$[\ (P \vee \neg Q) \wedge (\neg P \vee R)\ ,\ (\neg P \vee Q \vee \neg R) \wedge (P \vee \neg R)\ \ ,\quad \text{therefore } Q \wedge \neg R\ ]$$

is **not** valid. In other words

$$\{(P \vee \neg Q) \wedge (\neg P \vee R)\ ,\ (\neg P \vee Q \vee \neg R) \wedge (P \vee \neg R)\ \} \not\models Q \wedge \neg R\ .$$

How would we use resolution instead, to show that this example is satisfiable? The first step is to get rid of some notation, and just write down the five displayed lines above as sets called **clauses** :

$$\{P, \neg Q\}\ \ ;\ \ \{\neg P, R\}\ \ ;\ \ \{\neg P, Q, \neg R\}\ \ ;\ \ \{P, \neg R\}\ \ ;\ \ \{\neg Q, R\}\ .$$

Recall that each literal is either a propositional variable or the negation of one.

Definition. A *clause* is a finite set none of whose members is not a literal.

This doubly negative way of phrasing it is rather odd. But it serves to remind you that the empty set *is* a clause. And that empty clause will certainly come up in what follows. Really, the definition is just this:

A *clause* is a finite set whose members are literals.

So the display just above is a set of clauses. (Thus, it's a set whose members are themselves sets.) We should really have a large set of squiggly brackets around the whole display, but usually that will be omitted, as will the cancer of semi-colons between the clauses.

Definition. A set of clauses is *satisfiable* if and only if there is at least one truth assignment at which every clause in the set contains at least one literal which is true. Symbolically,

$$\mathcal{C} \text{ is satisfiable} \iff \exists \mathsf{e},\ \forall C \in \mathcal{C},\ \exists L \in C \text{ with } \mathsf{e}(L) = \mathsf{Tr}\ .$$

Thus we have two uses of the adjective "satisfiable". This last definition was chosen so that a set of formulae is satisfiable in the earlier sense exactly when a set of clauses, obtained by putting those formulae in conjunctive form, is satisfiable in the sense just defined.

Now let's get down to the details of the **Davis-Putnam procedure** (called **DPP** from now on). The main step in that procedure is called **resolution**. Here is what you do :

RESOLUTION on S:

> Take a pair of clauses, one containing S but not containing $\neg S$, denoted as, say, $\{S\} \cup \mathcal{C}$, and the other containing $\neg S$ but not containing S, say $\{\neg S\} \cup \mathcal{D}$. Produce the new clause $\mathcal{C} \cup \mathcal{D}$. Do this for all pairs as above for the chosen variable S. Keep all the new clauses. Throw away all the old clauses involving S or $\neg S$.

That is, the new clause is the *'union of the two old clauses with S and $\neg S$ deleted'*. Sometimes logicians write this like a rule of inference :

$$\frac{\{S\} \cup \mathcal{C}\ ,\ \{\neg S\} \cup \mathcal{D}}{\mathcal{C} \cup \mathcal{D}}\ .$$

We are definitely assuming that neither $\mathcal{C}$ nor $\mathcal{D}$ contains S or $\neg S$.

Starting from a set of clauses, you would resolve on each propositional variable S. Resolving on a propositional variable S means choosing *all* pairs of clauses as above for the fixed S and replacing each such pair by a single

clause. You can resolve (i.e. choose the S) in any order you wish, and you do 'clean-up' steps each time as well (explained below). So actually carrying this out, or programming a computer to do so, is not a major intellectual feat.

Note that each resolution step will get rid of one propositional variable, the variable S involved. (Because of the clean-up steps, there won't be any clauses containing *both* S and $\neg S$.) So the process will come to an end after a number of resolution steps not greater than the number of variables you started with.

Now although the number of variables decreases each time, the number of clauses may very well increase considerably after some resolution steps. For example, if there were originally 6 clauses of the form $\{S\} \cup \mathcal{C}$ as above, and 8 clauses of the form $\{\neg S\} \cup \mathcal{D}$, then there would be $6 \times 8 = 48$ ways of pairing them, so we would be deleting 14 (=6+8) clauses, but introducing possibly 48 new clauses (possibly less, because some of the new clauses may coincide with each other or with other clauses in the set.) But the new set of clauses would have one less variable (namely S) involved.

It is important to understand why a resolution step as above preserves satisfiability—that is, the new set is satisfiable if and only if the old set is satisfiable. (See **Theorem 4.4** below.)

But first, let me explain the clean-up steps, so that we can do the example. It is easy to see that each of the following processes will preserve satisfiability.

Clean-up steps.

(1) When a literal appears more than once in a clause, throw away the extra copies—this is just a notational cleanup, since we are dealing with sets, not multisets.

(2) When a clause appears more than once in our set of clauses, throw away the extra copies.

(3) Delete any clause which contains both T and $\neg T$ for any variable T. (This means to remove the clause entirely from the set of clauses, not just to remove those two literals or to make the clause empty.) This preserves satisfiability since any truth assignment whatsoever is true at one those two literals above.

(4) For any variable T, if no clause contains T, delete all the clauses which contain $\neg T$. The point is that we could adjust, if necessary, any truth assignment so as to make T false; that will make all those deleted clauses 'be satisfied', and won't affect the remaining clauses.

(5) Same as (4), except interchange T and $\neg T$.

Actually, (4) and (5) could be regarded also as special cases of a resolution step. If you resolve on the variable T as in (4) or (5), there are no **pairs** of clauses

$$(\{T\} \cup \mathcal{C} \ , \ \{\neg T\} \cup \mathcal{D}) \ ,$$

so nothing new is produced. But all the clauses $\{\neg T\} \cup \mathcal{D}$ in (4), or $\{T\} \cup \mathcal{C}$ in (5), are erased in that resolution step.

Now let's do our previous example,

$$\{P, \neg Q\} \ ; \ \{\neg P, R\} \ ; \ \{\neg P, Q, \neg R\} \ ; \ \{P, \neg R\} \ ; \ \{\neg Q, R\} \ ,$$

using the DPP :

After resolving on Q :

$$\{P, \neg P, \neg R\} \ ; \ \{\neg P, \neg R, R\} \ ; \ \{\neg P, R\} \ ; \ \{P, \neg R\} \ .$$

After clean-up :

$$\{\neg P, R\} \ ; \ \{P, \neg R\} \ .$$

After resolving on R :

$$\{\neg P, P\} \ .$$

At this point, it is trivial to see that we have a satisfiable set of clauses (containing only one clause). And so, because both resolution and clean-up steps preserve satisfiability, the original set of clauses has been shown to be satisfiable. This conclusion agrees, as it must, with our ad hoc argument earlier showing satisfiability.

The point is that, if a large set involving many propositional variables had been presented, an ad hoc method would often be very tedious to find, whereas the DPP is more systematic, and often much faster. (But DPP is definitely not a polynomial-time algorithm, as was admitted at the end of the last section.)

We could go all the way with the previous example and do a last clean-up step, deleting that one clause. That would produce the empty set of clauses (no clauses at all). Reading the definition of *satisfiable*, we see that the empty set of clauses is satisfiable (since something must be true for every clause in it; but there are no clauses in it—so no problem!) This agrees with our earlier answer in the example.

Note that the only things you can end up with after doing all possible clean-up and resolution steps (leaving no propositional variables at all) is either the empty set of clauses, or a set containing one clause, namely the empty clause with no literals in it. Again, reading the definition, this last set, consisting of a single (empty) clause, is unsatisfiable, because to be satisfiable, in every clause you'd have to be able to find at least one literal for which a certain thing is true, but you cannot find any literal in our one (empty) clause.

To summarize:

[DPP yields **empty set of clauses**] $\Longrightarrow$ [original set of clauses is **satisfiable**] (and if it came from checking an argument, that argument is **NOT valid**).

[DPP yields **set of clauses containing only one clause, namely the empty clause**] $\Longrightarrow$ [original set of clauses is **UNsatisfiable**] (and if it came from checking an argument, that argument is **valid**).

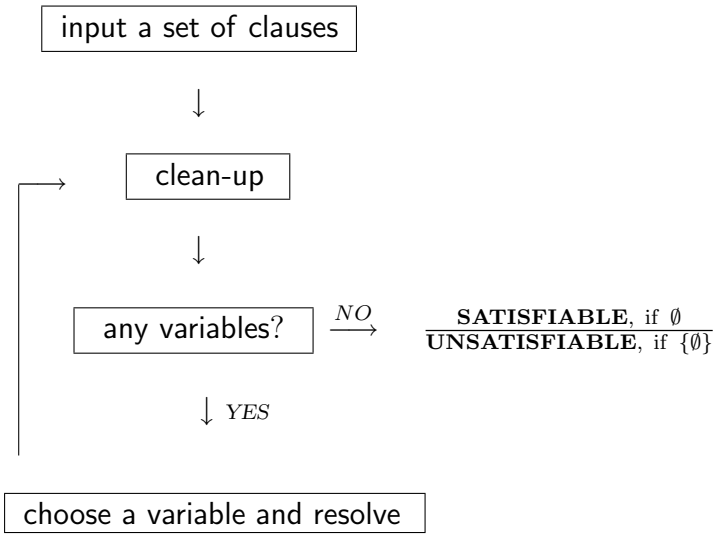
Rather than straining our brains trying to remember or figure out which is which for the zero variable case, most of us find it easier to go down just to one variable, at which point it's always easy to decide SAT or UNSAT—just think about the definition of *satisfiable*. If that one remaining variable is Q, then you only get UNSAT if the set of clauses contains both $\{Q\}$ and $\{\neg Q\}$, or if at some earlier stage we had produced the empty clause (but see the next paragraph).

When doing the DPP, it is sometimes easy to see, several steps before getting down to one or zero variables, whether you have satisfiability or not. In particular, if an empty clause appears part ways through the calculation, there's no way of deleting it, so you're in the unsatisfiable case. And that follows from an argument above anyway—a set of clauses which contains the empty clause is unsatisfiable. In fact, the only way to produce the empty clause is by resolution on some S, applied to a set of clauses which contains both singleton clauses $\{\neg S\}$ and $\{S\}$. So you can stop when you see that—and again it's obvious from the definition that a set of clauses containing those two singletons is unsatisfiable.

In which order should the variables be resolved?

In the above example we resolved in the order: Q, then R, then P (though the latter step was unneeded in that example). The reader should try all the other five possibilities for the order of resolution, and see how satisfying it is to get 'satisfiable' as the conclusion each time. (That had better be the conclusion, or something would be terribly wrong with the procedure!) There is no simple set of rules for deciding which variable to choose at any stage. One good choice is often a variable which appears on its own in a singleton clause, or one whose negation appears that way.

Here's a flowchart summary of DPP as a whole:



Be careful in reading the *empty set notations* in the above chart. The $\emptyset$ in the SAT case is referring to 'the empty set of clauses' , NOT the set of clauses containing only the empty clause. That's what the $\{\emptyset\}$ refers to. It should be pointed out that there is only one empty set in mathematics—there is no difference between 'the empty set of clauses' (the upper $\emptyset$), and 'the empty set of literals' (the lower $\emptyset$) i.e. the empty clause , or any 'other' empty set, like the empty set of numbers! There could be confusion here because, in examples we get sloppy and don't use a big pair of braces around our entire list of clauses to indicate that it's a *set* of clauses being considered. But the braces just above around the $\emptyset$ do play that role. As mentioned earlier, it seems to be easier for most of us to go down to one variable, rather than zero variables, when doing DPP by hand.

Before doing another example all the way from the argument stage to the "unsatisfiable, therefore valid argument" conclusion, let's return to deal with the important question of why a resolution step preserves satisfiability (and therefore, the DPP as a whole always gives the correct answer—in many books this is referred to as a completeness theorem for DPP).

Theorem 4.4 *Resolution preserves satisfiability. That is: the old set of clauses, before a resolution step, is satisfiable if and only if the new set, after resolving, is satisfiable.*

Proof. To see this one way, *assume that* e *shows that the old set of clauses* [just before the (resolution on S)–step] *is satisfiable*. Then I claim that the same truth assignment shows the new set of clauses also to be satisfiable. For if $\mathsf{e}(S) = \mathsf{Tr}$, then each clause $\mathcal{D}$, where $\mathcal{D} \cup \{\neg S\}$ is in the old set, must contain a literal true at e, and so $\mathcal{C} \cup \mathcal{D}$ also does; and, on the other hand, if $\mathsf{e}(S) = \mathsf{Fs}$, then each clause $\mathcal{C}$, where $\mathcal{C} \cup \{S\}$ is in the old set, must contain a literal true at e, and so, again, $\mathcal{C} \cup \mathcal{D}$ also does.

To see that *the new set being satisfiable implies that the old set of clauses is*, suppose that e does the trick for the new set. If even one of the clauses $\mathcal{C}$ as above has no literal true at e, then *every* clause $\mathcal{D}$ must have a literal true at e, since all the sets $\mathcal{C} \cup \mathcal{D}$ do. Thus, in this case, if we modify (if necessary) e making it true on S (and that won't alter its effect on $\mathcal{C} \cup \mathcal{D}$, nor on any clause not containing S or $\neg S$), we now have a truth assignment which is true at some literal in every clause of the initial set of clauses. On the other hand, if every clause $\mathcal{C}$ has a literal true at e, then modify (if necessary) e making it false on S, and that will do the trick for the initial set of clauses.

This completes the proof.

Now here is that promised example. (Another, larger example of producing a clause set, and applying DPP to show an argument to be valid, is given at the very end of this chapter.) Consider the argument:

$$(R \to S) \to R \ , \ \neg P \vee Q \ , \ \neg(Q \wedge R) \ , \ (R \to P) \wedge S \ ; \ \text{ therefore } \neg(Q \to R) \, .$$

To prove validity (which turns out to be the case, though that doesn't appear obvious), we need only, by **4.3**, show that the set of formulae

$$\{ \ (R \to S) \to R \ , \ \neg P \vee Q \ , \ \neg(Q \wedge R) \ , \ (R \to P) \wedge S \ , \ Q \to R \ \}$$

is unsatisfiable. (We cancelled a double negation on the last formula.) Getting a conjunctive form for each formula, we reduce to showing that the following set of clauses is unsatisfiable.

$$\{R\} \ , \ \{\neg S, R\} \ , \ \{\neg P, Q\} \ , \ \{\neg Q, \neg R\} \ , \ \{\neg R, P\} \ , \ \{S\} \ , \ \{\neg Q, R\} \ .$$

Resolving on S yields

$$\{R\} \ , \ \{\neg P, Q\} \ , \ \{\neg Q, \neg R\} \ , \ \{\neg R, P\} \ , \ \{\neg Q, R\} \ .$$

Resolving on P and cleaning yields

$$\{Q, \neg R\} \ , \ \{\neg Q, \neg R\} \ , \ \{R\} \ , \ \{\neg Q, R\} \ .$$

Resolving on R now yields at least the two clauses

$$\{Q\}\ ,\ \{\neg Q\}\ ,\quad \cdots\ ,$$

so we do have unsatisfiability, and the argument is valid, as claimed.

Actually, going back to the original set of clauses, it can be quickly seen, in that order, that the following variables would have to be true in any satisfying truth assignment : S, then R, then P, then Q. But now the clause $\{\neg Q, \neg R\}$ makes this impossible. So that's a quicker check of unsatisfiability, but only because of the small size of the example.

Exercises

4.14 Use the DPP to check the following sets of clauses for satisfiability—and where the answer is positive, find a truth assignment satisfying the set. Experience shows that it is best to separate the resolution steps from the clean-up steps. Trying to write them both down simultaneously can lead to errors.

(i)

$$\{\neg Q, T\} \quad \{P, \neg Q\} \quad \{\neg Q, \neg S\}$$
$$\{\neg P, \neg R\} \quad \{P, \neg R, S\} \quad \{Q, S, \neg T\}$$
$$\{\neg P, S, \neg T\} \quad \{Q, \neg S\} \quad \{Q, R, T\}$$

(ii)

$$\{\neg Q, R, T\} \quad \{\neg P, \neg R\} \quad \{\neg P, S, \neg T\}$$
$$\{P, \neg Q\} \quad \{P, \neg R, S\} \quad \{Q, S, \neg T\}$$
$$\{\neg Q, \neg S\} \quad \{\neg Q, \neg S\} \quad \{\neg Q, T\}$$

(Exercise 4.25 gives 3 more examples, if you want more practice.)

4.15 Check the following arguments for validity by reducing to clauses and using resolution. For all but (vi), give also a more direct reason, confirming your answer. These five are all pretty obvious, and perhaps reinforce your 'belief' in the method, if the proofs given didn't convince you; they should have!

(i) $P \to Q$, $Q \to R$, $R \to S$, $S \to T$; therefore $P \to T$.

(ii) $P \vee Q$, $Q \vee R$, $R \vee S$, $S \vee T$; therefore $P \vee T$.

(iii) $P \to Q$, $Q \to R$, $\neg R$; therefore $\neg P$.

(iv) $P \to Q$, therefore $P \wedge T \to Q \wedge T$.

(v) $P \wedge Q$, $Q \wedge R$, $R \wedge S$, $S \wedge T$; therefore $P \wedge T$.

(vi) $Q \leftrightarrow R \wedge (S \leftrightarrow Q)$, $Q \vee (S \wedge (R \leftrightarrow T))$, $(T \to \neg R) \leftrightarrow S$,
$S \leftrightarrow (S \wedge R) \vee (T \wedge Q)$; therefore $Q \wedge T \wedge (S \leftrightarrow Q)$.

4.9 (A hopefully satisfactory discussion of) SATISFIABILITY, (including the computational problem 3-SAT).

The word "3-SAT" in this heading is the name of a computational problem which is prominent in complexity theory. It is a specialized form of the satisfiability problem—the form in which *every clause contains exactly three literals*. Here, one of our minor objectives will be to show that this problem is no easier than the general satisfiability problem. So either there is an efficient algorithm for *both*, or (what many think to be more likely) for *neither* ; at present (2006), no one knows which. This equivalence of problems is demonstrated by giving an efficient method for converting

any instance of the SAT problem
into
an instance of the 3-SAT problem which has the same answer.

The box below illustrates a way of going from arbitrary finite clauses to clauses with at most 3 literals each, for which it is very easy to see that the old and new sets of clauses are either both or neither satisfiable. To keep it readable, on the left in the box only two of the old clauses, and on the right the corresponding three plus four new clauses, have been displayed. The pattern is easy to discern.

We take new propositional variables: X_3 and X_4; and then Y_3, Y_4 and Y_5; and then other new variables corresponding to other clauses in the old set of clauses. In each case, take three fewer new variables than the size of the clause. (Just copy, without alteration, any old clauses which have fewer than four literals.) When we say *new* propositional variables, we mean that

all these are distinct from each other and from all the variables involved in the old (given) set of clauses. The literals L_i and M_j, etc. may have many overlaps with each other.

OLD	NEW
$\{L_1, L_2, L_3, L_4, L_5\}$	$\{L_1, L_2, \neg X_3\}\ \{X_3, L_3, \neg X_4\}\ \{X_4, L_4, L_5\}$
$\{M_1, M_2, M_3, M_4, M_5, M_6\}$	$\{M_1, M_2, \neg Y_3\}\{Y_3, M_3, \neg Y_4\}\{Y_4, M_4, \neg Y_5\}\{Y_5, M_5, M_6\}$
$\bullet$ $\bullet$ $\bullet$	$\bullet$ $\bullet$ $\bullet$

Now it is easy but tedious to argue *directly* both ways that the old set is satisfiable if and only if the new set is.

But having done the theory of resolution, there is a *very short proof* of this. Just take the new set, and resolve on all the new variables, in any order you please. This will clearly produce the old set. So by Theorem **4.4** near the end of the last section, either both sets are satisfiable, or neither set is.

Ex. 4.21 The 3-SAT problem usually means deciding satisfiability for a finite set of clauses containing *exactly* three literals each. To change "at most three" (which the above process gives) to "exactly three" is quite easy. You just replace $\{L\}$ by four clauses

$$\{L, X, Y\}\ \ \{L, X, \neg Y\}\ \ \{L, \neg X, Y\}\ \ \{L, \neg X, \neg Y\}\ .$$

And replace $\{L, M\}$ by two clauses $\{L, M, Z\}\ \ \{L, M, \neg Z\}$. Here X, Y and Z are 'brand-new' variables. Prove the 'satisfiability-equivalence' of the new with the old when one does either one of these 1-step processes.

(If you have several clauses with only two or one literals, you of course do such a process for each of them, making sure the new variables are all different from each other, as well as from all the variables in the set of clauses you started with.)

Finally here, it is important to note that the rather trivial processes above are *themselves* efficient (not something we can rigorously prove here, since we haven't defined efficient in strict mathematical terms in this book). But, given that, it is clear that any supposed efficient method for deciding 3-SAT

combined with the methods above would give an efficient method for deciding SAT in general, as required.

Next we want to talk about one or two efficient ways of getting from formulae to clauses, because the standard ways of producing those conjunctive forms which are truth equivalent to a given formula are not efficient. First here are a couple of observations.

Clearly, when dealing with the satisfiability of a *finite* set of formulae, we can just talk about the single formula which is their conjunction. All that's being said here is that $\{H_1, \cdots, H_k\}$ is an unsatisfiable set of formulae if and only if $H_1 \wedge \cdots \wedge H_k$ is an unsatisfiable formula. For a single formula, being unsatisfiable means the same as being a contradiction. We shall therefore stick to *single* formulas when talking about unsatisfiability of formulae (as opposed to *sets* of clauses).

More interestingly, you might have been wondering why we didn't use *disjunctive* form, instead of conjunctive form, when we started to seriously discuss whether there is an efficient method for satisfiability. Deciding satisfiability is totally trivial for a disjunctive-form formula, since every such formula, except the "empty DNF", is satisfiable. (Just pick any constituent, and set all its literals true.) So why have we been desperately seeking a way to efficiently decide satisfiability for a formula in *conjunctive* form?

You may also be wondering whether, by "the SAT problem", we mean "the efficient satisfiability" of an arbitrary formula, or of an arbitrary set of clauses (essentially, of a formula already in conjunctive form).

The explanation for both is that, although there is no known **efficient** *method to produce disjunctive form, there are such algorithms related to conjunctive form.*

The last statement needs to be qualified. The 'bad' news (not really bad) is that below we don't produce a conjunctive form which is truth equivalent to the given formula. But we produce a formula in conjunctive form which is satisfiable if and only if the given formula is. In fact, the new formula usually uses many more propositional variables than the old, and the two are seldom even close to being truth equivalent.

The good news is that the process for producing the new formula is *very efficient*.

The method discussed just below is justified by Theorem **4.5**, which we state

and prove after giving the method and illustrating it.

The process (which turns out to be linear-time) is for converting an arbitrary formula into a "satisfiable-equivalent" conjunctive form. And this conjunctive form necessarily produces clauses with three or fewer literals, so the method at the start of this section for getting down to such small clauses is redundant, when the method below is used for getting from a formula to a set of clauses.

Here is the method, starting from some formula :

First of all, erase any double negations in the given (abbreviated) formula. It is clearly sufficient to work with the resulting simplified formula.

Next write down a verification column for that resulting formula, in which the initial lines are $Q_1, \cdots, Q_n$, the list of all propositional variables occurring. Now extract from the rest of that column just those steps which are justified as as one of the four *binary* connectives applied to earlier things in the column. That is, don't include the initial terms which are propositional variables, and don't include the lower steps which are justified as negation (the *unary* connective) applied to an earlier thing in the column.

Denote what you've just written down as $H_1, \cdots, H_k$, each of which is a formula of length greater than 1 in (some of) the variables $Q_1, \cdots, Q_n$.

Now make a list of "k" new variables $X_1, \cdots, X_k$. (X_i will play the role of a sort of **alias** for H_i.)

(You might prefer to follow the example below this question before spending too much time puzzling over the general recipe below. This sort of thing is easier done than said!)

Now write down the formula we want as

$$J_1 \wedge \cdots \wedge J_k \wedge X_k \quad \text{or} \quad J_1 \wedge \cdots \wedge J_k \wedge \neg X_k \quad ;$$

where J_i is a conjunction of four or fewer disjunctions of three or fewer literals, as defined below; and *where the negation is used on X_k only in the case where the whole formula being dealt with is a negation.* That is, use the right-hand form in the display above when the initial formula is $\neg H_k$, not H_k itself, so that H_k is the *second* last entry of the above verification column, not the last one.

(Note that, besides being very efficient, getting the J_i as below will require no manipulation of the original formula. Furthermore, if you inspect the 2nd section of Appendix B to Chapter 8, you will agree that getting the verification column presents no problem for the efficiency here. Thus the

overall method here does in fact give a quick way to essentially get down to the 3-SAT situation, as claimed.)

To define the J_i, first write down $G_i \leftrightarrow X_i$, where $G_1 = H_1$, and inductively :

if H_i is justified as $H_a * H_b$, where a and b are both less than i, we take G_i to be $X_a * X_b$. If the justification for H_i uses the binary connective $*$, but one or both of the ingredients is just a propositional variable Q_p, just use that variable instead of an X_p ;

if H_i is justified as $\neg H_a * H_b$, just take G_i to be $\neg X_a * X_b$, again where Q's are used instead of X's sometimes ;

and similarly for the cases $H_a * \neg H_b$ and $\neg H_a * \neg H_b$.

Note now that G_i involves at most two variables, so $G_i \leftrightarrow X_i$ involves at most three. Finally,

obtain J_i from the equivalent formula $(G_i \rightarrow X_i) \wedge (X_i \rightarrow G_i)$ by manipulations which express it as a conjunction of a very small number of disjunctions of literals in X_i and the two variables (or one) used in G_i. The results of these truth equivalence manipulations are given explicitly in the proof of **4.5** ahead, as four consecutive lines labelled as $\underline{* \ = \ \wedge}$, etc. But it is a waste of time to memorize the results—the manipulations are extremely simple compared to some that you have been doing earlier in the chapter.

The theorem, **4.5**, after the examples tells us two things:

By (i) in the theorem, the method above constructs a formula which is satisfiable if and only if the original formula is satisfiable.

Using (iii) in the theorem, the method above can be shown to be a linear time algorithm, once **efficiency, linear-time, quadratic-time***, etc.. have been given rigorous mathematical definitions.*

Example of the above process. If the given formula is

$$((P \rightarrow Q) \wedge P) \wedge \neg Q \ ,$$

we'll take the formula verification column (written sideways!)

$$P \quad , \quad Q \quad , \quad P \rightarrow Q \quad , \quad (P \rightarrow Q) \wedge P \quad , \quad \neg Q \quad , \quad ((P \rightarrow Q) \wedge P) \wedge \neg Q \ .$$

Now just write down the formula

$$((P \rightarrow Q) \leftrightarrow X_1) \ \wedge \ ((X_1 \wedge P) \leftrightarrow X_2) \ \wedge \ ((X_2 \wedge \neg Q) \leftrightarrow X_3) \ \wedge \ X_3 \ .$$

This last formula is definitely unsatisfiable

Ex. Prove this!

as it better be, since it's truth equivalent to our final answer below, and since our original formula is unsatisfiable.

Converting the last formula into a conjunctive form is a minor triviality now. First

$$((\neg P \vee Q) \to X_1) \wedge (X_1 \to (\neg P \vee Q)) \wedge ((X_1 \wedge P) \to X_2) \wedge (X_2 \to (X_1 \wedge P)) \wedge$$
$$((X_2 \wedge \neg Q) \to X_3) \ \wedge \ (X_3 \to (X_2 \wedge \neg Q)) \ \wedge \ X_3 \ .$$

Then

$$(\neg(\neg P \vee Q) \vee X_1) \wedge (\neg X_1 \vee (\neg P \vee Q)) \wedge (\neg(X_1 \wedge P) \vee X_2) \wedge (\neg X_2 \vee (X_1 \wedge P)) \wedge$$
$$(\neg(X_2 \wedge \neg Q) \vee X_3) \ \wedge \ (\neg X_3 \vee (X_2 \wedge \neg Q)) \ \wedge \ X_3 \ .$$

Finally

$$(P \vee X_1) \wedge (\neg Q \vee X_1) \wedge (\neg X_1 \vee \neg P \vee Q) \wedge (\neg X_1 \vee \neg P \vee X_2) \wedge (\neg X_2 \vee X_1) \wedge (\neg X_2 \vee P) \wedge$$
$$(\neg X_2 \vee Q \vee X_3) \ \wedge \ (\neg X_3 \vee X_2) \wedge (\neg X_3 \vee \neg Q) \ \wedge \ X_3 \ .$$

This is now a conjunctive form whose constituents are disjunctions of 1, 2 or 3 literals, but never more. It is indeed unsatisfiable, as it must be, if the general process is correct. The labour involved to produce it is not that large, but as before, a small example is inadequate to illustrate either the efficiency or really the 'satisfiability-equivalence'.

Note that if we had started with the negation of the formula which was illustrated, the answer would have been identical except for using $\neg X_3$, not X_3 itself, on the far right on each displayed line above.

It is important to realize that you cannot get a sense of the efficiency of this method by looking at examples that are done by hand. Much larger examples are needed for this. The above, and the examples in 4.22 below are undoubtedly quicker with the old methods of just getting partways to the conjunctive normal form. Some of them are even already in conjunctive form!

4.22 (a) Use the method just above to find conjunctive-form formulae which are satisfiability-equivalent to the following formulae from 4.4.

(b) Then write down the corresponding set of clauses.

(c) Then write down a satisfiability equivalent set of clauses from the 3-SAT problem.

(i) $R \leftrightarrow P \wedge (Q \rightarrow R)$

(ii) $P \rightarrow (Q \rightarrow (R \rightarrow S))$

(iii) $(P \rightarrow Q) \wedge (P \wedge \neg Q)$

(iv) $P \rightarrow \neg(Q \rightarrow \neg(R \rightarrow S))$

(v) $(P_1 \wedge P_2 \wedge P_3) \vee (Q_1 \wedge Q_2 \wedge Q_3)$.

(vi) $(P_1 \wedge P_2) \vee (Q_1 \wedge Q_2) \vee (R_1 \wedge R_2)$.

4.23 Are you now convinced that a better method no longer needs to be found, for getting from 'formula logic' down to 'clause logic'?

Now we'll state and prove the theorem which justifies the method above. The efficiency question is phrased in terms of formula length : the new, conjunctive form, formula length is at most a constant times the old formula length. This is the main fact needed in complexity theory to establish a linear-time algorithm. We cannot discuss details of algorithm efficiency with any precision, because the basic definitions from the theory of algorithms would take us too far from logic at the level treated in this book.

Some readers may prefer to skip this theorem, possibly altogether, or at least until they have read the rest of the section. Others may wish to just learn the theorem and its proof, and ignore the remainder of this section.

Theorem 4.5 *Given a formula F, probably in abbreviated form, there is another formula H with the following properties:*

(i) *F is satisfiable $\iff$ H is satisfiable ;*

(ii) *H is in conjunctive form $C_1 \wedge C_2 \wedge \cdots \wedge C_k$, where each C_i is a disjunction of three or fewer literals ;*

(iii) *length$(H) \leq 15$ length$(F) - 14$, where* length *means 'number of symbols other than brackets'.*

Proof. First cancel any double negations in F, and work with the result. The H produced will clearly work also for the original F. Thus we may assume that $\neg\neg$ appears nowhere in F.

Next write down a sequence,

$$Q_1\ ,\ \neg Q_1\ ,\ Q_2\ ,\ \neg Q_2\ ,\ \cdots\ ,\ Q_a\ ,\ \neg Q_a\ ,\ F_1\ ,\ \neg F_1\ ,\ F_2\ ,\ \neg F_2\ ,\ \cdots\ ,\ F_b\ ,\ \neg F_b\ ,$$

as follows. The Q_i are a complete list of the propositional variables occurring in F. The F_j are a complete list of those subformulae of F which are neither literals nor negations of other formulae, and occurring in the order in which they would appear in some formula verification column for F.

Thus F itself is F_b or $\neg F_b$. And for each j, we have $F_j = U_j * V_j$ for some U_j and V_j which appear to the left of F_j in the list, and with $*$ as one of the four binary connectives $\wedge$, $\vee$, $\rightarrow$ or $\leftrightarrow$ (probably varying for different j).

Now write down a matching sequence

$$Q_1\ ,\ \neg Q_1\ ,\ Q_2\ ,\ \neg Q_2\ ,\ \cdots\ ,\ Q_a\ ,\ \neg Q_a\ ,\ X_1\ ,\ \neg X_1\ ,\ X_2\ ,\ \neg X_2\ ,\ \cdots\ ,\ X_b\ ,\ \neg X_b\ .$$

The X_j are to be a list of distinct new variables, new in the sense of not having any occurrence in F.

Call the first sequence α. Let ϕ be the bijection which maps it to the second sequence, term-by-term. That is,

$$\phi(Q_i) := Q_i \qquad \phi(F_j) := X_j \qquad \text{and} \qquad \phi(\neg K) := \neg\phi(K) \quad \text{for all } K\ .$$

Note that $\phi(F)$ exists, and is either X_b or $\neg X_b$. We think of X_j as an 'alias' for F_j .

The following formula G will be satisfiable if and only if F is, as we show in the last part of the proof:

$$G\ :=\ (X_1 \leftrightarrow \phi(U_1){*}\phi(V_1)) \wedge\ (X_2 \leftrightarrow \phi(U_2){*}\phi(V_2)) \wedge\ \cdots\ \wedge\ (X_b \leftrightarrow \phi(U_b){*}\phi(V_b)) \wedge \phi(F)\ .$$

In the jth factor, the connective $*$ used is the one in $U_j * V_j$ above—it seemed a bit heavy-handed to use $*_1$, $*_2$, etc.

Now we shall define H so that it is truth equivalent to G, and has the following form:

$$H\ :=\ H_1 \wedge H_2 \wedge \cdots \wedge H_b \wedge \phi(F)\ .$$

For each j, the formula H_j will be truth equivalent to $X_j \leftrightarrow \phi(U_j) * \phi(V_j)$, the jth factor in G. The latter has the form

$$L \leftrightarrow M * N \quad \text{treq} \quad (L \vee \neg(M * N)) \wedge (\neg L \vee (M * N))$$

for literals L, M and N. To get the H_j, use the following four elementary equivalences, one for each choice of connective $*$:

$*\ =\ \wedge$ gives $L \leftrightarrow M{\wedge}N$ treq $(L{\vee}\neg M{\vee}\neg N) \wedge (\neg L{\vee}M) \wedge (\neg L{\vee}N)$;

$*\ =\ \vee$ gives $L \leftrightarrow M{\vee}N$ treq $(L{\vee}\neg M) \wedge (L{\vee}\neg N) \wedge (\neg L{\vee}M{\vee}N)$;

$*\ =\ \rightarrow$ gives $L \leftrightarrow (M \rightarrow N)$ treq $(L{\vee}M) \wedge (L{\vee}\neg N) \wedge (\neg L{\vee}\neg M{\vee}N)$; and

$*\ =\ \leftrightarrow$ gives $L \leftrightarrow (M \leftrightarrow N)$ treq $(L{\vee}\neg M{\vee}\neg N){\wedge}(L{\vee}M{\vee}N){\wedge}(\neg L{\vee}M{\vee}\neg N){\wedge}(\neg L{\vee}\neg M{\vee}N)$.

We use the right-hand sides in these four truth equivalences to produce H_j as the conjunction of three or four disjunctions of two or three literals. Of course the negation of a literal which itself is the negation of a variable is replaced by just that variable itself—i.e. cancel the double negation.

It is immediate that H has the form required by (ii) in the theorem, with a suitable k between $3b+1$ and $4b+1$, depending on how many of the right-hand sides in the display above with three 'components' are used, and how many with four.

It is also immediate that H is truth equivalent to G. And so H is 'satisfiable-equivalent' to F, once we see below that G is 'satisfiable-equivalent' to F.

First let's check the length condition in (iii) of the theorem. By far the worst case for H_j's are those which arise from the last of the four displayed truth equivalences. In those cases, for such a j with $*\ =\ \leftrightarrow$, we have 12 literals appearing in H_j, half of which are unnegated, giving 18 symbols, plus 8 $\vee$'s, plus 4 $\wedge$'s, including the $\wedge$ to the right, connecting up to H_{j+1} (or to $\phi(F)$ when $j = b$). When F is F_b (unnegated), this gives an upper bound $30b + 1$ for the length of H (and $30b + 2$ when F is negated, but that won't be needed).

Now a formula with exactly "b" occurrences of binary connectives clearly has length at least $2b + 1$—this may be proved readily by induction. Thus, in the case of unnegated F, we get our required inequality

$$\ell(H)\ \leq\ 30b+1\ \leq\ 30(\frac{\ell(F)-1}{2})\ +\ 1\ =\ 15\ell(F)\ -\ 14\ .$$

On the other hand, if $F = \neg F_b$, then the same process exactly as if F were F_b is used to produce H, except for the negation sign in $\phi(F)$, occurring as the penultimate symbol in H. Thus by the above result for unnegated F, we get

$$\ell(H)\ \leq\ 15[\ell(F)-1]\ -\ 14\ +\ 1\ <\ 15\ell(F)\ -\ 14\ ,$$

as required.

Here is an example before finishing the proof. Take $F\ =\ (P \leftrightarrow Q)$, producing the sequence α as

$$P\ ,\ \ \neg P\ ,\ \ Q\ ,\ \ \neg Q\ ,\ \ P \leftrightarrow Q\ ,\ \ \neg(P \leftrightarrow Q)$$

So $b = 1$, and the answer is

$$H\ =\ (X_1 \vee \neg P \vee \neg Q)\ \wedge\ (X_1 \vee P \vee Q)\ \wedge\ (\neg X_1 \vee P \vee \neg Q)\ \wedge\ (\neg X_1 \vee \neg P \vee Q)\ \wedge\ X_1\ .$$

This example, with $\ell(F) = 3$, $\ell(H) = 31$, shows the length inequality to be sharp, unless we find a more sophisticated process to get an H satisfying (i) and (ii) in the theorem.

To finish the proof, we must show that G is 'satisfiable-equivalent' to F.

Below we'll use the Hilbert truth morphism notation from Section 4.2, applying connectives to truth values themselves to produce truth values—for example, $(\mathsf{Fs} \rightarrow \mathsf{Tr})\ =\ \mathsf{Tr}$.

First suppose that G is satisfiable, and so e, say, is a truth assignment with $\mathsf{e}(G) = \mathsf{Tr}$. This is equivalent to saying that

$$\mathsf{e}(X_j) = \mathsf{e}(\phi(U_j) * \phi(V_j))\ \text{ for all } j \qquad \text{and} \qquad \mathsf{e}(\phi(F)) = \mathsf{Tr}\ .$$

By induction 'from left to right', we show that $\mathsf{e}(\phi(K)) = \mathsf{e}(K)$ for all K in the first sequence α. Starting the induction is trivial, since actually $\phi(K) = K$ for the literals at the left end of the sequence. Then, inductively, when $K = F_j$, we have

$$\begin{aligned}\mathsf{e}(\phi(F_j))\ =\ \mathsf{e}(X_j)\ &=\ \mathsf{e}(\phi(U_j) * \phi(V_j))\ =\ \mathsf{e}(\phi(U_j)) * \mathsf{e}(\phi(V_j)) \\ &=\ \mathsf{e}(U_j) * \mathsf{e}(V_j)\ =\ \mathsf{e}(U_j * V_j)\ =\ \mathsf{e}(F_j)\ \ ,\end{aligned}$$

as required. The equality beginning the lower line uses the inductive hypothesis. And when $K = \neg F_j$, we have

$$\mathsf{e}(\phi(\neg F_j)) \;=\; \mathsf{e}(\neg\phi(F_j)) \;=\; \neg\mathsf{e}(\phi(F_j)) \;=\; \neg\mathsf{e}(F_j) \;=\; \mathsf{e}(\neg F_j)$$

as required, completing the induction.

But F occurs as the last or second last term in the sequence α, so we have

$$\mathsf{e}(F) \;=\; \mathsf{e}(\phi(F)) \;=\; \mathsf{Tr}\ ,$$

showing that F is satisfiable.

Conversely, suppose that F is satisfiable. And so f, say, is a truth assignment with $\mathsf{f}(F) = \mathsf{Tr}$. Define a truth assignment e on variables so that it agrees with f except possibly on the X_j. On them, we may then define it inductively on j by requiring that

$$\mathsf{e}(X_j) \;=\; \mathsf{e}(\phi(U_j) * \phi(V_j))\ .$$

To show that $\mathsf{e}(G) = \mathsf{Tr}$, and so prove that G is satisfiable, as required, the equality displayed immediately above deals with all but the last conjunctive factor in the definition of G. So it remains only to show that $\mathsf{e}(\phi(F)) \;=\; \mathsf{Tr}$.

First we prove by induction 'from left to right' that $\mathsf{e}(\phi(K)) = \mathsf{f}(K)$, for all K in the first sequence α. Starting the induction is trivial, since actually $\phi(K) = K$ for the literals at the left end of the sequence, and e agrees with f on them. Then, inductively, when $K = F_j$, we have

$$\begin{aligned}\mathsf{e}(\phi(F_j)) \;=\; \mathsf{e}(X_j) \;&=\; \mathsf{e}(\phi(U_j) * \phi(V_j)) \;=\; \mathsf{e}(\phi(U_j)) * \mathsf{e}(\phi(V_j)) \\ &=\; \mathsf{f}(U_j) * \mathsf{f}(V_j) \;=\; \mathsf{f}(U_j * V_j) \;=\; \mathsf{f}(F_j)\ ,\end{aligned}$$

as required. And when $K = \neg F_j$, we have

$$\mathsf{e}(\phi(\neg F_j)) \;=\; \mathsf{e}(\neg\phi(F_j)) \;=\; \neg\mathsf{e}(\phi(F_j)) \;=\; \neg\mathsf{f}(F_j) \;=\; \mathsf{f}(\neg F_j)$$

as required, completing the induction.

So we have $\mathsf{e}(\phi(F)) \;=\; \mathsf{f}(F) \;=\; \mathsf{Tr}$, as required to complete the proof.

The existence of this algorithm shows why the question of getting an efficient algorithm for satisfiability of sets of clauses (the SAT problem) is of such interest, at least to logicians interested in automated theorem proving. Computer scientists interested in complexity theory will often start immediately with clauses, and study the question of whether there is an efficient method for deciding satisfiability, without worrying about other formulae which aren't already in conjunctive form. Their reason for interest is mainly that SAT and 3-SAT are about the simplest $\mathcal{NP}$-complete problems, and it is convenient to reduce other problems (travelling salesman, etc.) to them.

The answer is still unknown, but the conjecture is that there isn't a polynomial-time algorithm for this class of computational problems (as we said earlier).

To finish, here is a table followed by some comments. It summarizes several general points of the preceding several sections of this chapter.

input	remark	computational problem	remark
finite set of formulae $\{F_1, \cdots, F_n\}$, plus one extra formula G	(premisses) (conclusion)	argument valid or invalid? i.e. $\{F_1, \cdots, F_n\} \models G$? (yes or no)	*logicians' main problem*
efficient process $\downarrow$ (*trivial*)			
finite set of formulae $\{H_1, \cdots, H_k\}$	process produces $\{F_1, \cdots, F_n, \neg G\}$	UNSATisfiable or SATisfiable?	(as a set of formulae)
efficient process $\downarrow$ (**4.5**, *or appendix*)			
single formula in conjunctive form	use process on $H_1 \wedge \cdots \wedge H_k$	UNSATisfiable or NOT?	(as a formula)
efficient process $\downarrow$ (*trivial*)			
set of clauses	*computists'* (SAT) *problem*	UNSAT or SAT?	(as a set of clauses)
efficient process $\updownarrow$ (*beginning of section*)			
set of clauses, each containing 3 literals	*computists'* (3–SAT) *problem*	UNSAT or SAT?	(as a set of clauses)

Because of the efficient processes for passing back and forth between these five problems, either none of them admits an efficient algorithm for solving the problem, or they all do. The conjecture is that none of them does. But the only evidence for that conjecture seems to be that humans have a certain self-reverence, and no one has yet found an efficient algorithm, despite vigorous efforts. Truth tables are hopeless, and DPP is only better when applied to certain special forms of input, such as Horn clauses, described near the end of this chapter.

Appendix—Another method

We'll describe another way to produce, from an arbitrary formula, a new formula in conjunctive form, where
(i) the process is efficient; and
(ii) the new formula is satisfiable if and only if the old one is.

Here is a quick summary of the process. Suppose that F is a formula, possibly abbreviated. First produce a truth equivalent formula, $\hat{F}$, as follows:

By using their definitions, eliminate any occurrences of $\leftrightarrow$ and $\rightarrow$;

Then move all occurrences of $\neg$ 'as far inside as possible', so that they 'apply' only to propositional variables, not to larger subformulae, cancelling any double negations that come up.

(This is just a reiteration of the first stage in the 'manipulation' method for getting DNF's or CNF's.) Intuitively it is clear that producing $\hat{F}$ is efficient. Once the formal definitions in complexity theory were given, it would be mathematically provable to be a polynomial-time algorithm. This formula $\hat{F}$ is a 'polynomial in literals using operations $(\wedge, \vee)$'. (It is amusing that "polynomial" gets used in two quite unrelated ways.)

The main step is now to produce a conjunctive-form formula $\hat{F}^*$ [more simply, ... to produce such a G^*, for any polynomial G in literals using operations $(\wedge, \vee)$].

The one new idea is that the formulae $H \vee J$ and $(H \vee X) \wedge (\neg X \vee J)$, where X is a new propositional variable not occurring in H or J, are either both satisfiable, or neither is satisfiable. (This is closely related to the methods at the beginning of this section for replacing a big clause by several clauses of length three.)

Using the fact above, you just work inductively, converting larger and larger subformulae to conjunctive form. This proceeds by successively using the above idea, once for each occurrence of $\vee$ in the original formula; and then distributing the new variable inside. To do this systematically on a large formula, you need a formula verification column or some other listing of subformulae. An algorithm for doing this is given in the first two sections of **Appendix B** to Chapter 8. As above, both the latter, and the inductive process of introducing new variables X, are efficient, i.e. polynomial-time.

The idea is easy to grasp from a few examples, so we'll do that first. These examples are here to illustrate the process, not to convince you of its value. For one thing, several of the examples are already in conjunctive form at a stage before any of the new variables have been introduced. We'd need examples which are much too large before it would look convincing, except that then they would be unreadable. Also, for specific small examples, it's hard to convincingly illustrate the equivalence of satisfiability with respect to the new and old formulae, because the answer will be almost immediate for both by inspection.

So after the examples, I'll describe the above process more formally, with a general formulation and indication of the proof of satisfiability equivalence. I'll have to be vague about the efficiency side, since the necessary definitions haven't been given here. A course on complexity theory should treat those matters with precision.

The first two are examples of formulae which are unsatisfiable— ones which are contradictions.

(I) Consider the formula

$$F \;=\; ((P \rightarrow Q) \wedge P) \wedge \neg Q \;.$$

[Unsatisfiable for this is really the soundness of the rule of inference (MP).]

There are no $\leftrightarrow$'s to replace.

Replace $P \to Q$ by $\neg(P \wedge \neg Q)$, giving

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

Now move that leftmost $\neg$ inside, and cancel the double-negation, to produce

$$\hat{F} \ = \ ((\neg P \vee Q) \wedge P) \wedge \neg Q \ .$$

(Usually, it's better to change $F \to G$ to $\neg F \vee G$ right away, instead of using the definition, to save a few steps with moving negations inside.)

Now we only have one $\vee$, so apply the new idea to that to produce the formula

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

where X is a propositional variable different from both P and Q. No more needs to be done except throw away a few brackets in our usual manner of ignoring the distinction between formulae which differ only by different bracketing coming from associativity. This gives our final conjunctive-form formula (*not* truth equivalent to the original)

$$\hat{F}^* \ = \ (\neg P \vee X) \wedge (\neg X \vee Q) \wedge P \wedge \neg Q \ .$$

The corresponding clauses are of course

$$\{\neg P, X\} \ \ \{\neg X, Q\} \ \ \{P\} \ \ \{\neg Q\} \ .$$

Above, we have clauses with less than three literals each. If we'd had more than three literals in some clauses, the earlier process from this section could now be used to get it down to three at most. However, the 3-SAT problem usually means deciding satisfiability for a finite set of clauses containing *exactly* three literals each. To change "at most three" to "exactly three" is quite easy. You can just replace $\{L\}$ by four clauses

$$\{L, X, Y\} \ \ \{L, X, \neg Y\} \ \ \{L, \neg X, Y\} \ \ \{L, \neg X, \neg Y\} \ .$$

And replace $\{L, M\}$ by two clauses $\{L, M, Z\} \ \ \{L, M, \neg Z\}$. Here X, Y and Z are 'brand-new' variables. It is very easy to see the 'satisfiability-equivalence' of the new with the old. So converting the above example to a 3-SAT problem this way would produce

$$\begin{gathered}\{\neg P, X, X_1\} \quad \{\neg P, X, \neg X_1\} \quad \{\neg X, Q, X_2\} \quad \{\neg X, Q, \neg X_2\} \\ \{P, X_3, X_4\} \quad \{P, X_3, \neg X_4\} \quad \{P, \neg X_3, X_4\} \quad \{P, \neg X_3, \neg X_4\} \\ \{\neg Q, X_5, X_6\} \quad \{\neg Q, X_5, \neg X_6\}\{\neg Q, \neg X_5, X_6\} \quad \{\neg Q, \neg X_5, \neg X_6\} \ .\end{gathered}$$

Evidently all of the above displayed clause sets or formulae are unsatisfiable.

Here is a just slightly more complicated unsatisfiable example. Again it illustrates the kind of thing one could do, but normally wouldn't, because it's already gotten to conjunctive form before starting the introduction of the new variables.

(II) Given

$$(P \leftrightarrow R) \wedge \neg P \wedge (Q \to R) \wedge \neg(\neg R \wedge \neg Q) \ ,$$

eliminating the $\leftrightarrow$, moving a negation inwards and cancelling a couple of double negations yields

$$(P \to R) \wedge (R \to P) \wedge \neg P \wedge (Q \to R) \wedge (R \vee Q) \ .$$

Eliminating all three $\to$'s yields

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

Now doing the major step with four new propositional variables yields the conjunctive form

$$(\neg P \vee X) \wedge (\neg X \vee R) \wedge (\neg R \vee Y) \wedge (\neg Y \vee P) \wedge \neg P \wedge (\neg Q \vee Z) \wedge (\neg Z \vee R) \wedge (R \vee U) \wedge (\neg U \vee Q) \ .$$

This has been another example where the answer is unsatisfiable. It is tedious to convert this into a 3-SAT problem, so I won't do it—the number of clauses would be twenty, if using the method above.

Both examples, (I) and (II), were already in conjunctive form before we even started on the main part of the process, so are not terribly convincing. But (III) below is more convincing. You might try producing a conjunctive form truth equivalent to it, and compare to what we do below. Particularly if we expanded this example in the obvious way so that the subscripts went up to 8 or 10, not just 4, you'd find the process here much faster than standard methods to actually produce a conjunctive form truth equivalent to the starting formula. (On the other hand, these aren't good examples for the problem of satisfiability, since they are obviously satisfiable, being in disjunctive form to begin with.)

(III) To illustrate both the distributing of the new variable, and that sometimes you must take several steps (i.e. not introduce the new variables for all the occurrences of $\vee$ simultaneously), consider

$$(P_1 \wedge Q_1) \vee (P_2 \wedge Q_2) \vee (P_3 \wedge Q_3) \vee (P_4 \wedge Q_4) \ .$$

This needs some 'associativity brackets' to be made into a specific formula. Treating it as though there were brackets around two equal halves, our first step gives

$$(((P_1 \wedge Q_1) \vee X) \wedge (\neg X \vee (P_2 \wedge Q_2))) \vee (((P_3 \wedge Q_3) \vee Y) \wedge (\neg Y \vee (P_4 \wedge Q_4))) \ .$$

Distributing writes this as a disjunction of two conjunctive forms:

$$((P_1 \vee X) \wedge (Q_1 \vee X) \wedge (\neg X \vee P_2) \wedge (\neg X \vee Q_2)) \vee ((P_3 \vee Y) \wedge (Q_3 \vee Y) \wedge (\neg Y \vee P_4) \wedge (\neg Y \vee Q_4)) \ .$$

Now introduce a third new variable Z for the $\vee$ in the middle to get

$$(((P_1 \vee X) \wedge (Q_1 \vee X) \wedge (\neg X \vee P_2) \wedge (\neg X \vee Q_2)) \vee Z)$$

$$\wedge$$

$$(\neg Z \vee ((P_3 \vee Y) \wedge (Q_3 \vee Y) \wedge (\neg Y \vee P_4) \wedge (\neg Y \vee Q_4))) \ .$$

Distributing again gives the required conjunctive-form formula:

$$(P_1 \vee X \vee Z) \wedge (Q_1 \vee X \vee Z) \wedge (\neg X \vee P_2 \vee Z) \wedge (\neg X \vee Q_2 \vee Z)$$

$$\wedge$$

$$(\neg Z \vee P_3 \vee Y) \wedge (\neg Z \vee Q_3 \vee Y) \wedge (\neg Z \vee \neg Y \vee P_4) \wedge (\neg Z \vee \neg Y \vee Q_4) \ .$$

(IV) The formula

$$(P \vee Q) \vee R \ ,$$

is already in conjunctive form, but let's proceed anyway. The first step yields

$$((P \vee X) \wedge (\neg X \vee Q)) \vee R \ .$$

Then we get

$$(((P \vee X) \wedge (\neg X \vee Q)) \vee Y) \wedge (\neg Y \vee R) \ .$$

Now distributing and ignoring unneeded brackets, we get

$$(P \vee X \vee Y) \wedge (\neg X \vee Q \vee Y) \wedge (\neg Y \vee R) \ .$$

(V) Similarly, the formula $((P \vee Q) \vee R) \vee S$ yields a 'satisfiability-equivalent' (*not* truth equivalent!) conjunctive-form formula

$$(P \vee X \vee Y \vee Z) \wedge (\neg X \vee Q \vee Y \vee Z) \wedge (\neg Y \vee R \vee Z) \wedge (\neg Z \vee R) \ .$$

Examples (IV) and (V) are the second and third in an infinite sequence of examples. Not counting brackets, in the mth example, the input formula has "$2m+1$" symbols, and the output formula has "m^2+6m+1" . These are basically the worst possible cases for the size of the output as a function of the size of the input. One could formally prove that the length of the output formula is never more than the square of the length of the input formula. See Exercise 4.18. Together with some remarks about brackets, about the moving negation signs etc., and about the scanning etc. needed to carry this out automatically, it can be proved that we're dealing with a *quadratic*-time algorithm here, so certainly efficient, i.e. polynomial-time. But it is inferior to the *linear*-time algorithm for doing the same thing from Theorem **4.5**.

To give a more formal definition of the part of the process which we earlier called 'going from G to G^*', it is convenient to use a language whose formulae are precisely those things we earlier referred loosely to as "polynomials in literals using operations $\wedge$ and $\vee$".

This language has as its symbols: the brackets; the connectives $\wedge$ and $\vee$; and two infinite sequences $\{P_i\}$ and $\{\neg P_i\}$, for $i \geq 1$. Formulae are defined inductively by asserting that they are finite sequences formed using the rules that

(i) each P_i and $\neg P_i$, for $i \geq 1$ is a formula; and

(ii) if F and G are formulae, then so are $(F \wedge G)$ and $(F \vee G)$.

This language is theoretically as expressive as that of propositional logic, but it would be in very bad taste to use it from the beginning. It doesn't allow the negation of anything bigger than a propositional variable, so it's not too good for translating from mathematics or from natural languages.

Working within this language, the process which we've been discussing is defined by induction on formulae:

$$P_i^* = P_i \;\; ; \;\; (\neg P_i)^* = \neg P_i \;\; ; \;\; (F \wedge G)^* = F^* \wedge G^* \;\; ;$$

$$(F \vee G)^* = (F_1 \vee X) \wedge (F_2 \vee X) \wedge \cdots \wedge (F_a \vee X) \wedge (\neg X \vee G_1) \wedge \cdots \wedge (\neg X \vee G_b) \;,$$

$$\text{where } F^* = F_1 \wedge F_2 \wedge \cdots \wedge F_a \quad ; \quad G^* = G_1 \wedge \cdots \wedge G_b$$

$$\text{with each } F_i \text{ and } G_j \text{ being a disjunction of literals, and}$$

$$X = P_m \text{ for the smallest } m \text{ with } P_m \text{ not occurring in } F^* \text{ or } G^* \;\; .$$

With these definitions, one can now prove by induction on formulae that

(a) *All H^* are in conjunctive form, and*

(b) *H is satisfiable if and only if H^* is satisfiable.*

Actually, the proof of (a) should be simultaneous with the inductive definition, since the definition of $(F \vee G)^*$ depends on knowing that both F^* and G^* are already in conjunctive form. This proof is brief and straightforward.

For making the induction work, (b) should be strengthened to

$(b)_+$ *A truth assignment satisfying H^* will also satisfy H; and any truth assignment satisfying H can be adjusted on the new variables to give one which satisfies H^*.*

The inductive proof of this is straightforward. Let's just do the inductive step for a formula $F \vee G$, where we assume that $(b)_+$ holds for both F and G.

In one direction, given a truth assignment which makes $(F \vee G)^*$ true, if it's true at X, then all G_j are true, and if it's false at X, then all F_i are true. Thus either G^* is true, or F^* is true. By the inductive hypothesis, either G is true, or F is true. And so, $F \vee G$ is true at this truth assignment, as required.

In the other direction, given a truth assignment which makes $F \vee G$ true, so either G is true or F is true, we can extend the evaluation to the new variables involved in G^* and F^* to make at least one of G^* or F^* true, by the inductive hypothesis. Now extend to X as follows. If G^* is true, make X true; if not and so F^* is true, make X false. Then it is immediate that $(F \vee G)^*$ is true, by simply examining the recipe defining it.

This process, going from G to G^* above, is obviously wasteful in some cases. For example, if G is already in conjunctive form, why bother changing it at all?

Below we describe what one might actually do to a formula 'by hand', using this method, so as not to waste time with extra steps after already arriving at a conjunctive form.

Step 1 : Convert your formula into a truth equivalent "polynomial in literals"—that is, eliminate all $\leftrightarrow$'s; then eliminate all $\rightarrow$'s; then move all $\neg$'s inside and cancel double negations.

Step 2 : Now use the following notation. Your "polynomial in literals" is uniquely of the form $F_1 \wedge \cdots \wedge F_n$ (where possibly $n = 1$), and no F_i is itself a conjunction of two formulae. That is, decompose your "polynomial in literals" into a conjunction of as many as possible smaller formulae.

(non)Step 3 : Leave as is any F_i which is already a disjunction of one or more literals.

Step 4 : For each i such that F_i is not a disjunction of one or more literals, now go ahead and apply the algorithm which we gave a more formal description of above. That is, alter F_i to F_i^*. Note that you work 'from the inside', changing subformulae of the form $G \vee H$ to $(G \vee X) \wedge (\neg X \vee H)$, then distribute, etc., etc.

Applying the steps above in examples (I), (II), (IV) and (V), nothing would happen after Step 1.

In example (III), Step 1 would do nothing. In Step 2, we'd have $n = 1$. Step 3 wouldn't apply, and Step 4 is just what we did in that example before.

Exercises.

4.24 Use the method of this appendix to find conjunctive-form formulae which are satisfiability-equivalent to the following formulae from 4.4. Then write down the corresponding set of clauses. Then write down a satisfiability equivalent set of clauses from the 3-SAT problem.

(i) $R \leftrightarrow P \wedge (Q \rightarrow R)$.

(ii) $P \rightarrow (Q \rightarrow (R \rightarrow S))$.

(iii) $(P \rightarrow Q) \wedge (P \wedge \neg Q)$.

(iv) $P \rightarrow \neg(Q \rightarrow \neg(R \rightarrow S))$.

(v) $(P_1 \wedge P_2 \wedge P_3) \vee (Q_1 \wedge Q_2 \wedge Q_3)$.

(vi) $(P_1 \wedge P_2) \vee (Q_1 \wedge Q_2) \vee (R_1 \wedge R_2)$.

4.25 Write out a complete proof of (a) from a few paragaphs above.

4.26 Here are some definitions and identities, leading up to a proof of the essential part of the "quadratic-time" claim we made about the main process from this section. We deal with formulae in the 'new' language, which uses only $\vee$ and $\wedge$ applied to literals. Define $\ell(F)$ to be the total number of occurrences of symbols other than brackets in the formula F. Define $m(F)$ to be the total number of occurrences of literals in F. Prove:

(i) The number of constituents in F^* is $m(F)$.

(ii) $m(F) \leq \ell(F)$.

(iii) $m(F \vee G) = m(F) + m(G) = m(F \wedge G)$.

(iv) $\ell(F \vee G) = \ell(F) + \ell(G) + 1 = \ell(F \wedge G)$.

(v) $\ell((F \wedge G)^*) = \ell(F^*) + \ell(G^*) + 1$.

(vi) $\ell((F \vee G)^*) = \ell(F^*) + \ell(G^*) + 2m(F) + 3m(G) + 1$.

Now proceed by induction, probably using several of the above, to show that for all formulae H

(vii) $\ell(H^*) \leq \ell(H)^2$.

4.10 Horn clauses and related examples.

> *A program in (the programming language) PROLOG is essentially a list of horn clauses. The PROLOG interpreter is then simply a horn clause theorem-prover.*
>
> Martin Davis

Horn clauses have nothing to do with sexual arousal (nor are they clauses with musical ability), but rather, Horn clauses are clauses which contain at most one propositional variable; i.e. either all, or all but one, of the clause members are *negated* propositional variables. These seem to arise quite a bit, when mathematics is formalized. Here we want to indicate how there is an efficient algorithm for deciding satisfiability of sets of Horn clauses, despite the Davis-Putnam procedure (and any other known algorithm) being slow for sets of arbitrary clauses.

To do this, we go back from clauses to formulae as follows:

(I) Each clause $\{B\}$, for a propositional variable B, simply corresponds to the formula B.

(II) Each clause $\{\neg A_1, \cdots, \neg A_n\}$ containing only (one or more) negated propositional variables corresponds to the formula

$$\neg(A_1 \wedge \cdots \wedge A_n) \ ,$$

it being truth equivalent to the usual formula $\neg A_1 \vee \cdots \vee \neg A_n$ which we associate with such a clause.

(III) Each clause $\{B, \neg A_1, \cdots, \neg A_n\}$ containing a propositional variable and one or more negated variables corresponds to the formula

$$A_1 \wedge \cdots \wedge A_n \rightarrow B \ ,$$

it being truth equivalent to the usual formula $B \vee \neg A_1 \vee \cdots \vee \neg A_n$ which we associate with such a clause.

Particularly this last associated type of formula is often seen in mathematics. For example,

(f is a real-valued function)$\wedge$ (f is continuous)$\wedge$ (f has domain a closed interval of reals) $\rightarrow$ the image of f is a bounded set.

or

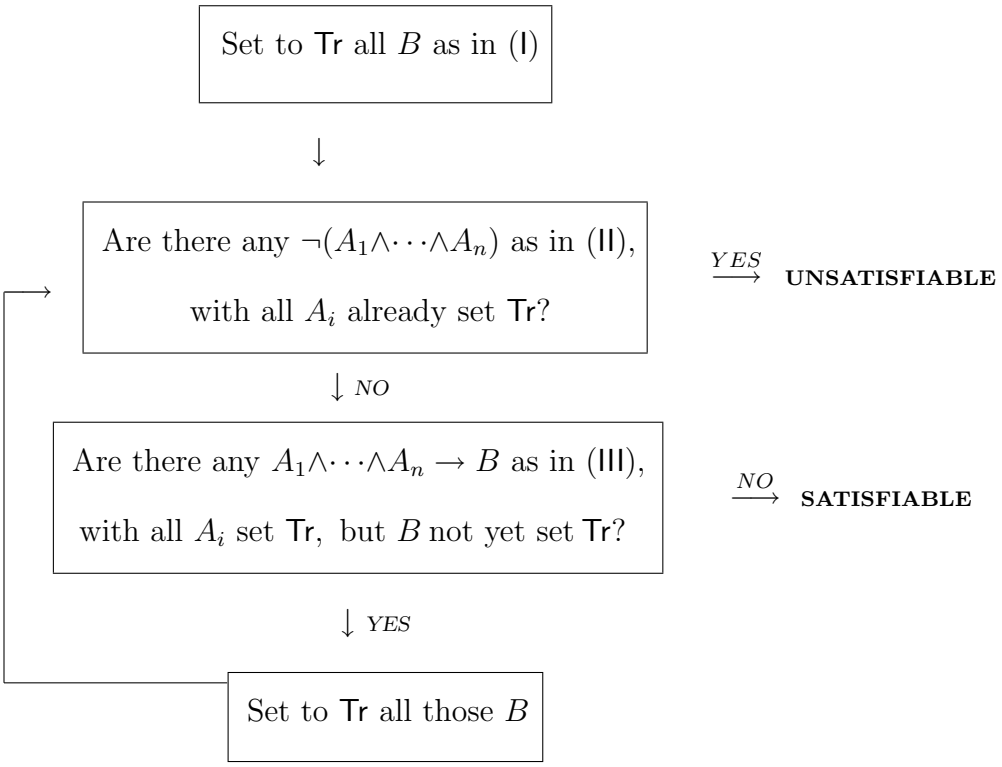
(G is a group)$\wedge$ (G is abelian)$\wedge$ (G is finitely generated) $\rightarrow$ G is isomorphic to a direct sum of cyclic groups.

The idea for deciding satisfiability for a set of formulae like these associated ones is not complicated. First set Tr all variables B as in (I), then begin a cycle in which at least one more of the variables is set Tr each time the cycle is completed. These will be forced for any truth assignment which supposedly makes all the formulae true. In the cycle, one first checks for formulae as in (II) where all the A_i have already been set Tr. If any exist, we must have unsatisfiability. If none exist, one next checks for formulae as in (III) where all the A_i have already been set Tr, but B hasn't. If any exist, then the B in each such formula is set Tr, and a new cycle begun. If

none exist, then the set of formulae is satisfiable. *One truth assignment* (of possibly many) *which does the trick is the one with all variables already set* Tr *being given that value, and all other variables set* Fs. (If there are other truth assignments which also work, they necessarily make Tr a larger set of variables.)

This algorithm terminates in a number of cycles not greater than the number of variables we had to begin with. It clearly doesn't suffer from any 'combinatorial explosion', so it's high efficiency is quite believable.

Here is a little flow chart, restating the algorithm.



As an example, the clause set

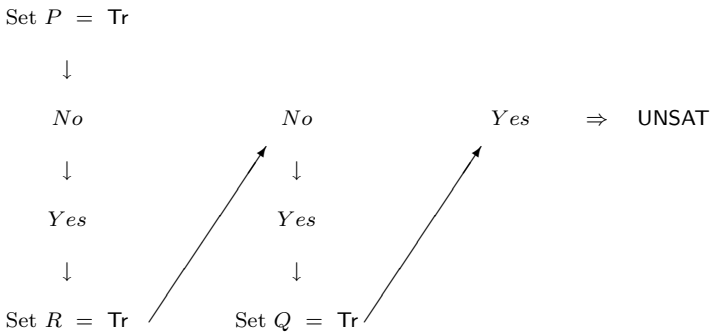
$$\{P\}\ ,\ \{\neg R, \neg Q\}\ ,\ \{\neg P, R\}\ ,\ \{\neg R, Q\}$$

consists of only Horn clauses. The process above is applied to the set of formulae

$$P\ ,\ \neg(R \wedge Q)\ ,\ P \to R\ ,\ R \to Q\ .$$

Initially P is set true. In the first cycle, R is set true. In the second cycle, Q is set true. The decision "UNSAT" is arrived at in the third cycle.

Here is a schematic diagram, following the same 'topology' as the chart above, re-doing this example. You might want to follow this when doing problems on Horn clauses, or invent your own pattern:



If we changed $\{\neg R, Q\}$ to $\{\neg R, P\}$ in the example above, and retained the other three clauses, the process would get through the first cycle as above, but then decide "SAT" in the second cycle. The truth assignment produced would set P and R true, but Q false.

Correctness of the algorithm.

To verify that this algorithm does give the correct answer, first note that, because the associated formulae are truth equivalent to the conjunctive constituents which usually generate the clauses, the original set of Horn clauses is a satisfiable set of clauses if and only if the set of associated formulae is a satisfiable set of formulae. As for the set of formulae, the UNSAT case of the algorithm is obvious. (In this case, the algorithm is essentially proceeding with a proof by contradiction, assuming it has a truth assignment showing satisfiability, finding a bunch of variables which must be true under this evaluation, and then deriving a contradiction.) In the SAT case, let's show

that the indicated truth assignment does indeed make each of the associated formulae true.

Type (I): We do set each such B true, as required.

Type (II): Note that, in such a formula $\neg(A_1 \wedge \cdots \wedge A_n)$, at least one of the A_i cannot have been set true, or the algorithm would not have reached the SAT conclusion. That particular A_i is Fs under the truth assignment, so the formula $\neg(A_1 \wedge \cdots \wedge A_n)$ is Tr, as required.

Type (III): For an associated formula $A_1 \wedge \cdots \wedge A_n \rightarrow B$, it could only be Fs under the truth assignment if B were Fs and all A_i were Tr. But such a situation would cause the algorithm to go into another cycle, not to decide SAT. So all such associated formulae are indeed Tr.

Passing from the set of clauses to the associated set of formulae is not really necessary. It is quite straightforward to see that the algorithm 'works'. when expressed directly in terms of the three types of Horn clauses.

You may have noticed that, if a set of Horn clauses has no singleton clause containing a variable (i.e. type (I)), then the algorithm terminates almost immediately, with the decision "SAT", setting every variable false. A form of this is true whether the clauses are all Horn or not, quite a bit more generally. See 4.26(i) below. Also from inspecting the flow chart, if a set of Horn clauses has no clause containing only negated variables (i.e. type (II)), then the algorithm will decide "SAT". Again, this is true whether the clauses are all Horn or not, quite a bit more generally. See 4.26(ii) below.

Exercises

4.27 Using the method of this section, decide the satisfiability question for the following sets of clauses (noting carefully first that they *are* all Horn clauses). If satisfiable, write down the truth assignment which the algorithm produces. In either case, apply resolution to double-check your SAT/UNSAT decision.

(i)

$$\{R, \neg S\} \quad \{\neg P, Q, \neg R, \neg T\}$$

$$\{S\} \quad \{Q, \neg S, \neg T\} \quad \{\neg P, \neg R, T\}$$

$$\{\neg S, \neg Q\} \quad \{P, \neg Q, \neg S, \neg T\} \quad \{T\}$$

(ii)

$$\{R\} \quad \{\neg P, Q, \neg R, \neg S\} \quad \{\neg P\}$$
$$\{\neg T\} \quad \{R, \neg T\} \quad \{\neg P, \neg R, \neg S, T\}$$
$$\{\neg P, T\} \quad \{S\} \quad \{P, \neg Q, \neg R\}$$

(iii)

$$\{R\} \quad \{\neg P, \neg R, S\} \quad \{\neg Q, \neg S\}$$
$$\{\neg S, T\} \quad \{P, \neg S, \neg T\}$$
$$\{P\} \quad \{\neg Q, \neg R\}$$

4.28 (i) Show directly from the definition of satisfiability that, if a set of clauses has at least one negated variable in each clause, then the set is satisfiable.

(ii) Show directly from the definition of satisfiability that, if a set of clauses has at least one variable (unnegated) in each clause, then the set is satisfiable.

4.29 Stephen New has pointed out the following: Generalize clean-up step (2) in DPP, which said to erase extra copies of clauses, so that now we also erase any clause which is a proper subset of another clause. First show that this process does preserve satisfiability/unsatisfiability. Then show that, if you start from a set of Horn clauses, and do this slightly modified DPP resolving singletons (i.e. always picking your resolving variable from a clause with only one literal), then you get the Horn algorithm.

When is DPP not efficient?

Below is a sequence of examples where DPP is definitely not efficient. But the proof that it's not polynomial-time is beyond the scope of this book.

One version of the pigeon-hole principle is simply that *there is no injective function* from $\{1, 2, \cdots, n+1\}$ to $\{1, 2, \cdots, n\}$. We use the suggestive notation $p_{i \mapsto j}$ as the name of a propositional variable which might be translated as saying i maps to j. Then the following set of formulae, depending on n, where all integers are positive,

$$\{p_{i\mapsto 1} \vee p_{i\mapsto 2} \vee \cdots \vee p_{i\mapsto n} \mid i \leq n+1\} \quad \cup \quad \{\neg(p_{i\mapsto k} \wedge p_{j\mapsto k}) \mid i < j \leq n+1 \; ; \; k \leq n\} \quad ,$$

would have to be satisfiable if such a function did exist. The set on the left is part of saying it maps from the domain $\{1, 2, \cdots, n+1\}$ to the codomain $\{1, 2, \cdots, n\}$. The one on the right translates as injectivity, that *distinct integers map to distinct integers*.

So one would expect that this set of formulae is unsatisfiable, and this is not hard to prove. (If e is supposed to show satisfiability, then the set on the left requires at least "$n+1$" variables to be true, whereas the one on the right requires at most "n" to be true, giving a contradiction.) It can be shown that, after converting it into a set of clauses, any application of DPP to that set will require more than n^K 'steps', for sufficiently large n, no matter how large the integer K, chosen beforehand, is.

Exercises

4.28 Write down the above formula sets when $n = 2$ and when $n = 3$ (dropping the $\mapsto$ symbols to save ink). Convert each formula set to a set of clauses. Then apply DPP in both cases as cleverly as you can (and perhaps convince yourself that you'd sooner not do the case $n = 4$).

4.29 Show that any proper subset of the one displayed above *is* satisfiable.

A not-quite-Horn example where DPP is efficient.

Recall, from Section **2.3** the following

Challenge. Prove, where the 13 capital letters denote 13 distinct propositional variables, that

$$\{\quad 1.\ P\wedge Q\wedge R\to S\ \ ,\ \ 2.\ \neg P\wedge A\to Z\ \ ,\ \ 3.\ U\wedge T\to\neg S\ ,$$
$$4.\ C\wedge A\to R\ \ ,\ \ 5.\ \neg Q\wedge U\to\neg V\ \ ,\ \ 6.\ \neg S\wedge Q\wedge T\to C\ ,$$
$$7.\ A\wedge\neg W\to X\ \ ,\ \ 8.\ V\wedge A\to Y\ \ ,\ \ 9.\ Y\wedge X\wedge\neg Z\to T\ ,$$
$$10.\ \neg V\wedge U\to Z\ ,\ 11.\ A\wedge Z\to\neg U\ ,\ 12.\ Y\wedge W\wedge P\to T\quad\}\quad\models\quad A\to\neg U$$

Here are two solutions to this.

The first is reminiscent of the Horn algorithm, but, because the set of formulae (or clauses) obtained by appending the negation of the conclusion is not entirely consisting of Horn objects, any of the many solutions along these lines must divide cases once or several times, as we do below. Can you find a shorter one than the following?

We show, as suffices, that no truth assignment e for which the conclusion is false can make all 12 of the premiss formulae true. So we start with $\mathsf{e}(A\to\neg U)=\mathsf{Fs}$, giving $\mathsf{e}(A)$=$\mathsf{e}(U)$=Tr. Now assuming the premisses all true at e, we successively are forced to have the following literals also true:

$\neg Z$ from 11 ; V from 10 ; Q from 5 ; P from 2 ; Y from 8.

Now divide cases according to the value of e on T :

If true, then $\neg S$ from 3 ; $\neg R$ from 1 ; but then we get a contradiction: C from 6 , but $\neg C$ from 4.

If $\neg T$ is true, then $\neg W$ from 12 ; but now also a contradiction: X from 7 , but $\neg X$ from 9.

So that proves it (and uses only a fraction of a page, as opposed to an absolutely brutal truth table)!

The second solution is to do DPP 'intelligently'. A clause set obtained from this argument is

$$\{S, \neg P, \neg Q, \neg R\} \quad , \quad \{Z, P, \neg A\} \quad , \quad \{\neg S, \neg U, \neg T\} \ ,$$

$$\{R, \neg C, \neg A\} \quad , \quad \{\neg V, Q, \neg U\} \quad , \quad \{C, S, \neg Q, \neg T\} \ ,$$

$$\{X, \neg A, W\} \quad , \quad \{Y, \neg V, \neg A\} \quad , \quad \{T, \neg Y, \neg X, Z\} \ ,$$

$$\{Z, V, \neg U\} \ , \quad \{\neg U, \neg A, \neg Z\} \ , \quad \{T, \neg Y, \neg W, \neg P\} \quad , \quad \{A\} \quad , \quad \{U\}$$

Now we can do seven successive resolution steps, on variables in the order they occurred in the first half of the previous solution; that is, A, then U, then Z, then V, then Q, then P , then Y. These produce no mess, because in each case, a singleton clause occurs involving the resolving variable, and all the other occurrences of that variable are the opposite literal. So, at each resolution step, all that happens is that one singleton clause disappears, and one literal is removed from a number of the other clauses. This quickly gets us down to the following set of clauses

$$\{S, \neg R\} \ , \ \{R, \neg C\} \ , \ \{\neg S, \neg T\} \ , \ \{C, S, \neg T\} \ , \ \{X, W\} \ , \ \{T, \neg X\} \ , \ \{T, \neg W\}$$

Finally, resolution, in that order, on T, R, W and C produces

$$\{\neg S, \neg X\} \ , \quad \{S, \neg X\} \ , \quad \{\neg S, X\} \ , \quad \{S, X\} \quad ,$$

which is clearly an unsatisfiable set, as required. Written out in detail, this solution perhaps doesn't satisfy the original challenge—to squeeze a solution onto at most one page. However this *is* an example where resolution is very quick. (I won't say *efficient*, because that usually refers to an algorithm's behaviour on a whole infinite class of possible inputs.)

5. 1ST ORDER LANGUAGES

As a means of communication, discovery and codification, no formal language can compete with the mixture of mathematical argot and formulas which is common to every working mathematician.
However, because they are so rigidly normalized, formal texts can themselves serve as an object for mathematical investigation. The results of this investigation are themselves **theorems of mathematics**. *They arouse great interest (and strong emotions) because they can be interpreted as* **theorems about mathematics**. *But it is precisely the possibility of these and still broader interpretations that determines the general philosophical and human value of mathematical logic.*

Yu. I. Manin

To introduce the next four chapters on 1st order logic, we'll begin with a few examples from primitive mathematics. The set of *natural numbers*, $\{0, 1, 2, 3, \cdots\}$, is denoted $\mathbf{N}$. (N.B. You may have been told in secondary school that 0 isn't a natural number. Let's just say that the set of positive integers is the 'high school teacher's natural numbers'. Don't forget that $0 \in \mathbf{N}$ here!) Let's write "n.number" to be brief.

The following statement is resoundingly easy to accept:

$$(3 < 0) \ \vee \ (3 = 0) \ \vee \ (0 < 3) \quad .$$

Acceptance is easy because the statement is still true when 3 is changed to any n.number. Or *is* it still true??

$$(\text{any n.number} < 0) \ \vee \ (\text{any n.number} = 0) \ \vee \ (0 < \text{any n.number}) \quad .$$

No, this is certainly false, since each of its three sub-statements is false. We need to re-phrase it:

$$\text{For any n.number } x, \ (x < 0 \ \vee \ x = 0 \ \vee \ 0 < x) \quad .$$

That's what we meant to say, and this last sentence is certainly true. (More often than not, a mathematician will say "for every" rather than "for any".) The displayed statement, written independently of the phrase "n.number", in the language we are about to introduce, is the following formula.

$$\forall x \ (x < 0 \ \vee \ (x \approx 0 \ \vee \ 0 < x))$$

(We have gone back to leaving in the "associativity brackets", but removed a pair of outside brackets.)

A statement like this can be translated to reasonably readable English. For example,

"*Given any n.number, if non-zero, it is either less than or greater than zero.*"

Pronouns (such as "it" above) can play the role of variables in natural languages such as English.

The statement above is not only true in the interpretation **N**, but also in **Z**, the set of *all integers*, and in **R**, the set of *real numbers*. Of course, in **N**, the stronger statement

$$\forall x\ (x \approx 0\ \vee\ 0 < x)$$

is true (though it is false in both **Z** and **R**).

The statement with which we started at the top could be written in 1st order number theory (with some abbreviations) as the following formula:

$$(1+1)+1 < 0\ \vee\ (1+1)+1 \approx 0\ \vee\ 0 < (1+1)+1$$

We won't have the symbol "3" as actually part of the language, but we will have brackets and the symbols $<$, $\approx$, $+$, 0 and 1 (as well as $\times$, which is normally interpreted as multiplication).

Since

$$\forall x\ (x \approx 0\ \vee\ 0 < x)$$

is true in **N**, but false in **Z** and in **R**, truth isn't just a property of a sentence on its own. In all three interpretations, the symbol $<$ is to be given the usual meaning, and the symbol $\approx$ is always to be interpreted as equality.

In a similar but somewhat opposite vein, consider the following statement.

$$(1 = 3)\ \wedge\ (1 < 3)$$

This is certainly false, but again it seems so not because of anything special about "3", but for a more general reason. That is, it is impossible to find some n.number to substitute for "3" and make it true. But again, if we wrote

$$(1 = \text{some n.number})\ \wedge\ (1 < \text{some n.number})\ ,$$

this would not be the more general false sentence we wished to write down. That one is true. What we want is the following:

$$\text{For some n.number } x,\ (1 = x\ \wedge\ 1 < x)\ .$$

This one is false, and by specializing, it follows that the earlier one involving "3" is also false. In our 1st order number theory language below, the last sentence would be written as the following formula:

$$\exists x\ (1 \approx x\ \wedge\ 1 < x)$$

This is false in all three of our interpretations, of course. However, it would be possible to invent a rather artificial interpretation in which it is true—we are not forced to interpret "$<$" in any familiar way. But that's for later.

The phrase "there exists a n.number x such that" is often used, rather than "for some n.number x", as the translation of "$\exists x$" . Of course "integer" or "real number" can be used (or nothing at all) in place of "natural number", depending on which interpretation the translators have in mind (but they might be uncommitted to any particular interpretation, so simply "there exists x such that" is fine).

Contrasts have emerged between our new 1st order languages below and the language of propositional logic from the first four chapters.

Firstly, we'll be dealing with the *quantifiers* $\forall$ and $\exists$. Their use can be a little bit subtle. From our earlier example, we see that there are big differences between

$$(\forall x\ F) \vee (\forall x\ G) \vee (\forall x\ H) \qquad \text{and} \qquad \forall x\ (F \vee G \vee H)\ ;$$

and also, from the later example, between

$$(\exists x\ F) \wedge (\exists x\ G) \qquad \text{and} \qquad \exists x\ (F \wedge G)\ .$$

Exercise 5.1 Do $\forall x(F \wedge G)$ and $(\forall xF) \wedge (\forall xG)$ 'really mean the same'? (Yes or no.) Same question for : $\exists x(F \vee G)$ and $(\exists xF) \vee (\exists xG)$?

Secondly, formulae in 1st order languages get truth values without any *truth assignment* being specified. But an *interpretation* (such as $\mathbf{N}$, $\mathbf{Z}$, or $\mathbf{R}$)

must be specified, and the truth value of a given sentence may differ from one interpretation to another.

We shall also have formulae which are not sentences, because they have 'free' variables not 'bound' by any quantifier, and these will often be neither true nor false in an interpretation. For example, $x < (1 + 1) + 1$ is such a formula. It 'puts a condition on x', or 'defines a property of x', but, in itself, is neither true nor false in $\mathbf{N}$. If we preceded it by $\exists x$, it would be changed into a sentence which is true in the interpretation $\mathbf{N}$. If we preceded it by $\forall x$, it would become a sentence which is false in that interpretation.

Truth will be dealt with informally in this chapter and more carefully in the next. Then in Chapter 7, just as with propositional logic, we'll have a proof system and a completeness theorem which tells us that 'what can be derived is exactly what is true'.

Before starting more carefully on the six or seven special $1^{\underline{\text{st}}}$order languages that we'll use for illustration, and on the case of a general $1^{\underline{\text{st}}}$order language, here are a few more examples concerning subtleties related to quantifiers.

The first is for readers who have seen the basic definitions in calculus. Occasionally, when speaking, we might sloppily say that the definition of $\lim_{x\to 0} f(x) = 0$ is something like

"there is a $\delta > 0$ with $|f(x)| < \epsilon$ for every $|x| < \delta$, for any $\epsilon > 0$ whatsoever."

The trouble with this is that it isn't clear whether some fixed δ works for every ϵ (NOT what's intended), or whether δ possibly depends on ϵ. The latter is certainly what is intended—usually, decreasing ϵ forces you to decrease δ. So we should have put "for any $\epsilon > 0$" at the very beginning of the sentence (**not** at the end, **nor** after "there is a $\delta > 0$"). The sentence

$$\exists\delta\forall\epsilon\forall x\ (|x| < \delta \to |f(x)| < \epsilon)$$

(which is *not* the definition) has very different meaning than

$$\forall\epsilon\exists\delta\forall x\ (|x| < \delta \to |f(x)| < \epsilon)\ ,$$

(which *is* the definition). The earlier 'wrong' definition would force $f(x) = 0$ for all x in some interval around 0. Here we are assuming ϵ and δ to range over positive real numbers. And the above two displays, as written, are

actually not quite in any of the formal 1$^{\underline{\text{st}}}$order languages which come up in this chapter. (See Exercise 5.12.) An **abomination** such as

$$\exists\delta\ (\forall|x| < \delta \rightarrow |f(x)| < \epsilon)\ , \forall\epsilon$$

has to be considered totally unacceptable, for at least two reasons.

This last discussion should convince you of the need to **always put a quantifier at the beginning of the subformula to which it applies** (sometimes referred to as its 'scope'). That doesn't necessarily mean at the very beginning of the formula. For example

$$x \approx 0\ \rightarrow \forall y\ x \times y \approx 0$$

is a fine formula (though not a sentence because of the free occurrence of x). It translates as: If x is zero, then x times any number is also zero. Notice that x must be part of the translation because it occurs freely.

Despite not being a sentence, this last formula does happen to have a definite truth value (namely true) in each of the three interpretations considered so far. But there is nothing to stop us from inventing a fourth interpretation where it is false; that is, displaying a set of objects, together with an 'artificial' multiplication and choice of element to interpret zero, for which that element times at least one element isn't that element. For example, take a two-element set $\{f, g\}$, specify f to be the zero element, and specify the product of f with itself to be g. To completely interpret the language, you'd also have to say what other products were, what sums were, and give a relation to interpret the symbol " $<$ ", as well as saying which of the two elements interprets the symbol "1". Chapter 6 is where interpretations and truth are discussed in general.

Returning to discussing just the language, note that **the order in which quantifiers are placed makes a big difference.** An example, simpler than the 'calculus' one, is the following in 1$^{\underline{\text{st}}}$order number theory:

$$\forall x \exists y\ x < y$$

is certainly true in $\mathbf{N}$, but

$$\exists y \forall x\ x < y$$

is false, since $\mathbf{N}$ has no member which is larger than everything (including itself!).

Next, it is clear from the following examples that the *existential* quantifier, $\exists$, can be 'expressed in terms of' the *universal* quantifier $\forall$: the sentence

$$\exists x\ x < 0$$

says in **Z** that *there is at least one negative integer*. (True there, but false in **N**.) A more awkward way of saying the same thing is that *not every integer is non-negative*, which translates to $1^{\underline{\text{st}}}$order as

$$\neg\forall x\neg\ x < 0\ .$$

You'll find that in every case, a similar conversion can be done. We shall exploit this fact to make our language more economical for theory by not taking $\exists$ as a basic symbol, but rather defining $\exists x$ to be an abbreviation for $\neg\forall x\neg$. This is analogous to how we didn't take the three famous connectives $\vee, \rightarrow$ and $\leftrightarrow$ as basic symbols in propositional logic. In a treatment where $\exists$ and $\forall$ are both taken as part of the basic language, the above fact comes out as the statement that the two formulae $\exists xF$ and $\neg\forall x\neg F$ are equivalent.

Symmetrically we'll have the fact that $\forall xF$ and $\neg\exists x\neg F$ are equivalent, once the word "equivalent" is given a precise definition with respect to $1^{\underline{\text{st}}}$order languages. We'll have some cool rules, such as the above : **pushing a "¬" past a quantifier means you must switch to the other quantifier to get an equivalent formula**. To illustrate, the sentences

$$\exists x\forall y\ x \times y \approx y\ ,$$

$$\neg\forall x\neg\forall y\ x \times y \approx y$$

and

$$\neg\forall x\exists y\ \neg x \times y \approx y$$

all say the same thing. They will turn out to be equivalent in the technical sense given later. The first one translates in **N** to say that *there is a n.number which, when multiplied into any n.number, produces that latter n.number.* It is true in **N**, because "1" is a n.number. The third one says: *it is not the case that for every n.number x, you can find a n.number y such that the product of x and y is different from y.* I'll leave it for you to try a literal translation of the second one. One point to note here is that there really is no single way to translate a given sentence in either direction.

Also, our translation of the third one just above leaves something to be desired, since **you should always try to translate a SENTENCE**

without using the bound variables x or y etc. as part of your math/English jargon. (However, in translating a non-sentence formula, the 'free' variables must come into the math/English. Also, if a $1^{\underline{\text{st}}}$order sentence has many quantifiers, it becomes virtually impossible to satisfy the desideratum above; there just ain't enough natural pronouns!)

The words to be used depend also on the interpretation that you have in mind, so that, if it's **N**, you'd say things like "for all n.numbers" or "for any n.number"; and similarly, "for at least one n.number" or "there exists a n.number". When you have **R** in mind, it would be "for all real numbers", etc. For example, the sentence

$$\forall x \forall y (x < y \to \exists z \ (x < z \wedge z < y))$$

says in **Z** that *between any two distinct integers there is a third integer* (definitely false!); whereas it says in **R** that *between any two distinct real numbers there is a third real number* (definitely true!).

Notice also how this translation depends on knowledge beforehand that you can always choose and name two distinct numbers so that the first is less than the second. Some would even quarrel with the correctness of the above translation, and insist on something more literal, such as : *For any two numbers, if the first is less than the second, then there is a third number which is less than the second, but greater than the first.* And to translate the English sentence : *Between any two distinct numbers there is a third number*, they would prefer

$$\forall x \forall y (\neg x \approx y \to \exists z ((x < z \wedge z < y) \vee (y < z \wedge z < x))) \ .$$

Certainly, translation is more an art than a science.

As a more trenchant illustration of this, and to introduce the question of whether there are genuine mathematical statements which *cannot* be translated to $1^{\underline{\text{st}}}$order, consider the following:

There are infinitely many even natural numbers.

If we are permitted to assume the fact that any set of n.numbers, which has a member larger than any given n.number, is an infinite set (that is, *not bounded above implies infinite*), then the following is a perfectly good translation :

$$\forall x \exists y \ x < (1 + 1) \times y \ .$$

In any case, a strict translator would say that this formula says that *there is at least one even n.number larger than any given n.number*. The trouble is that there is no way in $1^{\underline{\text{st}}}$order number theory to express the phrase "infinitely many numbers", and, in particular, no way to say that being bounded above and being finite are the same condition for sets of natural numbers. (They're not the same for sets of integers, nor for sets of real numbers.) Dealing with this type of question is very interesting, but beyond the scope of this book. See however Exercises 7.12 and 7.13.

As an exercise, you could express the first two relations below as $1^{\underline{\text{st}}}$order formulae. But don't necessarily expect to express the third/fourth convincingly.

Strictly between the n.numbers x and y, there are exactly two n.numbers.

Strictly between the n.numbers x and y, there are exactly three n.numbers.

Strictly between the n.numbers x and y, the number of n.numbers is a n.number.

Strictly between the n.numbers x and y, the number of n.numbers is finite.

Standard exercises can be found at the end of the next section (in which we define precisely some words used intuitively in this one). Those exercises will give you practice translating in both directions.

5.1 The language of $1^{\underline{\text{st}}}$order number theory.

Below, we shall painstakingly set up this language (from which many examples, some abbreviated, occur in the previous section). A slightly sloppy summary may be given rather quickly :

All **formulae** can be built up using $\neg F$, $F \wedge G$ and $\forall x F$ from smaller formulae F and G and variables x, starting with **atomic formulae**, which all have the form $t \approx s$ or $t < s$, where t and s are so-called **terms**, built up using $+$ and $\times$ from 0, 1 and variables.

Now let's do this carefully.

In this language, as in all $1^{\underline{\text{st}}}$order languages, the *symbols* come in two

sequences.

Firstly, there is **the infinite sequence of symbols common to all $1^{\underline{\text{st}}}$order languages**:

$$\forall \quad \approx \quad \neg \quad \wedge \quad) \quad (\quad x_{|} \quad x_{||} \quad x_{|||} \quad x_{||||} \quad \cdots$$

As we did with propositional variables, we'll immediately revert to $x_1, x_2, x_3, \cdots$ for the symbols from the 7th onwards. They are called *variables*. Other abbreviations for the first six variables are x, y, z, u, v and w. These letters are also used when we want to refer to one or a few variables in general, in a theoretical discussion. (Although it is not essential for merely describing the language, you should keep in mind that the variables here will correspond to 'things', such as numbers, when interpreted; they don't correspond to statements, as did the propositional variables from the earlier chapters.)

Secondly, there is **the sequence of symbols special to the particular language** (often a finite sequence, as it is in this case) :

$$0 \ \ (cs) \ ; \ \ 1 \ \ (cs) \ ; \ \ + \ \ (\text{2-ary } fs) \ ; \ \ \times \ \ (\text{2-ary } fs) \ ; \ \ < \ \ (\text{2-ary } rs) \ .$$

The designation (cs) is to tell you that these two symbols are *constant symbols*. The designation (2-ary fs) is to tell you that those two symbols are 2-ary (or binary) *function symbols*. The designation (2-ary rs) is to tell you that the symbol $<$ is a 2-ary (or binary) *relation symbol*. These designations will play a role in the rules below for creating terms and formulae.

The simplest definition of the word "term" is as follows :

(i) Any constant symbol or variable is a term.

(ii) If t and s are terms, then $(t + s)$ and $(t \times s)$ are terms.

(iii) Any term is obtained by finitely many applications of (i) and (ii).

As with the definition of *formula* below and in propositional logic, this can be mildly criticized in that some babbling is needed to justify excluding, for example, the infinite string $(1 \times (1 \times (1 \times \cdots\cdots \quad \cdots\cdots)))))$ from being a term.

Sometimes this is settled by saying that the set of terms is the *smallest* set which satisfies (i) and (ii). But then you need to think about why there should *be* a smallest such set. And in any case, it seems preferable not to get something as basic and finitistic as this entangled with set theory, especially

for readers who want to apply logic to the foundations of mathematics and don't have the experience to be sure they're not going round in circles here.

Another, more satisfactory way to settle this is to add to (i), (ii) and (iii) the requirement that a term is a *finite* string of symbols (and not empty).

I shall make the **official definition of "term"** as follows.

A *term* is a finite horizontal string of symbols which is the lowest line in a finite vertical array, each line of which satisfies *one of* the following.

(i) The line consists of a single symbol which is a constant symbol or a variable.

$(\text{ii})_+$ The line is obtained by placing the symbol + between two lines higher up in the vertical array, and enclosing this within a pair of brackets.

$(\text{ii})_\times$ The line is obtained by placing the symbol $\times$ between two lines higher up in the vertical array and enclosing this within a pair of brackets.

So a term is something that will correspond to an object (a n.number, an integer, a real number, etc.) when we talk about interpreting this particular language. (In a $1^{\underline{\text{st}}}$order language of geometry such as in the final section of this chapter, terms would be interpreted as points usually, sometimes lines as well.) A term plays the role of a *noun* or *pronoun* in the language, not a whole statement or assertion. In this particular $1^{\underline{\text{st}}}$order language, of number theory, a term is intuitively just any expression built up from 0, 1 and the variables by applying + and $\times$ as often as needed. For example, $(((1 + (x_3 \times 0)) \times x_{33}) + ((0 + 1) + 0))$ is a term.

Exercise 5.2 Write down a *term verification column*, as described in our official definition, for the term just above.

The answer will be a column having that string at the bottom and having every line satisfy one of the rules. Similar to what we've seen with both formula verification columns and derivations in propositional logic, deleting consecutive lines from the bottom of a term verification column still leaves a term verification column, so every line in such a column is a term. Later in the case of a general $1^{\underline{\text{st}}}$order language, we'll give a few explicit algorithms for checking whether a string of symbols is a term. See **Appendix B** to Chapter 8.

Often we can use the classical algebra convention, that 'multiplication

precedes addition', by using brackets with $+$, but not with $\times$. But it is better not to do this right at the beginning. The outside brackets can also be dropped.

Now we can write out the **rules for creating formulae**:

(i) For any terms t and s, the strings $s < t$ and $s \approx t$ are formulae, called *atomic formulae*.

(ii) For any formula F, the string $\neg F$ is also a formula.

(iii) For any formulae F and G, the string $(F \wedge G)$ is also a formula.

(iv) For any formula F and any variable x, the string $(\forall x F)$ is also a formula.

Then we could speak about only using these rules finitely often, or the smallest set satisfying these rules; then criticise these as being slightly distasteful; and finally give the official definition via formula verification columns. Since you've seen this both for propositional formulae and for terms (and also the definition of *derivation* is not entirely dissimilar), there is no need to write all this out.

The preceding section contains plenty of examples of formulae from 1st order number theory. But several of them are actually abbreviated formulae.

Exercise 5.3 Write down *formula verification columns* for these formulae.

Let's now write out something which is almost a verification column for the second last (abbreviated) formula from the introductory section.

$x \approx y$
$\neg x \approx y$
$z < x$
$y < z$
$(y < z \wedge z < x)$
$x < z$
$z < y$
$(x < z \wedge z < y)$
$((x < z \wedge z < y) \vee (y < z \wedge z < x))$
$(\exists z((x < z \wedge z < y) \vee (y < z \wedge z < x)))$
$(\neg x \approx y \rightarrow (\exists z((x < z \wedge z < y) \vee (y < z \wedge z < x))))$
$(\forall y(\neg x \approx y \rightarrow (\exists z((x < z \wedge z < y) \vee (y < z \wedge z < x)))))$
$(\forall x(\forall y(\neg x \approx y \rightarrow (\exists z((x < z \wedge z < y) \vee (y < z \wedge z < x))))))$

The first thing to note is that the abbreviations "$\vee$" and "$\rightarrow$" from propositional logic are used here. They are defined exactly as in propositional logic, so no more needs to be said. The same goes for "$\leftrightarrow$".

Secondly, we have used the quantifier "$\exists$", which is not part of the language. As explained in the preceding section, $\exists z$ is defined to be an abbreviation for the string $\neg\forall z\neg$.

Thirdly we have used x, y and z as variables. You can either think of them as nicknames for the variables with other names x_1, x_2 and x_3, or as names for any three distinct variables you wish, so that the last line is a name for lots of formulae, one for each choice of x, y and z. This is a very minor point.

Finally you will notice that there are three extra pairs of brackets on the bottom formula, as compared to its version in the introductory section. They are the ones where the " (" appears just to the left of the quantifier. I have avoided all bracket abbreviations in the above display. (N.B. In this book, we do **not** use brackets around an atomic formula.) Normally we would use our previous convention, and drop the pair of brackets on the outside of a formula. That explains one of the three, the outermost pair of brackets, because we shall also apply this convention to rule (iv) on building a formula which begins with a quantifier. As for the other two pairs of brackets,

in the case of a subformula beginning with a quantifier, we go further and **often do not restore the outside brackets on a formula beginning with a quantifier**, when it is used as a subformula within a bigger formula.

This is a great aid to readability, and never results in any ambiguity. This convention explains the non-appearance of the two pairs of brackets surrounding the subformulae beginning with $\forall y$ and $\exists z$ in writing this formula in the previous section, and their re-appearance in the last line and some earlier ones just above. We could have built this abbreviation right into the language from the beginning, by writing $\forall x F$, rather than $(\forall x F)$, in rule (iv) defining *formula*. But sometimes, for readability, leaving the brackets around such a subformula is best, as in the third line below. Consider:

$$\forall x \exists y (\neg x \approx y \wedge x < y) \quad = \quad (\forall x (\exists y (\neg x \approx y \wedge x < y))) \; ;$$

$$\forall x \exists y \neg (x \approx y \wedge x < y) \quad = \quad (\forall x (\exists y \neg (x \approx y \wedge x < y))) \; ;$$

$$\forall x \exists y \neg x \approx y \wedge x < y \;\; = \;\; (\forall x \exists y \neg x \approx y) \wedge x < y \;\; = \;\; ((\forall x (\exists y \neg x \approx y)) \wedge x < y) \, .$$

On the right end of each line we have three actual formulae (all different), or at least we would have 'actual' formulae as soon as x and y are changed to their real names, and $\exists$ is replaced by its definition in terms of $\forall$. On the left of the top and middle lines are the more readable abbreviations obtained by applying the bracket abbreviations above for both quantifiers. On the left of the bottom line we've applied both quantifier bracket abbreviations, as well as removing the outside brackets for this formula, which, in the last step, is built by a conjunction. But I think the middle formula is the most easily readable of the three on the bottom, by avoiding one of the two quantifier bracket abbreviations. In any case, a mathematician would probably rewrite it as the equivalent formula

$$x < y \;\; \wedge \;\; \forall x \exists y \neg x \approx y \; ,$$

and then rename the bound variables, as we do two paragraphs below.

To summarize,

the number of pairs of brackets used in a completely abbreviated formula is:
the total number of connectives $\wedge, \vee, \rightarrow$ and $\leftrightarrow$;
minus 1 (the outside pair) if the formula is built using one of these (as opposed to being atomic or being built using $\neg$ or a quantifier);
minus the number of bracket pairs erased from subformulae $(F \wedge G)$ or $(F \vee G)$ using the precedence conventions saying that $\wedge$ and $\vee$ are 'stickier' than $\rightarrow$ and $\leftrightarrow$.

Sometimes a few extra pairs will be left in, surrounding subformulae built using a quantifier, if that makes it more readable.

The three displayed formulae above, particularly the third line, also give examples for the next topic : **bound vs. free variables, scope, and sentences vs non-sentence formulae**. There is a major qualitative difference between the first two and the bottom lines, quite independent of 'abbreviation theory'. The first two are sentences, because every occurrence of either variable is bound (or governed) by a quantifier. On the bottom line, the rightmost occurrences of the variables x and y have nothing to do with the quantifiers, or with the earlier occurrences. This last formula is not a sentence, whereas the first two are, in the sense defined precisely below. In fact, it would probably have been better to have changed the last formula to

$$(\forall u \exists v \neg\ u \approx v\ ) \wedge\ \ x < y\ \ ;$$

or, even better,

$$x < y\ \ \wedge\ \ \forall u \exists v \neg\ u \approx v \quad .$$

These are different formulae, but ones which have exactly the same meaning as the original.

To explain rigorously about the two ways in which variables are used, *bound* and *free*, we need some definitions.

Definitions. Every occurrence of any variable **in an atomic formula** is a **free** occurrence. Then, inductively:

The free occurrences of a variable in $\neg F$ are its free occurrences in F;

The free occurrences of a variable in $F \wedge G$ are just its free occurrences in F together with its free occurrences in G;

The free occurrences of a variable other than x in $\forall x F$ are just its free occurrences in F; the variable x has no free occurrences in $\forall x F$.

The five statements above determine all the free occurrences of every variable in any formula.

The fact that one single variable might occur freely many times in a formula should not be misunderstood to mean that 'they are free of each other'! When we do so in Chapter 6, all such free occurrences of a single variable would have to be specialized to the *same* element in an interpretation. But we are 'free' to make that element any one we please. And the variable is 'free' of being governed by a quantifier.

Every occurrence of a variable which is not a free occurrence is said to be a **bound occurrence**.

A **sentence** is a formula in which **every occurrence** of every variable is **bound**; that is, no occurrences are free.

The **scope** of a given occurrence of "$\forall$" in a formula is, roughly speaking, the set of all occurrences (of the relevant variable) which it 'quantifies'. To define this precisely, let G be a formula, and consider a particular occurrence of "$\forall$" in G. By the inductive definition of *formula*, there is a unique subformula starting with that occurrence which we may write $\forall xF$. (This is dealt with generally and rather more carefully in the third and fourth sections of Appendix B to Chapter 8.) The *free* occurrences of x in F (which are therefore bound occurrences in $\forall xF$ and in G) are said to be *occurrences of x which are within the* **scope** *of the quantifier $\forall$ occurring at the left end of that string $\forall xF$*. By convention, the occurrence of x immediately after the "$\forall$" in $\forall xF$ is the only other occurrence within the scope of that quantifier occurrence. This then makes every bound occurrence of any variable have a unique quantifier occurrence whose scope it is within. (Exactly the same applies to $\exists xF$, using it as an abbreviation for $\neg\forall x\neg F$.)

For example, in the strange formula

$$(x < 0 \ \wedge \ (\forall x \ (0 \approx x \ \wedge \ (\forall x \ 0 < x)))) \ ,$$

which we would normally abbreviate as

$$x < 0 \ \wedge \ \forall x \ (0 \approx x \ \wedge \ \forall x \ 0 < x) \ ,$$

the leftmost occurrence of x is free, its middle occurrence is bound and within the scope of the first "$\forall$", and its rightmost occurrence is bound and within the scope of the last "$\forall$". This example shows that not all occurrences of x

in $\forall x F$ are necessarily within the scope of the "$\forall$" at its left-hand end—there might be another "$\forall x$" within F itself, though that would be an unusual way for a mathematician to write things.

A diagrammatic method to display free and bound variables, together with scopes, is illustrated on the previous example as follows.

$$\underset{\text{free}}{x} < 0 \ \wedge \ \forall x \ (0 \approx x \ \wedge \ \forall x \ 0 < x)$$

This leads to a brief diversion on **'strange but okay' formulae**. The one above would normally be rewritten by a mathematician as

$$x < 0 \ \wedge \ \forall y \ (0 \approx y \ \wedge \ \forall z \ 0 < z) \ ,$$

which is a different formula, but has the same meaning. Similarly, but stranger, we have perfectly acceptable formulae

$$\forall x \ y < z \qquad \text{and} \qquad \forall x \forall x \forall x \ 0 < x \ \ .$$

We could circumscribe the language to get rid of these formulae, but they are harmless, and doing so would just make the theory much messier. However, their existence does make it even more important for us to give quite precise rules for deciding about truth in an interpretation, validity of an argument, etc., as we do in the next chapter. One doesn't bother translating—assigning meaning—or whatever, to **'silly' formulae** like these.

To finish this section on the language of number theory, let's do some more informal translating both ways, finding meaning and truth (but not like a guru sitting cross-legged on top of a mountain!), expressing concepts, particularly with reference to the difference between sentences and non-sentence formulae.

Here is very important general point, which you probably have noticed, and which the examples below will reinforce :

A sentence translates as a definite statement.

A formula with one free variable x translates as a property of x.

A formula with two free variables x and y translates as a relation between x and y.

A formula with three free variables translates as a relation between those three—and so on—

For example, the relation of one integer dividing another is easily expressed. We'll write it as a relation between *terms*, not just variables.

Definition of "divides". If t and s are terms in the language of $1^{\underline{\text{st}}}$order number theory, the string "$t|s$" and the phrase "t divides s" will be **short forms** for any formula as follows in which the variable z occurs in neither t nor s :

$$\text{“}t|s\text{”} \quad = \quad \exists z \; s \approx z \times t \; .$$

We shall never in this book 'substitute' into short forms like this. It is possible to give rules for doing so, but they are complicated and can be confusing, since care needs to be taken with examples such as the following. The formula

$$\forall x \; \text{“}x|(y+1)\text{”} \quad = \quad \forall x \exists z \; (y+1) \approx z \times x \; ,$$

is a property of y, and neither true nor false in $\mathbf{Z}$. If we broke our vow and substituted x for y, it would become, the false (in $\mathbf{Z}$) sentence

$$\forall x \; \text{“}x|(x+1)\text{”} \quad = \quad \forall x \exists z \; (x+1) \approx z \times x \; .$$

Short forms should always be formulated using ingredients which are general terms, not just variables, so there is no temptation to substitute. Then **the only care we need to take, when converting any short form (which has particular terms) into an actual abbreviated formula, is to select bound variable(s) not occurring in any term which is part of the short form.** Different choices for these bound variables do give different formulae. But in the sense made precise in Chapter 8, all these different formulae are truth equivalent to each other. And that is all that we really want to define here: a formula 'up to equivalence'. Wherever these are used, it makes no difference which variable(s) are used, as long as the proviso above is not violated.

Using this short form for divisibility, the little theorem that the divisibility relation is transitive, namely

$$(\; t|s \quad \text{and} \quad s|r \;) \quad \text{implies that} \quad t|r \; ,$$

is easily translated into an abbreviated $1^{\underline{\text{st}}}$order sentence.

Exercise 5.4. Do this translation.

Definition of "prime". Now we can express in the $1^{\underline{\text{st}}}$order number theory language the short form "t is a prime", where t is any term:

$$\text{"}t \text{ is a prime"} \quad = \quad 1 < t \ \wedge \ \forall x((1 < x \ \wedge \ x < t) \rightarrow \neg \text{"}x|t\text{"}) \quad =$$

$$1 < t \ \wedge \ \forall x((1 < x \ \wedge \ x < t) \rightarrow \neg \exists z \ t \approx z \times x) \ .$$

This last abbreviated formula is a version of "t is a prime" for any variables z and x which are distinct and do not occur in the term t. (They are forced to be distinct by the earlier requirement for "$x|t$", which forces z to occur in neither of the terms x nor t.)

Notice, in the first line displayed, how we can use a short form, namely "$x|t$", right in the middle of what would otherwise be a $1^{\underline{\text{st}}}$order formula (or an abbreviated one). This is an indispensible tool, if we wish to show that interesting mathematics can be expressed in $1^{\underline{\text{st}}}$order languages (in a readable manner). It becomes an easy programming exercise to convert an expression containing short forms and abbreviations into an actual $1^{\underline{\text{st}}}$order formula. But that formula will often be almost unreadable by a normal human being. (For a striking example of this, look ahead to Section **5.6**, the one on $1^{\underline{\text{st}}}$order set theory. You'll have no trouble finding the formula to which I'm referring—it sticks out like two sore thumbs!) The important thing is to be able to establish that any mathematical statement can be converted into a formula in some $1^{\underline{\text{st}}}$order language, not to actually do the conversion explicitly, even less to try to work directly with the result.

Using short forms like these, many interesting theorems and conjectures about integers can be formulated as $1^{\underline{\text{st}}}$order sentences. And many properties and relations can be formulated as $1^{\underline{\text{st}}}$order formulae which aren't sentences.

For example, the famous theorem of the ancient Greeks that *there are infinitely many primes* can be written as the following $1^{\underline{\text{st}}}$order 'abbreviated short form sentence' :

$$\forall x \exists y(\text{"}y \text{ is prime"} \ \wedge x < y) \ .$$

(Notice that again we have identified a set of natural numbers as being infinite with such a set being not bounded above.)

The property of being a prime and the relation of divisibility were both formulated earlier. The property of a n.number x being **composite** could

be written either as

$$1 < x \ \wedge \ \neg\text{“}x \text{ is a prime”} \quad ,$$

or as

$$1 < x \ \wedge \ \exists y \exists z (1 < y \ \wedge \ 1 < z \ \wedge \ x \approx y \times z) \ .$$

(We omitted one pair of ‘associativity’ brackets.)

Exercises

5.5 Write out the theorem (about the existence of infinitely many primes) in all detail as a $1^{\underline{\text{st}}}$order sentence (with no short forms or abbreviations).

5.6 Show that it is possible to translate each of the following number theory statements as a $1^{\underline{\text{st}}}$order formula, by actually translating each at least as a $1^{\underline{\text{st}}}$order ‘abbreviated short form formula’.

(i) *The natural number x is odd.*

(ii) *All primes except 2 are odd.*

(iii) *There is an infinity of twin primes; that is, primes differing by 2.* (The truth of this is unknown—the Twin Prime Conjecture.)

(iv) *There is an infinity of primes of the form $4n + 3$.* (This is true, and not so hard to prove, but it’s somewhat harder for $4n + 1$.)

(v) *The n.numbers x and y have no common prime factor.*

(vi) *If a and b have no common prime factor, then there is an infinity of primes of the form $an + b$.* (Dirichlet’s Theorem)

(vii) *To be a prime, it suffices that a natural number have no prime factor between 2 and its square root, inclusive.* (This is true, and easy to prove.)

(viii) *All even positive n.numbers except 2 are writeable in at least one way as a sum of a pair of (not necessarily distinct) primes.* (This is unknown—the Goldbach Conjecture.)

(ix) *The sum of two cubes of positive n.numbers is not the cube of any n.number.* (This has long been known, but is also the very first case of Fermat’s Last Theorem, only recently proved. The general case of this theorem,

in which cubing is replaced by an arbitrary exponent larger than 2, can, in fact, be formulated in $1^{\underline{st}}$order number theory. But this is far from obvious, since we don't have a function symbol in this $1^{\underline{st}}$order language which corresponds to exponentiation! To paraphrase **Manin**, p. 14 :

we...find an atomic formula F with free variables $x, t, y, z_1, \cdots z_n$ such that the formula $\exists z_1 \cdots \exists z_n F$ is equivalent to "$x^t = y$". Then $x_1^t + x_2^t = x_3^t$ can be translated as

$$\exists y_1 \exists y_2 \exists y_3 (\text{“}x_1^t = y_1\text{”} \ \wedge \ \text{“}x_2^t = y_2\text{”} \ \wedge \ \text{“}x_3^t = y_3\text{”} \ \wedge \ y_1 + y_2 = y_3) \ .$$

The existence of such an F is a nontrivial number-theoretic fact, so that here the very possibility of performing a translation becomes a mathematical problem.

Manin doesn't use the word "problem" to refer to an exercise in an undergraduate text!)

(x) *Every n.number is a sum of four squares of n.numbers.* (This is known.)

(xi) *An odd prime is a sum of two squares of n.numbers if and only if it has the form* $4n + 1$*; and there are infinitely many such primes.* (This is known.)

(xii) x *is divisible by exactly one prime.*

(xiii) y *is a power of a prime.*

(xiv) x *can be written as a sum of two squares.*

(xv) (x, y, z) *is a pythagorean triple; that is, the sum of squares of the first two is the square of the third.*

(xvi) (abc conjecture) Use the method described by Manin in (ix) to show that the following can be translated to $1^{\underline{st}}$order number theory: *Working with non-zero natural numbers, for any* u*, there is a* v *so that, if* $x + y = z$ *and* $GCD(x, y, z) = 1$*, then, for all* t *divisible by all the primes dividing* xyz*, we have* $z < vt^{1+\frac{1}{u}}$.

Before going on to our other $1^{\underline{st}}$order languages, consider the abbreviated short form $1^{\underline{st}}$order formulae which are answers to the above exercises. Suppose that we then gave them to someone to translate back in to math/English. It would be necessary to also tell that person which interpretation they should be thinking about, so they'd know whether to use "n.number" or "integer" or "real number" or whatever in their translation. The translations back into

math/English will change their meanings depending on which interpretation is intended. In the subject called **number theory**, mathematicians are usually thinking about either **N** or **Z**, though some other interpretations which are **rings** are also of interest. (See Section **5.8**.)

In the language of ring theory, there is no relation symbol "$<$" . So a different definition of "t is a prime" must be used. For example, it could be defined to be short for

$$(\neg \text{“}t \text{ and } 1 \text{ are associates”} \wedge \neg \text{“}t \text{ and } 0 \text{ are associates”}) \wedge$$

$$\forall x(\text{“}x|t\text{”} \rightarrow (\text{“}x \text{ and } 1 \text{ are associates”} \quad \vee \quad \text{“}x \text{ and } t \text{ are associates”})) \ ,$$

where "r and s are associates" is short for

$$\exists x \ (r = x \times s) \ \wedge \ \exists y \ (s = y \times r) \ \ .$$

The latter is the same as $\text{“}s|r\text{”} \ \wedge \ \text{“}r|s\text{”}$.

Notice how this doesn't involve the symbol "$<$". It makes no difference which of the two you choose to define the short form when interpreting in **N**. However, in **Z**, it makes a small difference, in that the negatives of the usual primes become 'primes in **Z**', if the ring theory version is used.

If we were thinking about the interpretation **R**, the above concepts are of very little interest, and mathematicians waste little time thinking about them. As an exercise, you could show that no elements in **R** are primes in **R**, and that any pair of elements of **R** divide each other in **R** with the exception of when the first is zero and the second non-zero. Thus, using the ring theory definition of "t is a prime" immediately above (rather than the original one several pages back), the statement "-13 is a prime in **Z**" is true, whereas the statement "-13 is a prime in **R**" is false.

Note that the abbreviated short form formula

$$1 < t \ \wedge \ \forall x(\text{“}x|t\text{”} \rightarrow (x \approx 1 \ \vee \ x \approx t)$$

works as the definition of "t is a prime" pretty well only when we determinedly stick to just the interpretation **N**.

Exercises

5.7 Determine which of the following are formulae (possibly abbreviated), and which are just strings of symbols. Translate the non-silly formulae into

math/English, assuming the interpretation **N**. For bound variables, try not to use them in the translation—use pronouns like "it" and "them".

(i) $\forall x \exists y(((x \times y) + x) \to x \approx y)$

(ii) $x < x + 1 \to \forall x \forall z(z \times x \approx x \times z \ \wedge \ \exists x \ x + 1 < (x + 1) + 1)$

(iii) $(\forall x \ x \approx x \wedge \exists x) \ x \approx x$

(iv) $\forall x \ x \approx x \wedge \exists x \ x \approx x$

(v) $\forall x \ (x \approx x \wedge \exists x \ x \approx x)$

(vi) $\forall y \exists x((x \times y) + x)$

(vii) $\forall y \ x \approx z \ \to \ \exists z \exists z \ x < y$

(viii) $((\forall x(\exists y((x \times y) + x) < (x \times (y + 1)))) \to 1 \approx 0)$

(ix) $\forall x \forall y(x < y \to x + 1 \approx y \vee \exists z(x < z \wedge z < y))$

(x) $\forall x \forall y(x < y \to (\neg x + 1 \approx y \to \exists z(x < z \wedge z < y)))$

(xi) $\forall x \forall y(x < y \wedge \neg x + 1 \approx y \ \to \ \exists z(x < z \wedge z < y))$

5.8 Translate the following into $1^{\underline{\text{st}}}$order number theory. We are thinking of the interpretation **R**. You will probably want to invent some short forms of your own to use, and if so, use them to show that a translation exists, without necessarily writing it out as an actual formula.

(i) A real number has a real square root if and only if it is non-negative.

(ii) A quadratic polynomial has a real root if and only if its discriminant is non-negative. (Recall that the discriminant of $ax^2 + bx + c$ is $b^2 - 4ac$.)

(iii) The equation $ax + b = c$ has a solution as long as $a \neq 0$.

(iv) The system of equations

$$ax + by = e$$

$$cx + dy = f$$

has a solution as long as the determinant of the coefficient matrix (that is, $ad - bc$) is non-zero.

5.9 Express the relation "z is the GCD of x and y" as a $1^{\underline{\text{st}}}$order formula.

5.10 Determine the scopes of all the quantifier occurrences below, thereby listing which variable occurrences are free and which are bound. State which of the formulae are sentences.

(i) $x < x+1 \rightarrow \forall x \forall z (z \times x \approx x \times z \ \wedge \ \exists x \ x+1 < (x+1)+1)$

(ii) $\forall x \ x \approx x \wedge \exists x \ x \approx x$

(iii) $\forall x \ (x \approx x \wedge \exists x \ x \approx x)$

(iv) $\forall y (x \approx z \rightarrow \exists z \exists z \ x < y)$

(v) $\forall y \ x \approx z \rightarrow \exists z \exists z \ x < y$

(vi) $((\forall x (\exists y ((x \times y) + x) < (x \times (y+1)))) \rightarrow 1 \approx 0)$

Also abbreviate (vi) as much as possible by removing brackets, including the outside brackets on terms, where possible.

5.11 Write down 8 sentences in the language of $1^{\underline{\text{st}}}$order number theory with the following property : *when interpreted in the 3 interpretations* $\mathbf{N}$, $\mathbf{Z}$, *and* $\mathbf{R}$, *they will produce all 8 of the possible true-false combinations.* Explain your answer.
(Quite a bit more tedious would be to write down 27 formulae (mostly non-sentences) which will produce all the possible (true—false—neither) combinations for these three interpretations.)

5.12 Find a verification column for the formula below. It expresses, with some 'Greek variables', the limit from early in the chapter, with $f(x) = x^2$. That is, it's a $1^{\underline{\text{st}}}$order number theory version of the statement $\lim_{x \rightarrow 0} x^2 = 0$.

$$\forall \epsilon (0 < \epsilon \ \rightarrow \ \exists \delta (0 < \delta \wedge \forall x (x < \delta \wedge 0 < x + \delta \ \rightarrow \ x \times x < \epsilon \ \wedge \ 0 < (x \times x) + \epsilon))) \ .$$

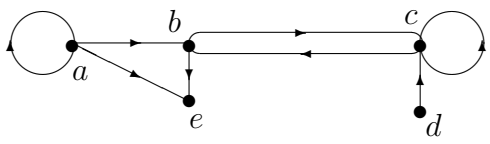
Actually, the last bit, $\cdots \wedge \ 0 < (x \times x) + \epsilon$, could be dispensed with over the reals.

5.2 The language of directed graph theory.

This language will have the same common symbols as every other $1^{\underline{\text{st}}}$order language, but its list of special symbols is much simpler than that of number theory. There are no constant or function symbols, and just one relation symbol, namely r, which is specified to be a *binary* (i.e. 2-*ary*) relation symbol.

A directed graph (each of which is an interpretation of this language) is typically pictured as below. We are imagining any collection of points together with arcs joining (some) points to points, with an arrow on each arc

giving its initial and final points (or 'vertices'). Given a pair of vertices, there is at most one directed edge from the first vertex to the second (possibly the same) vertex.



In this language, we'll write the binary relation as trs, to be read as saying that the term t is related to the term s. But since there are no constant nor function symbols, the only terms are the variables, so above I should have simply written xry, or maybe x_7rx_4.

The variables will be interpreted as the points (or vertices) in any particular directed graph, i.e. in any particular interpretation. The atomic formula xry will have the meaning (once the symbol x is interpreted (say) as the vertex labelled b, and y is interpreted as c) that *vertex b is joined to vertex c by an edge directed from b towards c.* Note that if you wanted to picture yrx as well, you'd draw a second edge between them with an arrow going the other way. And to picture xrx, you'd draw a loop at b.

It should be pretty clear how to specify what is a formula in directed graph theory. Here is a summary:

All **formulae** in 1st order directed graph theory are built up using $\neg F$, $F \wedge G$ and $\forall x F$ from smaller formulae F and G and variables x, starting with **atomic formulae**, which all have the form $x \approx y$ or xry, where x and y are variables.

This turns out to be a bit simpler than 1st order number theory, because the language is simpler. There should be no need here to be more formal than this. In fact, the section after next gives the general specification for a 1st order language, so you can use that to derive the rules for forming terms and formulae in both the special cases of number theory and directed graph theory. But first read the next section, because the general case in the section after that uses **prefix** (or 'partial Polish') notation, whereas this and the previous section use **infix** notation.

To express the relation

"there is a path of length 2 from x to y"

in this language, we have

$$\exists z \ (xrz \wedge zry) \quad .$$

To express the relation

"there is a path of length 3 from x to y"

in this language, we have

$$\exists z \exists w \ ((xrz \wedge zrw) \wedge wry) \quad .$$

By '$\vee$ing' a few of these together, it is not hard to express the relation

"there is a path of length at most 3 from x to y".

And it's clear that similar things can be done with 3 replaced by any fixed positive integer

But, it turns out that the relation

"there is a path from x to y"

cannot be expressed in this language. Results like this are discussed in many books on logic more advanced than this one. However the basic ideas needed for the proof are sketched in **Appendix L**. See also Exercise 7.12.

Exercises

5.13 Show that it is possible to translate each of the following directed graph theory statements as a 1^{st}order formula.

(i) *There is an edge.*

(ii) *For every vertex, there is an edge going from it to some different vertex.*

(iii) *There is a pair of distinct vertices with edges going both ways between them.*

(iv) *There is a directed 'triangle', that is, an edge from one vertex to another, from that one to a third, and from the third back to the first.*

(v) *x has at least three edges coming into it.*

(vi) *For every three distinct vertices, there is at least one edge touching one of them.*

(vii) *x, y and z have edges going between any ordered pair of them.*

(viii) *Every ordered pair of distinct vertices is connected by a path of length at most three.*

(ix) *Some vertex has at least two edges incident on it.*

5.14 Determine which of the following are (possibly abbreviated) formulae in the language of $1^{\underline{\text{st}}}$order directed graph theory . Translate the formulae into math/English.

(i) $\forall x \exists y(xry + yrx)$.

(ii) $\forall x \forall y \forall z \forall w((xry \vee xrz \vee rxr) \rightarrow (y \approx z \vee y \approx w \vee w \approx z))$.

5.15 (i) Say that a vertex is *lonely* if no edge touches it. How would you define in $1^{\underline{\text{st}}}$order the short form "$x$ is lonely"?

(ii) Express the relation "z is joined to both x and y by edges in one or the other direction" in $1^{\underline{\text{st}}}$order.

5.16 Translate the following from the language of $1^{\underline{\text{st}}}$order directed graph theory into math/English.

(i) $\forall x \exists y(xry \vee yrx)$.

(ii) $\forall x \forall y \forall z \forall w((xry \wedge xrz \wedge xrw) \rightarrow (y \approx z \vee y \approx w \vee w \approx z))$.
(We have dropped a couple of associativity bracket pairs here.)

(iii) $\exists x \exists y \neg y \approx x$.

(iv) $\forall x \forall y(\neg y \approx x \rightarrow (xry \vee \exists z(xrz \wedge zry)))$
(By our conventions, the middle bracket pair is unneeded here.)

5.17 (Compare 5.11) Give 3 directed graphs and then write down 8 sentences in the language of $1^{\underline{\text{st}}}$order directed theory with the following property: *when interpreted in the 3 interpretations you just gave, they will produce all 8 of the possible true-false combinations.* Explain your answer.

8.22.5 (This is a preview of a later exercise, easy then, but perhaps fairly challenging now.) Divide the following into collections, with all formulae in each collection having the same meaning. Then, for each collection, provide a single translation into math/English for all the formulae in that collection. (As a hint, you might want to first contrast (v) with (x), and then work your way from top to bottom, comparing each to the next. See Sections **8.2** and **8.3** for related material.)

(i) $\forall x \forall y \forall z (xry \wedge xrz \ \rightarrow \ y \approx z)$.
(ii) $\forall y \forall z \forall x (xry \wedge xrz \ \rightarrow \ y \approx z)$.
(iii) $\forall y \forall z \forall x (\neg y \approx z \ \rightarrow \ \neg (xry \wedge xrz))$.
(iv) $\forall y \forall z (\neg y \approx z \ \rightarrow \ \forall x \neg (xry \wedge xrz))$.
(v) $\forall y \forall z (\neg y \approx z \ \rightarrow \ \neg \exists x (xry \wedge xrz))$.
(vi) $\forall y \forall z (\exists x (xry \wedge xrz) \ \rightarrow \ y \approx z)$.
(vii) $\forall y \forall z \exists x (xry \wedge xrz \ \rightarrow \ y \approx z)$.
(viii) $\forall y \forall z \exists x (\neg y \approx z \ \rightarrow \ \neg (xry \wedge xrz))$.
(ix) $\forall y \forall z (\neg y \approx z \ \rightarrow \ \exists x \neg (xry \wedge xrz))$.
(x) $\forall y \forall z (\neg y \approx z \ \rightarrow \ \neg \forall x (xry \wedge xrz))$.
(xi) $\forall y \forall z (\forall x (xry \wedge xrz) \ \rightarrow \ y \approx z)$.

5.3 Infix versus prefix notation for function and relation symbols.

Adding two real numbers can in general be thought of as applying a function $a^{\mathbf{R}}$ of pairs of real numbers to produce a real number. That is, I'm thinking of defining $a^{\mathbf{R}} : \mathbf{R} \times \mathbf{R} = \mathbf{R}^2 \rightarrow \mathbf{R}$ by $a^{\mathbf{R}}(x, y) = x + y$, surely one of the simplest functions from multivariable calculus. Similarly, we could rewrite multiplication in this style by defining $m^{\mathbf{R}}(x, y) = xy$. Considering our other standard interpretations of the language of $1^{\underline{\text{st}}}$order number theory, there are actual functions (as opposed to function symbols) $a^{\mathbf{Z}} : \mathbf{Z}^2 \rightarrow \mathbf{Z}$ for adding integers, $m^{\mathbf{N}} : \mathbf{N}^2 \rightarrow \mathbf{N}$ for multiplying n.numbers, etc., different from the above only because we restrict to integers and n.numbers.

This would not often be done for the language of $1^{\underline{\text{st}}}$order number theory, but mathematicians sometimes prefer to use a so-called *prefix* notation. They would replace, in the language, the symbol + by a 2-ary function symbol a, and the symbol $\times$ by a 2-ary function symbol m. Then, in defining the word *term*, they would say "if t and s are terms, then ast and mst are terms." Note that no brackets nor commas are used, and the function symbol comes at the front, before the terms 'to which it applies'.

Now when we come to the most general $1^{\underline{\text{st}}}$order theories, we have to allow for 3-ary, 4-ary, etc., and also 1-ary functions. Except for the 2-ary case, this prefix notation is always used. It is nice that no brackets at all are needed. A string of function symbols, constant symbols and variables may or may not determine a term, but if they do, they do so in only one way. See **Appendix B** of Chapter 8, particularly its third section, for much more on this.

The statement in the general case corresponding to the one above to give the inductive definition of *term* is : "If $t_1, t_2, \cdots, t_n$ are terms, and f is an n-ary function symbol, then $ft_1t_2 \ldots t_n$ is also a term."

Similarly, relation symbols can be set up in prefix notation, and especially for non-binary relation symbols, prefix notation is almost always used.

For example, in a $1^{\underline{\text{st}}}$order geometry theory, we might have constant symbols and variables which are interpreted intuitively as points (in a plane, in 3-space, etc.). One of the relation symbols, say b, is 3-ary, with $bxyz$ interpreted as saying that point y is on the line segment joining x to z. So b is the 'in-betweenness' relation symbol. We might also have a 4-ary relation symbol c with $cxyzu$ interpreted as saying that the line segment from x to y has the same length as the line segment from z to u. So c is the 'congruence' relation symbol. In fact, some classical geometry is done that way. See the final section of this chapter.

The logic we do today was partly generated by the successful attempts to understand why Euclid's famous "parallel postulate" could not be proved from the other postulates. This is discussed in detail in **Appendix G** of this book, but from a naive point of view in which the language is not formalized, as we are doing here.

Let's go back to number theory, and imagine that our 'prefix fixated' mathematician above has also decided to replace the binary relation symbol "$<$" by "ℓ", and always write ℓts instead of $t < s$.

Here are a few formulae in his $1^{\underline{\text{st}}}$order number theory language, along with the 'same' formulae in the usual notation, dropping outside brackets on terms :

$$\begin{array}{rcl} \ell axyz & \text{is the same as} & x + y < z \\ \forall u \ell mxayzu & \text{is the same as} & \forall u\ x \times (y + z) < u \\ \forall u \ell amxyzu & \text{is the same as} & \forall u\ (x \times y) + z < u \end{array}$$

Exercise 5.18 Convert each of the following terms into 'prefix notation $1^{\underline{\text{st}}}$order number theory language'.

(i) $((y + (x \times x)) \times (x + (y \times y)))$

(ii) $(0 + (z \times ((z + x) + ((1 \times y) \times y))))$

(iii) $((x \times x) + ((y \times y) + ((z \times z) + (w \times w))))$

You're probably wondering why we haven't used at least the abbreviation that the outside brackets on an infix notation term can (and should!) be removed. This is just to remind you that they'd need to be restored if the above terms occurred as subterms in a larger term, and to emphasize how prefix notation is far more efficient, if less readable.

There is of course also a 'prefix notation 1st order directed graph theory', where every time in the old language you see xry, in the new language you see rxy. This prefix notation looks more natural to most of us than the prefix notation in number theory. It is unclear which of prefix or infix notation is preferrable here.

5.4 Predicate Calculus—or better, the 'most general' 1st order language.

By now you probably have a good idea of what a 1st order language is like in general. The basic data consists of firstly, the infinite sequence of **symbols**—universal quantifier, equality symbol, propositional connectives, brackets, and variables—**which are common** to all such languages. Just to remind you, this sequence is

$$\forall \quad \approx \quad \neg \quad \wedge \quad) \quad (\quad x_{|} \quad x_{||} \quad x_{|||} \quad x_{||||} \quad \cdots$$

But we immediately revert to using the ordinary positive integer symbols as subscripts (licence plates!) on the variables.

Secondly, the particular language is then specified by listing its **special symbols** together with the arities of the function symbols (if any), and of relation symbols (if any). The only other kind of such special symbols are possibly some constant symbols.

The **terms** of the language are now defined inductively by saying the following:

(i) Each variable and each constant symbol is a term.

(ii) For each function symbol f, if its arity is n, then, for any sequence of terms $t_1, t_2, \cdots, t_n$, the string $ft_1t_2 \ldots t_n$ is also a term. (Note that we are using prefix notation.)

(iii) Every term is a finite string of symbols obtained by applying the statements (i) and (ii) a finite number of times.

Finally, the **formulae** of the language can be defined inductively by saying the following:

(i) Each string $t \approx s$ is a formula, for any terms t and s. (Such a formula is called **atomic**.)

(ii) For each relation symbol r, if its arity is n, then, for any sequence of terms $t_1, t_2, \cdots, t_n$, the string $rt_1t_2 \ldots t_n$ is a formula. (Such a formula is also called **atomic**. Note again that we are using prefix notation.)

(iii) For any formula F, the string $\neg F$ is also a formula.

(iv) For any formulae F and G, the string $(G \wedge F)$ is also a formula.

(v) For any formula F and any variable x, the string $(\forall x F)$ is also a formula.

(vi) Every formula is a finite string of symbols obtained by applying some of (i), (ii), (iii), (iv) and (v) a finite number of times.

Note again that we definitely do not use brackets around an atomic formula. Be careful when reading other texts, as the opposite convention is often adopted.

Just as applicable here in the general case are all the remarks from the section on $1^{\underline{\text{st}}}$order number theory concerning term and formula verification columns, concerning abbreviating formulae, both by removing brackets and by using the extra symbols $\rightarrow$, $\leftrightarrow$, $\vee$, and $\exists$, and finally concerning free

and bound variables, formulae in general versus sentences, and the scope of a quantifier. We shall freely use these concepts in all the examples of $1^{\underline{st}}$order languages in the sections to follow.

Sometimes at this point in the discussion of general $1^{\underline{st}}$order languages, you'll find a pile of rather formidable notation consisting of three infinite sequences, of constant symbols, of function symbols, and of relation symbols. There will often be several subscripts and several superscripts attached to each of them. This is then explicit notation for 'the general, abstract (countable) $1^{\underline{st}}$order language', about which many results are proved, such as those in the next three chapters. By specializing somehow, the results then apply to any $1^{\underline{st}}$order language. Phrases such as "**the predicate calculus**" or "pure $1^{\underline{st}}$**order predicate calculus**" are often used when referring to this language, especially in the absence of function and constant symbols or of particular sentences taken as premisses in the deductive theory of that language. This notation is a challenge to wade through, and fortunately we won't need to.

5.5 THE LANGUAGE OF ($1^{\underline{st}}$order) LOVE (theory).

This is a brief, frivolous section, included mainly so you can try the exercises below. The first is reverse translating, from a $1^{\underline{st}}$order language, a famous (mis)quotation. The second will be equally entertaining, I hope.

We are thinking of a language in which the main non-logical symbol is a binary relation symbol, ℓ, using infix notation. The interpretation will always be a set of people, and "ℓ" will always be interpreted as "loves". We shall also throw in a couple of constant symbols, a and b. They can be interpreted as two objects (in the interpretation set) whose usual names are Alice and Bob. We won't get too explicit about the kind of love to which we are referring. Let's use infix notation again, so that $x\ell y$ is translated as saying "x loves y". And $a\ell b$ is vocalized as "Alice loves Bob".

Exercise 5.19 (a) Try to translate each of the following into English in the most elegant way possible. To make it easier, do them in the order given. The final one can come out as a rather famous saying, which has only three words which are longer than one letter. Remember that only *free* occurrences of variables should occur explicitly in your translation. You might want to put a bracket pair around each $\exists z y \ell z$ below to be clearer, but with our conventions we don't need to.

$\exists z a \ell z$; $\exists z y \ell z \ \rightarrow \ x \ell y$; $\forall y(\exists z y \ell z \ \rightarrow \ x \ell y)$
$\forall x \forall y(\exists z y \ell z \ \rightarrow \ x \ell y)$ or $\forall x \forall y \forall z(y \ell z \ \rightarrow \ x \ell y)$
The last two are equivalent (by results in Chapter 8), so they can be given the same translation.

(b) Give a seven word translation, with high amusement value, which begins "None of Alice's ...", of the following:

$$\forall x(\exists y(x \ell y \wedge y \ell a) \ \rightarrow \ \neg x \ell a) \ .$$

Some of you may have noticed that this language would have been the same as the language of directed graph theory, if we hadn't added the two constant symbols. We just changed the name of the relation symbol from r to ℓ. So any directed graph you happen to draw can be thought of as geometrically describing some set of complicated love relationships. In particular, the loops, if any, indicate individuals who love themselves.

On the other hand, the standard Christian theology, and probably that of some other religions, would wish for an ideal world in which the set of all people would give the vertices of a graph in which there were directed edges both ways between any two distinct vertices (and perhaps no loops at all?).

5.6 The languages of $1^{\underline{\text{st}}}$order set theory.

This section is not essential for anything later in this book. You may want to take it fairly lightly on first reading. However, the topic is exceptionally important in the foundations of mathematics. We shall not emphasize this example of a $1^{\underline{\text{st}}}$order language much, for reasons explained mainly in the early paragraphs of the next chapter.

The idea is to try to express, in $1^{\underline{\text{st}}}$order, everything we would ever need when dealing with sets in mathematics. This is clearly of fundamental interest because there is a general belief that all of mathematics can be expressed purely in terms of sets and the 'belonging to' relation (and nothing else).

The main reason for not emphasizing this language much after this section is that it is hard to imagine more than the one interpretation of any real interest, namely that in which the terms are to be thought of as any sets whatsoever. However, talking about the 'set' of all sets leads to some famous paradoxes. So we'll just describe the language, and not get involved with talking about interpretations.

We should understand the need for certain types of self-restraint. However, intellectual asceticism (like all other forms of asceticism) cannot be the lot of many.

Yu. I. Manin

There are at least two slightly different ways to give such a language. In both of them, the language has no function symbols, and just one binary relation symbol, denoted $\in$. Infix notation is used, and $x \in y$ is vocalized as *"x is a member of y"* or *"x belongs to y"* or *"y has x as an element."* Now every term denotes a set, so we are thinking only of sets whose elements are themselves sets. Any interpretation is very 'pure'—unpolluted by any kind of mathematical object other than sets.

One $1^{\underline{\text{st}}}$order set theory language is just that—there are no constant symbols, so $\in$ is the only special symbol, and all terms are merely variables.

The other common $1^{\underline{\text{st}}}$order set theory language includes also one constant symbol, namely $\emptyset$, which one thinks of as denoting the empty set, of course.

The first language does work, because it is possible to have $\emptyset$ as a **defined** constant, rather than as a symbol which is part of the language itself. This is a use of short form which is somewhat different to the use illustrated at the end of the section on $1^{\underline{\text{st}}}$order number theory. There we defined both a short form binary relation symbol " | " (divides), as well as a short form unary (1-ary) relation symbol "is a prime". Here we are trying to define a short form constant symbol. This is done as follows. We don't wish to say what $\emptyset$ 'is' except to say what actual formula any "short form formula" using $\emptyset$ as a term actually is. (The style below is poor for something to do mathematics with, but we won't be pursuing this any further.)

$$\text{“}\cdots\emptyset\cdots\cdots\emptyset\cdots\text{”} \quad = \quad \exists z((\forall x\ \neg x \in z)\ \wedge\ (\cdots z\cdots\cdots z\cdots)) \ .$$

Here the left-hand side is a string of symbols from the language and also containing $\emptyset$ one or more times. On the right, we just changed all those occurrences to the variable z, which shouldn't occur elsewhere in the string; and that string on the right should be a formula. And x is any variable different from z.

For example,

$$\text{“}y \in \emptyset\text{”} \quad = \quad \exists z(\forall x\ \neg x \in z\ \wedge\ y \in z) \ ,$$

where x and y are both different than z.

Below are some examples of other short forms. This will culminate with a rather striking example of how complicated the actual (or even abbreviated, but without short forms) $1^{\underline{\text{st}}}$order formula can be, which expresses a very simple concept.

If Y and Z are sets as treated in everyday mathematics, then $Y \times Z$ is defined to be the set consisting of all ordered pairs (y, z) for which $y \in Y$ and $z \in Z$. This so-called *Cartesian product set* is very pervasive. For example, it's used to talk about graphs of functions, and even to define what the word *function* means.

Ordered pairs are often not explicitly defined, but it is understood that two ordered pairs are different if either their first elements differ, or their second elements differ (or both). For example, the ordered pairs $(2, 3)$ and $(3, 2)$ are different, so we're talking about more than just the 2-element set consisting of 2 and 3.

Sometimes a finite sequence of length n is called a n-tuple, and is defined to be a function from the set $\{1, 2, \cdots, n\}$ into the set from which the elements in the sequence are taken. This is fine. But, to avoid circularity, it's important to not define an ordered pair to be a 2-tuple. This is because, when you defined "function", you will have referred to ordered pairs.

So one neat little way to define "ordered pair" (entirely in terms of sets) is to define $(y, z) \ := \ \{ \ \{y\} \ , \ \{y, z\} \ \}$.

This is the following set. One of its elements is a set (containing y and containing no other elements), one of its elements is a set (containing y and containing z and containing no other elements), and it has no other elements. I've phrased this boringly and probably almost unreadably here (like a legal document!) so as not to imply by sloppy use of language that y and z are necessarily distinct—they aren't. You might even want to think sometime about whether or why y and $\{y\}$ are necessarily distinct!

Now let's go through this in the $1^{\underline{\text{st}}}$order set theory language. The short forms below really should be defined using formulae in the " $\exists z \cdots$ "-style which we used above for $\emptyset$. Done this way, and then written out in prefix style with all its brackets, the actual formula for "$X = Y \times Z$" fills up the entire top third of page 12 in **Manin** ! It is even more unreadable than the formula below which sticks out into both margins.

Just for the next few lines, the upper case letters X, Y and Z will denote

variables in the language, as will the usual lower case letters. Also we'll write down these short forms only for variables, despite what we said earlier, because we won't be going further with this here. But really they ought to be defined with ingredients which are terms in general, and with the understanding that the bound variables on the right-hand side are any ones which don't occur in those terms.

$$\text{“}X \approx Y \times Z\text{”} \quad = \quad \forall x \, (x \in X \; \leftrightarrow \; \exists y \exists z \, (y \in Y \; \wedge \; z \in Z \; \wedge \; \text{“}x \approx (y,z)\text{”})) \; ,$$

where

$$\text{“}x \approx (y,z)\text{”} \quad = \quad \forall u \, (u \in x \; \leftrightarrow \; \text{“}u \in \{ \; \{y\} \; , \; \{y,z\} \; \}\text{”}) \; ,$$

where

$$\text{“}u \in \{ \; s \; , \; t \; \}\text{”} \quad = \quad u \approx s \; \vee u \approx t \; ,$$

$$\text{“}u \approx \{y\}\text{”} \quad = \quad \forall v \, (v \in u \; \leftrightarrow \; v \approx y) \; .$$

and

$$\text{“}u \approx \{ \; y \; , \; z \; \}\text{”} \quad = \quad \forall v \, (v \in u \; \leftrightarrow \; v \approx y \; \vee \; v \approx z) \; .$$

Now starting from the top and substituting directly, in a few lines we arrive at the following formula which is a formula without short forms expressing “$X \approx Y \times Z$” :

$$\forall x(x \in X \leftrightarrow \exists y \exists z(y \in Y \wedge z \in Z \wedge \forall u(u \in x \leftrightarrow (\forall v(v \in u \leftrightarrow v \approx y) \vee \forall v(v \in u \leftrightarrow (v \approx y \vee v \approx z))))))$$

This is pretty complicated, and it already contains many abbreviations, including standard and associativity bracket removals, and use of symbols not part of the original language. It is written out here merely to convince you of the absolute necessity for short forms (as well as for our standard abbreviations).

Sometimes $\{ \; y \; , \; \{y,z\} \; \}$ is used, rather than $\{ \; \{y\} \; , \; \{y,z\} \; \}$, as the definition of (y,z). The lengthy formula above then gets changed to one which can be found at the very end of Appendix **B** to Chapter 8.

Exercises.

5.20 Write down the intermediate lines in getting this last formula, displayed above.

5.21 Give definitions for short forms to define *union, intersection* and *subset*; that is,

$$\text{“}x \cup y\text{”} \quad ; \quad \text{“}x \cap y\text{”} \quad ; \quad \text{“}x \subset y\text{”} \quad .$$

5.22 Try to figure out how to give a decent $1^{\underline{\text{st}}}$order set theory treatment of von Neumann's definition of the natural numbers. (This is a snippet in the direction of the claim that all mathematics is reducible to $1^{\underline{\text{st}}}$order set theory, sometimes called Hilbert's Thesis. In particular, mathematical statements from earlier in number theory and graph theory, which I claimed to be not expressible in those $1^{\underline{\text{st}}}$order languages, *are* expressible in $1^{\underline{\text{st}}}$order set theory.) The definition, in ordinary mathematical parlance, is

$$0 := \emptyset \;\; ; \;\; 1 := \{0\} \;\; ; \;\; 2 := \{0, 1\} \;\; ; \;\; 3 := \{0, 1, 2\} \;\; ; \;\; \cdots$$

Finally, notice again that $1^{\underline{\text{st}}}$order set theory language using just $\in$ as the only special symbol is really the same language as that of $1^{\underline{\text{st}}}$order directed graph theory, with r changed to $\in$. But I don't suppose any sane person would attempt to draw a picture supposedly containing every set, with each drawn as a point, and with a directed edge from a to b exactly when $a \in b$! See however [**Barwise and Etchemendy**], Ch. 3, on the universe of hypersets, where many directed graphs appear in explaining a theory of sets which denies the axiom of foundation.

5.7 The language of $1^{\underline{\text{st}}}$order group theory.

This is the language which, if we use prefix notation, has one binary function symbol μ, one unary function symbol ι, and one constant symbol 1. But it is much more common to use infix notation with μts replaced by $(t \times s)$, and a superscript $^{-1}$ in place of ι.

Some examples of corresponding formulae in the two notations:

$\forall x\, \forall y\, \forall z\, (x \times y) \times z \approx x \times (y \times z)$ is the same as $\forall x\, \forall y\, \forall z\, \mu\mu xyz \approx \mu x \mu yz$;

$\forall x\, (x \times 1 \approx x \;\wedge\; 1 \times x \approx x)$ is the same as $\forall x\, (\mu x 1 \approx x \;\wedge\; \mu 1 x \approx x)$;

$\forall x\, (x \times x^{-1} \approx 1 \wedge x^{-1} \times x \approx 1)$ is the same as $\forall x\, (\mu x \iota x \approx 1 \wedge \mu \iota x x \approx 1)$.

Emphasizing now the infix notation, those who have studied group theory or even just matrix theory know many interpretations. Most or all such interpretations satisfy the three sentences above; that is, those sentences are true

in the interpretation. Such an interpretation is a known to mathematicians as a **group**.

An example is the set of all square invertible matrices of a fixed size with, say, real entries. Part of interpreting a language is specifying the constants, functions and relations, once one has specified the set of objects. In this case, the binary function is matrix multiplication, the unary function is matrix inversion, and the constant is the identity matrix.

I don't know how
you were inverted;
no one alerted you.

G. Harrison

An example familiar to everyone is just the case of 1×1 matrices above, which is really just the set of all *non-zero* real numbers under ordinary multiplication and inversion.

There are many other important examples of groups in which the objects are permutations, or invertible linear transformations, or other kinds of invertible functions (actual ones, not symbols, of course!); and where the binary function symbol is interpreted as composition, the unary as inversion of actual functions, and the constant symbol 1 as the identity function.

It should be pointed out that there is no requirement for an interpretation to actually be a group. Any set of objects together with interpretations of the special symbols will do, independently of whether the three sentences above (known as the **axioms of group theory**) are satisfied. Also notice that, except for the non-zero reals, our interpretations above generally don't satisfy the **commutative law**:

$$\forall x \forall y \; x \times y \approx y \times x \; .$$

The three group axioms are known respectively as the **associative law**, the existence of an **identity element**, and the existence of **inverses**.

Exercises.

5.23 Translate the following into the language of group theory, doing each both in infix and in prefix notation.

(i) *Every element is its own inverse.*

(ii) *Squaring an element always produces the identity element.*

(iii) *Every element is a square.*

(iv) *The seventh power of some element is x.*

Remark. In contrast to the last two exercises, an example of a statement which cannot be translated into the language of $1^{\underline{\text{st}}}$order group theory (as a single formula, or finitely many) is the following.

For every positive integer n, every element is an nth power.
(An example of a group satisfying this is the set of non-zero complex numbers, under complex multiplication—or just those on the unit circle centred at the origin.)
An example of a statement which cannot be translated into the language of $1^{\underline{\text{st}}}$order group theory (even as infinitely many formulae) is the following.

There is some positive integer n such that the nth power of x is zero.
See Exercise 7.13.

Exercise 5.24. Try to invent a $1^{\underline{\text{st}}}$order language into which that last statement about groups can be translated. You may wish to read first the material in the final section ahead about languages dealing with more than one type of object.

5.8 The language of $1^{\underline{\text{st}}}$order ring theory.

This is also a major topic in modern algebra. The language is the same as for number theory, except that we drop the relation symbol $<$. So there are two binary function symbols $+$ and $\times$, together with two constant symbols 0 and 1. Examples of formulae in this language are any formulae from the introduction or Section 5.1 of this chapter which don't involve $<$.

In any book on modern algebra, you will find a list of properties called the axioms for rings. These can be translated into this $1^{\underline{\text{st}}}$order language. They are similar to the axioms for groups, but a bit more complicated due to the extra operation. So another way of expressing the definition of the word *ring* is to say that it is any interpretation of the language of ring theory which is a model for (i.e. satisfies) those axioms.

Recall that back in Section 5.1 we mentioned a formula which could be used as the definition of the short form "t is a prime", and which didn't

involve $<$. Yet another one is

$$\neg \text{“}t \text{ is invertible”} \ \wedge \ \neg t \approx 0 \ \wedge$$
$$\forall x \forall y \ (t \approx x \times y \ \rightarrow \ (\text{“}x \text{ is invertible”} \ \vee \ \text{“}y \text{ is invertible”})) \ .$$

Exercise 5.25 Figure out what 1st order formula should have “s is invertible” as a short form.

5.9 Languages of 1st order geometry and the question of what to do when you want to talk about more than one kind of object.

One of these (which we'll call Euclid-Forder-Coxeter geometry language) has already been given, near the end of the section on infix versus prefix notation. It has only two special symbols, a 3-ary relation which in interpretations corresponds to in-betweenness for points, and a 4-ary relation which in interpretations corresponds to congruence (equal lengths) of the intervals defined by the first two and the last two of the four points.

There are a couple of reasons to talk about this language:

The student who so desires can translate much of the informal mathematics in **Appendix G** into this 1st order language.

It demonstrates that there are indeed very interesting mathematical theories in which it is natural to use n-ary relations with $n > 2$.

I've used the three mathematicians' names above because the latter two have written books in which the informal axiomatization of Euclid's geometry is expressed using these two relations.

But perhaps the most famous axiomatization is due to Hilbert at the beginning of the 20th century. He used somewhat different relations, but for us, the interesting point is that, for plane geometry, he used *two types of undefined objects* , corresponding to our geometric intuition of *points* and *lines*. (For 3-dimensional geometry, he used three—*points, lines* and *planes*.) Does that mean we cannot expect to translate this type of mathematics into a 1st order language?

Another example of this occurring is the theory of vector spaces in linear algebra, where the two types of objects are *scalars* and *vectors*.

The answer to the question is that we certainly *can* do this type of stuff in 1$^{\underline{\text{st}}}$order. Basically all that is needed is a couple of unary relation symbols, plus whatever other special symbols might be needed. For example, in the case of Hilbert-Euclid geometry, the extra relation symbols might be p and ℓ. Then the formula px says that the variable x is a point, and similarly ℓy says that the variable y refers to a line. In vector space language, the symbols might be s for scalars and v for vectors (but then you'd have to be careful not to use v as a nickname for a variable!).

One of Hilbert's basic relations is incidence of points with lines. One axiom is that any two distinct points are incident with one and only one line. Here is a sentence which is a 1$^{\underline{\text{st}}}$order version of that axiom, where we use ι as a binary relation symbol to be interpreted as the incidence relation.

$$\forall x \forall y \ ((px \wedge py \wedge \neg x \approx y) \ \rightarrow \ \exists z \ (\ell z \wedge \iota xz \wedge \iota yz))$$

$$\wedge$$

$$\forall x \forall y \forall z \forall w ((px \wedge py \wedge \neg x \approx y \wedge \ell z \wedge \ell w \wedge \iota xz \wedge \iota yz \wedge \iota xw \wedge \iota yw) \rightarrow z \approx w) \ .$$

The upper part of the formula states the existence ("one"), and the lower part states uniqueness ("only one"). Here I have left out quite a few pairs of brackets where more than two formulae have been '$\wedge$ed' together. They really should be in there, as we didn't include dropping them as an abbreviation. In fact, different choices for putting them in result in strictly different formulae. But all these formulae are truth equivalent, as defined in Chapter 8, so it doesn't really matter which particular way they are included. These are what were referred to earlier as "associativity brackets"

A shorter formula which expresses the same axiom is

$$\forall x \forall y \ (px \wedge py \wedge \neg x \approx y \ \rightarrow \ \exists z \ (\ell z \wedge \iota xz \wedge \iota yz \wedge \neg \exists w (\neg z \approx w \wedge \ell w \wedge \iota xw \wedge \iota yw))) \ .$$

Exercises.

5.26 In the Euclid-Forder-Coxeter language, the notion of incidence must be defined. In wordy fashion, the line defined by two distinct points contains also the points between them, and any point P for which either point above is between P and the other one above. (See "Order Geometry" in **Appendix G**.) Show how to express incidence in this 1$^{\underline{\text{st}}}$order language; that is, translate "x is on the line defined by distinct points y and z."

5.27 In the Euclid-Hilbert 1st order language, express the following.

(i) *x and y are distinct lines which are parallel to each other.*

(ii) *For any line and any point not on it, there is one and only one line through the given point and parallel to the given line.*

(iii) *If two distinct lines are parallel, and a third line intersects one of them, then it intersects the other.*

The last two are two ways of expressing the famous parallel postulate of Euclid, which is the main topic of **Appendix G**. The answers to (ii) and (iii) are not likely to be truth equivalent, as defined in Chapter 8. Their mathematical equivalence depends on other (admittedly obvious) facts.

Readers with an interest in the theory of **special relativity** from physics may find it interesting to learn that the relevant geometry of 4-dimensional space, called **Minkowski geometry**, can be defined entirely using a single binary relation. The points in this geometry are correlated in physics to the idea of an **event**, such as a particle at a particular moment of time (though one of the main ideas in relativity is that "time" in an absolute sense becomes non-unique). The binary relation is correlated to so-called **causal order**. Event A is related under causal order to event B only when B can be physically affected by A. Thus, for example, if, in some inertial observer's experience, event B takes place a long distance from A, but only a little later in time than event A, then A cannot have affected B because of the upper bound postulated on the propogation of physical effects (the speed of light). And so A is not related to B under the basic relation in Minkowski geometry, where now A and B are denoting points in that geometry. The surprising thing is that the in-betweenness relation, as well as the congruence relations in both the spatial 3-space and the temporal 1-space corresponding to any inertial observer ('inertial frame'), can be expressed in terms of that causal order relation.

Appendix VQ : VERY QUEER SAYINGS KNOWN TO SOME AS PARADOXES, BUT TO GÖDEL AS EXPLOITABLE IDEAS

Many of you will have heard of the person from ancient Crete who is reputed to have said something like :

"All statements by all persons from Crete are false."

(More commonly stated : "All Cretans are liars", though the version in the New Testament is more elaborate.) It is easy to argue that if such a statement is true (and with knowledge that it is made by a person from Crete), then it must also be false. So it's no paradox; it's simply a necessarily false statement.

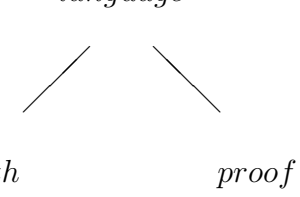
I have oversimplified above, and this so-called liar paradox can be sharpened to something more truly paradoxical :

"This statement is untrue." Or simply : "I am untrue."

This is uttered with the understanding that the subject of the sentence, "This statement", is actually that statement itself as displayed above (and not some other one to which the speaker is pointing). Now an easy argument shows that the statement is untrue (i.e. false) if and only if it is true.

There are lots of other well-known variations on the 'self-reference paradox'. For example, consider the resident adult male village barber who shaves every man in his village who doesn't shave himself. Who shaves the barber?

Now that we have begun $1^{\underline{st}}$order logic, I can give a brief indication of how it sheds some light on these self-referential statements (Is self-reference a greater or a lesser sin than self-reverence?), and an indication of how they can even be used to get some important theorems of logic. In $1^{\underline{st}}$order logic, we'll have the same setup as in propositional logic :



Truth will come from interpretations (not truth assignments as in propositional logic). This is treated in detail in Chapter 6.

Proof will come from a proof system, more elaborate than the one for propositional logic. This is done in Chapter 7.

Now we can ask a question. Take, for example, the language of $1^{\underline{st}}$order number theory, and its most famous interpretation, consisting of **N**, the set of natural numbers, with its usual addition, multiplication and ordering.

Is there a $1^{\underline{st}}$order sentence whose translation is: **"I am untrue"** ?

(You might think that this is a ridiculous question to ask. But please withhold your judgement on that till reading the next few paragraphs.) The mathematical logician Tarski used the liar paradox and some fundamental work of Gödel to show that *truth is* **not** *definable in the the language of* $1^{\underline{st}}$*order number theory*. For a contradiction, he assumed it was, and found a $1^{\underline{st}}$order sentence corresponding to "I am untrue." Then the 'liar paradox' allowed him to get a contradiction. See the last few pages of the **Addendum** to **Appendix L** at the end of this book. Above, "*truth*" means truth in the interpretation **N**. And that will be made almost painfully precise in the next chapter. Also "*definable*" was explained precisely using Gödel's work. There is nothing vague nor paradoxical about Tarski's work—no sophistry!

Now for something even better.

The statement "I am unprovable" has by no means the same meaning as "I am untrue". Now that we've learned to distinguish truth from proof (the semantical from the syntactical), this should be clear. And Gödel showed that *provability* **is** *definable, and even expressible, in the the language of* $1^{\underline{st}}$*order number theory*. He did this very precisely, and in a way that is independent of any particular interpretation. Because of this, he was able to show that the sentence "I am unprovable" does have a corresponding $1^{\underline{st}}$order sentence in number theory. The latter is interpretable as "I am unprovable" in many different interpretations, including the standard one, **N**.

Gödel produced here at least two all-time intellectual hall-of-famers—both the technical work of making sense of definability/expressibility using so-called Gödel numbers, and also the stroke of genius to think of replacing "I am untrue" by "I am unprovable." This work resulted in his famous incompleteness theorem. A good deal of **Appendix L** is taken up with dis-

cussing all this. You will, however, need to consult a more advanced book on logic for the technical details concerning definability/expressibility. In the **Addendum**, we explain what they mean, but don't prove Gödel's assertions about them. But, modulo that, we do give proofs of Gödel's and Tarski's theorems there. In the last section of this appendix, we sketch another, more recent proof of Gödel's theorem.

Exercises

VQ1 If I wrote in this book: "There is an untruth in this book", you really couldn't disagree, could you? Explain.

VQ2 One can use the 'Cretan argument' (not *cretin* argument!) to apparently give a proof that it is impossible for there to be a person anywhere who always lies. I will make the argument more convincing by making it more complicated than needs be. Firstly, it is enough to prove that, for every non-empty set of people, at least one of those people doesn't lie all the time—then just apply that result to singleton sets to get the desired result. So, suppose given a non-empty set of people. In one case, at least one of those people doesn't lie all the time, and so nothing needs to be proved. In the other case, given what you're trying to do, surely those people are happy to point out that you're wrong, and that their set provides a counterexample. So suggest that some member of the set utter the sentence: "All members of this set always lie." Now complete the argument by considering whether or not this utterance is a lie.

Digression: Zeno's 'Paradox'.

The best known of Zeno's paradoxes is sometimes lumped in with the above ancient logical/linguistic paradoxes. We shall Canadianize this, eh, by replacing 'Achilles chasing tortoise' with 'wolf chasing caribou'.

Check-times are defined inductively: the initial one is the start-time of the chase; and the *next* check-time is the moment of time when the wolf arrives at where the caribou was at the *previous* check-time. The preliminary (correct) conclusion is that there are infinitely many moments of time when the wolf has not caught the caribou. The actual (nonsensical) conclusion is that there is **no** moment of time when the wolf has caught the caribou.

So this is neither paradoxical, nor is it an invalid (logical) argument. It is a nonsensical physical argument, which does serve a purpose. Either our

abstract model of physical reality allows for infinitely many moments of time or it doesn't. (Usually it does!) If it doesn't, then the whole wolf vs. caribou discussion is near to impossible. If it does allow for infinitely many moments of time, then surely we don't expect that, for every infinite subset of moments of time, there is no moment of time which is later than all of them.

And I imagine that this is pretty much the point that Zeno was trying to make. Furthermore, his discussion is useful for illustrating the mathematical concept of *convergent sequence*.

Sometimes the claim (that there is a genuine paradox here) is to note that Zeno's description shows how our wolf would need to perform an *infinite* sequence of actions. On the contrary, it seems that his description goes some ways towards being an argument in favour of the existence of *completed infinities* (infinite sets). That is, denying their existence produces the paradox, if you want to call it that.

It is customary with philosophers to deny the reality of space and time, and to deny also that, if they were real, they would be aggregates. I shall endeavour to show ⋯ *that these denials are supported by a faulty logic, and by the now resolved difficulties of infinity. Since science and common sense join in the opposite view, it will therefore be accepted; and thus, since no argument* à priori *can now be adduced against infinite aggregates, we derive from space and time an argument in their favour.*

Bertrand Russell

Definable relations, Berry's paradox, and Boolos' proof of Gödel's incompleteness theorem

Think about the following 'object' : the smallest natural number which cannot be named using less than twenty-four syllables. We just named it using 23 syllables—count them!—so this seems to be another paradox, one which is called Berry's paradox. In order to give a taste of how Gödel and his disciples have been able to use this kind of thing as an idea for studying logic, we'll explain how George Boolos has used Berry's paradox to give a proof of a version of the famous incompleteness theorem of Gödel.

Several of the minor steps in the proof depend on technical material from the next two chapters, so we are jumping ahead of ourselves a bit here.

Each of these is dealt with in a later exercise which is referred to specifically below. In Chapter 6 we shall provide a painfully precise treatment of just what it means for a sentence in $1^{\underline{\text{st}}}$order number theory to be true in the interpretation $\mathbf{N}$ (and more generally— $\cdots$ a formula in a $1^{\underline{\text{st}}}$order language to be true some interpretation of it). In Chapter 7 we shall exhibit a proof system for $1^{\underline{\text{st}}}$order logic similar to the one in Chapter 3 for propositional logic. In particular, if Γ is a set of sentences in the language of $1^{\underline{\text{st}}}$order number theory, this will tell us what it means to have a derivation of F using Γ as the set of premisses. That is, it will define the phrase $\Gamma \vdash F$.

Let us now imagine Γ as a decidable set of sentences which are true in $\mathbf{N}$, and that Γ has perhaps been proposed by someone as a set of axioms for number theory. Recall that *decidable* here means that there is an algorithm whose input is an arbitrary sentence from the language, and whose output is yes or no, depending on whether or not the input is in Γ. The proposer's hope might be that every sentence true in $\mathbf{N}$ would be derivable from Γ.

The style of writing in treatments of this material can be somewhat 'loose'. Mainly this takes the form of statements in the style: G says '*such and such about* x *and* y'. Here the quoted phrase defines a relation on pairs (x, y) of natural numbers, and G is some $1^{\underline{\text{st}}}$order formula with two free variables, x and y. This is good for those who don't want any more precision (e.g. writers of elementary textbooks on the philosophy of mathematics), and for experts on logic (who know easily how to translate it into a precise form). Since we fall somewhere between those two groups, I'll use more precise language, as well as indicating exactly what has been left out of the proof. But I'll also include statements in the above style, because they really explain where the ideas come from. But such 'loose' statements are not actually part of the proof.

For each natural number $b > 1$, define a short form term in the language of $1^{\underline{\text{st}}}$order number theory as follows:

$$\text{“}b\text{”} \;:=\; (1 + (1 + (1 + \cdots\cdots (1 + 1) \cdots))) \;,$$

where exactly b copies of 1 are used.

Define *weight* in the language of $1^{\underline{\text{st}}}$order number theory as follows:

$$\text{weight}(x_n) := n \;\;;\;\; \text{weight}(\textit{any other symbol}) := 1 \;\;;$$

and weight(F) := sum of weights of all symbols occurring in F, counting once for each occurrence of the symbol.

Weight will be used as a replacement for counting syllables. We don't simply use the *length* of a string, because we want there to be only finitely many strings (and therefore finitely many formulae) of any given weight. This finiteness holds because, for weight n, we clearly cannot use any variables x_i with $i > n$, so there are only finitely many symbols which can be used, say s. But the length of a string of weight n cannot exceed n, and there are at most ns^n strings of length $\leq n$ using only symbols from a set of size s. Thus the number of strings of weight n cannot exceed ns^n, so is finite.

Now notice that the weight of "b" is $4b-3$, because there are $b-1$ copies of $+$ and of each bracket. So we get

$$\text{weight}((\text{“10”} \times \text{“}k\text{”})) \;=\; 4k + 37 \;\;,$$

which is almost always quite a bit less than the weight of "$10k$". This dumb little fact will turn out to play a crucial role below, in that we shall use "10" $\times$ "k" rather than "$10k$" at the appropriate place. (In the display above, we are careful to include, and count, the pair of brackets around the term, the inner ones. The outer ones are brackets in math/English, the metalanguage, as opposed to the inner ones in the formal language. Really fussy people would insist on using different notations. But many logicians don't even insist on different equality symbols in the two languages, as we are doing in this book.)

We assume fixed some decidable set Γ of sentences in $1^{\underline{\text{st}}}$order number theory, all of which are true in $\mathbf{N}$. It follows from basic properties of the proof system that only for formulae G which are true in $\mathbf{N}$ is it possible to have $\Gamma \vdash G$.

Everywhere below, we'll use x, y, z and w as abbreviations of x_1, x_2, x_3 and x_4, respectively.

Define a relation on pairs (a, c) of natural numbers, which is phrased

$$a \text{ is named by a formula of weight } c \;,$$

by saying that this holds for exactly those (a, c) for which there is a formula F of weight c such that

$$\Gamma \;\vdash\; \forall x(F \leftrightarrow x \approx \text{“}a\text{”}) \;.$$

The formula F would pretty surely have x as a free variable, and normally not have any other free variables, but we needn't dwell on that. Note however

that this naming relation definitely depends on which set Γ is being used as premisses/number theory axioms.

As an example, one would expect that most choices of Γ would have the property that

$$\Gamma \ \vdash \ \forall x(x \approx (\text{“}10\text{”} \times \text{“}k\text{”}) \quad \leftrightarrow \quad x \approx \text{“}10k\text{”}) \ .$$

In that case, we'd have that $10k$ can be named by a formula of weight $4k+39$, which is quite a bit less than the weight of “$10k$” itself.

A given formula F cannot name more than one number, since, if it named both a and b, then easily established properties of the proof system would give a derivation of “a” $\approx$ “b”. (This is a relatively elementary consequence of the definitions in Chapter 7, and is an example of Exercise 7.15 at the end of that chapter.) But “a” $\approx$ “b” is not true in $\mathbf{N}$ if a and b are different numbers.

It now follows from the finiteness of the set of formulae of a given weight that only finitely many numbers can be named by formulae of a given weight c.

Here is the major fact which you will have to take on faith— its proof is long and tedious, but the deep idea is just to think of *saying* this:

The naming relation is definable in 1^{st}order!

So far, that's a loose statement, and here's another one making this only a bit more precise:

There is a formula C with free variables x and z which says :
‘x is named by a formula of weight z'.

Well, let's do the precise version: For any formula C, let $C^{[x\to t,\ z\to s]}$ be the formula which is obtained from C by substituting the term t for each free occurrence of x, and the term s for each free occurrence of z. Then the precise version of the above pair of statements is the following:

There is a formula C such that $C^{[x\to\text{“}a\text{”},\ z\to\text{“}c\text{”}]}$ is true in $\mathbf{N}$ if and only if (that is, for exactly those a and c for which) a is named by a formula of weight c.

There is a general reason why this holds (but *its* proof is long and tedious too). That reason is the existence of an algorithm of Type L which lists all pairs (a, c) for which a is named by a formula of weight c (an assertion which is weaker than making the false claim that there is an algorithm of Type D which takes the input (a, c) and decides whether a is named by a

formula of weight c). The existence of a listing algorithm for a relation is all that's needed to be assured that the relation is definable in $1^{\underline{\text{st}}}$order. This is discussed in more detail in the final few paragraphs of this book. The existence of a listing algorithm above is quite straightforward to see from the fact that there is an algorithm for listing all formulae which are derivable from Γ. That is an absolutely fundamental property of the proof system. It is where the decidability of Γ is decisively used.

Now define $B \ := \ \exists z \ (z < y \ \wedge \ C)$. Then, loosely, B says :

'x is named by a formula of weight less than y'.

The precise version of the above statement is the following:

$B^{[x \mapsto \text{“}a\text{”},\ y \mapsto \text{“}b\text{”}]}$ is true in $\mathbf{N}$ *if and only if (that is, for exactly those a and b for which) a is named by a formula of weight less than b.*

Next define $A \ := \ \neg B \ \wedge \ \forall w(w < x \rightarrow B^{[x \mapsto w]})$. Then, loosely, A says :

'x is not named by any formula of weight less than y, but every number less than x is named by at least one formula of weight less than y'.

The precise version of the above statement is the following:

$A^{[x \mapsto \text{“}a\text{”},\ y \mapsto \text{“}b\text{”}]}$ is true in $\mathbf{N}$ *if and only if (that is, for exactly those a and b for which) a is not named by any formula of weight less than b, but every number less than a is named by at least one formula of weight less than b.*

The precise versions referring to B and A can easily be deduced from the one for C using the definitions in Chapter 6, and are examples of exercises 6.19 (i), (ii) at the end of that chapter. 6.19 (iii) gives the similar assertion about F just below.

Having chosen some particular C as above, and thence obtained the formulae B and A,

let k be the weight of A.

Do a little counting with A to see that $k > 9$, in fact, quite a bit bigger.

Finally define $F \ := \ \exists y(y \approx (\text{“}10\text{”} \times \text{“}k\text{”}) \ \wedge \ A)$. Then, loosely, F says :

'x is not named by any formula of weight less than $10k$, but every number less than x is named by at least one formula of weight less than $10k$'.

The precise version of the above statement is the following:

$F^{[x \mapsto \text{“}a\text{”}]}$ is true in $\mathbf{N}$ *if and only if (that is, for exactly that a for which) a is not named by any formula of weight less than $10k$, but every number less than a is named by at least one formula of weight less than $10k$.*

There is clearly at most one such a for which the latter statement holds.

That such a number a *does* exist follows from the facts that there are only finitely many formulae of weight $< 10k$, each one of which can name at most one number. So we are singling out the smallest member of a set of natural numbers which, for this purpose, we need to know to be non-empty (*very* non-empty, since it's the complement of a finite set).

Call this number n.

Notice that the weight of F is $5k + 46$, using the above calculation for the weight of ("10" $\times$ "k"), and the fact that y is x_2, which has weight 2.

Now for this choice of F,

$$\forall x(F \leftrightarrow x \approx \text{“}n\text{”}) \quad \text{is true in } \mathbf{N} \,. \tag{I}$$

This is intuitively clear from the fact that $F^{[x \mapsto \text{“}a\text{”}]}$ is true in $\mathbf{N}$ if and only if $a = n$, by the definition of n. Establishing it rigorously is an example of exercise 6.19 (iv) given at the end of the next chapter.

On the other hand, for this choice of F, we cannot have

$$\Gamma \vdash \forall x(F \leftrightarrow x \approx \text{“}n\text{”}) \tag{II}$$

Otherwise, we'd get (i) and (ii) below, a contradiction:

(i) n is named by a formula of weight $5k + 46$, by the above weight calculation and the definition of the naming relation; and

(ii) n is not named by any formula of weight less than $10k$, by the definition of n.

That combination is ridiculous, since $5k + 46 \geq 10k$ doesn't hold for any integer $k > 9$.

Clearly, combining (I) and (II) devastates the hope (of the proposer of Γ as a set of number theory axioms) that every sentence true in $\mathbf{N}$ would be derivable from Γ.

We have found a formula which is true but not derivable from Γ. This is one version of **Gödel's first incompleteness theorem :**

No **decidable** *set of* $1^{\underline{\text{st}}}$*order number theory sentences which are true in* $\mathbf{N}$ *has the property that every other sentence which is true in* $\mathbf{N}$ *can be derived from that set. In particular, the set of all sentences which are true in* $\mathbf{N}$ *is* not *decidable.*

The second statement in the theorem doesn't directly imply the first. But the stronger fact that the set of all sentences which are true in $\mathbf{N}$ is not even

a listable set, in the sense of existence of a Type L algorithm, does. See **Appendix L** for a lot more material on these matters.

The theorem shows that

there is a formula such that neither it nor its negation is derivable.

Gödel has also proved this with the assumption about truth in **N** replaced by a consistency assumption. Hypotheses and conclusion are then purely syntactical. In this form, the theorem is even free of philosophical quibblers' objections from those who worry about the existence of infinite sets like **N**.

Remark. The need to use "10"×"k", rather than "10k", which we noted earlier, seems at first a bit odd. One might wonder what this corresponds to in Berry's paradox itself. The answer would seem to be that "twenty-four syllables" is actually six syllables!

6. TRUTH RELATIVE TO AN INTERPRETATION

Here is where we get both precise and general about what an interpretation is, and about what it means for an argument to be valid and for a formula to be true. You may think : "The meaning of a sentence being true in an interpretation is obvious!" After all, we have been talking about that quite a bit in many examples from the last chapter. But conundrums can arise when just using your intuition when trying to decide which of [*true—false—indeterminate*] applies to some formulae which are not sentences. The same can be said for some of the 'silly' formulae, for example, in Section **5.1**. Furthermore, Tarski's definition of truth puts it in a form which is convenient for making theoretical arguments. (Recall that knowledgeable people, as opposed to local news reporters, use "theoretical" to mean "of a general nature", not to mean "questionable" ! They know the difference between "theoretical" and "hypothetical". If you consult the OED, you'll realize that I'm engaged in a bit of wishful thinking here.)

There will be some general families of formulae which play an important role in the proof system of the next chapter. We want to be able to prove rigorously that some of them are true completely generally (in every interpretation). For example, is this the case for $\forall x F \rightarrow F$ for every formula F and every variable x? How about $F \rightarrow \forall x F$? Tarski's definition allows us to decide such things.

On the other hand, you might ask: "How can we have a rocksolid method for deciding about the truth of any $1^{\underline{\text{st}}}$order formula, when there are such formulae that correspond to famous unsolved mathematical problems, such as the twin prime and Goldbach conjectures in number theory? Surely someone could just apply the definition and decide the conjecture!" The answer is that Tarski's definition *gives precise criteria for what it means for*, say, the Goldbach conjecture to be true. But *it does not give a procedure, or algorithm*, for deciding whether such a $1^{\underline{\text{st}}}$order sentence is true or not. Much more on this appears in Chapter 8.

In fact, the phrase "Tarski's definition of truth" is rather pompous and misleading. As you will see in the first section below, a more appropriate description would be: "Tarski's reduction of $1^{\underline{\text{st}}}$order truth-checking to acts of checking whether a couple of objects are actually the same object, and to acts of checking whether a given object is in a given set (including objects which are ordered n-tuples of basic individuals in the interpretation set)." More succinctly: "Tarski's reduction of $1^{\underline{\text{st}}}$order truth-checking to naive set-theoretic

acts." The brutal fact that *infinitely* many such acts are usually necessary explains the existence of an ever-expanding store of unsolved problems in mathematics.

Another preliminary remark is about the use of sets in interpretations. This should definitely be thought of in the intuitive sense. We did have a formal language for $1^{\underline{\text{st}}}$order set theory as an example in the last chapter. It could be quite confusing to think about a **set** which gives an interpretation of that language. (Once you have worked through this and the next chapter, a more sophisticated and thorough discussion of these matters is possible. A good one may be found in Chapter VII of **Ebbinghaus-Flum-Thomas**.) In fact, "the set of all sets" is a concept which can lead to paradoxes, if care is not taken, as Cantor showed. So we shall put the $1^{\underline{\text{st}}}$order language of set theory to one side, and not attempt to interpret it except in a vague intuitive sense. In a more advanced treatment of logic, using $1^{\underline{\text{st}}}$order set theory language is actually the best means of dealing with Cantor's paradox, and also with the famous one of Russell concerning "the set of all those sets which are not members of themselves". See Section **9.5** at the end of Chapter 9 for some discussion of this.

To re-iterate, when we say below that an interpretation of a $1^{\underline{\text{st}}}$order language is a set together with some extra data related to that set, we shall only be using very basic ideas about set membership, union of sets, sets of ordered pairs, etc., all of which are safe intuitive notions which don't lead anywhere near the difficulties in the foundations of mathematics which are associated with the set theory paradoxes, or disputes about such things as the infamous Axiom of Choice.

Finally, why do we insist below that an interpretation must involve a **non-empty** set V? If the language being interpreted has even one constant symbol, then V would necessarily be non-empty, because among the extra data must be a choice of some element of V to correspond to that constant symbol. But more generally, allowing $\emptyset$ as an 'interpretation' would lead to some confusing situations. For example, the formula $x \approx x$ would turn out to be **both true and false** in that 'interpretation'. It would be true because the sentence $\forall x\ x \approx x$ is true in any interpretation. And it would be false because the sentence $\exists x\ x \approx x$ is false in the empty "interpretation". Actually, the difficulty is more deep-seated than that. *Every* formula would be both true and false in the empty "interpretation". If you follow the definitions below carefully, you will see that. So we simply eliminate $\emptyset$ from consideration before even starting.

6.1 Tarski's definition of truth.

Let $\mathcal{L}$ be the set of special symbols (constant, function and relation) from some fixed $1^{\underline{\text{st}}}$order language.

Definition. An *interpretation* of $\mathcal{L}$ consists of

(i) a non-empty set V ;

(ii) for each constant symbol $c \in \mathcal{L}$, an element $c^V \in V$;

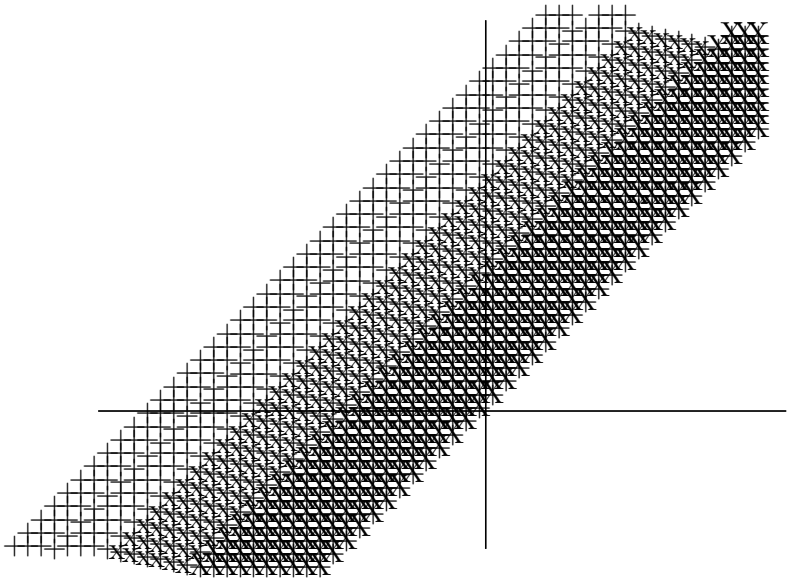
(iii) for each n-ary function symbol $f \in \mathcal{L}$, a function $f^V : V^n \to V$ (The set V^n is the set of all ordered n-tuples $(v_1, \cdots, v_n)$ with each v_i in V) ;

(iv) for each n-ary relation symbol $r \in \mathcal{L}$, a subset $r^V \subset V^n$.

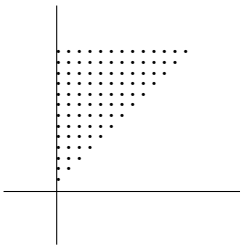
From the last chapter, we have many examples of this definition, and we used the word "interpretation" informally there, but in exactly the sense we defined above. In $1^{\underline{\text{st}}}$order number theory, most times, the interpretation had the set V as one of $\mathbf{N}$ or $\mathbf{Z}$ or $\mathbf{R}$. In $1^{\underline{\text{st}}}$order directed graph theory, each interpretation is a specific graph, with the set V as the set of vertices, as pictured in that interpretation. In $1^{\underline{\text{st}}}$order group theory, all interpretations had the set V as the set of elements of the particular group, for example, the set of invertible real 2×2 matrices. In $1^{\underline{\text{st}}}$order love theory, we had the set V as the set of all humans. In $1^{\underline{\text{st}}}$order geometry, we had the set V as the set of all points in the type of geometry being considered—the plane, 3-space, etc.

The motivation for (iv) is as follows. Rather than trying to define what a relation (as opposed to relation *symbol*) actually is in general by describing some sort of general way that a property or behaviour might determine a relation, we simply think of a relation on V as being the set of all n-tuples which are in fact related under that relation. (In the proper context, it is an important question to ask whether a relation defined more informally in mathematics can actually be defined by a $1^{\underline{\text{st}}}$order formula within some language and interpretation already given. We had examples of this earlier.) Here are three examples of relations on sets.

The relation $<^{\mathbf{R}}$ on $\mathbf{R}$ is the set $\{(x, y) \in \mathbf{R}^2 \ : \ x < y \}$, part of which is pictured below (shaded).



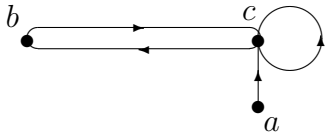
The relation $<^{\mathbf{N}}$ on $\mathbf{N}$ is the set $\{(x, y) \in \mathbf{N}^2 \ : \ x < y \}$, part of which is pictured below (with the dots).



The relation r^V of the directed graph, with $V = \{a, b, c\}$ as pictured below, is the set

$$\{ \ (a, c) \ , \ (b, c) \ , \ (c, b) \ , \ (c, c) \ \} \ ,$$

consisting of 4 of the 9 elements of V^2.



Notation. We'll use $\underline{v}$ quite often below to denote an infinite sequence $(v_1, v_2, v_3, \cdots)$ of elements from V. So $\underline{v}$ is kind of like a vector with infinitely many components, whose positions we'll call *slots*. It will always be used with the idea of substituting, in a formula F, the first slot's contents v_1 for each free occurrence of x_1, the second slot's v_2 for each free occurrence of x_2, etc. This would convert the formula into a definite statement about the set V, after we also replace each c by c^V, each f by f^V and each r by r^V. Exactly how this is done is explained just below. The resulting definite statement would then certainly be either true or false, and that is what saying "F is true (or false) at $\underline{v}$ from V" will be defined to mean. It will be necessary to make this definition by induction on formulae. Also, note that "$\underline{v}$ from V" doesn't mean $\underline{v}$ is a member of V. It isn't. Each 'slot-entry' v_i in $\underline{v}$ is a member of V.

N.B. Earlier we have occasionally used u, v and w as nicknames for variables. Here we'll be using them for something completely different. No conflicts will arise from this minor abuse of notation. It will always be quite clear from the context which way they are being used.

Let t be a term which starts with the function symbol f, which is, say k-ary. We are about to define an element of V called $t^{\underline{v}}$. All this really is, is what you'd get by writing down the string which is t, replacing each function symbol occurrence g in it by g^V, each constant symbol occurrence c by c^V, and each occurrence of each variable x_i by what is in the ith slot of $\underline{v}$, that is, v_i; and then putting in a bunch of brackets and commas, so that each g^V is being evaluated at the suitable 'tuple'. In particular, $t^{\underline{v}}$ is simply $f^V(z_1, \cdots, z_k)$ for suitable z_i's. For proofs, it is much more convenient to make the actual definition an inductive one, as follows.

Definition. Given $\underline{v}$ from V, define elements $t^{\underline{v}} \in V$ inductively for terms t in the language as follows :

(i) if t is c, a constant symbol, define $t^{\underline{v}}$ by setting $c^{\underline{v}} := c^V$;

(ii) if t is the ith variable x_i, define $t^{\underline{v}}$ by setting $x_i^{\underline{v}} := v_i$, the element in the ith slot of $\underline{v}$;

(iii) if t is $ft_1t_2 \ldots t_n$, where f is an n-ary function symbol, and $t_1, \cdots, t_n$ are terms for which $t_1^{\underline{v}}, \cdots, t_n^{\underline{v}}$ have all been defined, define $t^{\underline{v}} := f^V(t_1^{\underline{v}}, t_2^{\underline{v}}, \cdots, t_n^{\underline{v}})$.

In a concrete example, to calculate $t^{\underline{v}}$, you would write down a term verification column for t, and apply this definition one line at a time, working your way from top to bottom. For example,

$$(fx_3gdx_7x_2)^{\underline{v}} \;=\; f^V(v_3, g^V(d^V, v_7), v_2)$$

where d is a constant symbol, f is 3-ary, and g is 2-ary.

This definition is the precise version of the first part of what we meant above, when we spoke loosely about "replacing c by c^V and f by f^V in a formula F".

To finish the job, we now define the set F^V, which will be 'the set of $\underline{v}$ at which F is true'. This will also be done by induction. The result is just what you would expect. For example, let F be the formula $\forall x_2(x_5 < x_2 \vee x_5 \approx x_2)$. Suppose that the interpretation V is $\mathbf{N}$, together with its usual ordering, etc. Then the definition below will give the set $F^{\mathbf{N}}$ to consist of exactly those infinite sequences of natural numbers which have 0 in the 5th slot. The point is that this formula F is not a sentence. An interpretation on its own is not enough data to fix the truth value of F. We need to also give a $\underline{v}$. The formula F has x_5 as a free variable, so it 'expresses a property of x_5'. And that property holds only for $x_5 = 0$. So, once all the variables are fixed by choosing a specific sequence $\underline{v}$, only then does F 'take a definite truth value'. But how the variables other than x_5 are fixed has no effect on the answer. In general, it will turn out that the slots corresponding to variables which don't occur freely in a formula will not affect the truth value of that formula. This will be proved right after all the definitions are given.

Definition. Given an interpretation $(V\,;\ \cdots c^V \cdots\ ;\ \cdots f^V \cdots\ ;\ \cdots r^V \cdots)$ of our language, inductively define certain sets of $\underline{v}$'s, to be named F^V, for formulae F in the language, as follows.
(Really, F^V will be the set of $\underline{v}$ for which "F is true at $\underline{v}$ from V". But we'll define F^V first, then say that the quote above **means** $\underline{v} \in F^V$) .

(i) For any terms s and t, and when F is $t \approx s$, define

$$\underline{v} \in (t \approx s)^V \quad \Longleftrightarrow \quad t^{\underline{v}} = s^{\underline{v}} \,.$$

In other words, [$t \approx s$ is true at $\underline{v}$ from V] if and only if $t^{\underline{v}}$ and $s^{\underline{v}}$ are the same element of V.

(ii) For any terms $t_1, \cdots, t_n$ and n-ary relation symbol r, and when F is $rt_1t_2 \ldots t_n$, define

$$\underline{v} \in (rt_1 \ldots t_n)^V \quad \Longleftrightarrow \quad (t_1^{\underline{v}}, t_2^{\underline{v}}, \cdots, t_n^{\underline{v}}) \in r^V \ .$$

That is, $[rt_1 \ldots t_n$ is true at $\underline{v}$ from $V]$ if and only if $(t_1^{\underline{v}}, t_2^{\underline{v}}, \cdots, t_n^{\underline{v}})$ is a member of the set r^V. The latter just says that the n-tuple $(t_1^{\underline{v}}, t_2^{\underline{v}}, \cdots, t_n^{\underline{v}})$ of elements from V are related under the relation r^V .

So far, we have taken care of the initial cases, dealing with those F which are atomic formulae. Now we'll deal with the inductive cases of those F which are built from smaller formula by using "$\neg$" or "$\wedge$" or "$\forall x$".

(iii) Define

$$\underline{v} \in (\neg F)^V \quad \Longleftrightarrow \quad \underline{v} \notin F^V \ .$$

In other words, $\neg F$ is true at $\underline{v}$ from V if and only if F is not true at $\underline{v}$ from V .

(iv) Define

$$\underline{v} \in (F \wedge G)^V \quad \Longleftrightarrow \quad \underline{v} \in F^V \cap G^V \ ,$$

i.e. $\underline{v}$ is in both F^V and G^V
In other words, $F \wedge G$ is true at $\underline{v}$ from V if and only if both of F and G are true at $\underline{v}$ from V .

(v) Define

$$\underline{v} \in (\forall x_3 F)^V \quad \Longleftrightarrow \quad (v_1, v_2, u, v_4, v_5, \cdots) \in F^V \quad \text{for all } u \text{ in } V \ .$$

In other words, $\forall x_3 F$ is true at $\underline{v}$ from V if and only if F is true at $\underline{u}$ from V for all $\underline{u}$ which agree with $\underline{v}$ everywhere except possibly in the 3rd slot.

What does (v) say? It says to keep all the slots in $\underline{v}$ fixed except the 3rd, but allow anything, u, into the 3rd slot, and 'check' the truth of F at all these. Of course this is just what the universal quantifier $\forall x_3$ expects us to do. For defining the meaning of $\underline{v} \in (\forall x_{17} F)^V$, you make the obvious modification, changing "3rd" to "17th" above, or in the precise definition, change $(v_1, v_2, u, v_4, v_5, \cdots)$ to $(v_1, v_2, \cdots, v_{16}, u, v_{18}, \cdots)$. I'll single out x_3 this way many times below to make things more readable. It will always be

pretty obvious how to write it out correctly in general, by employing some extra " $\cdots\cdots$'s ".

In (iii), (iv) and (v), the definition by induction relies on you remembering that F is a 'smaller' formula, so the set F^V has 'already been defined'.

Now the hammer comes down several times.

Definition. Re-iterating: the statement "The formula F is true at $\underline{v}$ from V" is another way of saying $\underline{v} \in F^V$.

Definition. The statement "The formula F is false at $\underline{v}$ from V" is another way of saying $\underline{v} \notin F^V$.

So any formula, not just one which is a sentence, is either true or false (but not both) at $\underline{v}$ from V.

Definition. The statement "The formula F is true in V" means F is true at $\underline{v}$ from V for all $\underline{v}$. Thus F^V would contain every $\underline{v}$ in this case.

Definition. The statement "The formula F is false in V" means F is false at $\underline{v}$ from V for all $\underline{v}$. Thus F^V would be empty in this case.

Now for many formulae which are not sentences, there will be some $\underline{v}$ where it is true, and some where it is false.

Definition. The statement "The formula F is *truthwise indeterminate* in V" means that F is neither true nor false in V. Thus F^V would be neither empty nor contain all $\underline{v}$ in this case.

Remark. We shall show in the next section that a **sentence** is never truthwise indeterminate in V; it is **either true or false**, as you would expect from all the discussion in the previous chapter. This will be a special case of the fact that, for any formula F, whether or not F is true at $\underline{v}$ from V depends only on $(v_1, \cdots, v_n)$, if no x_i occurs freely in F for $i > n$. The slots in $\underline{v}$, beyond the point where F has no free variable with those subscripts, are irrelevant to the truth status of F.

Definition. A formula F is *logically valid* if and only if it is true in V for all interpretations V of the language.

Exercise 6.1. Find the place above where one can see that allowing the empty set to be an interpretation would result in all formulae being both true and false in that "interpretation". You don't need to go back to the technical stuff to see this.

6.2 Sentences are not indeterminate.

Since the slots in $\underline{v}$ are there to substitute for free variables, being true at $\underline{v}$ from V should depend only on those slots for which F actually has a free occurrence of the corresponding variable. To aid readability, we'll do the case when those slots include at most the first two.

Theorem 6.1 "for $i \geq 3$". *If x_i has no free occurrence in F for any $i \geq 3$, then with $\underline{u}= (v_1, v_2, u_3, u_4, u_5, \cdots)$ and $\underline{w}= (v_1, v_2, w_3, w_4, w_5, \cdots)$, we have*

$$\underline{u} \in F^V \quad \Longleftrightarrow \quad \underline{w} \in F^V \ .$$

In other words,

$$F \text{ is true at } \underline{u} \text{ from } V \quad \Longleftrightarrow \quad F \text{ is true at } \underline{w} \text{ from } V \ .$$

Theorem 6.1 "for $i = 3$". *If x_3 has no free occurrence in F, then with $\underline{u}= (v_1, v_2, u, v_4, v_5, \cdots)$ and $\underline{w}= (v_1, v_2, w, v_4, v_5, \cdots)$, we have*

$$\underline{u} \in F^V \quad \Longleftrightarrow \quad \underline{w} \in F^V \ .$$

Remark. The second version, "for $i = 3$", has a proof which is almost identical to the first, so it will be left as an exercise. The fact is that these results are rather obvious, and the proofs are technical, so the reader might wish to skip the proofs on first reading.

Proof. We proceed by induction. For atomic formulae we need the following lemma.

Lemma 6.2 *If t is a term which includes no x_i for any $i \geq 3$, then $t^{\underline{u}} = t^{\underline{w}}$, where $\underline{u}$ and $\underline{w}$ are as in the theorem—that is, $\underline{u}$ and $\underline{w}$ agree in their first two slots.*

Proof. When t is a constant symbol c, we have $t^{\underline{u}} = t^{\underline{w}} = c^V$, quite independently of any fact about $\underline{u}$ and $\underline{w}$.

When t is a variable x_i, then $i = 1$ or 2 by our assumption, and so $t^{\underline{u}} = t^{\underline{w}} = v_i$.

When t is $ft_1 \dots t_n$, where no t_j includes x_i for for any $i \geq 3$ and so, by induction, $t_j^{\underline{u}} = t_j^{\underline{w}}$, we have

$$t^{\underline{u}} = f^V(t_1^{\underline{u}}, \cdots, t_n^{\underline{u}}) = f^V(t_1^{\underline{w}}, \cdots, t_n^{\underline{w}}) = t^{\underline{w}} ,$$

completing the proof.

Continuation of proof of 6.1. To do the initial cases of atomic formulae, using the lemma,

$$\underline{u} \in (t \approx s)^V \iff t^{\underline{u}} = s^{\underline{u}} \iff t^{\underline{w}} = s^{\underline{w}} \iff \underline{w} \in (t \approx s)^V ,$$

and

$$\underline{u} \in (rt_1 \dots t_n)^V \Leftrightarrow (t_1^{\underline{u}}, \cdots, t_n^{\underline{u}}) \in r^V \Leftrightarrow (t_1^{\underline{w}}, \cdots, t_n^{\underline{w}}) \in r^V \Leftrightarrow \underline{w} \in (rt_1 \dots t_n)^V.$$

Now we do the inductive steps.

$$\underline{u} \in (\neg F)^V \iff \underline{u} \notin F^V \iff \underline{w} \notin F^V \iff \underline{w} \in (\neg F)^V .$$

$$\underline{u} \in (F \wedge G)^V \iff \underline{u} \in F^V \text{ and } \underline{u} \in G^V$$
$$\iff \underline{w} \in F^V \text{ and } \underline{w} \in G^V \iff \underline{w} \in (F \wedge G)^V .$$

$$\underline{u} \in (\forall x_2 F)^V \iff \text{for all } v \text{ we have } (v_1, v, u_3, u_4, \cdots) \in F^V$$
$$\iff \text{for all } v \text{ we have } (v_1, v, w_3, w_4, \cdots) \in F^V \iff \underline{w} \in (\forall x_2 F)^V .$$

The argument is the same for $\forall x_1 F$. Notice that this uses the inductive hypothesis for *all* v_1, v, not just for the 'fixed' v_1, v_2.

$$\underline{u} \in (\forall x_3 F)^V \iff \text{for all } v \text{ we have } (v_1, v_2, v, u_4, \cdots) \in F^V$$
$$\iff \text{for all } v \text{ we have } (v_1, v_2, v, w_4, \cdots) \in F^V \iff \underline{w} \in (\forall x_3 F)^V .$$

The argument is the same for $\forall x_i F$ for any $i \geq 4$. This one only uses the inductive hypothesis for the 'fixed' v_1, v_2.

This completes the proof.

Theorem 6.1—the general case "for $i \geq n$" or "for $i = n$" . The statement and proof of each should be clear, and will be left for the reader.

Corollary 6.3 *If F is a* **sentence**, *then F^V is either empty or consists of all $\underline{v}$; equivalently, F is either false in V or true in V.*

Proof. A sentence, by definition, has no free variables x_i for any $i \geq 1$, so that case of the theorem can be applied. It actually says that if *even one* $\underline{v}$ is in F^V, then *every* $\underline{v}$ is in F^V, as required.

Exercises.

6.2 Show that F is true in V if and only if $\forall x F$ is true in V.

6.3 A closure for a *formula* F is any *sentence* obtained by putting "$\forall x$" on the lefthand end of F a number of times, for variables x, which might as well be all different. Up to the order of appearance of these universally quantified variables, we can make the closure unique by appending only the variables which actually have a free occurrence in F. Deduce easily from (i) that F is true in V if and only if any of its closures is true in V.

6.4 Show that F is false in V if and only if $\exists x F$ is false in V.

Remarks. Just as 6.3 relates to 6.2, we could define an 'existential closure', and formulate a companion exercise to 6.4. These results show how truth in an interpretation for arbitrary formulae can be 'reduced', in a sense, to the truth (in that interpretation) of sentences.

Of course for non-sentence formulae F which happen to be truthwise indeterminate in V, it will be the case that the (universal) closure is false in V, and the existential closure is true in V. For example, take F to be $x_5 < x_2$. Obviously, $\forall x_2 \forall x_5\ x_5 < x_2$ is false in $\mathbf{N}$, and $\exists x_2 \exists x_5\ x_5 < x_2$ is true in $\mathbf{N}$.

6.3 A formula which is NOT logically valid, (but could be mistaken for one)

Much in the upcoming sections consists of showing certain formulae to be logically valid. Before doing that, here is one which isn't, from the language of 1$^{\underline{\text{st}}}$order number theory :

$$(1+1)+1 \ \approx\ 1+(1+1)$$

To verify this, we must find **an interpretation in which this sentence is false**.

Let $V = \{a, b\}$, where $a \neq b$; that is, V is a 2-element set. You can specify 0^V to be either of those elements. You can specify the relation $<^V$ on V and the function $\times^V : V \times V \to V$ however you like. But I insist that $1^V := a$, and that

$$a +^V a := b \quad ; \quad b +^V a := a \quad ; \quad a +^V b := b \quad .$$

You can specify $b +^V b$ to be whichever of the two elements you prefer.

Then the term $(1+1)^{\underline{v}}$ is

$$1^{\underline{v}} +^V 1^{\underline{v}} \;=\; a +^V a \;=\; b \quad .$$

Thus

$$((1+1)+1)^{\underline{v}} \;=\; (1+1)^{\underline{v}} +^V 1^{\underline{v}} \;=\; b +^V a \;=\; a \quad ,$$

and

$$(1+(1+1))^{\underline{v}} \;=\; 1^{\underline{v}} +^V (1+1)^{\underline{v}} \;=\; a +^V b \;=\; b \quad .$$

So the latter two terms produce different answers when '$\underline{v}$th powered'. From Tarski's definition, this says that $\underline{v} \notin [(1+1)+1 \;\approx\; 1+(1+1)]^V$ for any $\underline{v}$. Thus that given formula is false in this interpretation, as required. (We only needed *one* such $\underline{v}$, but because the formula is a sentence, it had to follow for all $\underline{v}$.)

Constructing finite examples of interpretations like this, often rather artificial, is quite popular with those who are primarily interested in using logic to do **model theory**. We won't do much of that. But it is important to emphasize that interpretations can be quite arbitrary. They need not have anything particularly natural about them. You will often see many finite 'multiplication' tables in such texts, specifying examples of binary functions tailor-made to provide counterexamples to one or another assertion, as we did above.

Without wishing to belabour the obvious, I'll state the following to emphasize that we're using (informal) language very carefully here.

The formula $(1+1)+1 \;\approx\; 1+(1+1)$ is not true; it's not false; it's not a fact of logic; it's not a theorem of mathematics. It's simply a formula, a special kind of string of symbols.

The statement "the formula $(1+1)+1 \;\approx\; 1+(1+1)$ is not logically valid, but it is true in the interpretations $\mathbf{N}$, $\mathbf{Z}$ and $\mathbf{R}$" is a true statement (whether within or about logic or mathematics, is not important here). It could be

argued that it's the first theorem of mathematics. It's as basic an example related to associativity as there is. That's also true for commutativity : we could define a short form term by saying that "2" means 1+1. Then the formula becomes "2" $+ 1 \approx 1 +$ "2" , which is a commutativity assertion.

6.4 Some logically valid formulae; checking truth with $\vee, \rightarrow$ and $\exists$.

Now the formulae $1 \approx 1$ and $1 + 1 \approx 1 + 1$ certainly **are** logically valid. They are special cases of the very easily checked fact (later) that $t \approx t$ is logically valid for any term t in any $1^{\underline{\text{st}}}$order language. So these probably shouldn't be called theorems of mathematics! But who cares where the boundary is between logic and mathematics? Russell believed that it doesn't exist.

To show directly from the definition that a formula F is logically valid, you show that, for each V, every $\underline{v}$ is in F^V . A simple example is $x_3 \approx x_3$. This makes sense in any $1^{\underline{\text{st}}}$order language, since all of ours have the equality symbol. It will of course hold for any other variable as well. Let's do the direct check :

$$\underline{v} \in (x_3 \approx x_3)^V \iff x_3^{\underline{v}} = x_3^{\underline{v}} \iff v_3 = v_3 \ ,$$

which clearly holds (without needing that last step!)

Exercise 6.5 Can you just substitute 1 for x in the logically valid formula $x \approx x$, and use that as an argument that $1 \approx 1$ is also logically valid ?

Proposition 6.4 *If F is logically valid, then so is $\forall x F$, for any x .*

Proof. For example, when x is x_3, we have

$$\underline{v} \in (\forall x_3 F)^V \iff \text{for all } u \text{ we have } (v_1, v_2, u, v_4, \cdots) \in F^V \ .$$

But the right-hand side holds because F is logically valid. The proof for other variables x is the 'same'.

So now we know that $\forall x_3 \ x_3 \approx x_3$ is logically valid—hardly a surprise.

Also $\forall x_7 \ x_3 \approx x_3$ is logically valid—one of our "silly" formulae.

What about the abbreviated formula $\exists x_3\ x_3 \approx x_3$? It *is* logically valid, as can be seen by writing the actual formula $\neg\forall x_3\neg\ x_3 \approx x_3$ and checking directly :

$$\underline{v} \in (\neg\forall x_3\neg\ x_3 \approx x_3)^V \iff \underline{v} \notin (\forall x_3\neg\ x_3 \approx x_3)^V$$

$$\iff \text{for some } u \text{ we have } (v_1, v_2, u, v_4, \cdots) \notin (\neg x_3 \approx x_3)^V$$

$$\iff \text{for some } u \text{ we have } (v_1, v_2, u, v_4, \cdots) \in (x_3 \approx x_3)^V\ ,$$

as checked just before **6.4**. But the last statement doesn't just hold for some u, it holds for all u. And V isn't empty!

Clearly it would be good to have some criteria, such as occurred in the definition of F^V, but applying directly to abbreviated formulae involving $\exists x$, $\rightarrow$, $\vee$, and $\leftrightarrow$. I'll leave the last one as an exercise for you to both formulate and prove the criterion. As for the other three :

Theorem 6.5 *We have the following criteria.*

(i)

$$\underline{v} \in (\exists x_3 F)^V \iff \text{there is at least one } u \in V \text{ with } (v_1, v_2, u, v_3, v_4, \cdots) \in F^V\ .$$

In other words, $\exists x_3 F$ is true at $\underline{v}$ from V if and only if F is true at $(v_1, v_2, u, v_3, v_4, \cdots)$ for some u .

(ii)

$$\underline{v} \in (F \rightarrow G)^V \iff \text{either } \underline{v} \notin F^V \text{ or } \underline{v} \in G^V \ \ (\text{or both}).$$

In other words, $F \rightarrow G$ is true at $\underline{v}$ from V if and only if [either F is false at $\underline{v}$, or G is true at $\underline{v}$, or both].

(iii)

$$\underline{v} \in (F \vee G)^V \iff \text{either } \underline{v} \in F^V \text{ or } \underline{v} \in G^V \ \ (\text{or both}).$$

In other words, $F \vee G$ is true at $\underline{v}$ from V if and only if [either F is true at $\underline{v}$, or G is true at $\underline{v}$, or both].

It is clear how to modify (i) for variables other than x_3, and that will be left to the reader.

Proof. (i) This mimics the example before the theorem.

$$\underline{v} \in (\exists x_3 F)^V \iff \underline{v} \in (\neg\forall x_3 \neg\, F)^V \iff \underline{v} \notin (\forall x_3 \neg\, F)^V$$

$$\iff \text{for some } u \text{ we have } (v_1, v_2, u, v_4, \cdots) \notin (\neg F)^V$$

$$\iff \text{for some } u \text{ we have } (v_1, v_2, u, v_4, \cdots) \in F^V \ .$$

(ii)

$$\underline{v} \in (F \to G)^V \iff \underline{v} \in (\neg(F \wedge \neg G))^V \iff \underline{v} \notin (F \wedge \neg G)^V$$

$$\iff \text{either } \underline{v} \notin F^V \text{or } \underline{v} \notin (\neg G)^V$$

$$\iff \text{either } \underline{v} \notin F^V \text{or } \underline{v} \in G^V \ .$$

(iii)

$$\underline{v} \in (F \vee G)^V \iff \underline{v} \in (\neg(\neg F \wedge \neg G))^V \iff \underline{v} \notin (\neg F \wedge \neg G)^V$$

$$\iff \text{either } \underline{v} \notin (\neg F)^V \text{or } \underline{v} \notin (\neg G)^V$$

$$\iff \text{either } \underline{v} \in F^V \text{or } \underline{v} \in G^V \ .$$

This completes the proof.

Now let's look at a couple of formulae mentioned in the first paragraph of this chapter, and answer the questions asked there.

Firstly, I claim that $\forall x F \to F$ *is* logically valid : Using **6.5**(ii),

$$\underline{v} \in (\forall x_3 F \to F)^V \iff \text{either } \underline{v} \notin (\forall x_3 F)^V \text{ or } \underline{v} \in F^V \quad \text{(or both)}$$

$$\iff \text{for some } u, (v_1, v_2, u, v_4, \cdots) \notin F^V \ , \text{ or else } (v_1, v_2, v_3, v_4, \cdots) \in F^V \ ,$$

which is clearly true, one could say profligately true, since both could easily hold. Modifying this for x other than x_3 is easy to do.

Proposition 6.6 *There are examples of F for which $F \to \forall x F$ is* **not** *logically valid.*

Proof. Take $x = x_3$ and $F = \forall x_2(x_3 < x_2 \vee x_3 \approx x_2)$. [Or if you prefer graphs, take $F = \forall x_2(x_3 r x_2 \vee x_3 \approx x_2)$.] Then the sentence $\forall x_3 F$ is clearly false in **N**. [Or ... in any graph which has a pair of distinct vertices

which are not connected by an edge in one or the other of the two directions.] You don't need Tarski's definition of truth to tell you that, but maybe you should check it in that format as an exercise. On the other hand F itself is true at $(v_1, v_2, 0, v_4, \cdots)$ from $\mathbf{N}$. [Or ... at any $(v_1, v_2, v_3, v_4, \cdots)$ from a graph in which there is a directed edge from v_3 to every other vertex.] Again the formal definition of truth in an interpretation isn't really needed, but again it's not a bad exercise for practice with the definition. So neither of the $\underline{v}$ mentioned just above are in $(F \to \forall x_3 F)^V$, as required to show that the formula is not logically valid. We have used the criterion in **6.5**(ii) in this last statement.

We shall come back to this example several times quite soon, because despite the result, $\forall x F$ can be 'inferred' from F, in one sense of that word. There is some subtlety here, despite the impression one gets that Tarski's definition is 'belabouring the obvious'. See the final section of this chapter, which can be read at this point with no loss of continuity.

6.5 Summary of the following four sections.

On first reading, you might prefer to skip the proofs in the next four sections, and proceed quickly to the last section of this chapter. These four sections are devoted to proving that the formulae in (i) to (iv) below are logically valid. We need to know this for the next chapter. There these formulae will be taken as axioms in the proof system. Combined with results of the last section of this chapter, we'll then immediately have soundness for the proof system, that it's not capable of 'proving something that's untrue'. The formulae are:

(i) The $1^{\underline{\text{st}}}$order formula G^*, for any propositional tautology G, where G^* is obtained by substituting, for each occurrence of each propositional variable P_i in G, some $1^{\underline{\text{st}}}$order formula F_i. These G^* are the $1^{\underline{\text{st}}}$*order tautologies.*

(ii) The formula $\forall x F \ \to \ F^{[x \to t]}$, where : x is any variable; F is any formula; $F^{[x \to t]}$ is obtained by substituting the term t for every *free* occurrence of x in F; and t is any term such that no occurrence of any variable involved in the term becomes a bound occurrence after the substitutions are made.

(iii) The formula $\forall x(F \ \to G) \ \to \ (F \to \forall x G)$, for any formula G, and any formula F which has no free occurrences of x.

(iv) The five 'equality formulae' : for all terms t, s, s_1, s_2, and s_3, all formulae F in which s and t are substitutable for x , and all strings A and B of variables, constant symbols and function symbols for which AsB is a term :

$$s \approx s \quad ; \quad s \approx t \ \rightarrow t \approx s \quad ; \quad (s_1 \approx s_2 \ \wedge \ s_2 \approx s_3) \ \rightarrow \ s_1 \approx s_3 \ ;$$

$$s \approx t \ \rightarrow \ (F^{[x \mapsto s]} \rightarrow F^{[x \mapsto t]}) \quad ; \quad s \approx t \rightarrow AsB \approx AtB \ .$$

6.6 $1^{\underline{\text{st}}}$order tautologies are logically valid.

Given a $1^{\underline{\text{st}}}$order language, the idea of a tautology in that language is intuitively fairly clear—they are obtained by replacing, in a tautology from propositional logic, the propositional variable P_1 by some formula F_1 from the $1^{\underline{\text{st}}}$order language, replacing P_2 by some formula F_2, etc. To prove that all such $1^{\underline{\text{st}}}$order tautologies are logically valid, it's best to define such a tautology more carefully by induction.

First make a list $F_1, F_2, F_3, \cdots$ of all the formulae in the given $1^{\underline{\text{st}}}$order language. This *can* be done, and it doesn't matter how you choose to do it. Now for any formula G in propositional logic, define a formula G^* in our $1^{\underline{\text{st}}}$order language inductively by requiring

$$P_i^* := F_i \quad ; \quad (\neg G)^* := \neg(G^*) \quad ; \quad (G \wedge H)^* := G^* \wedge H^* \ .$$

A little thought and you'll agree that G^* really is 'G with P_i replaced by F_i for all i', as in the previous paragraph. But the inductive definition is easier to use.

Now given an interpretation V of our $1^{\underline{\text{st}}}$order language, and given $\underline{v}$ from V, we can define something like a truth assignment, but on our $1^{\underline{\text{st}}}$order formulae, as follows :

$$\mathsf{f}_{\underline{v}}(F) \ := \ \begin{cases} \mathsf{Tr} & \text{if } \ \underline{v} \in F^V \ ; \\ \mathsf{Fs} & \text{if } \ \underline{v} \notin F^V \ . \end{cases}$$

It is easy to verify that

$$\mathsf{f}_{\underline{v}}(\neg F) = \mathsf{Tr} \iff \mathsf{f}_{\underline{v}}(F) = \mathsf{Fs} \ ,$$

and

$$\mathsf{f}_{\underline{v}}(F \wedge G) = \mathsf{Tr} \iff \mathsf{f}_{\underline{v}}(F) = \mathsf{Tr} = \mathsf{f}_{\underline{v}}(G) \ .$$

Now let's define an actual truth assignment by specifying it on the propositional variables by $\mathsf{e}_{\underline{v}}(P_i) := \mathsf{f}_{\underline{v}}(F_i)$. Then $\mathsf{e}_{\underline{v}}$ is determined on formulae in general as usual by induction.

Lemma 6.7 *For all propositional formulae G, we have* $\mathsf{f}_{\underline{v}}(G^*) = \mathsf{e}_{\underline{v}}(G)$.

Proof. Proceed by induction on propositional formulae G.
For $G = P_i$, we have

$$\mathsf{f}_{\underline{v}}(P_i^*) = \mathsf{f}_{\underline{v}}(F_i) = \mathsf{e}_{\underline{v}}(P_i) \ .$$

For $G = \neg G_1$, where the result holds for G_1, we have

$$\mathsf{f}_{\underline{v}}((\neg G_1)^*) = \mathsf{Tr} \iff \mathsf{f}_{\underline{v}}(\neg(G_1^*)) = \mathsf{Tr} \iff \mathsf{f}_{\underline{v}}(G_1^*) = \mathsf{Fs}$$

$$\iff \mathsf{e}_{\underline{v}}(G_1) = \mathsf{Fs} \iff \mathsf{e}_{\underline{v}}(\neg G_1) = \mathsf{Tr} \ .$$

Thus $\mathsf{f}_{\underline{v}}((\neg G_1)^*) = \mathsf{e}_{\underline{v}}(\neg G_1)$.
For $G = G_1 \wedge G_2$, where the result holds for G_1 and G_2 , we have

$$\mathsf{f}_{\underline{v}}((G_1 \wedge G_2)^*) = \mathsf{Tr} \iff \mathsf{f}_{\underline{v}}(G_1^* \wedge G_2^*) = \mathsf{Tr} \iff \mathsf{f}_{\underline{v}}(G_1^*) = \mathsf{Tr} = \mathsf{f}_{\underline{v}}(G_2^*)$$

$$\iff \mathsf{e}_{\underline{v}}(G_1) = \mathsf{Tr} = \mathsf{e}_{\underline{v}}(G_2) \iff \mathsf{e}_{\underline{v}}(G_1 \wedge G_2) = \mathsf{Tr} \ .$$

Thus $\mathsf{f}_{\underline{v}}((G_1 \wedge G_2)^*) = \mathsf{e}_{\underline{v}}(G_1 \wedge G_2)$.
This completes the proof.

Corollary 6.8 and Definition. *If G is a propositional tautology, then G^* is a logically valid formula in any* $1^{\underline{\text{st}}}$*order language*—and we call the formulae G^* (as G ranges over propositional tautologies) the **tautologies in the** $1^{\underline{\text{st}}}$**order language.** (It is straightforward to see that the set of $1^{\underline{\text{st}}}$order tautologies doesn't depend on the initial ordering of the $1^{\underline{\text{st}}}$order formulae into a sequence, basically because permuting the propositional variables does not change a propositional tautology into a non-tautology.)

Proof. Fix any V and any $\underline{v}$ from V. We have $\mathsf{e}_{\underline{v}}(G) = \mathsf{Tr}$ since it is a truth value of a propositional tautology. Thus, by the theorem, $\mathsf{f}_{\underline{v}}(G^*) = \mathsf{Tr}$. By the definition of $\mathsf{f}_{\underline{v}}$, we get $\underline{v} \in (G^*)^V$, that is, G^* is true at $\underline{v}$ from V, as required.

I suppose one might be tempted to say that the tautologies in a $1^{\underline{\text{st}}}$order language are more than just logically valid—they're tautologically valid !

Our earlier examples, $x \approx x$ and $\forall x F \rightarrow F$, of logically valid formulae are definitely not $1^{\underline{\text{st}}}$order tautologies. We are beginning to see some interesting distinctions. Note, for example, that

$$F \vee \neg F \qquad \text{and} \qquad (\forall x F) \vee \neg(\forall x F)$$

are $1^{\underline{\text{st}}}$order tautologies for any $1^{\underline{\text{st}}}$order formula F. But $\forall x(F \vee \neg F)$ is not a $1^{\underline{\text{st}}}$order tautology. However, by **6.4**, it is logically valid.

6.7 Substituting terms for variables.

We want to generalize the example mentioned just after the proof of **6.5**, and show that $\forall x F \rightarrow F^{[x \mapsto t]}$ is logically valid, where, as defined below, $F^{[x \mapsto t]}$ is the formula obtained by 'substituting the term t for all the free occurrences of the variable x in the formula F'.

It is possible to define such a substitution for just some occurrences, but we won't need that in this book, so won't consider it. There is a further restriction, concerning which terms will be substituted, basically that no variable in the term becomes 'quantified' by the substitution. It's simpler to leave that until after the definition. So, for now, let t be any term, and let F be any formula, both in any given $1^{\underline{\text{st}}}$order language. We define substitution by induction. First we must define substitution into terms.

Definition. Define $s^{[x_i \to t]}$ for all terms s by induction on s as follows.

If s is c, a constant symbol, define $c^{[x_i \to t]} := c$.

If s is x, a variable other than x_i, define $x^{[x_i \to t]} := x$.

If s is x_i , define $x_i^{[x_i \to t]} := t$.

If s is $f t_1 \dots t_n$, define

$$(f t_1 \dots t_n)^{[x_i \to t]} := f t_1^{[x_i \to t]} \dots t_n^{[x_i \to t]} \ .$$

Definition. Define $F^{[x_i \to t]}$ for all formulae F by induction on F as follows.

If F is $s_1 \approx s_2$, an atomic formula, define $(s_1 \approx s_2)^{[x_i \to t]} := s_1^{[x_i \to t]} \approx s_2^{[x_i \to t]}$.

If F is $r s_1 \dots s_n$, an atomic formula, define $(r s_1 \dots s_n)^{[x_i \to t]} := r s_1^{[x_i \to t]} \dots s_n^{[x_i \to t]}$.

Now inductively complete the definition :

If F is $\neg F_1$, define $(\neg F_1)^{[x_i \to t]} := \neg(F_1^{[x_i \to t]})$. Brackets <u>not</u> part of formula!

If F is $F_1 \wedge F_2$, define

$$(F_1 \wedge F_2)^{[x_i \mapsto t]} \;:=\; F_1^{[x_i \mapsto t]} \wedge F_2^{[x_i \mapsto t]} \;.$$

If F is $\forall x F_1$ for some variable x other than x_i , define

$$(\forall x F_1)^{[x_i \mapsto t]} \;:=\; \forall x (F_1^{[x_i \mapsto t]}) \;.$$

If F is $\forall x_i F_1$, define

$$(\forall x_i F_1)^{[x_i \mapsto t]} \;:=\; \forall x_i F_1 \;.$$

This completes the inductive definition.

Now for the **restriction on the use of substitution**.

First, here's an an example. The formula $F \;=\; \exists y \; x \approx y$ is easily shown to be logically valid. Now suppose we are in the language of number theory, and we substitute the term $y+1$ for x. The result is $F^{[x \mapsto y+1]} \;=\; \exists y \; y+1 \approx y$. This looks like nonsense. It is a perfectly good formula on the right-hand side, but it's very far from being logically valid. It's definitely false in the usual number theory interpretations. We want substitution to 'preserve truth', in particular to produce a logically valid formula when you substitute into a logically valid formula.

So from now on, **substitution will never be done unless t is substitutable for x in F** in the following sense :

Definition. We say that t *is substitutable for* x *in* F if and only if no variable which occurs in t has a new bound occurrence in the formula after the substitution.

Note that you never need to give this a second thought when the term being substituted only involves constants and functions (no variables). Note also that a degenerate case of substitution is substituting x for itself, which of course does satisfy the proviso and doesn't change F. Generalizing both comments is the fact that, if the term t involves no variable other than x, then it is substitutable for x in any formula. Recall the definition of $t^{\underline{v}}$ early in Section **6.1**.

Lemma 6.9 $\quad \underline{v} \in (F^{[x_3 \mapsto t]})^V \iff (v_1, v_2, t^{\underline{v}}, v_4, \cdots) \in F^V \;.$

This result is definitely false without the proviso on substitutability. As an exercise, find a counterexample.

Proof. This is proved by induction, and of course holds with x_3 changed to any x_i and the slot for $t^{\underline{v}}$ changed, with essentially the same proof.

First we must prove by induction on s that, for terms s and t,

$$(s^{[x_3 \to t]})^{\underline{v}} \;=\; s^{(v_1, v_2, t^{\underline{v}}, \cdots)} \; .$$

When s is a constant symbol c, both sides reduce to c^V. When it is the variable x_3, both sides reduce to $t^{\underline{v}}$. When it is any other variable x_i, both sides reduce to v_i. Inductively, when it is $f s_1 \dots s_n$, both sides reduce to

$$f^V(s_1^{(v_1, v_2, t^{\underline{v}}, v_4, \cdots)}, \cdots, s_n^{(v_1, v_2, t^{\underline{v}}, v_4, \cdots)}) \; .$$

Now for the induction to prove the lemma itself. Firstly,

$$\underline{v} \in ((s_1 \approx s_2)^{[x_3 \to t]})^V \iff \underline{v} \in (s_1^{[x_3 \to t]} \approx s_2^{[x_3 \to t]})^V \iff (s_1^{[x_3 \to t]})^{\underline{v}} = (s_2^{[x_3 \to t]})^{\underline{v}}$$

$$\iff s_1^{(v_1, v_2, t^{\underline{v}}, \cdots)} = s_2^{(v_1, v_2, t^{\underline{v}}, \cdots)} \iff (v_1, v_2, t^{\underline{v}}, \cdots) \in (s_1 \approx s_2)^V \; .$$

Also,

$$\underline{v} \in ((r s_1 \dots s_n)^{[x_3 \to t]})^V \iff \underline{v} \in (r s_1^{[x_3 \to t]} \dots s_n^{[x_3 \to t]})^V$$

$$\iff ((s_1^{[x_3 \to t]})^{\underline{v}}, \cdots, (s_n^{[x_3 \to t]})^{\underline{v}}) \in r^V \iff (s_1^{(v_1, v_2, t^{\underline{v}}, \cdots)}, \cdots, s_n^{(v_1, v_2, t^{\underline{v}}, \cdots)}) \in r^V$$

$$\iff (v_1, v_2, t^{\underline{v}}, \cdots) \in (r s_1 \dots s_n)^V \; .$$

That does the initial steps. Now for the inductive steps. Firstly,

$$\underline{v} \in ((\neg F)^{[x_3 \to t]})^V \iff \underline{v} \in (\neg (F^{[x_3 \to t]}))^V \iff \underline{v} \notin (F^{[x_3 \to t]})^V$$

$$\iff (v_1, v_2, t^{\underline{v}}, \cdots) \notin F^V \iff (v_1, v_2, t^{\underline{v}}, \cdots) \in (\neg F)^V \; .$$

Next,

$$\underline{v} \in ((F \wedge G)^{[x_3 \to t]})^V \iff \underline{v} \in (F^{[x_3 \to t]} \wedge G^{[x_3 \to t]})^V$$

$$\iff \underline{v} \in (F^{[x_3 \to t]})^V \text{ and } \underline{v} \in (G^{[x_3 \to t]})^V$$

$$\iff (v_1, v_2, t^{\underline{v}}, \cdots) \in F^V \text{ and } (v_1, v_2, t^{\underline{v}}, \cdots) \in G^V$$

$$\iff (v_1, v_2, t^{\underline{v}}, \cdots) \in (F \wedge G)^V \; .$$

When the formula is $\forall x_3 F$,

$$\underline{v} \in ((\forall x_3 F)^{[x_3 \to t]})^V \iff \underline{v} \in (\forall x_3 F)^V$$

$$\iff \text{ for all } u,\ (v_1, v_2, u, v_4, \cdots) \in F^V \iff (v_1, v_2, t^{\underline{v}}, \cdots) \in (\forall x_3 F)^V \ .$$

When the formula is $\forall x_j F$ for any $j \neq 3$, the argument is the 'same' as the one below with $j = 2$:

$$\underline{v} \in ((\forall x_2 F)^{[x_3 \to t]})^V \iff \underline{v} \in (\forall x_2 (F^{[x_3 \to t]}))^V$$

$$\iff \text{ for all } u,\ (v_1, u, v_3, \cdots) \in (F^{[x_3 \to t]})^V$$

$$\iff \text{ for all } u,\ (v_1, u, t^{(v_1, u, v_3, \cdots)}, v_4, \cdots) \in F^V$$

$$\iff (v_1, v_2, t^{(v_1, u, v_3, \cdots)}, v_4, \cdots) \in (\forall x_2 F)^V \iff (v_1, v_2, t^{\underline{v}}, \cdots) \in (\forall x_2 F)^V \ ,$$

the last because $t^{(v_1, u, v_3, \cdots)} = t^{\underline{v}}$, since t cannot involve x_2, being substitutable for x_3 in $\forall x_2 F$.

Theorem 6.10
The formula

$$\forall x F \to F^{[x \to t]}$$

is always logically valid—that is, for all F, all x, and all terms t, the latter being substitutable for x by our proviso above.

Note that $(\forall x \exists y\ x \approx y) \to \exists y\ y + 1 \approx y$ is definitely *not* a logically valid formula, so the proviso on substitutability is certainly needed in this result. We don't seem to use the proviso in the proof of **6.10** just after the following exercises. However it is definitely used in the proof of **6.9** above, and that result is used in the proof of **6.10**.

By **6.10**, $(\forall x \exists y\ x \approx y) \to \exists y\ t \approx y$ is a logically valid formula in any $1^{\underline{\text{st}}}$order language. But so is $\forall x \exists y\ x \approx y$.

N.B. Here and in the exercises below, the term t is substitutable for x in $\exists y\ x \approx y$. That just says that t has no occurrences of y in it.

Exercises.

6.6 Show directly that $\exists y\ x \approx y$ is logically valid. (Do it for $y = x_5$ and $x = x_3$, then wave your hands like I do !)

6.7 Show that $\forall x \exists y\ x \approx y$ is logically valid—it is easier to use 6.6 and a previous general result.

6.8 Show directly that $(\forall x \exists y\ x \approx y) \to \exists y\ t \approx y$ is logically valid, reinforcing the example above.

6.9 Show that if F and $F \to G$ are both logically valid, then so is G.

6.10 Deduce from 6.8 and 6.9 that $\exists y\ t \approx y$ is logically valid.

6.11 Prove the fundamental result that if F is logically valid, then so is $F^{[x \to t]}$. (Remember the proviso ! Also recall 6.5)

6.12 Deduce from 6.11 and 6.6 once again that $\exists y\ t \approx y$ is logically valid.

Proof of 6.10 (i) We'll prove it with $x = x_3$—the 'same' proof works in general.

$$\underline{v} \in (\forall x_3 F \to F^{[x_3 \to t]})^V \iff \underline{v} \notin (\forall x_3 F)^V \text{ or } \underline{v} \in (F^{[x_3 \to t]})^V$$

$$\iff \text{for some } u,\ (v_1, v_2, u, v_4, \cdots) \notin F^V \text{ or } (v_1, v_2, t^{\underline{v}}, v_4, \cdots) \in F^V \ .$$

This last condition (which used **6.9**) clearly holds for any $\underline{v}$ from V, for all V, as required.

Another proof of **6.10**(i) can be obtained from Exercise 6.11 above plus the following :

Exercise 6.13 Show that $(F \to G)^{[x \to t]} = F^{[x \to t]} \to G^{[x \to t]}$.

We know that $\forall x F \to F$ is logically valid from a long time ago (it is also a special case of **6.10**). So by Exercise 6.11, $(\forall x F \to F)^{[x \to t]}$ is also logically valid. Then by Exercise 6.13, $(\forall x F)^{[x \to t]} \to F^{[x \to t]}$ is logically valid. But $(\forall x F)^{[x \to t]}$ is just $\forall x F$ by definition, so we get our new proof of **6.10**(i).

Given a term t and a formula F, '**by renaming bound variables**', one can find a closely related formula, $\underline{\underline{F}}$, for which t is substitutable for any chosen free variable in $\underline{\underline{F}}$. Bound variables can be renamed, one at a time, one for each occurrence of a quantifier, with the only restriction on the new variable being that it doesn't have a free occurrence in the scope of the quantifier in question. By choosing to rename all bound variables using only variables not occurring in t, it is clear that the resulting formula $\underline{\underline{F}}$ has the property that t is substitutable into $\underline{\underline{F}}$ for any variable at all.

Proposition 6.11 *F is true at $\underline{v}$ if and only if $\underline{\underline{F}}$ is true at $\underline{v}$. That is, the formula $F \leftrightarrow \underline{\underline{F}}$ is logically valid.*

Proof. It is fairly clear that to prove inductively that $\underline{\underline{F}}$ inherits some property from F, it suffices to prove that $\forall y(G^{[x \to y]})$ inherits the property from $\forall xG$, for any formula G in which y has no free occurrence. (G would be the scope referred to above.) In the present situation, we have that F has the form $X(\forall xG)Y$ for some strings X and Y, and $\underline{\underline{F}}$ has the form $X(\forall y(G^{[x \to y]}))Y$ as above. Now in **6.15** at the end of this chapter we prove the analogue of **2.1**, the replacement theorem, namely that if $H \leftrightarrow J$ is logically valid, then so is $XHY \leftrightarrow XJY$, where X and Y are strings which make XHY into a formula.

So we are making use of that principle in this proof, since its proof is independent of this proposition. (The principle can be re-expressed in the form that XHY and XJY are truth equivalent as long as H and J are, once the appropriate definitions have been made in Section **8.2**.)

Below we shall also specialize the variables x and y to x_3 and x_2, as we often do, and let the reader formulate the general case.

Thus, we assume that x_2 has no free occurrence in G, and that $\forall x_3 G$ is true at $\underline{v}$. We must show that $\forall x_2(G^{[x_3 \to x_2]})$ is true at $\underline{v}$. Now

$$\forall x_3 G \text{ is true at } (v_1, v_2, v_3, \cdots)$$

$$\Longrightarrow \text{ for all } u,\ G \text{ is true at } (v_1, v_2, u, \cdots)$$

$$\Longrightarrow \text{ for all } u, w,\ G \text{ is true at } (v_1, w, u, \cdots)$$

$$\Longrightarrow \text{ for all } u,\ G \text{ is true at } (v_1, u, u, \cdots)$$

$$\Longrightarrow \text{ for all } u,\ G^{[x_3 \to x_2]} \text{ is true at } (v_1, u, v_3, \cdots)$$

$$\Longrightarrow \forall x_2(G^{[x_3 \to x_2]}) \text{ is true at } (v_1, v_2, v_3, \cdots)\ .$$

The second and fourth implications are justified by using **6.1**, the formulae involved being 'FOFOO' (this will mean **'free of free occurrences of'** from now on) some variable.

6.8 A logically valid formula relating $\rightarrow$ and quantifiers.

Here we check that

the formula : $\forall x(F \rightarrow G) \rightarrow (F \rightarrow \forall xG)$ *is logically valid, for any formula* G*, and any formula* F *which has no free occurrences of* x *.*

As often we do, the case $x = x_3$ must suffice to convince readers of their abilities to do it in general. We have

$$\underline{v} \in (\forall x_3(F \ \to G) \ \to \ (F \to \forall x_3 G))^V$$

$$\iff \underline{v} \notin (\forall x_3(F \ \to G))^V \ \text{ or } \ \underline{v} \in (F \to \forall x_3 G)^V$$

$$\iff \text{for some } u, \ (v_1, v_2, u, \cdots) \notin (F \ \to G)^V \ \text{ or}$$

$$\underline{v} \notin F^V \ \text{ or } \ \underline{v} \in (\forall x_3 G)^V$$

$$\iff \text{for some } u, \ [(v_1, v_2, u, \cdots) \in F^V \ \text{ and } \ (v_1, v_2, u, \cdots) \notin G^V] \ \text{ or}$$

$$\underline{v} \notin F^V \ \text{ or } \ \text{for all } w, \ (v_1, v_2, w, \cdots) \in G^V \ .$$

But, if the second and third conditions fail, let $(v_1, v_2, w, \cdots) \notin G^V$ for some choice of w. Then $(v_1, v_2, w, \cdots) \in F^V$ by **6.1**, since F is FOFOO x_3 and since $\underline{v} \in F^V$. Therefore, the first condition does hold, as required, by taking $u = w$.

Remark. This logically valid family is closely related to one of the 'rules for moving quantifiers' in the prenex section of Chapter 8.

6.9 Basic logically valid formulae involving $\approx$.

Having worked your way through the last three sections, you will have a good understanding of the phrase *logically valid*, and be perhaps getting a bit fed up of it. So it may be a relief to hear that this is the last of the sections devoted to establishing some general classes of logically valid formulae which can be used later as axioms in the proof system.

Firstly, $s \approx s$ is logically valid, simply because $s^{\underline{v}} = s^{\underline{v}}$ always.

Next, $s \approx t \ \to t \approx s$ is logically valid, because $s^{\underline{v}} = t^{\underline{v}}$ implies $t^{\underline{v}} = s^{\underline{v}}$.

The proof for $(s \approx t \ \wedge \ t \approx q) \ \to \ s \approx q$ is equally trivial.

As for $s \approx t \to (F^{[x \mapsto s]} \to F^{[x \mapsto t]})$, we are assuming that both the terms s and t are substitutable for x in F. Once again, we'll take $x = x_3$. We have

$$\begin{aligned}
&\underline{v} \in (s \approx t \to (F^{[x_3 \to s]} \to F^{[x_3 \to t]}))^V \\
\iff &\underline{v} \notin (s \approx t)^V \ \text{ or } \ \underline{v} \in (F^{[x_3 \to s]} \to F^{[x_3 \to t]})^V \\
\iff &s^{\underline{v}} \neq t^{\underline{v}} \ \text{ or } \ \underline{v} \notin (F^{[x_3 \to s]})^V \ \text{ or } \ \underline{v} \in (F^{[x_3 \to t]})^V \\
\iff &s^{\underline{v}} \neq t^{\underline{v}} \ \text{ or } \ (v_1, v_2, s^{\underline{v}}, \cdots) \notin F^V \ \text{ or } \ (v_1, v_2, t^{\underline{v}}, \cdots) \in F^V \ .
\end{aligned}$$

Clearly, if the first condition fails, then the second and third cannot both fail.

Finally we'll prove that $s \approx t \to AsB \approx AtB$ is logically valid. Here A and B are strings of symbols for which AsB is a term. From this it follows that, for any term t, the string AtB is also a term. (See the third section of Appendix B to Chapter 8.) Actually it is much better to rewrite AsB as $p^{[x \to s]}$, where p is the term AxB, and the variable x is chosen to be any variable not occurring in A nor B. (And so there's only the one occurrence of x for which to substitute.)

So we shall prove that $s \approx t \to p^{[x \to s]} \approx p^{[x \to t]}$ is logically valid, without any restriction on the term p. As usual, we'll do the case $x = x_3$, and leave the general case for you. Given any V and $\underline{v}$ from V, note that, by the first part of the proof of **6.9**,

$$(p^{[x_3 \to s]})^{\underline{v}} \ = \ p^{(v_1, v_2, s^{\underline{v}}, \cdots)} \quad \text{ and } \quad (p^{[x_3 \to t]})^{\underline{v}} \ = \ p^{(v_1, v_2, t^{\underline{v}}, \cdots)} \ .$$

Checking that $\underline{v} \in (s \approx t \to p^{[x_3 \to s]} \approx p^{[x_3 \to t]})^V$ amounts to checking that the left-hand sides in the display above agree whenever $s^{\underline{v}} = t^{\underline{v}}$. And that's immediate by looking at the right-hand sides.

Remark. In Exercise 7.11 at the end of the next chapter, you'll show that these logically valid formulae involving $\approx$ are actually quite a bit more than is needed as axioms in a proof system. We use them all as axioms at the beginning of that chapter in order to get on with more central issues.

6.10 Validity of arguments—the verbs $\models$ and $\models^*$.

Ultimately we are interested in sentences, and in studying the validity of arguments involving them.

Definition. Assume temporarily that F and all F_i are sentences in some $1^{\underline{\text{st}}}$order language. The statement

"$[F_1, F_2, \cdots, F_n$; therefore $F]$ *is a valid argument*"

is defined to mean that, for any interpretation V of the language, if all the F_i are true in V, then F is true in V .

However, to study this, it is very convenient to be able to work with non-sentence formulae as well. In particular just below we define two verbs which are analogous to " $\models$ " from propositional logic. They will be defined for data involving any formulae, not just sentences.

Definition—The two semantic conventions, namely $\models$ and $\models^*$.
Let F be a formula, and let Γ be a set of formulae, all from some $1^{\underline{\text{st}}}$order language.

Define

$$\Gamma \models F$$

to mean that, for any interpretation V of the language, if all the formulae in Γ are true in V, then F is true in V . In detail, this says that if $\underline{v} \in H^V$ for all $\underline{v}$ from V and for all $H \in \Gamma$, then $\underline{v} \in F^V$ for all $\underline{v}$ from V.

Define

$$\Gamma \models^* F$$

to mean that, for all V and all $\underline{v}$ from V, if all the formulae in Γ are true at $\underline{v}$ from V, then F is true at $\underline{v}$ from V . In detail, this says that if $\underline{v} \in H^V$ for some $\underline{v}$ from V for some interpretation V and for all $H \in \Gamma$, then $\underline{v} \in F^V$ for that particular $\underline{v}$.

As worded, the two verbs $\models$ and $\models^*$ appear to be different, and we'll see below by means of some examples that they are indeed different. It is very easy to see that $\Gamma \models^* F$ implies $\Gamma \models F$. The examples will show that the converse is false. However, if all the formulae in Γ are sentences and $\Gamma \models F$, then $\Gamma \models^* F$. So the two verbs coincide when applied only to sentences. In that case, they also coincide with the statement about validity of an argument from the previous definition, for $\Gamma = \{F_1, F_2, \cdots, F_n\}$.

The empty set consists entirely of sentences (and also entirely of purple unicorns), so the verbs coincide when Γ is empty. As in propositional logic, we abbreviate $\emptyset \models F$ to just $\models F$. Such F are exactly the logically valid formulae, as is immediately seen from the definitions.

Now let us look at the first of the examples promised above of a situation where $\Gamma \models G$, but $\Gamma \not\models^* G$. This is closely related to our example showing that $F \to \forall x F$ is not always logically valid. Despite that fact, I claim that

$$\{F\} \models \forall x F$$

holds for every F (and for every variable x, of course). For if F is true in V and we have any $\underline{v}$ from V, we need to show (when x is x_3) that $\forall x_3 F$ is true at $\underline{v}$ from V. This amounts to showing that F is true at $(v_1, v_2, u, v_4, \cdots)$ for every $u \in V$, which is certainly the case. Thus $\forall x_3 F$ is true in V, as required. The argument for other variables x is the 'same'.

On the other hand, there are examples of F where

$$\{F\} \not\models^* \ \forall x F \ .$$

Either of the earlier examples where $F \to \forall x_3 F$ is not logically valid, namely

$$F \ = \ \forall x_2 (x_3 < x_2 \ \vee \ x_3 \approx x_2) \quad \text{from the language of number theory ,}$$

or

$$F \ = \ \forall x_2 (x_3 r x_2 \vee x_3 \approx x_2) \ \text{ from the language of directed graph theory ,}$$

will do. The formula F itself, in the first case, is true at $(v_1, v_2, 0, v_4, \cdots)$ from $\mathbf{N}$ [or, in the second case, ... at any $(v_1, v_2, v_3, v_4, \cdots)$ from a graph in which there is a directed edge from v_3 to every other vertex.] But $\forall x_3 F$ is not true at those $\underline{v}$, because, if it were, F would have to be true however the 3rd slot was changed. But it isn't in the first case if the 3rd slot is chosen larger than v_2. And it isn't in the second case if the 3rd slot is chosen as a vertex with no edge directed towards the vertex v_2.

(Actually, these two examples are not different—the first is a special case of the second, where we choose our graph to be $\mathbf{N}$, with a directed edge from v to w whenever $v < w$. After all, "$<$" is simply a binary relation, so we can 'picture' it in terms of a directed graph.)

Before giving a few other examples to illustrate this point, we should write down the positive result involving " $\models$ " for future reference, and a corresponding positive result involving " $\models^*$ " .

Theorem 6.12 (i) *For all* $1^{\underline{\text{st}}}$*order languages, formulae* F *in that language, and variables* x*, we have*

$$\{F\} \models \forall x F \ , \quad \text{and so} \quad \Gamma \models F \implies \Gamma \models \forall x F \ .$$

(ii) *Furthermore, if we have a set of formulae* Γ *such that* x *has no free occurrence in any formula in* Γ*, then*

$$\Gamma \models^* F \implies \Gamma \models^* \forall x F \ .$$

(In particular, if F *contains no free occurrence of* x*, then* $\{F\} \models^* \forall x F$ *.)*

Proof. We have given the argument for (i) just above. As for (ii), take x to be x_3, for example, to be specific. Suppose that all H in Γ are true at $\underline{v}$ from V. By **6.1** 'for $i = 3$', they are in fact all true at $(v_1, v_2, u, v_4, \cdots)$ for any $u \in V$. But $\Gamma \models^* F$ then implies that F is true at $(v_1, v_2, u, v_4, \cdots)$ for any $u \in V$. Thus $\forall x_3 F$ is true at $\underline{v}$ from V, as required for the case when $x = x_3$. For every other variable, the argument is the 'same'.

Recall that the $1^{\underline{\text{st}}}$order language of ring theory is the same as that for number theory except that the relation symbol "$<$" is removed. And a ring is any interpretation which satisfies a certain set Γ of sentences asserting associativity, commutativity for addition, distributivity, behaviour of 0 and 1, and existence of negatives. There is a nice little result about rings which says that if every element r satisfies $r^2 = r$, then $r + r = 0$ for all r. The proof is quick:

$$1 = (r+1) - r = (r+1)^2 - r^2 = (r^2 + r + r + 1) - r^2 = r + r + 1 \ .$$

Now subtract 1 from both sides. This tells us that

$$\Gamma \cup \{F\} \ \models \ G \ ,$$

where F is $x \times x \approx x$, G is $x + x \approx 0$, and Γ is the set of (non-logical) axioms defining the word *ring* .

(Most mathematicians would prefer to write down sentences, and express the above as : $\Gamma \cup \{\forall x F\} \ \models \ \forall x G$, or, for that matter, as $\Gamma \cup \{\forall x F\} \ \models^* \ \forall x G$.)

But it is easy to find a ring and a particular element r with $r^2 = r$ but $r + r \neq 0$; just take $r = 1$ in $\mathbf{Z}$. This tells us that

$$\Gamma \cup \{F\} \quad \not\models^* \quad G \ .$$

This provides another example of the difference between $\models$ and $\models^*$.

Exercise 6.15 Show that the ring of all subsets of a given set satisfies $A^2 = A$ for all A, where multiplication is intersection, and addition is defined by

$$A + B \ := \ (A \cup B) \setminus (A \cap B) \ = \ (A \setminus B) \cup (B \setminus A) \ .$$

Deduce that you wouldn't get a ring, if addition was defined to be simply union of sets (except in one trivial case).

The texts which I have inspected all use one, but not both, of the verbs " $\models$ " and " $\models^*$ " . It seems to be a toss-up which is preferred. For example, **Barwise**, **Hodel**, **Manin** and **Mendelson** all use " $\models$ " . But **Bell& Machover**, **Burris**, **Ebinghaus-Flum-Thomas**, **Enderton**, **Kleene** and **Lyndon** all use " $\models^*$ " .

In the end, it is **sentences** which are important. So which verb is emphasized is not important. However, doing both cases serves to reinforce the various definitions, so we shall do that. In the next chapter are two proof systems, one giving the verb " $\vdash$ " and the other giving " $\vdash^*$ ". Then we'll have two completeness theorems, telling us that our Ch. 6 semantically defined verbs coincide with the Ch. 7 syntactically defined verbs. This is really just one theorem, the first of Gödel's numerous absolutely basic results.

The previous theorem, **6.12**, will be used to show that the one new rule of inference which we use in the proof systems of the next chapter makes sense, i.e. to help show that the systems are sound. We shall also use the old rules of inference from propositional logic, so the following facts will also be needed.

Theorem 6.13 *The following all hold in general in any* $1^{\underline{\text{st}}}$*order language. As we did in propositional logic,* $X \sqcap\!\sqcap Y$ *is always a formula, with* X *and* Y *as strings of symbols, and* $\sqcap\!\sqcap$ *as a subformula.*

$$\{F \ , \ F \to G\} \models G$$

$$\{X\neg\neg FY\} \models XFY$$

$$\{XFY\} \models X\neg\neg FY$$

$$\{X(F \wedge G)Y\} \models X(G \wedge F)Y$$

$$\{X((F \wedge G) \wedge H)Y\} \models X(F \wedge (G \wedge H))Y$$

$$\{X(F \wedge (G \wedge H))Y\} \models X((F \wedge G) \wedge H)Y$$

Furthermore, all the above hold with " $\models$ " changed to " $\models^*$ " .

These results can probably be proved a bit more quickly directly, but we shall use a portion of the following two general results, which motivate some parallel results in the next chapter. These could be called the **Semantic Deduction Theorem**.

Theorem 6.14 *If F is a sentence, then* $\Gamma \cup \{F\} \models G \Longleftrightarrow \Gamma \models F \to G$.

Proof. $\Longrightarrow$: Suppose that all H in Γ are true in V. For a contradiction, assume that $F \to G$ is not true in V. Choose some $\underline{v}$ from V such that $F \to G$ is not true at $\underline{v}$. Thus, at $\underline{v}$, F is true and G is false. But now, since it is a sentence, F is true in V. And so, all H in $\Gamma \cup \{F\}$ are true in V. Therefore, by the basic hypothesis of the theorem, G is true in V. In particular it is true at $\underline{v}$. This contradicts the statement above that it is false there, as required

$\Longleftarrow$: If all H in $\Gamma \cup \{F\}$ are true at $\underline{v}$ from V, but G isn't, then $F \to G$ isn't, but it must be, since all H in Γ are. This contradiction completes the proof.

Theorem 6.14* *For all formulae F, we have* $\Gamma \cup \{F\} \models^* G \iff \Gamma \models^* F \to G$.

Proof. $\Longrightarrow$: Suppose that all H in Γ are true at $\underline{v}$ from V. For a contradiction, assume that $F \to G$ is not true at $\underline{v}$ from V. Thus, at $\underline{v}$, F is true and G is false. And so, all H in $\Gamma \cup \{F\}$ are true at $\underline{v}$ from V. Therefore, by the basic hypothesis of the theorem, G is true at $\underline{v}$ from V. This contradicts the statement above that it is false there, and completes the proof.

The proof of the converse is identical to that in the previous theorem.

Notice how both the statement and proof are simpler for $\models^*$ than for $\models$. Some extra hypothesis certainly is needed for the latter, since we saw that $\{F\} \models \forall x F$ holds for all F, whereas $\models F \to \forall x F$ is sometimes false. So even with Γ empty, the $\Longrightarrow$ statement in **6.14** fails in general if F isn't a sentence.

Proof of 6.13 Since " $\models^*$ " is stronger than " $\models$ " , we need only prove the statements involving " $\models^*$ " .

For that, we need only show that each of the following is logically valid,

$$F \wedge (F \to G) \to G \quad \text{and} \quad X \sqcap \sqcap Y \leftrightarrow X \sqcup \sqcup Y$$

for each assertion $\{X \sqcap \sqcap Y\} \models X \sqcup \sqcup Y$ in the theorem statement. Then we apply the easy half of **6.14*** with Γ empty. That is, we're deducing

$$\{F_1, F_2, \cdots, F_n\} \models^* G$$

from knowing that

$$\models^* F_1 \wedge F_2 \wedge \cdots \wedge F_n \to G \ .$$

($n = 2$ for the topmost one, and $n = 1$ for all the others.)

As for that topmost one, $F \wedge (F \to G) \to G$, it's a $1^{\underline{\text{st}}}$order tautology (and so is logically valid). This is clear, since $P \wedge (P \to Q) \to Q$ is a propositional tautology—and that was the basis for the soundness of (MP) in propositional logic!

For the others, we can't use the same argument, since, for example,

$$\forall x(F \wedge G) \leftrightarrow \forall x(G \wedge F)$$

is *not* a $1^{\underline{\text{st}}}$order tautology.

So we must prove the following general result, which was already used once, in the proof of **6.11**.

Theorem 6.15 *If $F \leftrightarrow G$ is logically valid, then so is $XFY \leftrightarrow XGY$, for any choice of strings X and Y which makes XFY into a formula. (*Alternatively, *H and J are truth equivalent if one is obtained from the other by replacing a subformula by an equivalent subformula*—however, the formal definitions of truth equivalence and subformula have not been given yet in $1^{\underline{\text{st}}}$order, so we'll use the earlier notation).

Each of the formulae

$$F \leftrightarrow \neg\neg F \quad ; \quad F \wedge G \leftrightarrow G \wedge F \quad ; \quad F \wedge (G \wedge H) \leftrightarrow (F \wedge G) \wedge H$$

is a $1^{\underline{\text{st}}}$order tautology, and therefore logically valid. Thus **6.15** will complete the proof of **6.13**.

Proof of 6.15. We proceed by induction on the total length of the strings X and Y of symbols. When zero, there is nothing to prove.

For the inductive step, we show that $XFY \leftrightarrow XGY$ is logically valid, where X and Y taken together is a non-empty string. The formula XFY

is certainly not atomic, and so has one of the forms $\neg H$ or $H \wedge J$ or $\forall x H$. But then H (or perhaps J in the middle case) must have the form $X_1 F Y_1$, where the total length of X_1 and Y_1 is less than that of X and Y. (If you are skeptical about a subformula occurrence having to happen this way, see the third and fourth sections of Appendix B to Chapter 8 for proofs of some of the more boring facts about subformulae and subterms.) Now applying the inductive hypothesis, all we need prove is that if $H_1 \leftrightarrow H_2$ is logically valid, then so are all of

$$\neg H_2 \leftrightarrow \neg H_1 \;\; ; \;\; H_1 \wedge J \leftrightarrow H_2 \wedge J \;\; ; \;\; J \wedge H_1 \leftrightarrow J \wedge H_2 \;\; ; \;\; \forall x H_1 \leftrightarrow \forall x H_2 \; .$$

These are straightforward exercises.

Exercises.

6.16 Do the exercises referred to just above.

6.17 (i) Show that $\{F\} \models^* G \;\; \Rightarrow \;\; \{\neg G\} \models^* \neg F$.

(ii) Show that, in general, $\{F\} \models G \;\; \not\Rightarrow \;\; \{\neg G\} \models \neg F$, although, when F is a sentence, we have $\{F\} \models G \;\; \Rightarrow \;\; \{\neg G\} \models \neg F$.

(Many might prefer $\models^*$ to $\models$ because of this 'failure of the contrapositive'.)

6.18 Prove the following:

(i) In general $\{\exists x F\} \not\models F$.

(ii) $\{F\} \models \exists x F$.

(iii) $\{F\} \models^* \exists x F$.

(iv) In general, $\{\exists x F\} \not\models^* F$.

The usual symmetry between $\forall$ and $\exists$ is broken here, since there are counterexamples in only one of the four cases if $\exists$ is replaced by $\forall$. But, using only $\models^*$, the symmetry is restored.

6.19 This exercise takes care of several of the points needed in the Boolos' proof of Gödel's incompleteness theorem, given at the end of Chapter 5. Suppose given $R \subset \mathbf{N} \times \mathbf{N}$, and a formula C in the language of $1^{\underline{\text{st}}}$order number theory. Recall that for each natural number $k > 1$, a short form term is defined by

$$\text{“}k\text{”} \;\; := \;\; (1 + (1 + (1 + \cdots\cdots (1 + 1) \cdots))) \; ,$$

where exactly k copies of 1 are used. Assume that $C^{[x\mapsto \text{“}a\text{”},z\mapsto \text{“}c\text{”}]}$ is true in $\mathbf{N}$ if and only if $(a,c)\in R$. Let x,y,z,w be x_1,x_2,x_3,x_4 respectively.

(i) Define $B := \exists z(z<y\wedge C)$. Show that $B^{[x\mapsto \text{“}a\text{”},y\mapsto \text{“}b\text{”}]}$ is true in $\mathbf{N}$ if and only if $(a,c)\in R$ for some $c<b$.

(ii) Next define $A := \neg B\wedge\forall w(w<x\to B^{[x\mapsto w]})$. Show that $A^{[x\mapsto \text{“}a\text{”},y\mapsto \text{“}b\text{”}]}$ is true in $\mathbf{N}$ if and only if $(a,c)\notin R$ for any $c<b$, but, for every $d<a$, there is a $c<b$ with $(d,c)\in R$.

(iii) Finally define $F := \exists y(y\approx \text{“}e\text{”}\wedge A)$. Show that $F^{[x\mapsto \text{“}a\text{”}]}$ is true in $\mathbf{N}$ if and only if $(a,c)\notin R$ for any $c<e$, but, for every $d<a$, there is a $c<e$ with $(d,c)\in R$.

Now notice that the “b”, as b ranges over $\mathbf{N}$ is an example of the data in the final part of this question.

(iv) Assume that F is a formula in some $1^{\underline{\text{st}}}$order language. Assume also that we have an interpretation V of the language and a collection of constant terms (i.e. not involving variables) $\{t_v : v\in V\}$ such that $t_{\underline{v}}^{\underline{v}}$ (which is independent of $\underline{v}$ because of not involving variables) is always equal to v. Show that $\forall x(F\leftrightarrow x\approx t_{v_0})$ is true in V if and only if $F^{[x\mapsto t_v]}$ is never true in V except for when $v=v_0$.

7. FIRST ORDER PROOF SYSTEMS

This chapter provides the analogue for 1st order logic of the Chapter 3 (propositional logic) material on derivations. A good start would be to read again the last five or six paragraphs of that chapter.

The first two sections below give the two proof systems. These are in general just like the proof system for propositional logic in Chapter 3. They consist of some axioms and some rules of inference. The definition of a derivation is exactly as before. Examples of derivations can be found in Sections 7.7 (particularly Example 4) and 7.8. The first system defines the verb denoted " $\vdash$ ", and the second one the verb " $\vdash^*$ ". As with propositional logic, these verbs are defined to mean that *a derivation exists*. Then most of the remainder of the chapter is mainly devoted to proving the completeness theorems, saying that $\vdash$ coincides with $\models$, and that $\vdash^*$ coincides with $\models^*$. More casually, these say *derivability coincides with being true*. The definitions of $\models$ and $\models^*$ are in the last section of the previous chapter.

On first reading, most readers will presumably want to concentrate on just one of the two systems. Either one is fine for doing 1st order logic. Our labelling makes this relatively easy to do. Doing this will shorten the chapter by about 40%. We have used a smaller print for all material only involving the system*, which defines $\vdash^*$. This should make it easier reading, if you wish to study only the other system. But it was really a coin-flip to decide which system to treat as a second-class citizen.

For this chapter, assume that a particular 1st order language has been fixed.

7.1 The System

The **axioms** are listed below.

(i) For all 1st order formulae F, G and H, the three following formulae are axioms, the *tautology axioms*:

$$F \to F \wedge F \quad ; \quad F \wedge G \to F \quad ; \quad (F \to G) \to (F \wedge H \to G \wedge H) \ .$$

They are certainly 1st order tautologies, and so are logically valid, by **6.8**. They look identical to the axioms in Chapter 3 for propositional logic, and

are different only in that the 'ingredients', F, G and H, are $1^{\underline{\text{st}}}$order formulae, rather than propositional formulae.

(ii) For any formula F and any variable x, the formulae

$$\forall x F \to F^{[x \to t]} \quad \text{(for any term } t \text{ which is substitutable for } x \text{ in } F)$$

are axioms (called *substitution axioms*). These are all logically valid by Theorem **6.10**.

(iii) For all variables x, all formulae F which are FOFOO x (i.e. free of free occurrences of x), and all formulae G, the formulae

$$\forall x(F \to G) \ \to \ (F \to \forall x G)$$

are axioms (called *quantifier axioms*). These are all logically valid, proof of which was the subject of Section 6.8.

(iv) The following are axioms (called the *equality axioms*) for all terms t, s, and q, all formulae F in which s and t are substitutable for x , and all strings A and B of variables, constant symbols and function symbols for which AsB is a term :

$$s \approx s \quad ; \quad s \approx t \ \to t \approx s \quad ; \quad (s \approx t \ \wedge \ t \approx q) \to s \approx q \ ;$$

$$s \approx t \to (F^{[x \to s]} \to F^{[x \to t]}) \quad ; \quad s \approx t \to AsB \approx AtB \ .$$

These equality axioms are logically valid by the results in Section **6.9**. Actually, we could generalize the last one slightly, and write it in the following form, where s, t, p are any terms and x is any variable:

$$s \approx t \ \to \ p^{[x \to s]} \approx p^{[x \to t]} \ .$$

As indicated in Exercise 7.11 at the end of the chapter, most of these equality axioms are not really necessary; they can be derived from a more economical set. But fooling around showing this would divert us away from the main points (and also it makes a nice exercise for you).

The **rules of inference** have one addition to the analogues of those for propositional logic (see the early pages of Chapter 3). To be explicit, we use our familiar modus ponens and three (or perhaps five) replacement rules (for

double negation, and commutativity and associativity of conjunction), plus one extra rule, (GEN), below. More specifically:

(i) the formulae F, G in (MP), and $X \sqcap \sqcap \sqcap Y$ in the replacement rules of inference, must now of course be formulae in the given $1^{\underline{\text{st}}}$order language (not propositional); and

(ii) the new of rule of inference, called (GEN), says that, for any formula F and any variable x :

from F, one may immediately infer $\forall x F$.

By **6.12(i)** and **6.13**, these rules *preserve semantic entailment* as defined by the verb " $\models$ " . To be explicit, for any immediate inference from a rule, we have that,

if $\Gamma \models F$ *for (the one or two)* F*, then* $\Gamma \models G$*,*

where G *is the new line inferred, and the* F *involved are the line—or two lines, when the rule is* (MP)*—referred to in the direct inference from the rule.*

Thus, with $\Gamma \vdash F$ defined to mean that there is a derivation, within this system, of F from premiss set Γ, the logical validity of the axioms and the truth preservation of the rules immediately give the

Soundness Theorem : $If \quad \Gamma \vdash F \ , \quad then \ \ \Gamma \models F \ .$

To see this, write down a derivation of F from Γ. Proceeding from the top line to the bottom line, the facts quoted above, when compared to the justifications for these lines, show that each line is true in any model for Γ . In particular, this holds for the bottom line, as required.

Soundness just says that you can't derive something which isn't true.

7.2 The System*

Except for one change to one rule of inference, this is the same as the system above. The exception is that (GEN) is replaced by (GEN*), which says that

from F, one may infer $\forall x F$ **only at steps in a derivation before**

any steps in which a premiss, having a free occurrence of x , is used.

In other words, one must be able to fill in the justifications for the steps in a derivation so that all justifications (GEN*) which involve the variable x occur before any appeal to a premiss which has a free occurrence of x . There is something slightly ugly about this. But the deduction theorem (Section 7.5) for $\vdash^*$ is nice, whereas that theorem for $\vdash$ exhibits this slight ugliness—perhaps there is a law of conservation of ugliness.

By **6.12(ii)** and **6.13**, these rules *preserve semantic entailment* as defined by the verb " $\models^*$ " . To be explicit, for any immediate inference from a rule, we have that, *if* $\Gamma \models^* F$ *for the one* F *(or both* F*), then* $\Gamma \models^* G$*, where* G *is the new line produced, and the* F *involved are the line—or two lines, for the rule* (MP)*—referred to in the direct inference from the rule.*

Thus, with $\Gamma \vdash^* F$ defined to mean that there is a derivation in system* of F from premiss set Γ, the logical validity of the axioms and the truth preservation of the rules immediately give the

Soundness* Theorem : If $\Gamma \vdash^* F$, then $\Gamma \models^* F$.

The proof is the same as that for the previous soundness theorem, except that it refers to formulae being true "at $\underline{v}$ from V", rather than "in V".

Remarks. Use of the restricted generalization rule can be a bit tricky. **Lyndon** uses the semantic convention $\models^*$, and is the only reference I know which uses a rule closely similar to (GEN*). He formulates the restricted generalization rule as follows:

from F, one may infer $\forall x F$

only in derivations from a set of premisses containing no free x .

Then $\Gamma \vdash_L F$ is redefined to mean that there is a derivation of F from some **subset** of Γ, not necessarily from Γ itself. That sounds as though it might be the same as (GEN*). But suppose we take $\Gamma = \{ F , \forall x F \to G \}$, where, oddly enough, F is FOFOO x, but G isn't. Then certainly $\Gamma \vdash^* G$. A derivation is the sequence F , $\forall x F$, $\forall x F \to G$, G where justifications may be given as (premiss), (GEN*), (premiss), (MP). Note that the premiss which comes after (GEN*) has free occurrences of x, which is okay for $\vdash^*$ but not for $\vdash_L$. So this derivation is inadmissable in the formulation just above. In that formulation, one has $\Gamma \vdash_L \forall x F$ and $\{ \forall x F , \forall x F \to G \} \vdash_L G$, but the argument for 'transitivity' of $\vdash_L$ fails on this example. It seems doubtful that $\Gamma \vdash_L G$ in general.

There are a number of 'awkwardnesses' in system* related to the fact that one cannot just insert derivations into other ones, in the style of "shortcut derivations" as in Chapter 3, without being careful about the restriction on the use of (GEN*). We'll deal with these after proving the Deduction Lemma in Section 7.5. This is tied to the 'transitivity' referred to above. It holds, but needs to be proved.

It is arguable that a rule of inference ought not to have any dependency on the set of premisses being used. The system in **Enderton** (used also by **Bell&Machover**, having originated with **Rosser**, I believe) which is for our $\vdash^*$, has this good property. On the

other hand, for comparing system to system*, it is convenient to have them almost the same, with that one restriction on one rule as the only difference.

7.3 Comments on the two systems.

Notice that, when the set of premisses is empty, and we drop the $\emptyset$ as usual, we have $\vdash F \iff \vdash^* F$, since (GEN) and (GEN*) coincide when there are no premisses. (They coincide more generally when all the premisses are sentences.) This is as expected if both completeness theorems are to hold. By soundness, such F are necessarily logically valid. The special case of completeness in which the set of premisses is empty will give the converse—that every logically valid formula can be derived from the logical axioms alone.

Since the main interest is almost entirely in validity of arguments where the conclusion and all the premisses are **sentences**, there is very little ultimate need, from that point of view, for going through the details for *both* of our systems. However, it does seem worthwhile from a pedagogical point of view. Students who move on to the next level in studying logic should be aware of the differences between these two approaches. Among the best texts at the next level are **Mendelson** (which has a system dealing with what we denote by $\vdash$ and $\models$), and **Enderton** (whose system deals with what we denote by $\vdash^*$ and $\models^*$, though of course he doesn't use the $*$'s in his notation). For one more level beyond that, **Manin** is among the best.

A second pedagogical reason to see all the details of both approaches is to provide some reinforcement. There are close parallels between the two, but interesting differences as well.

Just as with systems for propositional logic, seldom do two authors have exactly the same system, and probably for the same reason as mentioned in Chapter 3 on the much simpler situation of propositional logic. But these systems are always chosen to be sound and complete. No authors disagree on the essentials in the treatment of semantics (truth, validity, interpretations, etc. ...), except for the distinction between $\models$ and $\models^*$. Therefore, all these different systems 'do the same thing' in the end, again except for the distinction between $\vdash$ and $\vdash^*$.

The system which we chose for propositional logic was (in *my* opinion, of course) the best compromise for beginning students to have easily remembered, 'natural' and few enough axioms and rules, but sufficient to make the path to proving completeness not too long and tortuous. Certainly a few of the axioms and rules can be dispensed with, both there in Chapter 3 and here in this chapter. For example, see Exercise 7.11, which shows how to dispense with most of the equality axioms.

Any proof of completeness for 1$^{\underline{\text{st}}}$order logic is going to be among the more difficult proofs taught to undergraduates. So you may find the remainder of this chapter to be a fairly exhilarating challenge. Again, the systems have been chosen to give a good compromise between economy of the system itself, and simplicity of the proofs for the important results. The axioms and rules are as natural as I could make them, though the naturality of our axiom scheme (iii),

$$\forall x(F \to G) \ \to \ (F \to \forall x G) \qquad \text{for } F \text{ FOFOO } x \quad ,$$

is not so clear. This axiom, or something very close, is used in all other systems of which

I'm aware. The following appears to me to be slightly more natural, since it bears some relation to (MP) (and needs fewer brackets!):

$$F \wedge \forall x(F \to G) \ \to \ \forall x G \qquad \text{for } F \text{ FOFOO } x \quad .$$

We could use it instead, since the two are derivable from each other 'purely propositionally'—see **7.6** in the next section. See also the remarks one paragraph below.

A final point for the more experienced reader is to notice that a sound proposed system with only (MP) as the single rule of inference (together with any set of logically valid formulae as axioms, decidable or not) must necessarily produce a derivation of F from Γ only if $\Gamma \models^* F$. This follows immediately once we know that our system* is complete, but is easy to see anyway. Thus no such proof system can exist for the semantic convention using $\models$, since examples exist where $\Gamma \models F$ but $\Gamma \not\models^* F$. In some sense, the rule (GEN) is essential.

On the other hand, the system in **Enderton** is precisely such a "proof system à la Hilbert" for the semantic convention using $\models^*$. Getting rid of our replacement rules of inference (by adding some tautologies to the axioms) is easy. To get rid of (GEN*), Enderton replaces our axiom (iii) by the following:

$$F \to \forall x F \quad \text{for } F \text{ which are FOFOO } x \ , \ \text{and}$$

$$\forall x(F \to G) \ \to \ (\forall x F \to \forall x G) \qquad \text{with no restriction on } F \text{ and } G \ ,$$

plus 'partial closures', i.e. any universal quantification, of all the axioms as well. (Not relevant to the point here, he is less parsimonious with tautologies, but much more so with equality axioms.) Besides the criteria mentioned earlier, our choice of systems is partly dictated by having the two as parallel as possible.

7.4 Propositions 7.1 to 7.8, 7.10 and 7.11 (and 7.1* to 7.8*, 7.10*, 7.11*).

The results in Chapter 3 with the corresponding numbers (that is, 3.1 to 3.8, 3.10 and 3.11) are proved with the axioms and rules of inference from propositional logic. It is quite clear that the 'same' axioms and rules will prove the 'same' results, but this time with all the formulae involved being 1st order formulae in our fixed language. Since our system does include all these axioms and rules from propositional logic, we can now assume that those results are established.

The ones labelled with a "$*$" need to have each $\vdash$ changed to a $\vdash^*$, of course.

For convenience, we list below the non-"$*$" versions.

Prop 7.1 $\{F \to G\} \vdash \neg G \to \neg F$.

Prop 7.2 $\{F \to G \ , \ \neg G\} \vdash \neg F$.

Prop 7.3 $\{F \to G\ ,\ G \to H\} \vdash F \to H$.

Prop 7.4 $\vdash F \to F$.

Prop 7.5 $\{F \to G\} \vdash F \to F \wedge G$.

Prop 7.6 (i) $\{F \to (G \to H)\} \vdash F \wedge G \to H$.
(ii) $\{F \wedge G \to H\} \vdash F \to (G \to H)$.

Prop 7.7 $\{F\ ,\ G\} \vdash F \wedge G$ and *'conversely'*.

Prop 7.8 (i) $\{G\} \vdash F \to G$.
(ii) $\{F \to G\ ,\ F \to (G \to H)\} \vdash F \to H$.

Prop 7.10 (i) $\vdash F \wedge \neg F \to G$.
(ii) $\{F\ ,\ \neg F\}\ \vdash\ G$.

Theorem 7.11 *For a set of formulae* Γ*, we have* $\Gamma \vdash G$ *for all* G *if and only if, for some formula* F*, both* F *and* $\neg F$ *can be inferred from* Γ .

7.5 Herbrand's Deduction Lemmas and Lemma 7.0* .

Here we deal with the appropriate analogues of **3.9**.

Theorem 7.9 (Herbrand Deduction Lemma for the system.) *If* F *is a sentence, and* $\Gamma \cup \{F\} \vdash G$*, then* $\Gamma \vdash F \to G$.

Theorem 7.9* (Herbrand Deduction Lemma for the system*.) *For any formula* F*, if* $\Gamma \cup \{F\} \vdash^* G$*, then* $\Gamma \vdash^* F \to G$.

The assumption in **7.9** that F is a sentence can be weakened, but not dispensed with altogether. To see an example of this, go back to Proposition **6.6**. Take Γ to be empty, and G to be $\forall x F$. Because of our new rule of inference, we certainly have $\{F\} \vdash \forall x F$ for any formula F. But by the soundness theorem, there can be no derivation of $F \to \forall x F$ from the empty set of premisses, for any choice of F for which the former formula is not logically valid.

There is a similar extra condition in **7.12** below. See also the display immediately before Exercise 7.0* . These are aesthetic blemishes on the

system, as compared to the system*, and seem to even things out, since the system* is the one with the uglier generalization rule of inference.

Proof of 7.9 This proceeds exactly as with that of **3.9**, with one extra case at the end (because we have one extra rule of inference now). Thus we assume failure of the result for a given Γ and F. We pick a G and a derivation of G from $\Gamma \cup \{F\}$ which is shortest possible among all such derivations for all G for which $F \to G$ cannot be derived from Γ. If the derivation is $H_1, \cdots, H_n = G$, we have that $F \to H_i$ *can* be derived from Γ for all $i < n$, by the minimum length choice above. Next we consider all the possibilities for justifying the last line, and, in each case, show that, in fact $F \to G$ *can* be derived from Γ.

All of these, except when the last line $H_n = G$ comes from our new rule of inference, are done exactly as in **3.9**. But I'll copy those five paragraphs here for your convenience.

We have $G = F$; or G is an axiom or is in Γ; or G is justified by a replacement rule of inference from some earlier H_i ; or by (MP) using a pair of earlier lines.

In the first case, $F \to G$ is just $F \to F$, so by **7.4**, we'd certainly have $\Gamma \vdash F \to G$, as required.

In the second case, we also get $\Gamma \vdash F \to G$, since $\{G\} \vdash F \to G$ by the first part of **7.8**, and since $G \in \Gamma$ or is an axiom.

In the third case, G is obtainable from an H_i for $i < n$ by one of the replacements in those five rules of inference. But then $F \to G$ is obtainable from $F \to H_i$ by the 'same' replacement, so again we get $\Gamma \vdash F \to G$, by adding just one line to a derivation of $F \to H_i$ from Γ.

In the last case, we have that G arises by using (MP). Thus there are two lines: H_i say, and H_j which is $H_i \to G$, where both i and j are less than n. Now both $F \to H_i$ and $F \to (H_i \to G)$ can be derived from Γ. (This is just from the business of "shortest", as noted above.) But, by **7.8**(ii), $F \to G$ can be derived from the two formulae in the previous sentence. (Just change G in **7.8**(ii) to H_i, and change H to G.) Thus $F \to G$ *can* be derived from Γ, completing the 'old part' of the proof.

The new case for **7.9** is when G is $\forall x H_i$ for some $i < n$. Here we can just append to a derivation showing $\Gamma \vdash F \to H_i$ these three lines at the bottom: firstly, the formula $\forall x(F \to H_i)$ using the new rule; then

$$\forall x(F \to H_i) \ \to \ (F \to \forall x H_i) \ ,$$

which is an axiom, since F was assumed to be a sentence, and, in particular, it is FOFOO x; finally, write down $F \to \forall x H_i$ (which is $F \to G$, as required) applying (MP) to the previous two lines. This shows that $\Gamma \vdash F \to G$, completing this new case, and the proof.

Proof of 7.9* Here the method of proof from **3.9** is sandwiched between extra discussion at both ends.

Assume that **7.9*** is false, so that, for some Γ, F, G, we have $\Gamma \cup \{F\} \vdash^* G$, but $\Gamma \nvdash^* F \to G$. Then there is a *finite* such Γ, for the following reason. Write down a derivation showing $\Gamma \cup \{F\} \vdash^* G$. Let Λ be the (finite!) subset of those premisses from Γ which actually appear as lines in this derivation. Then $\Lambda \cup \{F\} \vdash^* G$. Also $\Lambda \nvdash^* F \to G$ since $\Gamma \nvdash^* F \to G$.

So we may assume that Γ is finite, and thus pick a smallest possible Γ for which there is an F and G such that $\Gamma \cup \{F\} \vdash^* G$, but $\Gamma \nvdash^* F \to G$. (The argument above for finiteness was needed simply to be able to do this—it's not always so obvious or even true that we can get a smallest possible from a collection of infinite sets. By *smallest possible* Γ we mean that no proper subset of Γ has the property.)

Now for this smallest Γ and F, G as above, I claim that the following holds for every derivation showing $\Gamma \cup \{F\} \vdash^* G$:

(aha!) F and every $G \in \Gamma$ must appear as lines in such a derivation,

and none of these lines can be justified except as a premiss .

For, if F didn't appear in this way , we'd have $\Gamma \vdash^* G$. But now observe that, by the first part of **7.8***, we have $\{G\} \vdash^* F \to G$. And so, putting the two derivations next to each other (which is okay because the latter never uses (GEN*)), we get $\Gamma \vdash^* F \to G$, which was assumed false to begin with.
And if some G didn't appear, we could remove it from Γ and contradict the choice of Γ as smallest possible.

Now proceed with the proof just as in **3.9**. That is, for the minimal finite Γ above and a given F, choose G such that $\Gamma \nvdash^* F \to G$, but $\Gamma \cup \{F\} \vdash^* G$, and a derivation of shortest length showing the latter exists for that particular G. Let $H_1, \cdots, H_n = G$ be such a shortest derivation, and argue as in **3.9** that, in fact, $\Gamma \vdash^* F \to G$, if G, the last line, can be justified by an axiom, a premiss, (MP), or by a replacement rule of inference. This time I won't copy out those five paragraphs again. These arguments use the fact that $F \to H_i$ *can* be derived from Γ for all $i < n$, by the minimum length choice of the derivation, but do not use (aha!).

So we are left with the case where the last line, G, can be justified only because it is $\forall x H_i$ for some $i < n$ and some variable x. But because of the extra conditions on inferring $\forall x H$ from H in system*, no earlier line can have as a justification a premiss which contains a free occurrence of this x. But, by (aha!), every premiss in Γ and F itself all do occur as earlier lines not justifiable otherwise than as premisses. Therefore, neither F nor any G in Γ contains x freely. So we can finish the proof just like we did for **7.9** by appending at the bottom to a derivation showing $\Gamma \vdash F \to H_i$ these three lines :

$\forall x(F \to H_i)$
$\forall x(F \to H_i) \ \to \ (F \to \forall x H_i)$
$F \to \forall x H_i \ = \ F \to G$

Since x doesn't occur freely in any premiss $G \in \Gamma$, using the new rule of inference to justify the first of these three is okay. Since x doesn't occur freely in F, the second line *is* an axiom. And the last line is justified by applying (MP) to the previous two.

This completes the proof.

One must concede that systems á la Hilbert for the semantic convention $\models^*$ do have an easier time proving the deduction lemma. Those references which deal with the $\models^*$ convention, and have a quantifier rule with restriction such as we have, do give shorter proofs than above, but some of the proofs as stated seem hard to follow.

A fairly extreme example of the last part of the proof just above comes from converting the obvious two line derivation, using (GEN*), showing $\{F\} \vdash^* \forall x F$ for F FOFOO x, into a premiss-free derivation showing $\vdash^* F \to \forall x F$. This would start with a 'propositional' derivation of $F \to F$ (as in **7.4***), followed by the line $\forall x(F \to F)$ justified by (GEN*), then the axiom $\forall x(F \to F) \ \to \ (F \to \forall x F)$ (by FOFOOness of F), and finally the line $F \to \forall x F$ justified by (MP).

This brings us to the first of a number of 'awkwardnesses' in system* related to the fact that one cannot just insert derivations into other ones, in the style of "shortcut derivations" as in Chapter 3, without being careful about the restriction on the use of (GEN*).

For example, for F FOFOO x,

$$\Gamma \vdash^* F \quad \Rightarrow \quad \Gamma \vdash^* \forall x F \ ,$$

but appending the single line $\forall x F$ justified by (GEN*) might not do it, because the derivation of F from Γ to which this is appended might have a premiss with a free occurrence of x.

Let H be the conjunction of all the premisses used in some derivation of F. To get one of $\forall x F$, first write down the derivation from no premisses of $F \to \forall x F$ from a couple of paragraphs above, then again one for $H \to F$ by iterating the deduction lemma. Now a purely propositional derivation of H from Γ is written down, basically iterating **7.7***. This has no appeals to (GEN*). Now finish with two lines, F then $\forall x F$, both times appealing to (MP).

The iteration of the deduction lemma referred to above goes as follows. Let H be $H_1 \wedge \cdots \wedge H_k$. Strictly speaking, iteration of the deduction lemma gives

$$\vdash^* H_1 \to (H_2 \to (\cdots \to (H_k \to F) \cdots)) \ ,$$

but we can get from that to $H_1 \wedge \cdots \wedge H_k \to F$ 'purely propositionally', via **7.6***.

The fact two displays above is a special case of the following 'transitivity' property (which is obvious for $\vdash$).

$$\textit{If } \Gamma \vdash^* G \textit{ for all } G \in \Lambda \textit{ and } \Lambda \vdash^* F, \ \textit{ then } \ \Gamma \vdash^* F \ .$$

Again simply inserting derivations from Γ of the premisses used in a derivation of F from Λ would possibly violate the restriction on (GEN*). Instead, let $G_1, \cdots, G_\ell$ be those premisses, and define G' to be their conjunction. Let H be the conjunction of all the premisses from Γ used in "ℓ" chosen derivations of the G_j. To get a derivation of F from Γ, first use the Deduction Lemma "ℓ" times, and write down derivations with no premisses of each $H \to G_j$. Follow that by a purely propositional derivation of $H \to G'$. Use the Deduction Lemma again and append a derivation of $G' \to F$. Now append a derivation of H from Γ, where of course those premisses occur, but purely propositionally, so no appeals to (GEN*) occur. Now finish with two lines, G' then F, both times appealing to (MP).

Yet another example is the following.

$$If\ \Gamma \vdash^* F\ and\ \Gamma \vdash^* F',\ \ then\ \ \Gamma \vdash^* F \wedge F'\ \ .$$

Again if H is the conjunction of all the premisses appealed to in two derivations (of F and F'), produce a new derivation as follows. First a derivation of $H \to F$, then one of $H \to F'$, both with no premisses by the Deduction Lemma, then a derivation of H from Γ, then two lines, F and F' both using (MP), and finally a purely propositional derivation of $F \wedge F'$, namely that in **7.7***. However this can also be done using a simpler idea. Take the two derivations (of F and F'), and 'slot' them together, making sure that the restriction on (GEN*) isn't violated. That is, don't necessarily just write them down one after the other. Then follow that with the purely propositional derivation of $F \wedge F'$. See also Exercise 7.16

Generally speaking, one can more-or-less bypass the restriction on (GEN*) by noting that, if $\Gamma \vdash^* F$, then there exists a derivation showing this which has the form $\mathcal{G}$ followed by $\mathcal{P}$, where $\mathcal{G}$ is a bunch of lines with no appeals to premisses (but might have to (GEN*)), whereas $\mathcal{P}$ has no appeals to (GEN*) (but might have to premisses). To see this, take a given derivation, and let H be the conjunction of all premisses used. Let $\mathcal{G}$ be a derivation of $H \to F$ using the Deduction Lemma. Now let $\mathcal{P}$ be a purely propositional derivation of H from Γ, followed by the line F, using (MP).

We could have used this last paragraph to derive the second last one. Using, with obvious notation, derivations $[\mathcal{G}, \mathcal{P}]$ and $[\mathcal{G}', \mathcal{P}']$, the derivation $[\mathcal{G}, \mathcal{G}', \mathcal{P}, \mathcal{P}', \cdots F \wedge F']$ does the trick.

Recall Ex. 6.17 , which, if completeness is to hold implies that

$$\{F\} \vdash^* G \quad \Longrightarrow \quad \{\neg G\} \vdash^* \neg F\ ,$$

but, in general,

$$\{F\} \vdash G \quad \not\Longrightarrow \quad \{\neg G\} \vdash \neg F\ .$$

Exercise 7.0* Prove the first inference just above, perhaps using the Deduction Lemma and then 'purely propositional' considerations.

A technical result which we'll need later is the following. We need it in the $*$-case only. This seems fortunate, in view of the comments immediately above, and the exercise, which is used in the proof. (However, the result does hold in the other case as well, of course, as would follow from completeness.) See the discussion concerning renaming bound variables just before **6.13**.

Lemma 7.0* *If* $\underline{\underline{F}}$ *is obtained from the formula* F *by renaming some or all of the bound variables, then* $\{F\} \vdash^* \underline{\underline{F}}$.

Proof. By the 'transitivity' remarks above, it suffices to prove the case of renaming the variable in the scope of a single quantifier occurrence. Proceeding by contradiction, we assume that F is a formula of shortest length where $\{F\} \nvdash^* \underline{\underline{F}}$. Certainly F is not atomic, since it has a quantifier in it.

Thus F has the form $G \wedge H$ or $\neg G$ or $\forall x G$, where the result holds for G and H.

In the first case, $\underline{\underline{F}}$ is $\underline{\underline{G}} \wedge H$ or $G \wedge \underline{\underline{H}}$, with obvious notation. By **7.7***, we infer $\underline{\underline{F}}$ from its two components above, which are themselves inferrable from $\{G, H\}$ by induction, and hence from F by the converse in **7.7***. (So we have used transitivity a few times!)

In the second case, $\underline{\underline{F}}$ is just $\neg\underline{\underline{G}}$, and, G being shorter than F, it may be inferred from $\{\underline{\underline{G}}\}$. So by the Ex. 7.0*, $\neg\underline{\underline{G}}$ may be derived from $\{\neg G\}$, as required.

In the third case, if the leftmost symbol is not the quantifier whose bound variable is being renamed, we get $\underline{\underline{F}} = \forall x \underline{\underline{G}}$. So it comes down to the fact that $\{G\} \vdash^* H$ implies $\{\forall x G\} \vdash^* \forall x H$, which is easy to see: just sandwich a derivation showing $\{G\} \vdash^* H$ between $\forall x G$ and $\forall x H$. The premiss has no free x, so the last step is no problem. (The middle part is no problem because any free variables in $\forall x F$ are already free in F itself.)

Finally in the third case where the 'outside' quantifier *is* the one having its variable renamed, it amounts to showing

$$(\%) \qquad \{\forall x G\} \vdash^* \forall y(G^{[x \to y]}) \,,$$

where G is any formula in which the variable y has no free occurrences. A derivation showing this has four lines : first the premiss $\forall x G$; then the axiom $\forall x G \to G^{[x \to y]}$; then $G^{[x \to y]}$, by (MP); and finally $\forall y(G^{[x \to y]})$ by our new rule of inference. Again, note that our premiss has no free occurrences of y.

7.6 The overall structure of the proof of completeness.

Our aim is to prove:

Theorem 7.16 (Adequacy of the system.) *If* $\Gamma \models F$, *then* $\Gamma \vdash F$.

Theorem 7.16* (Adequacy of the system*.) *If* $\Gamma \models^* F$, *then* $\Gamma \vdash^* F$.

We are continuing to keep the numbering of results correlated to the corresponding result in propositional logic. Indeed, **7.16** as stated appears identical to **3.16**, but of course refers to 1$^{\underline{\text{st}}}$order , not propositional, formulae, and to the system of this chapter. Combining the adequacy theorem with the soundness theorem then gives the completeness theorem, telling us that the semantically defined verbs of Chapter 6 coincide with the syntactically defined verbs of this chapter :

Theorem. (Completeness of the system.)

$$\Gamma \models F \quad \Longleftrightarrow \quad \Gamma \vdash F \ .$$

Theorem***.** (Completeness of the system*).

$$\Gamma \models^* F \quad \Longleftrightarrow \quad \Gamma \vdash^* F \ .$$

These fundamental results were originally due to Gödel in his thesis, around 1930.

The overall structure of the proofs of these theorems is also essentially the same as that of **3.16**. But some of the details are much more intricate. This is mainly because truth assignments must be replaced by interpretations.

Definition. A set of 1$^{\underline{\text{st}}}$order formulae from our fixed language is said to be $\vdash$-*consistent* if and only if it does **not** satisfy one (and therefore either) of the conditions in **7.11**. Thus Γ is $\vdash$-*inconsistent* if and only if, for some F, we have $\Gamma \vdash F$ and $\Gamma \vdash \neg F$, or equivalently, ... if and only if we have $\Gamma \vdash G$ for every formula G .

Definition. A set of 1$^{\underline{\text{st}}}$order formulae is said to be $\vdash^*$-*consistent* exactly as above, except with **7.11** changed to **7.11***. Thus Γ is $\vdash^*$-*inconsistent* if and only if, for some F,

we have $\Gamma \vdash^* F$ and $\Gamma \vdash^* \neg F$, or equivalently, ... if and only if we have $\Gamma \vdash^* G$ for every formula G .

Lemma 7.12
(i) *If* $\Gamma \vdash F$ *and* Γ *is* $\vdash$*-consistent, then* $\Gamma \cup \{F\}$ *is* $\vdash$*-consistent.*
(ii) *If* F *is a* **sentence**, *and* $\Gamma \nvdash F$ *(so* Γ **is** $\vdash$*-consistent!), then* $\Gamma \cup \{\neg F\}$ *is* $\vdash$*-consistent.*

Lemma 7.12*
(i) *If* $\Gamma \vdash^* F$ *and* Γ *is* $\vdash^*$*-consistent, then* $\Gamma \cup \{F\}$ *is* $\vdash^*$*-consistent.*
(ii) *For* **any** F, *if* $\Gamma \nvdash^* F$ *(so* Γ **is** $\vdash^*$*-consistent!), then* $\Gamma \cup \{\neg F\}$ *is* $\vdash^*$*-consistent.*

Parts (i) of these last two results are almost obvious, with the same proof as that for **3.12**(i).

Deduction of (ii) in 7.12 from 7.9 (and of (ii) in 7.12* from 7.9*). The proof proceeds exactly as with that of **3.12**(ii). That is, assuming that $\Gamma \cup \{\neg F\}$ is $\vdash$-inconsistent (respectively, $\vdash^*$-inconsistent), we can deduce anything from it, in particular, F itself. Then we can apply the Deduction Lemmas, in the first case because we assumed F to be a sentence. Thus we have $\Gamma \vdash \neg F \rightarrow F$ (resp. $\Gamma \vdash^* \neg F \rightarrow F$). The rest of the proof is just 'propositional', appealing to **7.2** (resp. **7.2***) to infer $\neg\neg F$, and thence F, thereby showing that F *is* derivable from Γ .

It is fairly clear how **satisfiable** ought to be defined here, recalling that the analogue of a truth assignment is an interpretation V (respectively, in the $*$-case, a pair $(V, \underline{v})$—that is, an interpretation V and some $\underline{v}$ from V).

Definition. A set of $1^{\underline{\text{st}}}$order formulae is said to be $\models$-satisfiable if and only if there is an interpretation V of our language such that all formulae in the set are true in V.

Definition*. A set of $1^{\underline{\text{st}}}$order formulae is said to be $\models^*$-satisfiable if and only if there is an interpretation V of our language and some $\underline{v}$ from V such that all formulae in the set are true at $\underline{v}$ from V.

Theorem 7.15 *If* Γ *is* $\vdash$*-consistent, then it is* $\models$*-satisfiable.*

Theorem 7.15* *If* Γ *is* $\vdash^*$*-consistent, then it is* $\models^*$*-satisfiable.*

Deduction of the completeness result 7.16 from 7.12 and 7.15 (and of 7.16* from 7.12* and 7.15*).

These are essentially identical to the proof of **3.16**, but there is a small wrinkle in the non-$*$-case, so let's do the $*$-case first.

Assume that $\Gamma \nvdash^* F$. Then, by **7.12*(ii)**, $\Gamma \cup \{\neg F\}$ is $\vdash^*$-consistent, and therefore is $\models^*$-satisfiable by **7.15***. But that means we have an interpretation V and some $\underline{v}$ from V such that $\neg F$ as well as all $G \in \Gamma$ are true at $\underline{v}$ from V. But then F is not true at $\underline{v}$ from V. This shows that $\Gamma \not\models^* F$, as required.

As for the other case, we'd like to just copy this proof, only altering "at $\underline{v}$ from V" to "in V". The trouble is that F is not necessarily a sentence—this proof is fine if F is a sentence. So just replace F by one of its closures $\bar{F}$, which is a sentence formed by putting $\forall x$'s in front of F for various variables x, enough to convert it into a sentence. One needs to do this for all variables x which have a free occurrence in F. (And if you do it only for those variables, then the closure is unique up to altering the order of the $\forall x$'s in front of F. See Exercise 6.3.) Now all we need, in order to make the proof work, are the facts that $\Gamma \vdash \bar{F} \implies \Gamma \vdash F$ and $\Gamma \models F \implies \Gamma \models \bar{F}$. The first of these is immediate because $\{\forall x H\} \vdash H$, using (MP) and that $\forall x H \to H$ is an axiom. The second is immediate from the fact that $\{H\} \models \forall x H$. (This was proved in Chapter 6—after all, it is the basis for our new rule of inference.)

To express the proof of the non-$*$ case in a more succinct manner :

$$\Gamma \nvdash F \implies \Gamma \nvdash \bar{F} \implies \Gamma \cup \{\neg \bar{F}\} is \vdash\text{-}consistent \implies \Gamma \cup \{\neg \bar{F}\} is \models\text{-}satisfiable$$

using **7.12 (ii)** and **7.15**. But that means we have an interpretation V such that $\neg \bar{F}$ as well as all $G \in \Gamma$ are true in V. But then $\bar{F}$ is not true in V. This shows that $\Gamma \not\models \bar{F}$, and therefore $\Gamma \not\models F$, completing the proof.

All that is left to prove is the "consistency implies that a model exists" result, namely **7.15** (resp. **7.15***). This is **Henkin's Theorem**. It requires a thorough discussion.

7.7 Preliminary discussion of Henkin's Theorem.

From one viewpoint, Henkin's theorem seems a bit miraculous. How can you expect to construct some mathematical objects (an interpretation)

merely out of a bunch of statements which happen to be mutually consistent?

You might wish to read **Appendix G**. It is written from a naive viewpoint in which no analysis of language (no formal language) is introduced. It concerns the very seminal affair in mathematics of showing that *the parallel postulate of Euclid cannot be proved*. This is done by producing a model for a set of axioms for non-Euclidean geometry. There and elsewhere that appendix illustrates more than once an informal version of the converse of Henkin's Theorem, basically that "existence of a model implies consistency". (This is immediate from, and more-or-less equivalent to, soundness.)

From the naive viewpoint prior to introducing formal languages, getting the converse of this, namely Henkin's Theorem, seems to present insuperable problems.

On the other hand, in abstract algebra you may have seen a situation which is analogous to what we are about to do (in fact, it can be regarded as a special case): for example, the construction of *the free group on two generators x and y*. You are just given the two symbols, the three axioms defining *group*, and are asked to produce an example of a group in which all the elements are "words in x, x^{-1}, y and y^{-1}", and such that no two such words give the same group element "unless that can be proved from the three axioms". Now this is of course not done to prove the consistency of the set of three axioms by coming up with an actual group—there are plenty of groups which occur in our 'natural mathematical surroundings', such as symmetry groups. In any case there's always the trivial group with only one element. But the free group is needed at a more sophisticated level, for example, to deal with groups given by generators and relations. And it doesn't occur very easily in our natural surroundings. (Actually it does occur as the fundamental group, of continuous loops up to continuous deformations, of the figure eight or the doubly-punctured plane.) But a careful formal construction of the free group can be given, simply using the given symbols. And that's just what Henkin will have us do: construct a model for a consistent set of premisses directly out of the symbols in the language, more specifically, out of the terms. Actually, if you are familiar with the construction of free groups, it can be an instructive exercise here to add one or two sentences to the axioms for groups so that you have an inconsistent set. Then try to see what happens when you mimic the free group construction. For example, add $\exists x\ \neg x^4 \approx 1\ \wedge\ \forall y\ y^2 \approx 1$.

Before doing the construction of the interpretations, let's think very briefly about what might be termed a philosophical question (or maybe an unanswerable question), risking the wrath of most mathematicians, and probably most philosophers as well. The question is: "What really *is* a mathematical object?" This book began with a fairly blatant attitude that the symbols used in the formal languages are almost physical objects. But the physical objects are really the *names* of these symbols, those physical objects being the ink marks, chalk marks, noises made by human vocal chords, etc., which we use to communicate (sometimes with other parts of our own brain!). We'll take the simple-minded attitude that we are naming actual but abstract things which exist in some sort of 'ideal world' or 'world of abstract objects'. But we won't get into any philosophical discussion of linguistics or ontology— names, names of names, 'do they really exist?", meaning of the verb "to exist", and so on. They have no real effect on what we're doing, and some would even say they have no effect on its significance. In any case, you have to start somewhere, and assume that some kind of objects exist. So we think of a world of abstract objects where all our symbols live, where all the mathematical objects live, and where Beethoven's symphonies probably also live. (After all, "the 5th" must be something, and it's certainly not any particular actual performance of the symphony, nor any particular conductor's written score, nor trombonist's—so it seems that it is an abstract object—for jazz pieces, you'd presumably need equivalence classes!) Thus it appears to be rather an excellent idea to construct a model for a set of formulae directly out of the symbols in the language where those formulae live.

Let's get on with it, I'm sure you're saying. Fix a $1^{\underline{\text{st}}}$order language, as we've been doing, and also fix a set, Γ, of formulae from it.

Definition of V_Γ^*. The *autological**-Γ-*interpretation*, denoted V_Γ^*, of the language, is defined as follows. As a set, V_Γ^* is a set whose members are equivalence classes of terms in the language. We use all terms, and define the equivalence relation on the set of all terms by

$$t \sim s \qquad \Longleftrightarrow \qquad \Gamma \ \vdash^* \ t \approx s \ .$$

The equivalence class of a term t, consisting of all terms equivalent to it, is denoted $[t]$. To summarize

$$V_\Gamma^* \ := \ \{ \ [t] \ : \ t \text{ is a term } ; \ [t] = [s] \Leftrightarrow t \sim s \Leftrightarrow \Gamma \ \vdash^* \ t \approx s \ \} \ .$$

The remaining data for this interpretation is given as follows :

$$\text{For each constant symbol } c, \text{ let } c^{V_\Gamma^*} := [c] \ .$$

For each $n-$ary function symbol f, let $f^{V_\Gamma^*}$ be the function $(V_\Gamma^*)^n \to V_\Gamma^*$
which maps $([t_1], \cdots, [t_n])$ to $[ft_1 \dots t_n]$.
For each $n-$ary relation symbol r, let $r^{V_\Gamma^*}$ be the subset of $(V_\Gamma^*)^n$ defined by
$$([t_1], \cdots, [t_n]) \in r^{V_\Gamma^*} \qquad \Longleftrightarrow \qquad \Gamma \ \vdash^* \ rt_1 \dots t_n \ .$$

There are several things to check before we can be sure that this is a bona fide interpretation: that $\sim$ is an equivalence relation, and that both of the last two definitions are independent of the choices of representatives for equivalence classes. The reader should probably skip this on first reading. The straightforward but tedious details of these verifications are given in two sections immediately after this section.

As you may suspect from the small print and the $*$'s in the notation, the interpretation V_Γ^* is the one which will be relevant to system*. More precisely, if $\underline{[x]}$ from V_Γ^* is defined by $\underline{[x]} \ := \ ([x_1], [x_2], [x_3], \cdots)$, then $\underline{[x]}$ from V_Γ^* will show that Γ is $\models^*$-satisfiable. But we'll need some extra conditions on Γ for this to work directly (as done in the section after verifying the meaningfulness of the above definition). And then (in the section after that) we'll do some squirming around to prove **7.15*** for arbitrary Γ.

The interpretation which is relevant in the other case (i.e. the system, rather than the system*) is the following.

Definition of V_Γ. Assume that our fixed $1^{\underline{\text{st}}}$order language has at least one constant symbol, so that the set of objects just below is non-empty. The *autological-Γ-interpretation*, denoted V_Γ, of the language, is defined in the same way that V_Γ^* was defined just above, with two changes.

Firstly, **use only terms which contain no variables**. We'll call these *fixed terms*. They can be defined inductively by starting with constant symbols, and inductively building $ft_1 \dots t_n$, where each t_i is a fixed term.

The only other change is that, in the definition of the equivalence relation, and in the definition of the relation component of the interpretation, use $\vdash$, rather than $\vdash^*$.

Checking that V_Γ is a well-defined interpretation is virtually identical to doing the same for V_Γ^*, and will be omitted.

Exercise 7.1 Write out the definition of V_Γ in detail. Then, using as a guide the two sections after this one, prove in detail that V_Γ is a well-defined interpretation.

For suitable Γ, it will be shown later that V_Γ is a model for Γ. Then the details of getting a proof of **7.15** for arbitrary Γ will be similar to those in

the $*$-case. These details will in particular involve enriching the language by adding lots of new constant symbols, so there will be no problem with the restriction above to languages with at least one.

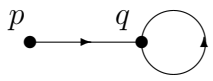
The rest of this section is taken up with examples of these definitions from the languages of groups and directed graphs. (Here we shall move back and forth between the $*$- and non-$*$-case without using this smaller print, but then revert to that convention for the the $*$-case in the rest of this chapter.)

The examples below should also give you a taste of how one actually comes up with specific derivations within our two systems. In general, that's a bit of a "mug's game". Neither mathematicians nor logicians spend much time doing that. They spend their time proving in some general situations that derivations do exist, or don't exist, and other things. Even checking that something claimed to be a derivation actually is one, is far more suited to the type of intelligence (if you can call it that) possessed by a computer, than to human intelligence.

Example 1. Take the language to be that of directed graphs. Take Γ to contain only the single formula $\forall x \exists y\ xry$, which is a sentence saying that every vertex has at least one directed edge emanating from it. Being a sentence, it doesn't matter which specific variables x and y are, as long as they are distinct. Since there are no constant nor function symbols, the terms are simply the variables, and so we have, as a set,

$$V_\Gamma^* \ = \ \{[x_1], [x_2], [x_3], \cdots\} \ .$$

The various $[x_i]$ are all distinct here, but this must be proved. Consider the following graph :



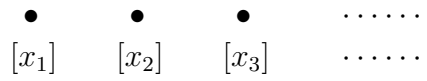
The one formula in Γ is true in this interpretation. (So we're certainly not looking at V_Γ^* to show that this Γ has a model, since we just wrote one down, and it will turn out that V_Γ^* actually is **not** a model in this case !)

To show that $[x_i] \neq [x_j]$ for $i \neq j$, we must show $x_i \not\sim x_j$; that is, $\Gamma \not\vdash^* x_i \approx x_j$. This follows by soundness as long as $\Gamma \not\models^* x_i \approx x_j$. So we

need an interpretation V which is a model for Γ, and such that, for some $\underline{v}$ from V, the formula $x_i \approx x_j$ is not true there. But the above interpretation clearly does the trick, by taking $\underline{v}$ to have p in the ith slot, and q in the jth.

To complete the determination of V_Γ^*, we must calculate $r^{V_\Gamma^*}$. I claim that it is *empty*; i.e. there are no directed edges at all. For this we must show that $\Gamma \nvdash^* x_i r x_j$ for *every* i and j, not necessarily distinct. And again by soundness, it suffices to show that $\Gamma \not\models^* x_i r x_j$. The model above will again suffice for this. For $i \neq j$, put q in the ith slot, and p in the jth. For $i = j$, put p in that slot. (Actually, we could have just put p in both slots, the ith and the jth, and not divided cases.)

Therefore, V_Γ^* is pictured as an infinite sequence of isolated vertices, with no edges between them (and no loops).



In particular, V_Γ^* is clearly not a model for Γ in this case. The conditions of maximal consistency and Henkinianity (in the main theorems ahead which tell us when V_Γ^* and V_Γ *are* models) do not hold for this particular Γ. Despite this, these autological interpretations turn out to be just the job for the general proof of Henkin's theorem. Of course, we have no trouble seeing that the set Γ in this particular example is satisfiable in both senses, since we wrote down a simple model for it above.

You should also notice the following from the discussion. The best way to prove that a derivation of some formula doesn't exist usually is to reduce it to a semantic question; that is, find a model for the set of premisses, but such that the formula in question is false in that interpretation, and so cannot have a derivation, by soundness. But this is what you've been doing all your life, in a less formal setting: finding counterexamples to justify your claim of falseness for some proposed 'theorems'.

Example 2. We cannot illustrate V_Γ with the data in the previous example, because the set of constant symbols is empty. So enrich the language of directed graph theory with two extra constant symbols c and d. But again use the same singleton Γ as in the previous example. It is clear that, as a

set,

$$V_\Gamma \;=\; \{[c],[d]\}\ .$$

We claim that $[c] \neq [d]$. We must show that $c \not\sim d$, i.e. that $\Gamma \nvdash\ c \approx d$. By soundness, it suffices to see that $\Gamma \not\models\ c \approx d$. But the 'same' interpretation as in Example 1 does this, except that we must enrich the interpretation by specifying c^V and d^V. So take $c^V = p$ and $d^V = q$. (We could also do it the other way round. But if we took c^V and d^V to be the same, we'd get an interpretation which wouldn't do what we want.) Now we have $c^V \neq d^V$, so the sentence $c \approx d$ is false in the interpretation which has been specified. This shows, as required, that $\Gamma \not\models\ c \approx d$.

To complete the explicit determination of V_Γ, we must determine r^{V_Γ} ; that is, we must figure out whether there are any directed edges, including loops. Just as in Example 1, there aren't any:

Exercise 7.2 Show that r^{V_Γ} is indeed empty.

Thus V_Γ here just consists of two vertices, with no edges (or loops). In particular, the one sentence in Γ is false in V_Γ. So, although V_Γ is an interpretation of the language of (directed graphs enriched with two constant symbols), it is not a model for Γ.

Example 3. Now take the language to be that of group theory. Take any example of Γ which includes at least the three sentences which are the *group axioms*. These three are

$$\forall x_1 \forall x_2 \forall x_3\ (x_1 \times x_2) \times x_3\ \approx\ x_1 \times (x_2 \times x_3)\ ,$$

$$\forall x_1\ (x_1 \times 1 \approx x_1\ \wedge\ 1 \times x_1 \approx x_1)\ ,$$

and

$$\forall x_1\ (x_1 \times x_1^{-1} \approx 1\ \wedge\ x_1^{-1} \times x_1 \approx 1)\ .$$

Then the only possibility for V_Γ is that it contains only the single element [1]. And so, if Γ also includes some interesting existential statement, even as simple as $\exists x\ \neg x \approx 1$, or perhaps asserting the existence of an element of some particular order bigger than 1, then V_Γ will definitely not be a model for Γ.

Exercise 7.3 Show in detail that, indeed, $V_\Gamma = \{[1]\}$ here.

Example 4. Again let's use the language of group theory, and take Γ to be the union of the set of three axioms defining *group* with

$$\{\ \forall x_1\ x_1 \times x_1 \approx 1\ ,\ \forall x_1\ x_1 \times (x_1 \times x_1) \approx 1\ \}\ .$$

Note that the last two say that the square and the cube of every element are both the identity element.

By the previous example, in this case $V_\Gamma = \{[1]\}$ is in fact a model for the five-element set Γ, though a very trivial one.

What about V_Γ^* ? I claim that it also reduces to simply $\{[1]\}$. The proof of this will illustrate how to go through the hoops of converting a very simple mathematical argument into a series of $1^{\underline{\text{st}}}$order derivations.

The mathematical argument is proving the fact that a group G, in which every element g satisfies $g^3 = 1 = g^2$, is necessarily the trivial group, consisting of only the identity element. A group theorist would just say: Each element has order dividing both 2 and 3, and so dividing $\text{GCD}(2,3) = 1$, and so equal to 1, as required. The proof from scratch is pretty quick too: For each $g \in G$, we have $g = (g^3)(g^2)^{-1} = 1 \cdot 1^{-1} = 1$.

To do this explicitly in $1^{\underline{\text{st}}}$order is basically to give derivations to show $V_\Gamma^* = \{[1]\}$. We must show, for each term t, that $[t] = [1]$; i.e. that $t \sim 1$; i.e. that $\Gamma \vdash^* t \approx 1$. This can be done by induction on terms.

The only constant symbol is 1, and $\Gamma \vdash^* 1 \approx 1$, since $1 \approx 1$ is one of the logical axioms.

As for variables, let x denote any one of the variables x_i. If we can show that

$$\text{(i)} \qquad \Gamma\ \vdash^*\ (x \times (x \times x)) \times (x \times x)^{-1} \approx 1\ ,$$

and

$$\text{(ii)} \qquad \Gamma\ \vdash^*\ x \approx (x \times (x \times x)) \times (x \times x)^{-1}\ ,$$

then the logical axiom

$$(s_1 \approx s_2\ \wedge\ s_2 \approx s_3)\ \rightarrow\ s_1 \approx s_3\ ,$$

together with (MP), will give us $\Gamma \vdash^* x \approx 1$, as required, by taking

$$s_1 = x\ ,\ s_2 = (x \times (x \times x)) \times (x \times x)^{-1}\ \text{ and }\ s_3 = 1\ .$$

In dealing with (i) and (ii), let's simply write x^3 for $(x \times (x \times x))$ and x^2 for $x \times x$. It is easy to see that $\Gamma \vdash^* x^3 \approx 1$ as follows:

One of the premisses is $\forall x_1\ F$, where $F = x_1^3 \approx 1$. A logical axiom is $\forall x_1\ F \rightarrow F^{[x_1 \mapsto x]}$. Now just apply (MP), since the formula to be derived is $F^{[x_1 \mapsto x]}$.

Similarly, we have $\Gamma \vdash^* x^2 \approx 1$.

Showing that a derivation exists to establish (i) can now be done by successively showing that each of the following can be derived from Γ in the $\vdash^*$ sense of "derived":

(I) $x^3 \times (x^2)^{-1} \approx 1 \times (x^2)^{-1}$, from $x^3 \approx 1$, a logical axiom, and (MP).

(II) $1 \times (x^2)^{-1} \approx (x^2)^{-1}$, from a group axiom, etc.

(III) $(x^2)^{-1} \approx 1^{-1}$, from $x^2 \approx 1$, a logical axiom, and (MP).

(IV) $1 \times 1^{-1} \approx 1$, from a group axiom.

(V) $1^{-1} \approx 1$, from (IV), a group axiom and a logical axiom.

(VI) $(x^2)^{-1} \approx 1$, from (V), (III) and a logical axiom.

(VII) $1 \times (x^2)^{-1} \approx 1$ from (VI), (II) and a logical axiom.

(VIII) $x^3 \times (x^2)^{-1} \approx 1$ from (VII), (I) and a logical axiom.

This last one is finally what we want.

Exercise 7.4 Write down in all detail the derivations and shortcut derivations in the above argument, including justifications, where the shortcutted ones may include references to previous derivations in your list.

Exercise 7.5 Write down in all detail the complete derivation establishing (i), with no shortcuts, and with justifications for each line.

Sorry, this last one probably resembles computer science assignment questions which sometimes come from those profs who never do them nor mark them. But it does make the point of how tedious writing out complete derivations can be, and how unreadable the final product is for a normal human.

Showing that a derivation exists to establish (ii) is slightly shorter than for (i). It uses the associativity group axiom and then cancels x^2 with its 'inverse'. It uses the group axioms in Γ, but not the other two sentences.

Exercise 7.6 Write down such an argument, if you wish in the style preceding 7.4, or else explicitly, as in 7.4 or 7.5.

At this point, we have established the initial cases in the induction on terms to show $\Gamma \vdash^* t \approx 1$ for any term t. For the inductive step, we assume that $\Gamma \vdash^* s \approx 1$ and $\Gamma \vdash^* t \approx 1$, and must deduce that $\Gamma \vdash^* (s \times t) \approx 1$ and $\Gamma \vdash^* t^{-1} \approx 1$.

For the former, two logical axioms are $t \approx 1 \ \rightarrow \ (s \times t) \approx (s \times 1)$ and $s \approx 1 \ \rightarrow \ (s \times 1) \approx (1 \times 1)$. Using (MP) a couple of times, etc. then does it.

Exercise 7.7 Write down in all detail the derivations and shortcut derivations in the above argument, including justifications, where the shortcutted ones may include references to previous derivations in your list.

Exercise 7.8 Give an argument for the other half of the inductive step.

Taken together, this completes the argument to establish that V_Γ^* here just has the one element. One lesson to be gleaned is that working out these autological interpretations in specific cases, with careful arguments, is not a job which is particularly appealing. We shall state the answers in the remaining examples, but be considerably less detailed in the arguments.

Example 5. Now let's take Γ as above, except change the second set to

$$\{ \ \forall x_1 \ (x_1 \times x_1) \times (x_1 \times x_1) \approx 1 \ , \ \forall x_1 \ (x_1 \times (x_1 \times x_1)) \times (x_1 \times (x_1 \times x_1)) \approx 1 \ \} \ .$$

So all we've done is change the last two formulae of the previous example so that now the 4th and 6th powers of any element are both the identity element. In a group with this property, it follows that the square of every element is 1, by an argument much like that in the previous example.

But there is also an argument that, in a group where every element g satisfies $g^2 = 1$, i.e. $g^{-1} = g$, the operation is commutative:

$$gh = (gh)^{-1} = h^{-1}g^{-1} = hg,$$

the middle equality being a general property of groups.

By converting all this into $1^{\underline{\text{st}}}$order derivations, or at least into arguments that certain $1^{\underline{\text{st}}}$order derivations exist, it can then be established that here V_Γ^* is the set

$$\{[1], [x_1], [x_2], [x_1 \times x_2], [x_3], [x_1 \times x_3], [x_2 \times x_3], [x_1 \times x_2 \times x_3], [x_4],$$

$$[x_1 \times x_4], [x_2 \times x_4], [x_3 \times x_4], [x_1 \times x_2 \times x_4], [x_1 \times x_3 \times x_4], \cdots\cdots \} \ .$$

The 'fatter' elements above should have a pair of associativity brackets added, but it doesn't matter which way you add them. The pattern should

be fairly clear. We are also claiming that all the elements in the list are distinct. (A group theorist would say that this group is the direct sum of countably many copies of $\mathbf{Z}_2$.)

Example 6. Now enrich the language of group theory by adding an extra constant symbol c, and take Γ to contain only the three group axioms. Then I claim that

$$V_\Gamma \;=\; \{\, [1]\,,\, [c]\,,\, [c^{-1}]\,,\, [c{\times}c]\,,\, [c^{-1}{\times}c^{-1}]\,,\, [c{\times}c{\times}c]\,,\, [c^{-1}{\times}c^{-1}{\times}c^{-1}]\,,\; \cdots \,\}\,,$$

and that the listed elements are all distinct. Again I have omitted a couple of pairs of brackets. Note how the elements of this set correspond in a natural way to integers. For those who have studied groups, this group is infinite cyclic, or equivalently, isomorphic to $\mathbf{Z}$ under ordinary addition of integers.

Example 7. Here, do the same as in Example 6, except for adding one extra formula, $c \times c \approx 1$, to Γ, which now has four formulae in it. In this case,

$$V_\Gamma \;=\; \{\, [1]\,,\; [c]\,\}\,,$$

which is called a cyclic group of order 2. It is the 'same' as (*isomorphic* to) the group $\{\, 1\,,\; -1\,\}$ under ordinary multiplication of numbers.

Example 8. Here, do the same as in Example 6, except for adding two new constant symbols, c and d, to the language of group theory. So Γ is back to just consisting of the three group axioms. In this case, V_Γ is actually the free group on two generators mentioned in the initial discussion of Henkin's theorem. We won't attempt to describe it here. This and Example 6 are the simplest cases of the claim there that the formal constructions of free groups are special cases of Henkin's theorem.

Example 9. Here, go back to just the language of group theory, but make Γ into an infinite set, consisting of the three group axioms, plus the following infinite set of formulae

$$x_2 \approx x_3\;,\; x_2 \approx x_4\;,\; x_2 \approx x_5\;,\; \cdots\quad .$$

In this case, V_Γ^* turns out also to be the free group on two generators, $[x_1]$ and $[x_2]$.

7.8 Verification that $\sim$ in the definition of V_Γ^* is an equivalence relation.

To show that $t \sim t$, note that $\Gamma \vdash^* t \approx t$, since $t \approx t$ is an axiom.

To show $s \sim t \Longrightarrow t \sim s$, use that $s \approx t \to t \approx s$ is an axiom. We are given $\Gamma \vdash^* s \approx t$, and can then conclude $\Gamma \vdash^* t \approx s$ using (MP).

To show $t_1 \sim t_2$ and $t_2 \sim t_3$ implies $t_1 \sim t_3$, we are given $\Gamma \vdash^* t_1 \approx t_2$ and $\Gamma \vdash^* t_2 \approx t_3$; and must show that $\Gamma \vdash^* t_1 \approx t_3$. From $\{F, G\} \vdash^* F \wedge G$, which is **7.7***, we see that $\Gamma \vdash^* (t_1 \approx t_2 \ \wedge \ t_2 \approx t_3)$. But we have $\vdash^* (t_1 \approx t_2 \ \wedge \ t_2 \approx t_3 \to t_1 \approx t_3)$, because the formula is an axiom. So (MP) does the job.

7.9 Verification that $f^{V_\Gamma^*}$ and $r^{V_\Gamma^*}$ are well defined.

We must show, for any positive integer n, that if

$$t_1 \sim s_1 \ , \ t_2 \sim s_2 \ , \ \cdots , \text{ and } t_n \sim s_n \ ,$$

then for all n-ary function symbols f, and all n-ary relation symbols r,

$$ft_1 \dots t_n \ \sim \ fs_1 \dots s_n \quad \text{and} \quad \Gamma \vdash^* rt_1 \dots t_n \implies \Gamma \vdash^* rs_1 \dots s_n \ . \qquad (*)$$

For the first of these, we shall show by induction on j, for fixed n, that if $t_i \sim s_i$ for $1 \le i \le j \le n$, then $ft_1 \dots t_n \ \sim \ fs_1 \dots s_j t_{j+1} \dots t_n$. The end case, $j = n$, is then the result we want.

When $j = 1$, we are given $\Gamma \vdash^* t_1 \approx s_1$, and we know that

$$t_1 \approx s_1 \ \to \ ft_1 \dots t_n \approx fs_1 t_2 \dots t_n$$

is an axiom. Applying (MP), we get $\Gamma \vdash^* ft_1 \dots t_n \approx fs_1 t_2 \dots t_n$, as required.

For the inductive step, we are given $\Gamma \vdash^* t_i \approx s_i$ for $1 \le i \le j$. Using the first "$j-1$" of these and the inductive hypothesis, we have

$$\Gamma \vdash^* ft_1 \dots t_n \ \approx \ fs_1 \dots s_{j-1} t_j t_{j+1} \dots t_n,$$

or, equivalently, $ft_1 \dots t_n \ \sim \ fs_1 \dots s_{j-1} t_j t_{j+1} \dots t_n$. But

$$t_j \approx s_j \ \to \ fs_1 \dots s_{j-1} t_j t_{j+1} \dots t_n \approx fs_1 \dots s_j t_{j+1} \dots t_n$$

is an axiom. Since $\Gamma \vdash^* t_j \approx s_j$ is given, we conclude from (MP) that

$$\Gamma \vdash^* fs_1 \dots s_{j-1} t_j t_{j+1} \dots t_n \approx fs_1 \dots s_j t_{j+1} \dots t_n \ ,$$

or, equivalently, $fs_1 \dots s_{j-1} t_j t_{j+1} \dots t_n \sim fs_1 \dots s_j t_{j+1} \dots t_n$. By the transitivity of $\sim$, we get $ft_1 \dots t_n \ \sim \ fs_1 \dots s_j t_{j+1} \dots t_n$, completing the proof.

The proof of the second one in the display $(*)$ is formally very similar, replacing the equality symbol by the relation symbol r; using the 4th rather than the 5th of the equality axioms everywhere; and noting the effect of substitution, as we do below.

So fix n, and we shall show by induction on j, that if, for $1 \le i \le j \le n$, we have $t_i \sim s_i$, then $\Gamma \vdash^* rt_1 \dots t_n$ implies $\Gamma \vdash^* rs_1 \dots s_j t_{j+1} \dots t_n$. The end case, $j = n$, gives the result we want.

When $j = 1$, we are given $\Gamma \vdash^* t_1 \approx s_1$, and we know that

$$t_1 \approx s_1 \ \rightarrow \ (rt_1 \dots t_n \rightarrow rs_1 t_2 \dots t_n)$$

is an axiom, since

$$(rx_1 t_2 \dots t_n)^{[x_1 \to t_1]} = rt_1 \dots t_n \ ,$$

and

$$(rx_1 t_2 \dots t_n)^{[x_1 \to s_1]} = rs_1 t_2 \dots t_n \ .$$

Applying (MP), we get $\Gamma \vdash^* (rt_1 \dots t_n \rightarrow rs_1 t_2 \dots t_n)$, Thus, if $\Gamma \vdash^* rt_1 \dots t_n$, then another application of (MP) yields $\Gamma \vdash^* rs_1 t_2 \dots t_n$, as required.

For the inductive step, we are given $\Gamma \vdash^* t_i \approx s_i$, for $1 \le i \le j$, and we know that

$$t_j \approx s_j \ \rightarrow \ (rs_1 \dots s_{j-1} t_j \dots t_n \rightarrow rs_1 \dots s_j t_{j+1} \dots t_n)$$

is an axiom, since

$$(rs_1 \dots s_{j-1} x_j t_{j+1} \dots \dots t_n)^{[x_j \to t_j]} \ = \ rs_1 \dots s_{j-1} t_j \dots t_n \ ,$$

and

$$(rs_1 \dots s_{j-1} x_j t_{j+1} \dots \dots t_n)^{[x_j \to s_j]} \ = \ rs_1 \dots s_j t_{j+1} \dots t_n \ .$$

Applying (MP), we get $\Gamma \vdash^* (rs_1 \dots s_{j-1} t_j \dots t_n \rightarrow rs_1 \dots s_j t_{j+1} \dots t_n)$. Now we also have $\Gamma \vdash^* (rt_1 \dots t_n \rightarrow rs_1 \dots s_{j-1} t_j \dots t_n)$, by the inductive hypothesis. Thus, given that $\Gamma \vdash^* rt_1 \dots t_n$, two applications of (MP) yield $\Gamma \vdash^* rs_1 \dots s_j t_{j+1} \dots t_n$, as required.

7.10 Proof of Henkin's Theorem

for Γ which are maximally consistent and Henkinian.

First we'll do the entire analysis for system, and then do the corresponding analysis for system*. So for the next few pages we are working in the interpretation V_Γ.

Lemma 7.10.1. *For any* **fixed** *terms s and t_i, we have* $s^{([t_1],[t_2],\cdots)} = [s]$.

Proof. For a constant symbol c, we have $c^{([t_1],[t_2],\cdots)} = c^{V_\Gamma} = [c]$, first using the definition of $c^{\underline{v}}$ in general, then the definition of V_Γ .
Proceeding inductively,

$$(fs_1 \dots s_n)^{([t_1],[t_2],\cdots)} \;=\; f^{V_\Gamma}(s_1^{([t_1],[t_2],\cdots)}, \cdots, s_n^{([t_1],[t_2],\cdots)})$$
$$=\; f^{V_\Gamma}([s_1], \cdots, [s_n]) \;=\; [fs_1 \dots s_n]\ .$$

Lemma 7.10.2. *If F is an atomic* **sentence***, then*

$$F \text{ is true in } V_\Gamma \quad \Longleftrightarrow \quad \Gamma \vdash F\ .$$

Proof. (The significance of restricting to sentences is that all terms in the proof below are fixed terms.)

When F is $s_1 \approx s_2$, we have, for every $\underline{[t]}$ from V_Γ,

$$s_1 \approx s_2 \text{ is true at } \underline{[t]} \text{ from } V_\Gamma \iff s_1^{\underline{[t]}} = s_2^{\underline{[t]}}$$

$$\iff [s_1] = [s_2] \iff s_1 \sim s_2 \iff \Gamma \vdash s_1 \approx s_2\ .$$

When F is $rs_1 \dots s_n$, we have, for every $\underline{[t]}$ from V_Γ,

$$rs_1 \dots s_n \text{ is true at } \underline{[t]} \text{ from } V_\Gamma \iff (s_1^{\underline{[t]}}, \cdots, s_n^{\underline{[t]}}) \in r^{V_\Gamma}$$

$$\iff ([s_1], \cdots, [s_n]) \in r^{V_\Gamma} \iff \Gamma \vdash rs_1 \dots s_n\ .$$

In both of the above displays, the first step is Tarski's definition, the second is **7.10.1**, and the last is the overall definition of the interpretation V_Γ . This completes the proof.

Now, for any **sentence** F, relabel as $(\mathsf{YES})_F$ the statement in the previous lemma; that is,

$\underline{\underline{(\mathsf{YES})}}_F$: $\quad F \text{ is true in } V_\Gamma \quad \Longleftrightarrow \quad \Gamma \vdash F\ .$

Lemma 7.10.3. *If $(\mathsf{YES})_F$ and $(\mathsf{YES})_G$ both hold, then $(\mathsf{YES})_{F\wedge G}$ also holds (for* **sentences** *F and G).*

Proof. $F \wedge G$ is true in $V_\Gamma \iff$ both F and G are true in V_Γ

$$\iff \Gamma \vdash F \text{ and } \Gamma \vdash G \iff \Gamma \vdash F \wedge G\ .$$

The last step uses **7.7**.

Definition. A set Γ of formulae is said to be *maximally $\vdash$-consistent* if and only if, first of all, it *is* $\vdash$-consistent, and secondly, for all **sentences** F, exactly one of $\Gamma \vdash F$ and $\Gamma \vdash \neg F$ holds. (Despite the fact that, for any **formula** G, $\Gamma \vdash G \iff \Gamma \vdash \bar{G}$, where $\bar{G}$ is the closure of G, the consistency of a set Γ does not follow from "secondly $\cdots$" above, basically because the closure of the negation can be very different from the negation of the closure!)

Lemma 7.10.4. *If Γ is maximally $\vdash$-consistent, and $(\mathsf{YES})_F$ holds, then $(\mathsf{YES})_{\neg F}$ holds (where F is a* **sentence***).*

Proof. $\neg F$ is true in $V_\Gamma \;\Leftrightarrow\; F$ is not true in $V_\Gamma \;\Leftrightarrow\; \Gamma \nvdash F \;\Leftrightarrow\; \Gamma \vdash \neg F$.

Definition. A set Γ of formulae is said to be *Henkinian* if and only if, for all formulae G of the language, and all variables x, such that $\forall x G$ is a **sentence**, there is a constant symbol c in the language (probably depending on G and x) for which

$$(\neg \forall x G \;\rightarrow\; \neg G^{[x \to c]}) \;\in\; \Gamma \;.$$

We have written this with the basic symbol $\forall$, not the abbreviation $\exists$. But if G is $\neg H$, it could have been written more simply as

$$(\exists x H \;\rightarrow\; H^{[x \to c]}) \;\in\; \Gamma \;.$$

So c becomes a kind of 'witness' to the fact that $\exists x H$ is 'true', if that fact does hold. The point is that, using (MP), once we know that $\Gamma \vdash \exists x H$, then we know that, for some c, we have $\Gamma \vdash H^{[x \to c]}$. Even better,

Lemma 7.10.5. *If Γ is both Henkinian and maximally $\vdash$-consistent, then for any* **formula** *F and variable x such that $\forall x F$ is a* **sentence***,*

$$\Gamma \vdash \forall x F \iff \text{for all fixed terms } t, \text{ we have } \Gamma \vdash F^{[x \to t]} \;.$$

Proof. For $\Longrightarrow$, we use the fact that $\forall x F \rightarrow F^{[x \to t]}$ is an axiom, and apply (MP).

For $\Longleftarrow$, if $\Gamma \nvdash \forall x F$, then $\Gamma \vdash \neg \forall x F$, by maximal consistency. But then by Henkinianity, for some constant symbol c, the formula $(\neg \forall x F \;\rightarrow\; \neg F^{[x \to c]})$ is in Γ , so applying (MP) we get $\Gamma \vdash \neg F^{[x \to c]}$. So $\Gamma \nvdash F^{[x \to c]}$ by consistency of Γ. Thus the condition, $\Gamma \vdash F^{[x \to t]}$, fails to hold for at least one fixed t, namely $t = c$, as required.

Lemma 7.10.6. *For all* Γ, x_i *and* **formulae** F*, the formula* $\forall x_i F$ *is true in* V_Γ *if and only if for all fixed terms* t*, we have that* $F^{[x_i \to t]}$ *is true in* V_Γ*.*

Proof. Taking $i = 3$ (the general case is the 'same'), we have the following sequence of mathematically equivalent statements :

$\forall x_3 F$ is true in V_Γ ;

F is true in V_Γ ;

for all fixed terms t_i, F is true at $([t_1], [t_2], [t_3], \cdots)$ from V_Γ ;

for all fixed terms t and t_i, F is true at $([t_1], [t_2], [t], \cdots)$ from V_Γ ;

for all fixed terms t and t_i, $F^{[x_3 \to t]}$ is true at $([t_1], [t_2], [t_3], \cdots)$ from V_Γ ;

for all fixed terms t, $F^{[x_3 \to t]}$ is true in V_Γ.

(Since t is fixed, we don't need to fuss about substitutability here.)

Lemma 7.10.7. *If* Γ *is both Henkinian and maximally* $\vdash$*-consistent, then* $(\mathsf{YES})_F$ *holds for all* **sentences** F *from the language.*

Proof. Proceed by induction on the total number of symbols "$\neg$", "$\wedge$" and "$\forall$" in the actual (not abbreviated) **sentence** F. (This is more than slightly different from induction on **formulae**.) When that number is zero, F is an atomic sentence, so **7.10.2** does it. For the inductive step, F is one of $\neg G$, $G \wedge H$ or $\forall x G$, where $(\mathsf{YES})_G$ and $(\mathsf{YES})_H$ hold in the first two cases. **Lemmas 7.10.3** and **7.10.4** do the those two. As for the other, note that for any fixed term t, we have $(\mathsf{YES})_{G^{[x \to t]}}$ holding, because $G^{[x \to t]}$ has one less "$\forall$" than $\forall x G$, and the same number of "$\neg$" and "$\wedge$", and also because it is in fact a sentence. (G can have no free occurrences of any variable other than possibly x, since $\forall x G$ is a sentence.) Now combining **7.10.5** with **7.10.6** yields the result :

$$\Gamma \vdash \forall x G \iff \text{for all fixed terms } t, \text{ we have } \Gamma \vdash G^{[x \to t]}$$

$$\iff \text{for all fixed terms } t, \text{ we have } G^{[x \to t]} \text{ true in } V_\Gamma$$

$$\iff \forall x G \text{ is true in } V_\Gamma \ .$$

Theorem 7.10.8. *If* Γ *is both Henkinian and maximally* $\vdash$*-consistent, then every* **formula** *in* Γ *is true in* V_Γ*. And so, by definition,* Γ *is* $\vdash$*-satisfiable, which is the conclusion of* **7.15**

Proof. This is nearly immediate from the previous lemma. We have, with $\bar{F}$ as a closure of F,

$$F \in \Gamma \implies \Gamma \vdash F \implies \Gamma \vdash \bar{F} \implies \bar{F} \text{ true in } V_\Gamma \implies F \text{ true in } V_\Gamma \ .$$

(All but the first "$\Longrightarrow$" could be "$\Longleftrightarrow$" , as it happens.)

7.10* The similar analysis for system*.

Now we'll do the analysis for system*. The numbering of results matches exactly with the previous. We had $\underline{[x]} := ([x_1], [x_2], [x_3], \cdots)$ from V_Γ^*.

Lemma 7.10.1* *For any term s, we have $s^{\underline{[x]}} = [s]$.*

Proof. For a constant symbol c, we have $c^{\underline{[x]}} = c^{V_\Gamma^*} = [c]$, first using the definition of $c^{\underline{v}}$ in general, then the definition of V_Γ^* .
Just using the definition of $x_j^{\underline{v}}$ in general, we get $x_j^{\underline{[x]}} = [x_j]$.
Proceeding inductively,

$$(f s_1 \dots s_n)^{\underline{[x]}} = f^{V_\Gamma^*}(s_1^{\underline{[x]}}, \cdots, s_n^{\underline{[x]}}) = f^{V_\Gamma^*}([s_1], \cdots, [s_n]) = [f s_1 \dots s_n] \ .$$

Lemma 7.10.2* *If F is an atomic formula, then*

$$F \text{ is true at } \underline{[x]} \text{ from } V_\Gamma^* \quad \Longleftrightarrow \quad \Gamma \vdash^* F \ .$$

Proof. When F is $s \approx t$, we have

$$s \approx t \text{ is true at } \underline{[x]} \text{ from } V_\Gamma^* \iff s^{\underline{[x]}} = t^{\underline{[x]}}$$

$$\iff [s] = [t] \iff s \sim t \iff \Gamma \vdash^* s \approx t \ .$$

When F is $r s_1 \dots s_n$, we have

$$r s_1 \dots s_n \text{ is true at } \underline{[x]} \text{ from } V_\Gamma^* \iff (s_1^{\underline{[x]}}, \cdots, s_n^{\underline{[x]}}) \in r^{V_\Gamma^*}$$

$$\iff ([s_1], \cdots, [s_n]) \in r^{V_\Gamma^*} \iff \Gamma \vdash^* r s_1 \dots s_n \ .$$

In both displays, the first step is Tarski's definition, the second is **7.10.1***, and the last is the overall definition of the interpretation V_Γ^*. This completes the proof.

Now, for any formula F, label as $(\mathsf{YES})_F$ the statement in the previous lemma; that is,

$\underline{\underline{(\mathsf{YES})}}_F$: F is true at $\underline{[x]}$ from $V_\Gamma^* \quad \Longleftrightarrow \quad \Gamma \vdash^* F$.

Lemma 7.10.3* *If $(\mathsf{YES})_F$ and $(\mathsf{YES})_G$ both hold, then $(\mathsf{YES})_{F \wedge G}$ also holds.*

Proof. $F \wedge G$ is true at $\underline{[x]}$ from $V^*_\Gamma \iff$ both F and G are true at $\underline{[x]}$ from V^*_Γ

$$\iff \Gamma \vdash^* F \text{ and } \Gamma \vdash^* G \iff \Gamma \vdash^* F \wedge G \ .$$

The last step uses **7.7***.

Definition. A set Γ of formulae is said to be *maximally $\vdash^*$-consistent* if and only if, for **all formulae** F, exactly one of $\Gamma \vdash^* F$ and $\Gamma \vdash^* \neg F$ holds. (Such a Γ is certainly $\vdash^*$-consistent, since not both hold.)

Lemma 7.10.4* *Assuming Γ is maximally $\vdash^*$-consistent, if $(\mathsf{YES})_F$ holds, then so does $(\mathsf{YES})_{\neg F}$.*

Proof. $\neg F$ is true at $\underline{[x]}$ from $V^*_\Gamma \iff F$ is not true at $\underline{[x]}$ from V^*_Γ

$$\iff \Gamma \nvdash^* F \iff \Gamma \vdash^* \neg F \ .$$

Definition. A set Γ of formulae is said to be $*$-*Henkinian* if and only if, for all formulae G of the language, and all variables x, there is a variable y in the language (probably depending on G and x) for which y is substitutable for x in G, and

$$(\neg\forall x G \ \rightarrow \ \neg G^{[x \mapsto y]}) \ \in \ \Gamma \ .$$

We have written this with the basic symbol $\forall$, not the abbreviation $\exists$. But if G is $\neg H$, it could have been be written more simply as

$$(\exists x H \ \rightarrow \ H^{[x \mapsto y]}) \ \in \ \Gamma \ .$$

So y becomes a kind of 'witness' to the fact that $\exists x H$ is 'true', if that fact does hold. The point is that, using (MP), once we know that $\Gamma \vdash^* \exists x H$, then we know that, for some y, we have $\Gamma \vdash^* H^{[x \mapsto y]}$. Even better,

Lemma 7.10.5* *If Γ is both $*$-Henkinian and maximally $\vdash^*$-consistent, then for any x and any formula F,*

$$\Gamma \vdash^* \forall x F \iff \textit{for all terms } t, \textit{ substitutable for } x \text{ in } F, \textit{ we have } \Gamma \vdash^* F^{[x \mapsto t]} \ .$$

Proof. For $\Longrightarrow$, we use the fact that $\forall x F \rightarrow F^{[x \mapsto t]}$ is an axiom, and apply (MP).

For $\Longleftarrow$, if $\Gamma \nvdash^* \forall x F$, then $\Gamma \vdash^* \neg\forall x F$, by maximal consistency. But then, for some variable y which is substitutable for x in F, by $*$-Henkinianity, the formula $(\neg\forall x F \ \rightarrow \ \neg F^{[x \mapsto y]})$ is in Γ . Thus, applying (MP), we get $\Gamma \vdash^* \neg F^{[x \mapsto y]}$. Thus $\Gamma \nvdash^* F^{[x \mapsto y]}$ by consistency of Γ. So the condition $\Gamma \vdash^* F^{[x \mapsto t]}$ fails to hold for at least one t, namely $t = y$, as required.

Lemma 7.10.6* *For all Γ and F, the formula $\forall x_3 F$ is true at $\underline{[x]}$ from V^*_Γ if and only if F is true at $([x_1], [x_2], \underline{t^{[x]}}, \cdots)$ from V^*_Γ for all terms t.*

(We have taken $i = 3$ but the general case of x_i is the 'same'.)

Proof. The first four of the statements following the Remark/Exercise below are mathematically equivalent statements, using Tarski's definition, **7.10.1***, and the meaning of "true". The equivalence of the first three gives the result.

Remark/Exercise 7.9 The last three statements are also equivalent. (Use **6.9**). Furthermore, by **7.10.5*** and **7.10.7*** below, the first and seventh statements are equivalent when Γ is $*$-Henkinian and maximally $\vdash^*$-consistent. However, for certain Γ, none of the first four is equivalent to any of the last three. Find a counterexample to demonstrate this.

$\forall x_3 F$ is true at $\underline{[x]}$ from V^*_Γ ;

for all terms t, F is true at $([x_1], [x_2], [t], \cdots)$ from V^*_Γ ;

for all terms t, F is true at $([x_1], [x_2], t^{\underline{[x]}}, \cdots)$ from V^*_Γ ;

for all terms t, $([x_1], [x_2], t^{\underline{[x]}}, \cdots) \in F^{V^*_\Gamma}$;

for all terms t, substitutable for x_3 in F, $([x_1], [x_2], t^{\underline{[x]}}, \cdots) \in F^{V^*_\Gamma}$;

for all terms t, substitutable for x_3 in F, $\underline{[x]} \in (F^{[x_3 \to t]})^{V^*_\Gamma}$;

for all terms t, substitutable for x_3 in F, $F^{[x_3 \to t]}$ is true at $\underline{[x]}$ from V^*_Γ .

Lemma 7.10.7* *If Γ is both $*$-Henkinian and maximally $\vdash^*$-consistent, then* $(\mathsf{YES})_F$ *holds for all formulae F from the language.*

Proof. Proceed by induction on the total number of symbols "$\neg$", "$\wedge$" and "$\forall$" in the actual (not abbreviated) formula F. (This is slightly different from induction on formulae.)

When that number is zero, F is an atomic formula, so **7.10.2*** does it.

For the inductive step, F is one of $\neg G$, $G \wedge H$ or $\forall y G$, where $(\mathsf{YES})_G$ and $(\mathsf{YES})_H$ hold.

To do the first two of these, use **7.10.3*** and **7.10.4***

As for the other, note that for any term t which is substitutable for y, we have $(\mathsf{YES})_{G^{[y \to t]}}$ holding, because $G^{[y \to t]}$, has one less "$\forall$" than $\forall y G$, and the same number of "$\neg$" and "$\wedge$".

We shall do the case $y = x_3$; the general case is the 'same'.

Combining **7.10.5*** with **7.10.6*** yields the result in one direction :

$$\forall x_3 G \text{ is true at } \underline{[x]} \text{ from } V_\Gamma^*$$

$$\Longrightarrow \text{ (by } \mathbf{7.10.6}^*\text{) for all terms } t, \text{ we have } G \text{ true at } ([x_1],[x_2],t^{\underline{[x]}},\cdots) \text{ from } V_\Gamma^*$$

$$\Longrightarrow \text{ for all terms } t, \text{ substitutable for } x_3 \text{ in } G,$$
$$\text{we have } G \text{ true at } ([x_1],[x_2],t^{\underline{[x]}},\cdots)$$

$$\Longrightarrow \text{ (by } \mathbf{6.9}\text{) for all terms } t, \text{ substitutable for } x_3 \text{ in } G,$$
$$\text{we have } G^{[x_3 \rightarrowtail t]} \text{ true at } \underline{[x]}$$

$$\Longrightarrow \text{ (by the inductive hypothesis) for all terms } t, \text{ substitutable for } x_3 \text{ in } G,$$
$$\text{we have } \Gamma \vdash^* G^{[x_3 \rightarrowtail t]}$$

$$\Longrightarrow \text{ (by } \mathbf{7.10.5}^*\text{) } \Gamma \vdash^* \forall x_3 G \ .$$

As for the converse,

$$\forall x_3 G \text{ is not true at } \underline{[x]} \text{ from } V_\Gamma^*$$

$$\Longrightarrow \text{ for some term } t_0, \ G \text{ is not true at } ([x_1],[x_2],[t_0],[x_4],\cdots)$$

$$\Longrightarrow \text{ (by } \mathbf{6.11} \text{ and } \mathbf{7.10.1}^*\text{) for some term } t_0, \ \underline{G} \text{ is not true at } ([x_1],[x_2],t_0^{\underline{[x]}},[x_4],\cdots)$$
$$\text{where } \underline{G} \text{ is obtained from } G \text{ by renaming the bound variables}$$
$$\text{so that } t_0 \text{ is substitutable for } x_3 \text{ in } \underline{G}$$

$$\Longrightarrow \text{ (by } \mathbf{6.9}\text{) for some term } t_0 \text{ which is substitutable for } x_3 \text{ in } \underline{G} \text{ as above,}$$
$$\underline{G}^{[x_3 \rightarrowtail t_0]} \text{ is not true at } \underline{[x]} \text{ from } V_\Gamma^*$$

$$\Longrightarrow \text{ (by the inductive hypothesis) for some term } t_0 \text{ which is substitutable for}$$
$$x_3 \text{ in } \underline{G} \text{ as above, } \Gamma \nvdash^* \underline{G}^{[x_3 \rightarrowtail t_0]}$$

$$\Longrightarrow \text{ (by } \mathbf{7.10.5}^*\text{) } \Gamma \nvdash^* \forall x_3 \underline{G}$$

$$\Longrightarrow \text{ (by } \mathbf{7.0}^* \text{ at the end of Section 7.5) } \Gamma \nvdash^* \forall x_3 G \ .$$

This completes the proof.

Theorem 7.10.8* *If* Γ *is both* $*$*-Henkinian and maximally* $\vdash^*$*-consistent, then every formula in* Γ *is true at* $\underline{[x]}$ *from* V_Γ^**. Thus,* Γ *is* $\vdash^*$*-satisfiable (by definition) which is the conclusion of* **7.15***

Proof. This is immediate from the previous lemma.

7.11 Puffing Γ up, and proving Henkin's Theorem in general.

There are two steps needed. First we expand Γ (and the language as well) to a set Λ which is Henkinian (resp. $*$-Henkinian). Then we expand Λ to a set Δ which is maximally consistent (and still Henkinian, because this time the language doesn't change).

Theorem H (Henkinizing) *Let Γ be a $\vdash$-consistent set of formulae, from a language $\mathcal{L}$. Then, in the language $\mathcal{L}_+$, which adds to $\mathcal{L}$ an infinite sequence of new constant symbols $b_1, b_2, \cdots$, there is a larger set Λ (containing Γ) which is Henkinian and still $\vdash$-consistent.*

Theorem H* (Henkinizing) *Let Γ be a $\vdash^*$-consistent set of formulae, from a language $\mathcal{L}$. Then, in the language $\mathcal{L}_+$, which adds to $\mathcal{L}$ an infinite sequence of new variables $y_1, y_2, \cdots$, there is a larger set Λ (containing Γ) which is $*$-Henkinian and still $\vdash^*$-consistent.*

Theorem 7.13 *Let Λ be any $\vdash$-consistent set. Then, from the same $1^{\underline{\text{st}}}$order language, there is a larger set Δ (containing Λ) which is maximally $\vdash$-consistent.*

Theorem 7.13* *Let Λ be any $\vdash^*$-consistent set. Then, from the same $1^{\underline{\text{st}}}$order language, there is a larger set Δ (containing Λ) which is maximally $\vdash^*$-consistent.*

Note that **7.13** and **7.13*** are analogues of **3.13**, and are also called **Lindenbaum's Theorem**.

Deduction of Henkin's Theorem, 7.15, from H and 7.13

Given a $\vdash$-consistent set Γ, expand it to Λ, and then to Δ, as in **H** and **7.13**. By **Theorem 7.10.8** of the previous section, there is an interpretation V (i.e. V_Δ) of $\mathcal{L}_+$ in which all formulae of Δ are true. We can regard V as an interpretation of $\mathcal{L}$ by simply ignoring the choices of elements b_i^V corresponding to the extra constant symbols b_i which are not in $\mathcal{L}$. But all formulae in Γ, since they are also in Δ, are true in V. So the existence of V shows that Γ is $\models$-satisfiable, as required.

Deduction of Henkin's Theorem 7.15* from H* and 7.13*

This is almost identical to the above, except that we use **Theorem 7.10.8*** to produce V and $\underline{v}$ from V (i.e. $[x]$ from V^*_Δ) such that all formulae in Δ are true at $\underline{v}$ from V. (Interestingly, the slots of $[x]$ include all the new variables, as well as the ones in $\mathcal{L}$.) So we have what we want: a $\underline{v}$ from V which shows that Γ is $\models^*$-satisfiable, as required.

Proof of 7.13* This is nearly the same as the proof of **3.13**.
List all the formulae of the language as $F_0, F_1, F_2, \cdots$.
Inductively define sets of formulae, $\Delta_0, \Delta_1, \Delta_2, \cdots$ as follows :

$$\Delta_0 := \Lambda \quad \text{and, given } \Delta_n, \quad \text{let} \quad \Delta_{n+1} := \begin{cases} \Delta_n \cup \{F_n\} & \text{if} \quad \Delta_n \vdash^* F_n \ ; \\ \Delta_n \cup \{\neg F_n\} & \text{if} \quad \Delta_n \nvdash^* F_n \ . \end{cases}$$

It is immediate from Lemma **7.12*** in Section 7.6 that, if Δ_n is $\vdash^*$-consistent, then Δ_{n+1} is also. Since $\Delta_0 = \Lambda$ is assumed $\vdash^*$-consistent, it follows that all the Δ_n are $\vdash^*$-consistent.

Now let Δ be the union of all the Δ_n. Any finite subset of Δ is certainly a subset of Δ_n for some n, since the Δ_n form an increasing sequence of sets. Thus, if there were derivations of both some F, and its negation $\neg F$, from Δ, these derivations would contain only finitely many formulae, so we'd get derivations of both F and $\neg F$ from some Δ_n. This is impossible by the $\vdash^*$-consistency of Δ_n. So Δ is $\vdash^*$-consistent, as required.

The set Δ certainly contains Λ, so it suffices now to show that it contains (and so $\vdash^*$'s) at least one of F or $\neg F$, for each F. But F occurs in the list, say $F = F_n$. From the definition of Δ_{n+1}, either F_n or $\neg F_n$ is in Δ_{n+1}, and therefore in Δ. This completes the proof.

Proof of 7.13 We would like to do the same thing as above. But first of all, Lemma **7.12** has an extra hypothesis about some formula being a sentence. And secondly, maximal $\vdash$-consistency is defined slightly more subtly than maximal $\vdash^*$-consistency.
So here we list all the **sentences** $F_0, F_1, F_2, \cdots$.
Inductively define sets of formulae, $\Delta_0, \Delta_1, \Delta_2, \cdots$ as follows :

$$\Delta_0 := \Lambda \quad \text{and, given } \Delta_n, \quad \text{let} \quad \Delta_{n+1} := \begin{cases} \Delta_n \cup \{F_n\} & \text{if} \quad \Delta_n \vdash F_n \ ; \\ \Delta_n \cup \{\neg F_n\} & \text{if} \quad \Delta_n \nvdash F_n \ . \end{cases}$$

Taking $\Delta := \cup_{n=0}^{\infty} \Delta_n$. As above, either $\Delta \vdash F_k$ or $\Delta \vdash \neg F_k$ for all k, so either $\Delta \vdash F$ or $\Delta \vdash \neg F$ for all **sentences** F, as the definition of maximal $\vdash$-consistency requires.

We must also show that the set Δ is consistent. This proof goes exactly as above, quoting **7.12**, since, in the construction of Δ_n, a **sentence** was added to Δ_{n-1}, so the extra hypothesis in **7.12** (as opposed to **7.12***) is taken care of.

First Part of the Proof of Theorem H (and of H*).

Begin by dredging up from somewhere :

in the non-$*$-case, the list $b_1, b_2, \cdots$, of new constant symbols;

in the $*$-case, the list $y_1, y_2, \cdots$, of new variables;

with the new symbols distinct from all symbols in $\mathcal{L}$.

We are going to add formulae

$$\neg\forall x F \ \rightarrow \ \neg F^{[x\rightarrow z]}$$

to Γ in a systematic way, for various $z = b_i$ (or y_i) . The main problem is to show that the resulting set remains consistent.

But right from the start we should check that Γ is still consistent after the language has been expanded. The point is that, with new symbols b_i (or y_i), there are lots of new formulae, including new logical axioms, and so, in a sense, new 'methods of proof'. Could these lead to a contradiction when the old ones on their own do not? The answer is no : Suppose given a derivation, consisting of formulae from $\mathcal{L}_+$, and showing that $\Gamma \vdash H \wedge \neg H$ (or $\Gamma \vdash^* H \wedge \neg H$) for some H from $\mathcal{L}_+$. For each i, pick an old variable $x_{j(i)}$ (all different for different i and none occurring in the derivation). Each time b_i (or y_i) appears in the derivation, replace it with $x_{j(i)}$. After this replacement is done, you have a derivation *within the language* $\mathcal{L}$ showing $\Gamma \vdash J \wedge \neg J$ (or $\Gamma \vdash^* J \wedge \neg J$) for some J from $\mathcal{L}$. This contradicts the consistency of Γ regarded as a set of formulae from $\mathcal{L}$. Therefore Γ is consistent, regarded as a set of formulae from $\mathcal{L}_+$.

Now we need two lemmas, after which this proof will be completed.

Lemma A. *Assume that, from some* $1^{\underline{\text{st}}}$*order language,* Ω *is a set of formulae,* F *is a formula, and* c *is a constant symbol not occurring in* $\Omega \cup \{F\}$*. If* $\Omega \vdash F^{[x\rightarrow c]}$*, then* $\Omega \vdash \forall x F$ *.*

Lemma A*. *Assume that, from some* $1^{\underline{\text{st}}}$*order language,* Ω *is a set of formulae,* F *is a formula, and* y *is a variable not occurring in* $\Omega \cup \{F\}$*. If* $\Omega \vdash^* F^{[x\rightarrow y]}$*, then* $\Omega \vdash^* \forall x F$ *.*

Proof of A. Pick a derivation showing $\Omega \vdash F^{[x\rightarrow c]}$, and then choose a variable y not occurring in it. Everywhere in that derivation and its justifications, replace c by y. Then we get a derivation from Ω of $F^{[x\rightarrow y]}$. (Note that c doesn't occur in Ω, so no premiss gets changed when we do the replacements in the justifications.) By (GEN), $\Omega \vdash \forall y F^{[x\rightarrow y]}$. Now

$$\forall y F^{[x\rightarrow y]} \ \rightarrow \ (F^{[x\rightarrow y]})^{[y\rightarrow x]}$$

is an axiom, and $(F^{[x\rightarrow y]})^{[y\rightarrow x]} \ = \ F$. Thus $\Omega \vdash F$, and so, by (GEN), $\Omega \vdash \forall x F$.

Proof of A*. Start with a derivation showing $\Omega \vdash^* F^{[x\rightarrow y]}$. Since y doesn't appear at all in Ω, we may apply (GEN*) to give $\Omega \vdash^* \forall y F^{[x\rightarrow y]}$. The display also labelled (%)

in the proof of **7.0*** at the end of Section 7.5 gives

(%) $$\{\forall y G\} \vdash^* \forall x(G^{[y\to x]}) ,$$

where G is any formula in which the variable x has no free occurrences. So we get $\Omega \vdash^* \forall x F^{[x\to y][y\to x]}$. But $F^{[x\to y][y\to x]} = F$, so that does it.

Lemma B. *Assume that G is a sentence and $\Omega \cup \{F\}$ contains no occurrences of the constant symbol b, where $G = (\neg\forall x F \to \neg F^{[x\to b]})$. Then*

$$\Omega \text{ is } \vdash\text{-consistent} \quad \Longrightarrow \quad \Omega \cup \{G\} \text{ is } \vdash\text{-consistent} .$$

Lemma B*. *Assume that $\Omega \cup \{F\}$ contains no occurrences of the variable y, where $G = (\neg\forall x F \to \neg F^{[x\to y]})$. Then*

$$\Omega \text{ is } \vdash^*\text{-consistent} \quad \Longrightarrow \quad \Omega \cup \{G\} \text{ is } \vdash^*\text{-consistent} .$$

Proof of B. If $\Omega \cup \{G\}$ were inconsistent , then $\Omega \cup \{\neg\neg G\}$ would be inconsistent. So, by **7.12** in Section 7.6 (applicable because G is a sentence),

$$\Omega \vdash \neg G = \neg(\neg\forall x F \to \neg F^{[x\to b]}) .$$

Now $\neg(H \to J) = \neg\neg(H \wedge \neg J)$, so

$$\Omega \vdash \neg(H \to J) \implies \Omega \vdash H \text{ and } \Omega \vdash \neg J .$$

So we get $\Omega \vdash \neg\forall x F$ and $\Omega \vdash \neg\neg F^{[x\to b]}$. An application of the rule (NEG) gives $\Omega \vdash F^{[x\to b]}$. Now $\Omega \cup \{F\}$ has no occurrence of b, so Lemma **A** applies, yielding $\Omega \vdash \forall x F$. Combined with $\Omega \vdash \neg\forall x F$, we have direct evidence that Ω in inconsistent, as required.

Proof of B*. This is almost exactly as above—b is replaced by y, and **A*** is applied, and **7.12*** is used, so G needn't be a sentence.

Completion of the proof of Theorem H (and of H*).

Let $\mathcal{P}_1, \mathcal{P}_2, \cdots$ be a list of pairs (F, x), each pair consisting of a formula from, and a variable in, the expanded language $\mathcal{L}_+$.

(In the $*$-case, use all such pairs.)

In the non-$*$- case, use all the pairs (F, x) for which $\forall x F$ is a sentence (and therefore so is $F^{[x\to b]}$ for any constant symbol b). Inductively on n, define an increasing sequence of sets, Λ_n, as follows:

$$\Lambda_0 := \Gamma , \text{ and } \Lambda_{n+1} := \Lambda_n \cup \{ \neg\forall x F \to \neg F^{[x\to z_i]} \} ,$$

where :
(F, x) is the pair $\mathcal{P}_n$;
$z_i = b_i$ in the non-$*$-case (resp. $z_i = y_i$ in the $*$-case); and
i (depending on n) is chosen as the smallest subscript for which z_i does not occur in any formula in $\Lambda_n \cup \{F\}$.

(In the $*$-case, we never run out of y_i's because Γ has no y_i's. Note however that the pairs (F, x) include $x = y_j$ as well as x_j.)

Let $\Lambda := \cup_{n=0}^{\infty} \Lambda_n$. Then certainly $\Gamma \subset \Lambda$, and the set Λ is Henkinian (or $*$-Henkinian), by definition, since for every pair $\mathcal{P} = (F, x)$, it contains $(\neg\forall x F \rightarrow \neg F^{[x \rightarrow z]})$ for some appropriate z. It remains only to prove that Λ is consistent. Because of the finiteness of proofs, it suffices to prove by induction that all Λ_n are consistent. We started the induction in the first half of this proof of **H** (and $\mathbf{H}^*$) by showing that $\Lambda_0 = \Gamma$ is consistent. For the inductive step, in Lemma **B** (or $\mathbf{B}^*$) take $\Omega = \Lambda_n$ and $G = (\neg\forall x F \rightarrow \neg F^{[x \rightarrow z_i]})$ to see that

$$\Lambda_n \text{ consistent} \quad \Longrightarrow \quad \Lambda_{n+1} \text{ consistent} .$$

This completes the proof of **Theorem H** (and $\mathbf{H}^*$), and thereby of Henkin's Theorem **7.15** (**and 7.15***), and thereby of Gödel's Completeness Theorem (finally!)

You may be wondering about a couple of related things. First of all, if we are claiming that the proof systems used here are slightly different than those elsewhere, why do we call these Gödel's theorem? And in any case, what is the 'real core' of Gödel's Completeness Theorem, independent of the particular choice, from many possibilities, of an acceptable proof system for 1$^{\underline{\text{st}}}$order logic? Please re-read for a second time the four penultimate paragraphs of Chapter 3, mainly concerning what is within the realm of acceptability for a proof system. In that context, the theorem can be stated as follows, independent of any particular choice.

Gödel's Completeness Theorem. *There do exist choices of* complete *proof systems for* 1$^{\underline{\text{st}}}$*order logic such that the set of* axioms *and the set of* rules of inference *are both* decidable sets.

This is a highly non-trivial theorem apparently, but it would become

trivial without the phrase concerning decidability, or if "1st-*order logic*" were replaced by "*propositional logic*". Strictly speaking, we haven't given a precise definition of decidability, since we haven't defined *algorithm*. But that's crucial mainly when it comes to proving *non*-decidability, or more generally, *non*-existence of an algorithm. Here it is very easy to become convinced that one can program a computer (at least an ideal one with no memory limitations) to recognize whether or not a string of symbols is indeed one of the axioms in our proof system (and similarly for the rules of inference of the system).

A slightly different way of phrasing the theorem, which takes account of types of proof system which aren't exactly in the 'axioms—rules' style is this:

There is an algorithm which lists all and only the logically valid consequences of its input, where the input is a decidable set of formulae.

Exercises

7.10 Combining Gödel's completeness theorem with **6.14** (the semantic deduction theorem) appears to give a very easy proof of **7.9** (Herbrand's deduction lemma). Explain why this is nonsense, at least in the context of this book.

7.11 This 11-part exercise is for showing that the five equality axiom schemes (in the system or system*) can be replaced by the following two, each of which is essentially a special case of one of those five.

(A) *The following is a single axiom:* $x_1 \approx x_1$.

(B) *The following is a family of axioms, one for each pair* (F, x) *consisting of an atomic formula* F *and a variable* x *:*

$$F \wedge\ x \approx y \ \longrightarrow\ F^{[x \to y]} ,$$

where $y = x_i$ *for the smallest* i *such that* $x_i \neq x$ *and* x_i *does not occur in* F*.* (If x doesn't occur in F then this is not a new axiom; it's one of the tautology axioms.)

Make that replacement to produce a new, leaner proof system. Now redefine the symbol $\vdash$ to assert the existence of a derivation in this new

system. Everything below concerns premiss-free derivations, so it applies to either of our systems. Thus, an appeal to (GEN) below can be regarded as an appeal to (GEN*), if you prefer to regard yourself as studying system*.

The main objective will be to show $\vdash G$ for each of the 'old axioms' G from the five original families of equality axioms. We get a few other results as well, related to substituting for some but not all occurrences.

For efficiency of communication, here are three labels for conditions on a fixed pair (F, x), where now F is not necessarily atomic.

(I)$_{(F,x)}$: For some variable y not occurring in F, with $y \neq x$, we have

$$\vdash \quad F \wedge \ x \approx y \ \longrightarrow \ F^{[x \to y]} \ .$$

(II)$_{(F,x)}$: For all terms t substitutable for x in F, we have

$$\vdash \quad F \wedge \ x \approx t \ \longrightarrow \ F^{[x \to t]} \ .$$

(III)$_{(F,x)}$: For all terms t and s, both substitutable for x in F, we have

$$\vdash \quad s \approx t \ \longrightarrow \ (F^{[x \to s]} \to F^{[x \to t]}) \ .$$

(i) Show that, for all terms t, we have $\vdash t \approx t$, perhaps with a 4-line derivation using an axiom, generalization, an old axiom from the family (ii), and (MP).

(ii) Show that (I)$_{(F,x)}$ implies (II)$_{(F,x)}$, again using generalization, an old axiom from the family (ii), and (MP).

(iii) Show that (II)$_{(F,x)}$ implies (III)$_{(F,x)}$.

Hint: First prove it for t not involving x by a similar method to the last two, also noting that there are 'purely propositional' derivation fragments to show

$$\vdash \quad G \wedge H \to K \quad \Longleftrightarrow \quad \vdash \quad G \to (H \to K) \ .$$

See **7.6**. When t does involve x, take the formula just established, but with some variable z rather than t, where $z \neq x$ and z does not occur in F nor s nor t. Now again apply the method in (i) and (ii), this time substituting the given t for z.

(iv) Show that, for all atomic F and all x, we have

$(\mathrm{IV})_{(F\ ,\ x\ ,\ \text{some free occurrences of } x)}$: for all terms s and t, both substitutable for x in F,

$$\vdash \quad s \approx t \ \longrightarrow \ (F^{[\text{some}x \to s]} \to F^{[\text{same}x \to t]}) \ .$$

Explanation and hint: Informally the formula $F^{[\text{some}x \to s]}$ is obtained by singling out some, but not necessarily all, free occurrences of x in F, and replacing them by s. If G is the formula obtained by replacing them instead by some variable v which has no occurrences in F, then

$$F = G^{[v \to x]} \ \text{ and } \ F^{[\text{some}x \to s]} := G^{[v \to s]} \ \text{ and } \ F^{[\text{same}x \to t]} := G^{[v \to t]} \ .$$

The last two are the more formal definitions. Now consider $(\mathrm{III})_{(G,v)}$. Note that atomicity and axiom (B) allow (ii) and (iii) to be applied.

(v) Show that $\vdash \quad s \approx t \ \to t \approx s$.

Hint: In $(\mathrm{IV})_{(F\ ,\ x\ ,\ \text{some free occurrences of } x)}$, try F as $x \approx s$ with "some x" as the first, perhaps but not necessarily the only, occurrence of x. Then use 'purely propositional' arguments as in the display in (iii) just above, plus $\vdash \ s \approx s$ from (i).

(vi) Show that $\vdash \quad s \approx t \ \wedge \ t \approx q \ \longrightarrow \ s \approx q$ for all terms s, t and q.
Hint: Start from the following case of (IV):

$$t \approx q \ \longrightarrow \ (\ (s \approx x)^{[\text{visible}x \to t]} \to (s \approx x)^{[\text{visible}x \to q]} \ ) \ .$$

The rest is 'purely propositional'.

(vii) Show that

$$\vdash \quad s \approx t \ \to \ AsB \approx AtB \ .$$

for all terms s and t, where A and B are strings of symbols as in the last of the five old, original equality axioms.

Hint: In $(\mathrm{IV})_{(F\ ,\ x\ ,\ \text{some free occurrences of } x)}$, take F to be $AsB \approx AxB$ with "some x" as the only visible one. (There might be some hidden in s, A or B.) Perhaps use also that (i) gives $\vdash \ AsB \approx AsB$.

At this point we have deduced $\vdash G$ for four of the five original 'old equality axioms'—see (i), (v), (vi) and (vii). The fifth one is $(\mathrm{III})_{(F,x)}$ for all F and x. Combining axiom (B) with (ii) and (iii), we have this when F is atomic. [Alternatively, take "some" to be "all" in (iv).] We could just change (B) by dropping the word "atomic", and be done. Something like that form of the equality axioms could be used as the best compromise between minimal

assumptions and maximal pedagogical virtuosity. But the final four parts of this exercise allow us to go from atomic to all F by induction. That's worth recording, since we completely ignored minimal assumptions originally. We simply wrote down as equality axioms exactly what was needed to verify that the autological interpretation was one. Once finished we'll also have (IV) for non-atomic F, as well. That's because we got (IV) directly from (III).

(viii) Show that $(\mathrm{III})_{(\neg F,x)}$ follows from assuming $(\mathrm{III})_{(F^{[x\to y]},y)}$ for some $y \neq x$ which does not occur in F.

Hint: To get $(\mathrm{I})_{(\neg F,x)}$, apply the assumption for $s = y$ and $t = x$, and also apply (v) with $s = x$ and $t = y$, plus 'purely propositional' considerations; then use (ii) and (iii) to get from (I) to (III) .

(ix) Show that $(\mathrm{III})_{(F,x)}$ and $(\mathrm{III})_{(G,x)}$ together imply $(\mathrm{III})_{(F\wedge G,x)}$.

(x) Show that $(\mathrm{III})_{(F,x)}$ implies $(\mathrm{III})_{(\forall z F,x)}$ for any z.

Hint: To get $(\mathrm{I})_{(\forall z F,x)}$, apply the assumption for $s = y$ and $t = x$ together with the fact that

$$\{H \to K\} \ \vdash \ \forall z H \to \forall z K \ ,$$

plus 'purely propositional' considerations; then use (ii) and (iii) to get from (I) to (III).

(xi) Finally, starting from axiom (B) and (ii) and (iii), deduce from these last three by induction on the length of F that $(\mathrm{III})_{(F,x)}$ holds for all F and x, giving our remaining original 'old equality axiom'.

Hilbert was keen on *independent* sets of axioms. Using our new pair, (A) and (B), we now have seven axiom schemes: three tautology, one substitution, one quantifier and two equality. This is an independent set of *schemes*, in the sense that no one of the seven can be dropped without losing completeness.

The infinite set of *individual* axioms is not independent, and that question is not particularly interesting. For example, one can replace (B) by this subscheme:

(B$_-$) For all atomic F in which x_1 occurs but x_2 doesn't, take as an axiom the following:

$$F \wedge \ x_1 \approx x_2 \ \longrightarrow \ F^{[x_1 \mapsto x_2]} \ .$$

The proof is another application of the substitution axiom scheme, (ii), but let's not get carried away.

7.12 The Compactness and Löwenheim-Skolem Theorems.

In this section, it is assumed that the reader knows something about the cardinality of sets. We'll drop the distinction between the "$*$-case" and the other. Everything below applies to both.

Compactness Theorem for $1^{\underline{\text{st}}}$order logic. *Let Γ be a set of formulae from a $1^{\underline{\text{st}}}$order language* (possibly even with more than countably many symbols, though we've been assuming $\mathcal{L}$ is at most countable)*. Then Γ is satisfiable if and only if every finite subset of Γ is satisfiable.*

Proof. One way is trivial, so assume that every finite subset is satisfiable, and therefore consistent. By the finiteness of proofs, Γ itself is consistent. (The premisses used in a supposed proof of $F \wedge \neg F$ form a finite subset.) Therefore, by Henkin's theorem (possibly generalized to larger languages), Γ itself is satisfiable.

Löwenheim-Skolem Theorem. *If a set of formulae from a $1^{\underline{\text{st}}}$order language* (with countably many symbols, as we've been assuming all along) *has a model, then it has a model which is countable or finite.*

Proof. If Γ has a model, then it is consistent, and therefore has a model V_Δ or V^*_Δ as in the proof of Henkin's theorem. But those models, as sets, consist of equivalence classes from a countable set (some terms from the language), and are therefore countable at most in size.

This last theorem is thought by some to lead to a bit of a paradox, when you apply it to the language of $1^{\underline{\text{st}}}$order set theory. But we have been avoiding that language like the plague for the last two chapters !

Exercises

7.12 Use the compactness theorem to prove the claim in Section **5.2** that there is no translation into the $1^{\underline{\text{st}}}$order directed graph theory language of the statement *"there is a path from x to y"*. Hint: Consider the infinite set consisting of that statement together with the negations of all the statements for all n that there is a path of length n from x to y.

7.13 (a) Use the compactness theorem to prove the claim in Section **5.7** that there is no translation into the $1^{\underline{\text{st}}}$order group theory language of the statement *"x has a power which is the identity element"* or, in group theory jargon, *"x has finite order"*.

(b) Give an infinite set of formulae which state that x is an nth power for every n. Can you see why no single formula (nor finite set of formulae) will do?

7.14 The system in **Enderton** is described briefly in Section 7.3. (It is a system with only (MP) as rule of inference, a substitute for system* here, since he uses the $\models^*$-semantic

convention.) Give a specific derivation in each of these two systems of each of the axioms from the other system. (Don't just argue that, since both systems are complete, such derivations necessarily exist!) You could do the same for any other reputable book dealing with $1^{\underline{\text{st}}}$order logic, first deciding which of our two systems is the appropriate one (that is, which semantic convention for arguments not just involving sentences is being used, that given by $\models$, or that by $\models^*$).

7.15 This exercise takes care of one of the points needed in the Boolos' proof of Gödel's incompleteness theorem, given at the end of Chapter 5. The following apply to any formulae F, G, H and any terms s, t in any $1^{\underline{\text{st}}}$order language.

(i) Show that if $\Gamma \vdash \forall x(F \leftrightarrow G)$ and $\Gamma \vdash \forall x(F \leftrightarrow H)$, then $\Gamma \vdash \forall x(G \leftrightarrow H)$.

(ii) Show that if $\Gamma \vdash \forall x(x \approx s \leftrightarrow x \approx t)$. then $\Gamma \vdash s \approx t$.

(It is more elegant *not* to use completeness!)

7.16 Prove directly from the deduction lemma, not using completeness, the following form of proof by cases: if $\Gamma \cup \{G\} \vdash^* F$ and $\Gamma \cup \{\neg G\} \vdash^* F$, then $\Gamma \vdash^* F$. Do the same for $\vdash$, this time assuming that G is a sentence.

8. FIRST-ORDER CALCULATIONS AND ALGORITHMS

The basic techniques to be learned here are presented in Sections **8.2, 8.3** and **8.4**. The motivation for these is given in the first and the last two sections, where we discuss general results about trying to make Leibniz' dream come alive. That is, can one find an automatic, 'machine-like' method for deciding whether an argument in 1st order logic is valid?

8.1 Deciding about validity of 1st order arguments

The first question is whether there is a good algorithm, or even any algorithm at all, for checking validity of arguments in 1st order logic. The answer is very negative. For most interesting 1st order languages and sets of premisses, there can be no algorithm at all—not just no known algorithm, but actually a proof that no algorithm can exist. Proofs of results like this depend on defining precisely what is meant by the word "algorithm", and so are beyond the scope of this book. (See however Church's theorem, **L5**, in **Appendix L**.) So we're saying that, for many languages and premiss sets Γ, there is no Type D algorithm, whose input is any formula, and whose output is a decision as to whether the input is a valid consequence of Γ.

This is a good occasion for comparing and relating the two somewhat vague notions from Chapter 4 of "Type D" versus "Type L" algorithm. Suppose as above that we have a fixed 1st order language, and a set Γ of premisses, which might as well be sentences, though that isn't crucial. Assume that Γ comes also equipped with a Type D algorithm which checks whether a given formula is **in** Γ—not, as above, whether it's **derivable from** Γ. So the input is an arbitrary formula from the language, and the output is "yes" or "no" according to whether the input is or is not a member of Γ. The existence of such an algorithm is definitely a restriction on the premiss set Γ, because, just as with propositional logic, 'most' sets of formulae do *not* admit such an algorithm (though it's more-or-less impossible to write down a *specific* set which doesn't). A set of premisses which wasn't 'decidable' in this sense would be of dubious interest as a premiss set. Certainly any *finite* set is decidable.

In contrast to the lack of Type D algorithms for checking validity of arguments, for a decidable Γ as above, it is quite easy to believe that there is

a Type L algorithm that **lists** all the formulae which are valid consequences of Γ. That is, there is a procedure which spits out, one at a time, all the formulae in the set $\{ F : \Gamma \vdash F \}$. (We assumed that Γ consists of sentences so that above it made no difference whether $\vdash$ or $\vdash^*$ was used. And by completeness, you could change them to $\models$ or $\models^*$, if preferred, for those who skipped Chapter 7.)

Roughly speaking, to produce the listing algorithm, first make use of a Type L algorithm which spits out all finite sequences of formulae from the language. (See the last paragraph of **Appendix B** to this chapter.) But divert each piece of output to a Type D algorithm which checks whether that finite sequence is or is not a derivation with premisses Γ. This is where we would use the algorithm which checks whether a given formula is in Γ. It is also necessary to be aware that for this we'd use an algorithm which checked whether a given formula was a logical axiom, and also one which checked whether an 'input' formula was directly inferrable by a rule of inference from another input formula or pair of formulae. After inspecting our proof system from Chapter 7, I trust that you'll agree that such algorithms certainly exist. If the finite sequence turns out *not* to be a derivation, you do not produce any output, but go on to check the next finite sequence. If it *is* a derivation, you output its last line, and then go on to the next sequence. This algorithm will clearly list all formulae which are logical consequences of Γ, as required.

(Such formulae will appear more than once in the list. So, if you don't like that, you can insert a procedure which prevents any output from being repeated. It is noteworthy that, without the completeness theorem and 'proof theory' of Chapter 7, although the question can still be asked, using $\models$ rather than $\vdash$, the existence of the listing algorithm would be much harder, if not impossible, to see.)

This is quite interesting, when contrasted with the claim further up that often there is no Type D algorithm for deciding whether a given formula appears in the list which the Type L algorithm just above would produce. By simply inspecting the list, you could give a 'Type semi-D algorithm' which outputs "yes" when that's the answer, but won't halt ['infinite loop'] if the input is not a logical consequence of Γ. In a case like this, there will be no Type L algorithm for listing $\{ F : \Gamma \nvdash F \}$, since otherwise the two listing algorithms could be inspected, producing a 'deciding' algorithm.

One case (of this failure of decidability for logical consequence) is when

the language is that of 1st order number theory. The theorem is due to Church. The first page of **Appendix L** has a list of nine simple 1st order sentences which are obviously true in **N**. (There we use a slightly different language for number theory. The constant symbol 1 is not used. But a 1-ary function symbol s is used, which in the interpretation **N** is interpreted as the successor function $k \mapsto k + 1$. But either language could be used.)

Church's theorem says that for no set Γ, whose union with these nine sentences is consistent, is there such a Type D algorithm, i.e. for no such set is the set of logical consequences of Γ decidable. In particular this applies both to the empty set and to the nine-sentence set itself. See **Appendix L** and especially its **Addendum** for more details. Of course, if such an algorithm actually had turned out to exist for some nice little set Γ of 'non-logical axioms for number theory', then in principle you would have been able to use it to decide such things as the Goldbach and twin prime conjectures.

Note that the set above of logical consequences of $\emptyset$ is just another name for the set of logically valid formulae in the 1st order language of number theory. To the extent that you regard this set as *specific*, maybe Church's theorem contadicts my statement above about not being able to "write down" a specific undecidable set.

Note also that this **non-existence of a derivability/satisfiability algorithm** is a feature which **distinguishes 1st order logic from propositional logic**. In the latter logic, the simplest (but very inefficient) algorithm is inspection of truth tables. Why can't we just inspect interpretations to get an algorithm for any given 1st order language? The answer is that many interpretations are not finite, and, in any case, many languages have infinitely many essentially different interpretations. For a given propositional formula, we need only look at finitely many truth assignments—"2^n", where n is the number of propositional variables in the formula.

There is much which could be added here, but that must be left to **Appendix L** and to your further study of logic elsewhere. Despite the general lack of an algorithm for deciding derivability or satisfiability (the connection between these two is the same as in propositional logic), the next several sections will be devoted to calculational methods which help in many particular cases to decide these things, and which are parts of systems for automated theorem-proving in 1st order. These systems are of course only partially successful at best. This general discussion is continued in the last two sections

of the present chapter.

8.2 Equivalence of $1^{\underline{\text{st}}}$order Formulae

For the rest of the book, except when mentioned explicitly, we won't distinguish between what is a formula and what is really an abbreviated formula. It is important to be comfortable doing the kinds of calculations considered below with the symbols $\exists, \vee, \rightarrow$ and $\leftrightarrow$, as well as the symbols which are 'officially' part of the language.

Fix a $1^{\underline{\text{st}}}$order language, and let F and G be formulae from it.

Definition. We say that F and G are truth equivalent if and only if, *for every interpretation V of the language, and every $\underline{v}$ from V, F is true at $\underline{v}$ if and only if G is true at $\underline{v}$.*

So this says simply that $F^V = G^V$, making it obvious, if it isn't already, that we've defined an equivalence relation on the set of all formulae. Informally, it says that *F and G define the same relation on the variables that occur freely in them*—a bit more precisely, *... the same relation, in every interpretation, on the 'slots' corresponding to those variables*. If there aren't any such variables, it says that F and G are sentences which are true in exactly the same interpretations.

Remark. This definition is very much the analogue of the one with the same name in propositional logic. It seems to be more in the spirit of our '$*$-ed' definitions from the last section of Chapter 6 than of our 'un-$*$-ed' definitions. The exercises 8.3 and 8.4 below relate to this.

Examples. For any $1^{\underline{\text{st}}}$order formulae F and G, all three of

$$F \rightarrow \neg G \quad , \quad G \rightarrow \neg F \quad \text{and} \quad \neg(F \wedge G)$$

are truth equivalent, basically because, in propositional logic

$$P \rightarrow \neg Q \quad \text{treq} \quad Q \rightarrow \neg P \quad \text{treq} \quad \neg(P \wedge Q) \ .$$

See Exercise 8.6 for the general justification of this.

A closely related specific example of a pair of equivalent sentences in the language of $1^{\underline{\text{st}}}$order number theory is

$$\forall x \forall y \ (x \approx y \ \rightarrow \ \neg x < y) \quad \text{and} \quad \forall u \forall v \ (u < v \ \rightarrow \ \neg u \approx v) \ .$$

It seems very clear that these two sentences say the same thing; and so they are equivalent.

A more formal proof of equivalence in these examples can be given, using the definitions in Chapter 6, but that seems hardly necessary. However, the following example is more subtle.

If G is FOFOO x_3, then $(\forall x_3 F) \rightarrow G$ and $\exists x_3 (F \rightarrow G)$ are equivalent.

This is one of a list of 'rules for moving quantifiers' below, where it has an indirect proof. Here is the direct proof, using the Chapter 6 definitions.

$$\underline{v} \in ((\forall x_3 F) \rightarrow G)^V \iff \underline{v} \in G^V \quad \text{or} \quad \underline{v} \notin (\forall x_3 F)^V$$

$$\iff \underline{v} \in G^V \quad \text{or} \quad \text{for some } u, \ (v_1, v_2, u, \cdots) \notin F^V \ .$$

But,

$$\underline{v} \in (\exists x_3 (F \rightarrow G))^V \iff \text{for some } u, \ (v_1, v_2, u, \cdots) \in (F \rightarrow G)^V$$

$$\iff \text{for some } u, \ (v_1, v_2, u, \cdots) \in G^V \quad \text{or} \quad \text{for some } u, \ (v_1, v_2, u, \cdots) \notin F^V \ .$$

But now use the fact that G is FOFOO x_3 and **6.1** to see that the lower lines in the two displays amount to the same thing. Thus the sets $((\forall x_3 F) \rightarrow G)^V$ and $(\exists x_3 (F \rightarrow G))^V$ are the same, as required to show that the two formulae are truth equivalent.

On the other hand, the formulae $(\forall x F) \rightarrow G$ and $\forall x (F \rightarrow G)$ are in general NOT equivalent, even when G is FOFOO x. For example, using the language of number theory, take $F \ = \ x \approx x + 1$ and $G \ = \ 0 \approx 1$. Here is an example of a rather silly little finite interpretation, which does what we need. Let V be the 2-element set $\{p, n\}$, and let $0^V = p$ and $1^V = n$. Define multiplication and the order relation $<^V$ however you please (they're clearly irrelevant here). And define addition in V by taking $p + n = n$, $n + n = n$, and letting $p + p$ and $n + p$ be anything you please (in V, of course). Then $\forall x \ x \approx x + 1$ is false in V since $p \neq p + n$. Thus

$(\forall x\ x \approx x+1) \to G$ is true in V for *any* formula G in the language. In particular, $(\forall x\ x \approx x+1) \to 0 \approx 1$ is true, but not for the obvious reason. On the other hand, the sentence $\forall x(x \approx x+1 \to\ 0 \approx 1)$ is false in V by taking the slot for the variable x to contain n. To explain this, note that $x \approx x+1$ holds with x and 1 changed to n, and $\approx$ changed to $=$. But $0 \approx 1$ is false in V because $p \neq n$. So $x \approx x+1 \to\ 0 \approx 1$ is false at any $\underline{v}$ which has n in the slot for x. And so the sentence $\forall x\ (x \approx x+1\ \to\ 0 \approx 1)$ is false in V, as required.

Exercise 8.1 Check all this carefully, using the definitions of Chapter 6.

Prop 8.1. *Formulae F and G are truth equivalent if and only if the formula $F \leftrightarrow G$ is logically valid.*

Exercises

8.2 Prove **8.1**.

8.3 Show that if F and G are truth equivalent then, for every interpretation V of the language, F is true in $V \iff G$ is true in V.

8.4 Show that the converse of the previous exercise is false; specifically, find formulae F and G which are not truth equivalent, and yet , for every interpretation V of the language, F is true in $V \iff G$ is true in V. (You might want to look at the next exercise, and then at some examples in Chapter 6. The converse does hold when restricted to sentences, of course.)

8.5 Show that $F \wedge G$ is logically valid if and only if both F and G are logically valid. Deduce that F and G are truth equivalent if and only if both $F \to G$ and $G \to F$ are logically valid.

8.6 Let H and J be formulae in propositional logic. Recall the definition of the $1\underline{\text{st}}$order formula H^* in Section **6.6**—roughly, H^* is obtained by substituting a $1\underline{\text{st}}$order formula for each propositional variable in H. Prove that, if $H \overset{\text{treq}}{} J$ in the sense of propositional logic, then H^* and J^* are truth equivalent in the sense of $1\underline{\text{st}}$order logic. (Use **8.1** and **6.8**.)

From now on we'll also use the notation $F \overset{\text{treq}}{} G$ to denote truth equivalence in $1\underline{\text{st}}$order logic.

The last exercise gives us plenty of examples of truth equivalence, but not very interesting ones at this stage of our knowledge. Much more interesting are the following **basic quantifier truth equivalences**.

Prop 8.2. *The following truth equivalences hold, for any formulae F and G and variable x :*

(i) $\neg\forall xF \quad {}^{\text{treq}} \quad \exists x\neg F$;

(ii) $\neg\exists xF \quad {}^{\text{treq}} \quad \forall x\neg F$;

(iii) $\forall x(F \wedge G) \quad {}^{\text{treq}} \quad (\forall xF) \wedge (\forall xG)$.

(iv) *If either F or G is FOFOO x, then* $\exists x(F \wedge G) \quad {}^{\text{treq}} \quad (\exists xF) \wedge (\exists xG)$.

Exercise 8.7 Show that the FOFOO condition in (iv) cannot be removed.

Proof. Thinking about what these formulae mean, it is pretty clear that, in each case, they do mean the same thing (though without the FOFOO assumption in (iv) they wouldn't, as an example on the 3rd page of Chap. 5 shows). This is probably adequate as a proof for most of us.

However, we'll give direct proofs below, using the basic definitions from Chapter 6. (A more elegant deduction of (i) and (ii) is perhaps to do exercises 8.9, 8.10 and 8.12 below directly, then proceed as suggested in 8.14.)

Take $x = x_3$; for other variables, the proof is 'the same'.

(i) We have $\underline{v} \in (\neg\forall x_3F)^V \iff \underline{v} \notin (\forall x_3F)^V$

$$\iff \text{for some } u,\ (v_1, v_2, u, \cdots) \notin F^V \ .$$

But, $\underline{v} \in (\exists x_3\neg F)^V \iff$ for some u, $(v_1, v_2, u, \cdots) \in (\neg F)^V$

$$\iff \text{for some } u,\ (v_1, v_2, u, \cdots) \notin F^V \ .$$

Thus $(\exists x_3\neg F)^V$ and $(\neg\forall x_3F)^V$ are the same set, as required.

(ii) is proved in a very similar manner.

(iii) We have

$$\underline{v} \in (\forall x_3(F \wedge G))^V \iff \text{for all } u,\ (v_1, v_2, u, \cdots) \in (F \wedge G)^V$$

$$\iff \text{for all } u,\ (v_1, v_2, u, \cdots) \in F^V \cap G^V \ .$$

But,

$$\underline{v} \in ((\forall x_3 F) \wedge (\forall x_3 G))^V \iff \underline{v} \in (\forall x_3 F)^V \text{ and } \underline{v} \in (\forall x_3 G)^V$$

$$\iff \text{for all } u,\ (v_1, v_2, u, \cdots) \in F^V \text{ and for all } w,\ (v_1, v_2, w, \cdots) \in G^V$$

$$\iff \text{for all } u,\ (v_1, v_2, u, \cdots) \in F^V \cap G^V\ ,$$

as above, and as required.

(iv) We have

$$\underline{v} \in (\exists x_3 (F \wedge G))^V \iff \text{for some } u,\ (v_1, v_2, u, \cdots) \in (F \wedge G)^V$$

$$\iff \text{for some } u,\ [\,(v_1, v_2, u, \cdots) \in F^V \text{ and } (v_1, v_2, u, \cdots) \in G^V\,]\,.$$

But,

$$\underline{v} \in ((\exists x_3 F) \wedge (\exists x_3 G))^V \iff \underline{v} \in (\exists x_3 F)^V \text{ and } \underline{v} \in (\exists x_3 G)^V$$

$$\iff \text{for some } w,\ (v_1, v_2, w, \cdots) \in F^V \text{ and for some } z,\ (v_1, v_2, z, \cdots) \in G^V\ .$$

But, using **6.1**, in the case that F is FOFOO x_3, both bottom lines become

$$\underline{v} \in F^V \text{ and } \text{for some } u,\ (v_1, v_2, u, \cdots) \in G^V\ .$$

Just reverse the roles of F and G for the other case.

Exercises

8.8 Prove that, if H is FOFOO x, then $\forall x H \ \overset{\text{treq}}{\ }\ H$. Give two proofs that, if H is FOFOO x, then $\exists x H \ \overset{\text{treq}}{\ }\ H$; one proof direct from Chapter 6 results, and one using the first part plus 8.10 and 8.12 below.

8.9 Show $[F_1 \ \overset{\text{treq}}{\ }\ G_1$ and $F_2 \ \overset{\text{treq}}{\ }\ G_2]$ implies that $F_1 \wedge F_2 \ \overset{\text{treq}}{\ }\ G_1 \wedge G_2$.

8.10 Show $F \overset{\text{treq}}{\ } G \implies \neg F \ \overset{\text{treq}}{\ }\ \neg G$.

8.11 Show

(i) $F \ \overset{\text{treq}}{\ }\ G \implies \forall x F \ \overset{\text{treq}}{\ }\ \forall x G$, but

(ii) the converse is false.

(iii) State and prove the analogue for 1st order logic of **2.1/2.1***.

(It will be used without mention frequently below!)

8.12 Show that $\neg\neg F \ \overset{\text{treq}}{\ }\ F$.

8.13 Show that the converse of 8.10 is true.

8.14 Combining the definition of $\exists$ (in terms of $\forall$) with the above exercises, give short snappy elegant new proofs of (i) and (ii) in **8.2**.

Now we can get away from the mildly ugly notation needed to use the basic definitions in Chapter 6, and use instead the results in **8.2** to give short snappy elegant proofs of the **derived quantifier truth equivalences** in the following three propositions. These quantifier truth equivalences can be used to 'move all the quantifiers to the front', as explained in the next section.

Prop 8.3. (i) For any F and G, we have $\exists x(F \to G) \ {}^{\text{treq}} \ \forall xF \to \exists xG$.
(ii) If either F or G is FOFOO x, then $\forall x(F \to G) \ {}^{\text{treq}} \ \exists xF \to \forall xG$.

Proof. For (i), using **8.2** and a few exercises,

$$\exists x(F \to G) \ = \ \exists x\neg(F \wedge \neg G) \ {}^{\text{treq}} \ \neg\forall x(F \wedge \neg G)$$

$$ {}^{\text{treq}} \ \neg((\forall xF) \wedge (\forall x\neg G)) \ {}^{\text{treq}} \ \neg((\forall xF) \wedge (\neg\exists xG)) \ = \ \forall xF \to \exists xG \, .$$

The proof for (ii) is quite similar:

$$\forall x(F \to G) \ = \ \forall x\neg(F \wedge \neg G) \ {}^{\text{treq}} \ \neg\exists x(F \wedge \neg G)$$

$$ {}^{\text{treq}} \ \neg((\exists xF) \wedge (\exists x\neg G)) \ {}^{\text{treq}} \ \neg((\exists xF) \wedge (\neg\forall xG)) \ = \ \exists xF \to \forall xG \, .$$

Exercise 8.15 Point out where the "FOFOO" assumption is used in this last proof. Give an example to show that this assumption cannot be deleted.

Prop 8.4. (i) *For any F and G, $\exists x(F \vee G) \ {}^{\text{treq}} \ (\exists xF) \vee (\exists xG)$.*
(ii) *If either F or G is FOFOO x, then $\forall x(F \vee G) \ {}^{\text{treq}} \ (\forall xF) \vee (\forall xG)$.*

Exercise 8.16 Prove **8.4.**—you may also need to prove [$F_1 \ {}^{\text{treq}} \ G_1$ and $F_2 \ {}^{\text{treq}} \ G_2$] implies that $F_1 \vee F_2 \ {}^{\text{treq}} \ G_1 \vee G_2$; or use 8.11(iii) a few times.

Prop 8.5. (i) *If F is FOFOO x, then $\forall x(F \wedge G) \ {}^{\text{treq}} \ F \wedge (\forall xG)$.*
(ii) *If G is FOFOO x, then $\forall x(F \wedge G) \ {}^{\text{treq}} \ (\forall xF) \wedge G$.*

Proof. For (i), we have

$$\forall x(F \wedge G) \ {}^{\text{treq}} \ (\forall xF) \wedge (\forall xG) \ {}^{\text{treq}} \ F \wedge (\forall xG) \ ,$$

first using **8.2 (iii)**, and then using exercises 8.8 and 8.9. The proof for (ii) is almost the same.

There are analogous truth equivalences, with $\wedge$ replaced by $\vee$ or $\rightarrow$, in which one of the component formulae is assumed to be free of free occurrences of the variable. The reader can easily formulate and prove these. We shall use them in the next section.

8.3 Prenex Form

The following two sentences are two different ways of expessing in the language of $1^{\underline{\text{st}}}$order ring theory the algebraic assertion that, *if every non-zero element has a multiplicative inverse, then no two non-zero elements have product zero.* (For those who have studied abstract algebra : they each express, when combined with the axioms defining commutative rings, the theorem that *any field is an integral domain.*)

$$\forall x(\neg x \approx 0 \rightarrow \exists y\ x \times y \approx 1)\ \rightarrow\ \forall u \forall v\ (\neg u \approx 0 \wedge \neg v \approx 0\ \rightarrow\ \neg u \times v \approx 0)\quad ;$$

$$\exists x \forall y \forall u \forall v\ ((\neg x \approx 0 \rightarrow x \times y \approx 1)\ \rightarrow\ (\neg u \approx 0 \wedge \neg v \approx 0\ \rightarrow\ \neg u \times v \approx 0))\quad .$$

It's not obvious, but these two do say the same thing, so they are truth equivalent. That can be verified mechanically from our quantifier truth equivalences in the last section. See the paragraphs below the next.

Now (to me at least) the first of these is by far the more natural way to express in $1^{\underline{\text{st}}}$order this algebraic assertion. However, in this section we'll concentrate on the process of converting formulae, particularly sentences, into something like the second one above. That is, 'get all quantifiers to the front'. Although not always the most natural for understanding what is being expressed, this **prenex form** is very convenient in logic and for automated theorem proving.

The equivalence of the above two sentences is more easily checked in the following general form:

If F is FOFOO all of y, u and v; and G is FOFOO both u and v; and H is FOFOO both x and y; then

$$\forall x(F \rightarrow \exists y G)\ \rightarrow\ \forall u \forall v\ H \qquad \text{treq} \qquad \exists x \forall y \forall u \forall v\ ((F \rightarrow G) \rightarrow H)\quad .$$

The claimed truth equivalence further up then comes from taking

$$F \;=\; \neg x \approx 0 \quad ; \quad G \;=\; x \times y \approx 1 \;\; ; \quad \text{and}$$
$$H \;=\; \neg u \approx 0 \wedge \neg v \approx 0 \;\rightarrow\; \neg u \times v \approx 0 \; .$$

Before doing the calculation to prove this, here is a list of the most useful truth equivalences involved in producing the prenex form of a formula. (Recall that we're not distinguishing here between "formula" and "abbreviated formula".) The main thing you should learn from this section is to get skillful at the process of using this list to produce the prenex form of a formula.

$$\neg\forall x F \quad \text{treq} \quad \exists x \neg F$$
$$\neg\exists x F \quad \text{treq} \quad \forall x \neg F$$
$$F \wedge (\forall x G) \quad \text{treq} \quad \forall x (F \wedge G) \qquad [F \text{ FOFOO } x]$$
$$(\forall x F) \wedge G \quad \text{treq} \quad \forall x (F \wedge G) \qquad [G \text{ FOFOO } x]$$
$$F \wedge (\exists x G) \quad \text{treq} \quad \exists x (F \wedge G) \qquad [F \text{ FOFOO } x]$$
$$(\exists x F) \wedge G \quad \text{treq} \quad \exists x (F \wedge G) \qquad [G \text{ FOFOO } x]$$

Change $\wedge$ to $\vee$ in the previous four equivalences, giving four more.

$$F \rightarrow \forall x G \quad \text{treq} \quad \forall x (F \rightarrow G) \qquad [F \text{ FOFOO } x]$$
$$F \rightarrow \exists x G \quad \text{treq} \quad \exists x (F \rightarrow G) \qquad [F \text{ FOFOO } x]$$
$$(\forall x F) \rightarrow G \quad \text{treq} \quad \exists x (F \rightarrow G) \qquad [G \text{ FOFOO } x]$$
$$(\exists x F) \rightarrow G \quad \text{treq} \quad \forall x (F \rightarrow G) \qquad [G \text{ FOFOO } x]$$

Notice that all the right-hand sides have the quantifier 'at the front'. The first two and last two of these are the most interesting, due to the switch of quantifiers. The correctness of these is immediate from the results in the previous section. When dealing with "silly" formulae, you get rid of unneeded quantifiers by using the 'identity'

$$\forall x H \;\; \text{treq} \;\; H \;\; \text{treq} \;\; \exists x H \; , \qquad \text{for } H \text{ FOFOO } x \; .$$

Exercises

8.17 Explain the relationship between the 4th last equivalence in the box and logical axiom scheme (iii) in the proof system early in Chapter 7.

8.18 The last two in the box above are the only ones that seem a bit mysterious, partly because formulae in prenex form are often less easy to translate than ones where the quantifiers are placed more 'naturally'. But you can give another proof of the last two, using the preceding two and the top two, and making use of the fact that $(F \to G) \overset{\text{treq}}{} (\neg G \to \neg F)$. Do so.

Proving the correctness of the truth equivalence further up is now a typical calculation using this list :

$$(\forall x(F \to \exists y G)) \ \to \ \forall u \forall v \ H \qquad \overset{\text{treq}}{} \qquad (\forall x \exists y(F \to G)) \ \to \ \forall u \forall v \ H$$

$$\overset{\text{treq}}{} \quad \exists x \forall y((F \to G) \ \to \ \forall u \forall v \ H) \qquad \overset{\text{treq}}{} \quad \exists x \forall y \forall u \forall v((F \to G) \ \to \ H) \, .$$

The first step used one of the equivalences in the list above, and the other two each used two of them. Also Exercise 8.11(ii) and its existential analogue were used.

Exercise 8.19 Give careful details of the justifications sketched just above.

You may be wondering about whether we won't sometimes run into difficulties with the FOFOO conditions in the list above.

For example, what if the algebraic assertion at the beginning of this section had been written

$$(\forall x(\neg x \approx 0 \to \exists y \ x \times y \approx 1)) \ \to \ \forall x \forall y \ (\neg x \approx 0 \wedge \neg y \approx 0 \ \to \ \neg x \times y \approx 0) \quad ?$$

That has the same meaning as the way it was originally written using variables u and v on the 'right-hand side'. (We have been assuming all along that the four variables in the earlier version of the formula are all different from each other. In teaching ring theory, most mathematicians would probably use [a less formal statement of] the version just above, with two variables, each used twice.) In the new form, we cannot use the equivalences in the list directly in two cases, because the FOFOO conditions are not satisfied. But, up to truth equivalence, we can just switch back to the original form, and use u and v.

And that trick works perfectly well in general. In fact, **given any formula that you wish to 'put into prenex form', the first step should always be changing some or all of the bound variables to new ones.** Do it so as to have all occurrences of quantifiers binding different variables, and also all these bound variables distinct from any free variables that occur in the formula. We had discussed this renaming of bound variables in and before **6.11**. That result simply says that $F \overset{\text{treq}}{} \underline{F}$, where $\underline{F}$ is the result of renaming a bound variable in F. So renaming a whole bunch of bound variables won't change the formula up to truth equivalence, as required.

Once that is done, you just persistently push quantifiers to the front, without needing to think about the FOFOO conditions. In fact, **all you need to remember** about the previous list **is the switch in quantifier** which occurs **when pushing a negation sign past a quantifier** (the first two on the list) **and when pulling a quantifier out of the left side of a "→-subformula"** (the last two on the list).

Here is an example from the language of directed graphs to re-illustrate most of the points above. We want to find a prenex form for

$$(\forall u \exists x(\forall y\ xry\ \rightarrow\ xrx \wedge \forall x\ xru)) \vee uru\ \ .$$

(The outermost pair of brackets is not strictly necessary, according to an abbreviation convention—but it does serve to reinforce the fact that the two rightmost occurrences of u are free.) Since u occurs freely as well as bound, change the bound occurrences to v. (And now drop the outermost pair of brackets.) Since x occurs bound with two different quantifier occurrences, change the latter one to t. The resulting new formula, truth equivalent to the old, is

$$\forall v \exists x(\forall y\ xry\ \rightarrow\ xrx \wedge \forall t\ trv) \vee uru\ \ .$$

Now doing a double step sometimes (i.e. using two from the list of equivalences) the above formulae are truth equivalent to each of the following:

$$\forall v \exists x((\forall y\ xry\ \rightarrow\ xrx \wedge \forall t\ trv) \vee uru)$$

$$\forall v \exists x(\exists y(xry\ \rightarrow\ \forall t(xrx \wedge trv)) \vee uru)$$

$$\forall v \exists x(\exists y \forall t(xry\ \rightarrow\ xrx \wedge trv) \vee uru)$$

$$\forall v \exists x \exists y \forall t((xry\ \rightarrow\ xrx \wedge trv) \vee uru)$$

The last is our prenex form.

Exercise 8.20 Give careful details of the justifications needed just above, showing also the intermediate steps.

Note that a **prenex form** for a formula **is definitely not unique**. This is obvious when you think of all the possibilities for renaming bound variables without changing the formula up to truth equivalence. A slightly more interesting aspect of non-uniqueness is the order of the 'quantifier-followed-by-variable's at the front of such a formula.

YOU CANNOT JUST SWITCH THE ORDER OF AN ADJACENT PAIR $\forall$ AND $\exists$. The notorious distinction between "$\forall\epsilon\exists\delta\cdots$" and "$\exists\delta\forall\epsilon\cdots$" from the basic definitions in calculus was perhaps your first taste of this. There are also some simpler examples of this in the early part of Chapter 5.

However, two adjacent $\forall$'s can be switched, and similarly for two adjacent $\exists$'s. Stated more carefully, we always have

$$\forall x\forall y\ F \quad \text{treq} \quad \forall y\forall x\ F \qquad \text{and} \qquad \exists x\exists y\ F \quad \text{treq} \quad \exists y\exists x\ F\ .$$

Exercise 8.21 Prove these, using the basic definitions from Chapter 6.

Consider, for example, the formula

$$\forall x\exists y\ F\ \rightarrow\ \forall u\forall v\ G\ \ ,$$

where we assume that all needed renaming of bound variables has been done. Moving the variables out to the front in the order x, then y, then u, then v, produces the prenex form

$$\exists x\forall y\forall u\forall v(F \rightarrow G)\ .$$

But the order x, then u, then y, then v, produces the prenex form

$$\exists x\forall u\forall y\forall v(F \rightarrow G)\ .$$

These two prenex formulae are equivalent because they're both equivalent to the original formula, but also because of 8.21. However, we could use also the order u, then x, then y, then v, producing the prenex form

$$\forall u\exists x\forall y\forall v(F \rightarrow G)\ .$$

So even though $\exists x \forall u H$ and $\forall u \exists x H$ are not truth equivalent in general, they sometimes can be. It happens here because of the nature of H itself, particularly because of our assumption that variables have been renamed; in particular, because F is FOFOO u and v, and G is FOFOO x and y.

The connective $\leftrightarrow$ has not been explicitly dealt with, and this gives rise to a caution. To deal with it, just use its definition in terms of $\rightarrow$. But be careful at this point to do some more renaming of bound variables. For example, suppose we had

$$\forall x F \leftrightarrow \exists y G \ ,$$

renaming had been done, and no quantifiers appear other than the two you see. So F has no free y, and G has no free x. Replace this formula by

$$(\forall x F \rightarrow \exists y G) \ \wedge \ (\exists y G \rightarrow \forall x F) \ ,$$

but then immediately rename in the second half, giving

$$(\forall x F \rightarrow \exists y G) \ \wedge \ (\exists y' G' \rightarrow \forall x' F') \ ,$$

using new variables, so that x, x', y and y' are four distinct variables, and so that F' and G' are F and G with x and y replaced by x' and y', respectively. Now you can proceed with the main prenexing procedure, obtaining

$$\exists x \exists y \forall y' \forall x' ((F \rightarrow G) \ \wedge \ (G' \rightarrow F')) \ .$$

The main result of this section is the theorem below. The proof is just a matter of producing enough notation to express the ideas used in converting a formula to prenex form, so it will be omitted. Note that a proof by induction on formulae need only consider the connectives $\neg$ and $\wedge$, and can ignore $\vee$, $\rightarrow$ and $\leftrightarrow$.

Theorem 8.6 (Prenex Form) *Any formula is truth equivalent to some formula of the form* $\mathsf{Q}_1 y_1 \mathsf{Q}_2 y_2 \cdots \mathsf{Q}_n y_n \ F$ *, for some* $n \geq 0$*, where* F *is a formula containing no quantifiers, each* y_i *is a variable, all distinct from each other, and where each* Q_i *is* $\forall$ *or* $\exists$*. (Somewhat irrelevant, we can add the condition that every* y_i *should have a free occurrence in* F*, so that any silliness is removed from the prenex formula.)*

Exercises.

8.22 For each of the following formulae, find an equivalent prenex formula.

(i) $\exists x\ (xry \to \forall y\ (yry \wedge \exists z\ zrx))$

(ii) $x < x+1 \to \forall x \forall z(z \times x \approx x \times z\ \wedge\ \exists x\ x+1 < (x+1)+1)$

(iii) xry

(iv) $\exists xxrx \vee \forall xxrx$

(v) $\forall x\ (x \approx x \wedge \exists x\ x \approx x)$

(vi) $((\forall xyrx) \to \forall yyrx) \to \exists z(zry \vee xrz)$

(vii) $\forall y(x \approx z \to \exists z \exists z\ x < y)$

(viii) $\forall y\ x \approx z \to \exists z \exists z\ x < y$

(ix) $(\forall x \exists y\ (x \times y) + x < x \times (y+1)) \leftrightarrow 1 \approx 0$

8.22.5 (This exercise, which was previewed in Section **5.2**, should now be relatively routine.) Divide the following formulae, from 1st order directed graph theory, into collections, with all formulae in each collection having the same meaning. Then, for each collection, provide a single translation into math/English for all the formulae in that collection.

(i) $\forall x \forall y \forall z(xry \wedge xrz\ \to\ y \approx z)$.
(ii) $\forall y \forall z \forall x(xry \wedge xrz\ \to\ y \approx z)$.
(iii) $\forall y \forall z \forall x(\neg y \approx z\ \to\ \neg(xry \wedge xrz))$.
(iv) $\forall y \forall z(\neg y \approx z\ \to\ \forall x \neg(xry \wedge xrz))$.
(v) $\forall y \forall z(\neg y \approx z\ \to\ \neg \exists x(xry \wedge xrz))$.
(vi) $\forall y \forall z(\exists x(xry \wedge xrz)\ \to\ y \approx z)$.
(vii) $\forall y \forall z \exists x(xry \wedge xrz\ \to\ y \approx z)$.
(viii) $\forall y \forall z \exists x(\neg y \approx z\ \to\ \neg(xry \wedge xrz))$.
(ix) $\forall y \forall z(\neg y \approx z\ \to\ \exists x \neg(xry \wedge xrz))$.
(x) $\forall y \forall z(\neg y \approx z\ \to\ \neg \forall x(xry \wedge xrz))$.
(xi) $\forall y \forall z(\forall x(xry \wedge xrz)\ \to\ y \approx z)$.

8.4 Skolemizing

(It may not surprise you to hear that the verbs "to prenex" and "to skolemize" do not yet appear in the Oxford English Dictionary. A royal com-

mission is needed to investigate the barbarous practice of verbifying nouns and adjectives—or should it be "verbicating"?! At least I didn't write "verbing"!)

FOR THIS SECTION, ALL FORMULAE WILL BE SENTENCES. Then there is no distinction between the two kinds of satisfiability discussed in Chapter 7 : A sentence is satisfiable if and only if there is at least one interpretation of the language in which it is true. But that is equivalent to the existence of an interpretation V and some $\underline{v}$ from V with the sentence true at $\underline{v}$.

The process discussed in this section is another one directed towards deciding, by some semi-automatic method, about the validity of arguments in $1^{\underline{st}}$order logic, and about the satisfiability of a formula or set of formulae. Just as in propositional logic, one can **change a validity question into a satisfiability question by checking for the unsatisfiability of the set consisting of the argument's premisses together with the negation of its conclusion**. If the set of premisses is finite, then as before we can just conjunct all the formulae in the earlier set, and get down to studying the satisfiability of a single formula. As in propositional logic, the argument is valid if and only if the formula is unsatisfiable.

Now of course two *equivalent* formulae are either both satisfiable, or neither is. So the previous section gets us down to the question of the satisfiability of a single formula in prenex form.

Given such a formula, we would like to get rid of its existential quantifiers, producing a universally quantified formula which is satisfiable if and only if the given one is. That's what skolemizing is all about. In this section, we show **how to get rid of the existential quantifiers** (but a sentence will remain a sentence after skolemizing).

In contrast to the previous section, the new, (existential quantifier)-free formula will NOT be truth equivalent to the old one, in general. In fact, it's rarely even a formula in the same language. What must be done to **gain the simplicity of getting rid of some quantifiers** is to complicate things a bit **by adding new constant and function symbols to the language**. This is reminiscent of, and not unrelated to, the work we did in Chapter 7 to prove Henkin's Theorem. It's also somewhat reminiscent of the procedures studied in propositional logic in the penultimate section of Chapter 4, where a formula was produced (with lots of new propositional variables) which was

'satisfiability-equivalent', but not truth equivalent, to the input formula.

The procedure is surprisingly simple. Suppose given a formula in prenex form.

(0) If it has the form $\exists y\ F$, introduce a new constant symbol c into the language and let the new formula be $F^{[y \to c]}$.

(1) If it has the form $\forall x \exists y\ F$, introduce a new unary function symbol f into the language and let the new formula be $\forall x\ F^{[y \to fx]}$.

(2) If it has the form $\forall x_1 \forall x_2 \exists y\ F$, introduce a new binary function symbol f into the language and let the new formula be $\forall x_1 \forall x_2\ F^{[y \to f x_1 x_2]}$.

The general case is pretty clear :
(n) If it has the form $\forall x_1 \forall x_2 \cdots \forall x_n \exists y\ F$, introduce a new n-ary function symbol f into the language and let the new formula be

$$\forall x_1 \forall x_2 \cdots \forall x_n\ F^{[y \to f x_1 x_2 \ldots x_n]}\ .$$

For example, the one-step skolemization of the number theory sentence

$$\forall x \forall y \exists z \forall t\ (x + y \approx z \times t\ \wedge\ z < 1 + y)$$

is

$$\forall x \forall y \forall t\ (x + y \approx fxy \times t\ \wedge\ fxy < 1 + y)\ .$$

The enriched language is the same as number theory, except that it has *three* binary function symbols, $+$ and $\times$ with infix notation, and f with prefix notation.

A rough expression of the motivation for this procedure is this: A statement asserting the existence of something is replaced by a statement referring to a specific choice of that something. There is no difficulty with "substitutability" in the procedures above, as explained after the theorem statement at the end of this section. The variables $x_1, x_2, \cdots, x_n$ are just any old distinct variables, not necessarily the first "n" of our standard $1^{\underline{\text{st}}}$order language. Note that F itself may very well have some quantifiers as part of it, all at the front of course. All these quantifiers and the ones displayed above are distinct from each other. The given formula, the formula F, and the new

formula are all in prenex form. The net effect of the first step as above in skolemizing is to reduce the number of quantifiers "$\exists$" by one.

The main claim is that *the new formula is satisfiable as a formula from the new, enriched language, if and only if the old formula is satisfiable as a formula from the original language.* The proof is not hard, but will be left to the end of the section. It amounts to going back to the basic Tarski definitions of Chapter 6, and grinding away.

Now the step above, **eliminating one existential quantifier, can be repeatedly applied**, in the end producing a formula in prenex form which

(a) has only universal quantifiers, $\forall$, and no existential ones, $\exists$;

(b) is satisfiable in the new, multiply enriched language if and only if the originally given formula is satisfiable as a formula from the original language.

A prenex formula satisfying (a) is called a **universal prenex formula.**

This is then what is generally known as the process of skolemizing a given formula in prenex form. The new constant and/or function symbols introduced to enrich the language are known as skolem constants and skolem functions.

In the next section, we'll consider the matter of even getting rid of (or at least ignoring) the remaining, universal quantifiers.

This procedure of skolemizing is sufficiently simple that just a few examples ought to suffice.

Examples. (A) To skolemize the prenex formula

$$\exists x \forall y \forall z \exists u \forall v \exists w \ (xry \vee zru \vee vrw) \ \ ,$$

the steps done start, as always, from the 'outside' $\exists$, and proceed inwards. Here they successively produce

$$\forall y \forall z \exists u \forall v \exists w \ (cry \vee zru \vee vrw) \ ;$$

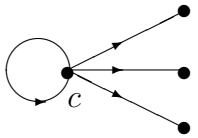
$$\forall y \forall z \forall v \exists w \ (cry \vee zrfyz \vee vrw) \ ;$$

$$\forall y \forall z \forall v \ (cry \vee zrfyz \vee vrgyzv) \ .$$

This last one is now 'fully skolemized'. The first formula is satisfiable by a directed graph if and only if this last one is satisfied in the enriched language of

directed graph theory plus one constant symbol c, plus two function symbols, binary f and 3-ary g. So to directly prove satisfiability of the latter, one must give a specific directed graph V (that includes specifying r^V, i.e.'drawing and arrowing the edges'), a particular vertex c^V of that graph, and two functions, f^V of two vertex variables, and g^V of three vertex variables. And then show that the last sentence is true of that structure.

For a human, it's often easier just to find a model for the original sentence. But here it's easy for both. Here is such a graph.



Exercise 8.23 Change all "$\vee$" to "$\wedge$" in **(A)** and decide about satisfiability of the corresponding sentences.

For an automated theorem proving 'device', attempting to use the sort of heuristics alluded to in the next section, the last, skolemized sentence in example **(A)** is easier to use— actually the first one is impossible to use.

(B) Now suppose we look at the long sentence beginning the previous section which expresses, in the $1^{\underline{\text{st}}}$order language of ring theory, the theorem about "all non-zeroes invertible implies no non-zeroes multiply to give zero":

$$(\forall x(\neg x \approx 0 \to \exists y\, x \times y \approx 1)) \;\to\; \forall u \forall v\, (\neg u \approx 0 \wedge \neg v \approx 0 \;\to\; \neg u \times v \approx 0) \quad .$$

The challenge is to give a universal prenex sentence which is satisfiable if and only if that one is. (They both *are* satisfiable as we know from algebra.) The first step is to produce a prenex sentence equivalent to the given one :

$$\exists x \forall y \forall u \forall v\, ((\neg x \approx 0 \to x \times y \approx 1) \;\to\; (\neg u \approx 0 \wedge \neg v \approx 0 \;\to\; \neg u \times v \approx 0)) \quad .$$

The other step is to skolemize, producing in the language of 'ring theory enriched by one extra constant symbol c', the following :

$$\forall y \forall u \forall v\, ((\neg c \approx 0 \to c \times y \approx 1) \;\to\; (\neg u \approx 0 \wedge \neg v \approx 0 \;\to\; \neg u \times v \approx 0)) \quad .$$

(C) Those of you who have studied some group theory may have noticed a slight difference between the axioms defining *group* there, and what we gave

in Chapter 5. Converted to 1$^{\underline{\text{st}}}$order, the axioms you may have seen could be written as follows.

$$\forall x\ \forall y\ \forall z\ (x \times y) \times z \approx x \times (y \times z)$$

$$\exists x \forall y \exists z\ (x \times y \approx y\ \wedge\ y \times x \approx y\ \wedge\ y \times z \approx x\ \wedge\ z \times y \approx x)$$

The first one is the associative law as before. The second one is the combination of the existence of the identity element with the existence of inverses. It was best to combine them here, since the existence of inverses refers to the identity element. In group theory you would often prove the uniqueness of the identity element before introducing the existence of inverses, so that there would be no ambiguity about which identity element was being referred to. Notice that here the language need only have the one special symbol $\times$; no constant symbol (for the identity) nor any unary function symbol (for inverses) are needed to state the group axioms this way.

Skolemizing the associativity axiom does nothing. But skolemizing that second axiom produces the following, where we use 1 for the constant symbol introduced to get rid of "$\exists x$", and ι is used as the unary function symbol to get rid of "$\exists z$".

$$\forall y\ (y \times 1 \approx y\ \wedge\ 1 \times y \approx y\ \wedge\ y \times \iota y \approx 1\ \wedge\ \iota y \times y \approx 1)$$

But this is just the two axioms from our original version, combined by more-or-less conjuncting them. (We also did a minor prenexing, and mixed the infix notation $\times$ with the prefix notation ι).

So you can see how prenexing is producing something which really says the same thing, but with no existential quantifiers, and a few extra special symbols.

Group theorists seem to prefer expressing things the first way, and logicians/universal algebraists the second way without "$\exists$". The difference is mainly just a matter of technical convenience. Similar things happen with respect to other algebraic objects, such as rings.

Exercise 8.24 For each of the following prenex formulae, find a universal prenex formula which is satisfiable iff the given one is. State clearly what the language of the new one is.

(i) $\exists x \forall y \exists z\ (xry \rightarrow\ yrz)$

(ii) $\forall y \exists x \forall z \exists t \exists u \exists v\ x + y + z + t + v \approx 0$

(iii) xry

(iv) $\exists x \forall y\ xrx$

(v) $\forall y \exists x\ xrx$

Exercise 8.25 For each of the following formulae (same ones as in 8.22), find a universal prenex formula which is satisfiable iff the given one is. State clearly what the language of the new one is.

(i) $\exists x\ (xry \to \forall y\ (yry \wedge \exists z\ zrx))$

(ii) $x < x+1 \to \forall x \forall z (z \times x \approx x \times z\ \wedge\ \exists x\ x+1 < (x+1)+1)$

(iii) xry

(iv) $\exists xxrx \vee \forall xxrx$

(v) $\forall x\ (x \approx x \wedge \exists x\ x \approx x)$

(vi) $((\forall xyrx) \to \forall yyrx) \to \exists z(zry \vee xrz)$

(vii) $\forall y(x \approx z \to \exists z \exists z\ x < y)$

(viii) $\forall y\ x \approx z \to \exists z \exists z\ x < y$

(ix) $(\forall x \exists y\ (x \times y) + x < x \times (y+1)) \leftrightarrow 1 \approx 0$

To conclude, here is the theorem which states precisely why skolemizing is a worthwhile process.

Theorem 8.7 *Suppose given a* $1^{\underline{\text{st}}}$*order language with set of special symbols* $\mathcal{L}$*, and a prenex formula* G *from it. Let* c *be a constant symbol and* f *be a function symbol, neither in* $\mathcal{L}$*.*

(0) If $G = \exists y\ F$ *, then* G *is satisfiable with respect to the language* $\mathcal{L}$ *if and only if* $F^{[y \to c]}$ *is satisfiable with respect to the language* $\mathcal{L} \cup \{c\}$ *.*

(n) If $G = \forall y_1 \forall y_2 \cdots \forall y_n \exists y\ F$ *(where* y *and all the* y_i *are distinct), then* G *is satisfiable with respect to the language* $\mathcal{L}$ *if and only if*

$$\forall y_1 \forall y_2 \cdots \forall y_n\ F^{[y \to f y_1 y_2 \ldots y_n]}\ .$$

is satisfiable with respect to the language $\mathcal{L} \cup \{f\}$*, where* f *is* n*-ary.*

Remarks. (i) In part (n), whether we first substitute into F and then apply the quantifiers, or else do it in the other order, the resulting formula

is the same. That is,

$$\forall y_1 \forall y_2 \cdots \forall y_n \ (F^{[y \to f y_1 y_2 \ldots y_n]}) \quad = \quad (\forall y_1 \forall y_2 \cdots \forall y_n \ F)^{[y \to f y_1 y_2 \ldots y_n]} \quad .$$

But on the right-hand side, our "substitutability" condition is violated. Therefore, in the proof below, we think of the new formula definitely as the left-hand side above, in order to be able to apply **6.9**. (There is no inherent ambiguity in the language plus substitution notation. The ambiguity is in combining the substitution notation with the abbreviation which drops brackets surrounding subformulae beginning with a quantifier.)

(ii) The division of the theorem into two parts isn't really necessary. In sophisticated circles, a constant symbol is treated as a '0-ary function symbol'. But there's no point here in getting wound up trying to figure out what "0-ary" should mean!

Sketch Proof. (0) Here we shall take y to be x_3, and trust as usual that you will become convinced of the general case. Suppose that $\exists x_3 F$ is satisfiable. Thus $\underline{v} \in (\exists x_3 F)^V$ for some V and $\underline{v}$ from V. So, for some u in V, we have $(v_1, v_2, u, \cdots) \in F^V$. We must show that $F^{[x_3 \to c]}$ is satisfiable. Let the interpretation W of the enriched language be the same in all respects as V, except that we must specify c^W, and we do so by letting $c^W = u$. It suffices to show that $\underline{v} \in (F^{[x_3 \to c]})^W$. By **6.9**, this holds if and only if $(v_1, v_2, c^{\underline{v}}, \cdots) \in F^W$. But since $F^W = F^V$ and $c^{\underline{v}} = c^W = u$, we are done.

Conversely, suppose that $F^{[x_3 \to c]}$ is satisfiable. All the steps in the paragraph above are "iff"s, so just reversing the argument there will do the trick.

(n) Here we shall be even more sketchy and take n to be 1, and take the sentence G to be $\forall x_2 \exists x_3 F$. (Recall that the y_i in the theorem statement are not necessarily the first "n" variables in the language; but now x_2 and x_3 definitely are the 2nd and 3rd variables, so we can concentrate on the 2nd and 3rd "slots" below.) Suppose that $\underline{v} \in (\forall x_2 \exists x_3 F)^V$ for some V and some $\underline{v}$ from V. Then, for all $u \in V$, we have $(v_1, u, v_3, \cdots) \in (\exists x_3 F)^V$. Thus, for all $u \in V$, there is a $w_u \in V$ with $(v_1, u, w_u, v_4, \cdots) \in F^V$. For each u, pick one particular w_u and define $f^W : V \to V$ by $f^W(u) = w_u$. Also let W be the interpretation of the new language (enriched by including f) which is the same in all respects as V, except that we additionally specify f^W as above. It is now straightforward to check, using **6.9**, that $\underline{v} \in (\forall x_2 F^{[x_3 \to f x_2]})^W$, which shows, as required, that $\forall x_2 F^{[x_3 \to f x_2]}$ is satisfiable.

Furthermore, as in the first part, the argument above is reversible, almost word-for-word, to give a proof of the converse.

8.5 Clauses and Resolution in $1^{\underline{st}}$order Logic.

To finish this chapter, here is a brief indication of how to take the previous three sections further. We reduce the unsatisfiability question for a sentence (and thereby the validity question for a $1^{\underline{st}}$order argument consisting of sentences) to something which looks somewhat like the material at the end of Chapter 4 on using resolution in propositional logic. Recall however the comments at the beginning of this chapter: In the end, it is certain for most languages and premiss sets that one will get a heuristic for solving this problem which works only for some input, definitely not an algorithm, much less an efficient algorithm. The best that can be hoped for in general is a procedure which, when confronted with a valid argument, will eventually agree that, yes, this is valid, but when confronted with an invalid argument, may bumble around forever, and never produce an answer.

So we start with a universal prenex sentence, and wish to decide satisfiability. First throw away the universal quantifiers, leaving a quantifier-free (probably non-sentence) formula. Now find an equivalent formula in $1^{\underline{st}}$**order conjunctive form**. This is the same as propositional conjunctive form, except that atomic formulae take the place propositional variables. So here a **literal** is any atomic formula or negated atomic formula. A **clause** is a set of such literals. The previous $1^{\underline{st}}$order conjunctive form is a conjunction of formulae (its **constituents**), each of which is a disjunction of literals. The methods of propositional logic are all that is needed to produce a conjunctive form. Producing a set of clauses is now done as before, with one clause for each constituent.

Efficiency here is already a question. The efficient methods of the penultimate section on 3-SAT in Chapter 4 can't be used in any obvious way here. Recall that those methods don't do exactly what we said above. Rather they produce a formula whose satisfiability is equivalent to that of the original formula, rather than the formula itself being truth equivalent to the original formula. But the introduction of auxiliary propositional variables there has no obvious analogue here of introducing of auxiliary atomic formulae.

In any case, assuming that a set of clauses has been found before the world comes to an end, the following is clearly the correct definition for satisfiability for a set of clauses, given that we began with a universally

quantified sentence.

A set of clauses is **satisfiable** if and only if there is an interpretation V of the language with the following property: *For each clause and each $\underline{v}$ from V, there is a literal in that clause which is true at $\underline{v}$ from V.*

It is then straightforward to see that the sentence we began with is satisfiable in the ordinary sense if and only if the resulting set of clauses is satisfiable in the sense of the definition above.

It is quite possible to give 'by hand' examples of arguments in $1^{\underline{\text{st}}}$order, with validity reduced to unsatisfiability of a set of clauses using the last three sections. But they need to be so simple that the validity of the argument is always obvious one way or the other. On the other hand, after beavering away to produce the set of clauses, you usually find that the satisfiability of the set of clauses is not so obvious at all to ordinary mathematical minds. So it is probably best to basically leave this topic here, for you to take up elsewhere when you study automated theorem proving.

A lot of good material on the previous paragraphs and the next four can be found in **Burris**. See also **Schöning**, pp.70–107, which is very informative.

There is an earlier approach along these lines, due to Löwenheim, Skolem, Gödel, and especially Herbrand. It reduces the satisfiability of a single formula in $1^{\underline{\text{st}}}$order logic to that of an infinite set of formulae in propositional logic. There are some subtleties due to the difference between $1^{\underline{\text{st}}}$order logic as we have treated it, and '$1^{\underline{\text{st}}}$order logic *without equality*'. These subtleties will be ignored in the rough description below.

It turns out that the fundamental problem arises because of the existence of formulae which are neither logically valid nor unsatisfiable, but for which no finite model exists. A construction due to Herbrand is made which is reminiscent of the construction of the autological interpretation in Henkin's theorem. But this construction leaves some free parameters. Using this, Herbrand proved that the Skolemized sentence is unsatisfiable if and only if a certain infinite set of propositional formulae has a finite subset which is unsatisfiable (in the propositional sense—using truth assignments, not interpretations, of course).

Let's make this a bit more precise. One removes the universal quantifiers from the front of the Skolemized sentence, and then substitutes constant terms (as defined in the proof of Henkin's theorem) for the variables in the

resulting quantifier-free formula. Doing this in every possible way produces an infinite set of 1st order formulae which contain no variables (nor quantifiers, of course). Now just ignore the 'internal structure' of the atomic subformulae of all these formulae. That is, choose a different propositional variable for every atomic formula which occurs, replace the atomic subformula with its corresponding propositional variable everywhere, and you now have the infinite set of propositional formulae referred to above. As it stands, the procedure indicated here can be staggeringly inefficient. And it isn't a decision procedure, because of the set being infinite. But it does 'semi-decide' (as you may suspect) in that, if the original 1st order formula is unsatisfiable, that can be determined in a (usually very large) finite number of steps.

This then gives rise to procedures which have Skolemized sentences as input, and which will terminate with the correct answer if the input is unsatisfiable, but run forever otherwise. This is in concordance with our knowledge that the validity of arguments in 1st order is in general undecidable, but that the set of valid consequences of a decidable premiss set can be algorithmically listed.

Exercise 8.26 Explain why one cannot use the method of resolution (as described in Chapter 4) directly to learn about the satisfiability of sets of 1st order clauses. The idea is to again treat atomic formulae as though they were propositional variables, ignoring their 'inner meaning'. In particular, explain why propositional resolution is inadequate as a method, giving an example of a set of clauses where it fails.

8.6 How close to the objective are we?

At the beginning of the book, it was stated that the main objective of mathematical logic was to analyze the validity of arguments. Here is a brief summary of the situation, as described in more or less detail in this and earlier chapters.

Propositional logic plays two roles here.

On the one hand, it can in a sense be said to be part of the main topic, 1st order logic—for example, dealing with 1st order tautologies. (As well, the fragment of 1st order logic consisting of formulae with no equality symbols, no free variables, and only one universal quantifier whose scope is the entire formula, where the language has a countable infinity of unary relation symbols and no constant or function symbols, is 'isomorphic to', i.e. 'essentially the same as', propositional logic—this is illustrated in Section **9.4**, and Exercise

9.13 there makes it more precise.)

On the other hand, the whole theory in the propositional case is a kind of 'toy theory', with analogues of most of the features of the $1^{\underline{st}}$order theory—tautologies vs. the logically valid, truth assignments vs. interpretations, satisfiability for both, proof systems for both, soundness, completeness, $\cdots$.

The biggest contrast comes when we consider finding algorithms for deciding whether or not an argument is valid.

In propositional logic, there are such algorithms—input: a finite set of premisses and a conclusion—output: yes or no according as to whether the argument is valid or not.

In $1^{\underline{st}}$order logic, the best we can hope for is an 'Type semi-D algorithm' of the following form—input: a finite set of premisses and a conclusion—output: yes if the argument is valid, and possibly runs forever with no decision if the argument is invalid.

Finally, it is arguable that one of the most important problems in mathematics at the beginning of the 21st century is the conjecture that no algorithm for propositional logic as above exists which is efficient, that is, runs in polynomial time (with no prior restriction on the type of formula in the input). This is equivalent to the $\mathcal{P} \neq \mathcal{NP}$ conjecture.

To finish the chapter (and the basic theoretical development of the book) with some history, the following satirical quote from a great mathematician perhaps betrays a bit of insecurity :

> *. . . we might imagine a machine where we should put in axioms at one end and take out theorems at the other, like that legendary machine in Chicago where pigs go in live and come out transformed into hams and sausages. It is no more necessary for the mathematician than it is for these machines to know what he is doing.*
>
> Henri Poincaré

Church's theorem, that no 'mechanical' method can exist to decide validity in $1^{\underline{st}}$order logic, presumably allows Poincaré to rest in peace. That theorem was rediscovered independently by Turing before Church published it. Since Turing had a more convincing analysis of what 'mechanical methods' (or algorithms) are, his version of the theorem was, on the face of it, also more convincing. However, Turing soon proved that his and Church's definitions of what is computable are equivalent. Church's Thesis (or the Church–Turing Thesis) says that no wider definition is possible. See also the last few pages of this book and **[CM]**.

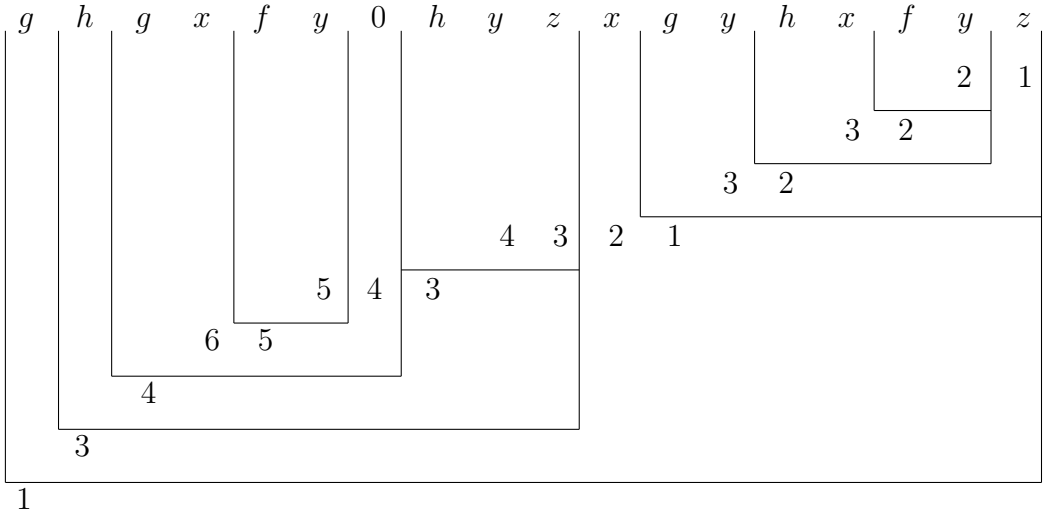
Appendix B : ALGORITHMS TO CHECK STRINGS FOR 'TERMHOOD' AND 'FORMULAHOOD'.

Here are some algorithms for determining which strings of symbols are terms or formulae; and also for automatically producing all occurrences of subterms and subformulae, producing verification columns, etc. I think this stuff is pretty boring, but there is one important understanding which comes out of it: if a string is a term or formula, then it is so 'in only one way'. There is no ambiguity. Basically this says that if we used trees in place of verification columns, the answer is unique. To prove this, some theory is given in the third section below, which as well deals with our two equivalent definitions of subformula and subterm occurrence, and which also can be used to justify the algorithms. The latter are given in the first (terms), second (propositional formulae), and fourth ($1^{\underline{st}}$order formulae) sections below.

A term checking procedure, the 'tub' algorithm.

Here we give one algorithm for checking, in a $1^{\underline{st}}$order language, whether a finite string of function symbols, constant symbols and variables is a term. This will be done for the prefix notation, with no brackets. Infix notation is dealt with in the next section.

It is probably easiest to get this across by means of a few examples. Consider a language whose special symbols include : 0 (a constant symbol), f (a unary function symbol), g (a 3-ary function symbol), and h (a binary function symbol). The top line of the display below is a string to be checked, and the rest consists of numbers and 'tubs' which do the checking, starting from the right-hand end and proceeding to the left. This is explained in more detail below this example, which *does* turn out to be a term.



Starting from the right with the integer 1, you place successive positive integers under the first bunch of constant symbols and variables. (If the rightmost symbol is a function symbol, you stop immediately and say it's not a term.) When encountering the first function symbol from the right, you make a tub, beginning with that function symbol and including as many of the symbols to its right as the arity of that function symbol. Under that tub you place the smallest integer which is inside the tub—the rightmost integer in the tub. Once a tub is formed, all the integers inside it are to be ignored; only the integer under it is used.

Keep moving from right to left in this manner. Each time you encounter a constant symbol or variable, you write down the next integer to the one immediately to its right which is still in use. Each time you encounter a function symbol, you attempt to make a tub of the correct size (the arity of that function symbol), using the integers in use immediately to its right. If this cannot be done because there are not enough integers in use immediately to its right, you stop and declare the string to be not a term. If you can continue all the way to the last (leftmost) symbol in the string, then, for the string to be a term, that last symbol must be a function symbol, except in the trivial case of a string of length 1. And the tub associated with the leftmost symbol must go across the whole string and end up exactly swallowing up the rightmost symbol. If it does, you have a term. If it falls short you don't. (And if it 'goes too far', i.e. the arity of the leftmost symbol is greater than the number of integers in use, then we are in the case above—"not enough integers"—so it's not a term.)

In the displayed example, we do have a term.

But suppose the leftmost symbol had been h rather than g, i.e. the string was $h \;\; h \;\; g \;\; x \;\; f \;\; y \;\; 0 \;\; \cdots$. Then the algorithm would have operated all the way from right to left, but the last tub would have ended between the x and g, seven symbols from the right-hand end, and we would not have a term.

On the other hand, if the second symbol from the left had been a g, i.e. the string was $g \;\; g \;\; g \;\; x \;\; f \;\; y \;\; 0 \;\; \cdots$, then the second last tub would have ended between the x and g, seven symbols from the right-hand end, and had a 2 under it. Then again we would not have a term, because the last tub couldn't be formed.

Suppose that you have checked a sequence in this manner, and found that you do have a term. Then by reading your tubs again from right to left, you can write down an expression for the term which looks like some (often quite complicated) function from multivariable calculus. That is, you can add in commas and pairs of brackets to the string, producing this 'function of several variables'. In the displayed example above, this expression is

$$g(h(g(x, f(y), 0), h(y, z)), x, g(y, h(x, f(y)), z)) \ .$$

Not terribly readable, is it?!

If, in such an expression, you changed f to f^V, similarly for g and h; changed each constant symbol c to c^V; and inserted the appropriate slot from some $\underline{v}$ for each variable, that expression would become $t^{\underline{v}}$ (where t is the string just verified as being a term, and V is the set component of some interpretation, with $\underline{v}$ from V).

This algorithm also makes it easy to produce a term verification column, when the answer is "Yes, this is a term". To do this you successively write down, each below the previous, one term for each integer that you produced in the tub algorithm. If the integer was under a constant symbol or variable, just write down that symbol. If the integer was under a tub, write down all the symbols in the tub, in their given order from left to right. Once you are finished, you might as well erase any repeats in this vertical list, though this is not strictly necessary. After erasing, you have a list of every *subterm*. Before erasing, you have a list of every *occurrence* of a subterm. So, for subterms of length > 1 (that is, not constant symbols or variables), each tub gives an occurrence, and these are all the occurrences.

To describe the algorithm above and carry it out by hand, both the integers and tubs inserted are handy. But actually to simply check if the string is a term, all you need do is insert integers. Start from the right with 1, and move left. Increase your integer by one each time you encounter a constant symbol or variable; and go down by one less than the arity each time you encounter a function symbol. If you ever get a non-positive integer, or if the last (leftmost) one is not 1, you don't have a term. Otherwise you do.

But this doesn't make the subterms and verification column all that transparent. However, you could alternatively dispense with the integers, and just

use tubs. (You can even use a little tub of length 1 under each constant symbol and variable, to keep track of how wide the other tubs should be).

Those with interests in that direction might enjoy writing out this algorithm properly, so that it can be easily coded for a computer.

Exercise B1 Check, for each of the following strings, whether or not it is a term. When the answer is yes, produce a term verification column, 'calculus function expression' for each line in that column, a list of all subterms and all occurrences of subterms. The function symbols are f—unary; g—3-ary; and h—binary.

(i) $ghfhxwcyhxgccy$

(ii) $hgxygwgghhxyxuvcyz$

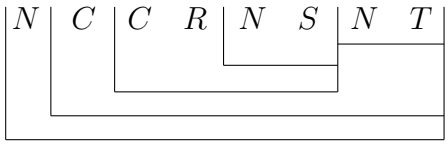
(iii) $hxgchxfy$

(iv) $hgxygfzhcccgxyghffxxxhyc$

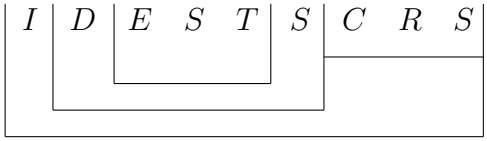
(v) $gcfhyxcz$

How to Polish your prose.

Suppose now that we use upper case letters for a short while, and that our language has a unary function symbol N, a binary function symbol C, and constants R, S and T. Then we have a term just below, as checked with tubs.



Suppose also it has three other binary function symbols I, E and D. Then again we have a term just below, as checked with tubs.



Now comes the little surprise perhaps. I don't like my notation, so I've decided to rename N as $\neg$, and call it the negation symbol; and also to

rename C as $\wedge$, and call it the conjunction symbol, and use infix notation. Then the first term above becomes

$$\neg((R \wedge \neg S) \wedge \neg T) \ .$$

I also decide to rename D as $\vee$, I as $\rightarrow$, E as $\leftrightarrow$, and to call the first two the disjunction symbol and the implication symbol, and to use infix notation for all three. Then the second term above becomes

$$((S \leftrightarrow T) \vee S) \rightarrow (R \wedge S) \ .$$

Well, I guess you can see what is going on. We are using the trick of regarding a formula in propositional logic as a term in some partially specified $1^{\underline{\text{st}}}$order language. Initially I tried to hide what was going on by writing the two formulae in Polish notation, changing the symbols for the connectives, which really are function symbols operating with propositional variables, which themselves really are constant symbols, from this point of view. By means of the two examples, you see that this can be done for both a (strict) formula, and for any formula abbreviated using the three other connectives.

The main point is that the algorithm(s) of the previous section can now be used to check strings of propositional logic symbols for being formulae, and to produce formula verification columns, subformula occurrences, etc.

There is just one question: those algorithms operate with prefix notation, which in this case we also call Polish notation. Therefore it is desirable to discuss algorithms which convert between Polish and ordinary notation in the language of propositional logic.

But we might as well do it in general for terms from any $1^{\underline{\text{st}}}$order language; that is, discuss algorithms which convert between infix and prefix notation, where some or all of the binary function symbols in the language can be used with either notation. This is really something you might prefer to figure out for yourself, before reading the next page or so.

The important half here is to get from a string, which contains some infix notation and brackets, to either a decision that it's not a term, or to a string which is entirely prefix and has no brackets. It's best that different names be used for the two versions of any binary function symbol, for example, $\times$ for the infix version, and m for the prefix version.

Here is a 1-step procedure, which, if repeated, will eventually achieve the above.

Read from left to right, searching for the first right bracket.

If none occurs, you are finished, and the the entire string isn't a term if any left brackets do occur; otherwise,

if no left bracket occurs before the first right bracket, then the entire string isn't a term; otherwise,

if, between the first right bracket and the left bracket immediately to its left, you don't have exactly one infix notation binary function symbol (say, $\times$), then the entire string isn't a term; otherwise,

erase that $\times$, and erase the left bracket and right bracket referred to above, and put an m in place of that left bracket.

Repeating this as often as needed, the algorithm will clearly terminate, since the number of brackets is finite. Even if no repetitions above lead to eliminating the string as a possible term, you still must use the procedure of the first section to check for termhood of the resulting bracket-free string.

For example, suppose also that f is a unary function symbol. Applying the one-step procedure to $f(x\times fyz)$ produces $fmxfyz$, but that then checks out to be not a term.

As for going the other way, from a term in the usual prefix notation, to introduce infix notation for some or all of its binary function symbols, proceed as follows.

First check that you actually have a term, by the tub algorithm. If so, here is an obvious 1-step procedure for introducing one extra occurrence of an infix notation binary function symbol and its pair of brackets, if that is possible.

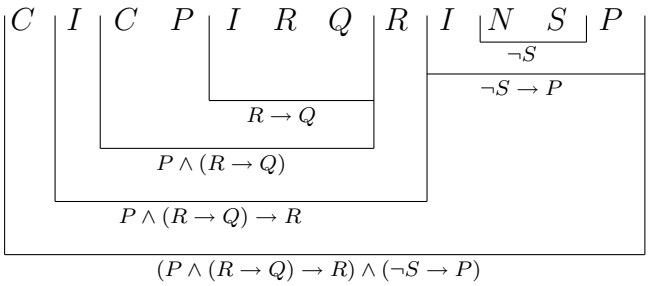
Find, if possible, a prefix notation binary function symbol, say m.

If there are none, nothing can be done; otherwise,

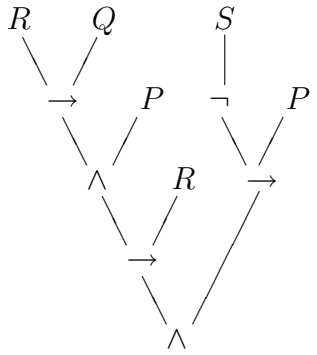
The symbol m will be followed immediately by a pair of terms, which can be found by algorithms above. (See also the 'left-to-right tub algorithm' in the next section.)

Erase m, put a pair of brackets around the string which is the pair of terms above, and insert $\times$ between them.

These conversions back and forth are very easy. One can even use the tubs to give a neat way to record the conversion from prefix to infix. One example suffices to explain it. We'll do one getting from Polish to normal notation in propositional logic.



This diagram is not far from the **tree** of the formula, a refinement of the verification column, mentioned briefly near the beginning of Chapter 1. One can make a more economical tree by just inserting the connective producing each subformula of length greater than 1. For example, the economical tree corresponding to the formula just above is *almost* as follows. There is a slight error, which would be more serious if one "$\wedge$" had been a "$\rightarrow$". Find the error.



More generally, every term in a 1st order language has a unique tree which is a refinement of any of its term verification columns.

Exercises.

B2 Convert these prefix notation terms to infix notation, but only if they are indeed terms. Here a and m are binary function symbols, whose corresponding infix mates should be called $+$ and $\times$, respectively.

(i) $mmammxyxmz10a0x$

(ii) $amzaazxmm1yy$

(iii) $aaaxxxx$

B3 Convert these infix notation terms to prefix notation, but only if they are indeed terms. Here $+$ and $\times$ are binary function symbols, whose corresponding prefix mates should be called a and m , respectively.

(i) $((y + (x \times x) \to (x + (y \times y)))$

(ii) $(0 + (z \times ((z + x) + ((1 \times y) \times y))))$

(iii) $((x \times x) + ((y \times y) + ((z \times z) + (w \times w))))$

(iv) $(x \times x + y)$

B4 Convert these abbreviated propositional formulae to Polish notation, using the symbols from the beginning of the section. Then go through the mechanical (tub) check that they are all formulae, and use this calculation to produce a formula verification column (converting back to usual notation), a list of all subformulae and of all subformula occurrences.

$$P \wedge \neg Q \to \neg(R \wedge S) \qquad P \wedge (\neg Q \to \neg R \wedge S)$$
$$P \wedge (\neg Q \to \neg(R \wedge S)) \qquad P \wedge ((\neg Q \to \neg R) \wedge S)$$
$$(P \wedge (\neg Q \to \neg R)) \wedge S \qquad (P \wedge \neg Q \to \neg R) \wedge S$$

Potentially boring facts on readability, subterms and subformulae.

There are a few matters to tidy up.

When dealing with replacement of a subformula occurrence by an equivalent formula, we had two ways of characterizing occurrences of subformulae, and later, more generally, of subterms in a term. The first was simply as a connected substring which happened to be a term; the other is the inductive

definition below, which is really the right way to do it and is all that's really needed. But the other definition is more accessible. It's better for examples and for notation when we formulate replacement rules of inference, whereas the inductive definition is essential for theory. So we should show that the two definitions are equivalent. This is done in the first and last of the four propositions below.

For anyone who feels impelled to formulate the previous algorithms of this appendix rigorously, and to prove their correctness, the propositions that follow are basically what is needed.

We shall deal with terms in prefix notation. A *subterm occurrence* is more than just a term that occurs within a bigger term; it also includes the 'location' within that bigger term, since the same smaller term can occur several times as subterm occurrences in a given term. This can be made more precise by defining a string of length n to be a function from $\{1, 2, \cdots, n\}$ to the set of symbols. In particular, a term is such a function. The above point is then made by having each subterm occurrence be a function from a *subset* of that domain. But we'll operate more informally, and speak 'physically', for clarity.

The *subterm occurrences* in a term t are defined inductively on t:

When t is a constant symbol or a variable, the only subterm occurrence in t is t itself.

When t is $ft_1t_2 \ldots t_n$, the subterm occurrences in t are t and all subterm occurrences in any of the t_i.

Prop. I. *Every subterm occurrence in a term t is a 'connected' substring which is itself a term when thought of as a string on its own. (If you must,* connected *means that the domain of the subterm occurrence is a set of* consecutive *integers).*

Proof. Proceed by induction on terms. When t is just a constant term or variable, there aren't any substrings which aren't connected. When t is $ft_1t_2 \ldots t_n$, both t itself (obviously), and all subterm occurrences in each of the t_i (by the inductive hypothesis) are connected substrings which form terms on their own, so that's it.

Prop. II. *If a term t is factored as juxtaposition of strings r and s, that is, $t = rs$, with s non-empty, then r is not a term.*

Proof. This time, proceed by induction on the *length* of t to show that

a factorization in which r is a term would force s to be empty. Since the empty string isn't a term, this takes care of the initial case, and forces r to have the form $fr_1r_2\ldots r_n$ when t is $ft_1t_2\ldots t_n$ in the inductive step (with, of course, f being an n-ary function symbol). Here we have chosen the r_i to be some terms (not assumed unique—that would be almost begging the question of this section). They exist because we are assuming r is a term. But now we'll show that $r = t$ as required, by showing $r_i = t_i$ for all i. If not, choose the least i with $r_i \neq t_i$. But then, one of the latter would be a proper initial substring of the other, with both being terms, contradicting the inductive assumption.

It follows immediately from this proposition and the argument in the last two sentences that a term is uniquely readable, that is, *if the strings $ft_1t_2\ldots t_n$ and $fr_1r_2\ldots r_n$ are identical, where all the t_i and r_i are terms and f is an n-ary function symbol, then $t_i = r_i$ for all i.*

Prop. III. *Every symbol occurring in a term is the beginning of a* unique *subterm occurrence. (And so, in a sense, that describes all the subterms.)*

Proof. Proceed by induction on terms, the initial case again being trivial. When t is $ft_1t_2\ldots t_n$, the unique subterm beginning with f is t itself, by the previous proposition and the inductive definition. As for subterm occurrences beginning with the second or a later symbol in t, they certainly exist, there being (by the inductive assumption) one (and only one) lying entirely within the t_i which contains that initial symbol. But could there be another subterm occurrence, starting at that initial symbol, which stretches past the end of t_i and into t_{i+1}? No, because by the inductive definition of *term*, any proper subterm occurrence lies entirely inside some t_i.

Prop. IV. *Any connected substring of a term t, which happens to be a term when just thought of as a string by itself, is actually a subterm occurrence in t.*

Proof. If it were not the unique subterm occurrence starting from the symbol which it starts from, the longer of those two strings would be a term in which the shorter of the two contadicted Prop II.

Now, as discussed ad nauseam in the previous section, all of this applies to Polish propositional logic, with *term* and *subterm occurrence* replaced by

formula and *subformula occurrence*. The discussion of conversion between that and ordinary propositional logic can easily be used to get the required results in propositional logic. In the next section we discuss formulae in $1^{\underline{\text{st}}}$order languages in a manner similar to the previous two sections. (For example, we treat $\forall x$ as though it were a unary connective.) That discussion can then easily be used to see that the same results go through in $1^{\underline{\text{st}}}$order. This was suppressed in the main text, but actually is needed—for example, when we spoke about the scope of a quantifier occurrence. All this will be left for you to provide the details.

Checking a string for formulahood in a $1^{\underline{\text{st}}}$order language.

Clearly this is going to be a bit more complicated than checking in the case of a propositional formula.

Firstly, it's probably best to change all occurrences of $\forall x$ to $\forall_x$, and similarly, when dealing with abbreviated formulae, change all occurrences of $\exists x$ to $\exists_x$.

Then these collections of ink molecules, $\forall_x$ and $\exists_x$, are thought of as just one symbol, which is a kind of unary connective, formally like negation.

It's probably a good idea now to reiterate the definition of $1^{\underline{\text{st}}}$*order formula* for a given language, with this new notation. We will call what this defines a **puritanical** $1^{\underline{\text{st}}}$order formula, since we later want to simultaneously consider the old style formulae, including abbreviations. The list of symbols is modified by adding the infinite sequence $\forall_{x_1}, \forall_{x_2}, \cdots, \forall_{x_i}, \cdots\cdots$, and removing $\forall$. Define terms as before, with all function symbols in the language in the prefix notation style (as well as all relation symbols, except for $\approx$).

Then a *puritanical formula* is defined to be any non-empty finite string of symbols which may be obtained from a 'blank page' by applying the following rules a finite number of times.

(i) Each string $t \approx s$ is a formula, for any terms t and s. (Such a formula is called **atomic**.)

(ii) For each relation symbol r, if its arity is n, then, for any sequence of terms $t_1, t_2, \cdots, t_n$, the string $rt_1t_2 \ldots t_n$ is a formula. (Such a formula is also called **atomic**.)

(iii) For any formula F, the string $\neg F$ is also a formula.

(iv) For any formulae F and G, the string $(G \wedge F)$ is also a formula.

(v) For any formula F, each string $\forall_{x_i} F$ is also a formula. (Notice that we

didn't use brackets here, just as we didn't with $\neg$, the other unary connective).

So these puritanical formulae live by very strict rules. Nicknames are not allowed for variables—they must be called x_i, not x or y, etc. Brackets are never allowed around subformulae formed by quantification or negation. No occurrences of $\vee, \rightarrow$ or $\leftrightarrow$ are allowed. Each occurrence of a subformula formed by the one binary connective $\wedge$ must be chaperoned, by having a pair of brackets surrounding it.

Actually, the only change from actual, as opposed to abbreviated, formulae earlier in the book is subscripting the quantifiers, and not having brackets for them. However, soon after beginning, we quickly slurred this by using x, etc. as variables, and then introducing the various abbreviations.

Now there are two questions to consider.

(i) Starting from a string of symbols in our new, puritanical style, how can we check automatically whether it's a puritanical formula?

(ii) Starting from what we are hoping is a (possibly abbreviated) formula in our old style, how can we convert it into a string in the new puritanical style, so as to use (i) to see whether it really was an abbreviated formula?

Before doing (i), it is useful to remark that there is a perfectly good 'tub algorithm' which reads from left to right, rather than right to left. So it's a term checking algorithm which we'll use to help single out the atomic subformula occurrences in a formula. But let's forget about tubs, and just concentrate on producing the integers under the symbols. When you actually have a term, the following will produce exactly the same integers as before, except in the opposite order.

Assume given a string (of length at least 2) of constant and function symbols and variables.

The leftmost symbol must be a function symbol, or the string is not a term. Under that first function symbol place a 1, and under the next symbol to the right, place the arity of that 1st function symbol.

Proceeding to the right, under each symbol place an integer by looking at the preceding symbol, to the left. If a function symbol, add (its arity-1) to what you had. If it's a variable or constant symbol, subtract 1 from what you had.

If you ever reach 0 before finishing, the string isn't a term. At the end,

you must reach 0 to place under the non-existent symbol after the rightmost symbol, which is not a function symbol. Then it's a term, but not otherwise.

It is quite easy to see that this is really producing the same thing as previously. Here is the first example we used for term checking, with its integers. You should go through the procedures, in both directions, to see how both give this answer.

g	h	g	x	f	y	0	h	y	z	x	g	y	h	x	f	y	z	
1	3	4	6	5	5	4	3	4	3	2	1	3	2	3	2	2	1	0

Now to do (i) above:

Suppose given a finite string of symbols from the puritanical list. The following will check if the string is a puritanical formula.

Search out any symbols $\approx$ or r, relation symbols.

Immediately to the right of each occurrence of an r, and to the right of each $\approx$, apply the algorithm above till a 0 is produced. If you can't produce a term immediately to the right of the relation symbol, you're finished, the string isn't a formula.

In case of binary relation symbols r other than $\approx$, do the same again immediately to the right of the term just produced, to produce, if possible, a second term. In case of n-ary relation symbols r for $n > 2$, repeat this as often as needed so as to produce "n" terms, if possible.

In case of symbols $\approx$, apply the original, right-to left, term checking algorithm immediately to its left. Produce the longest string you can which is a term.

Now assuming the correct number of terms has been produced for each occurrence of a relation symbol, check that there is no overlap at all, i.e. no symbols in common between all these terms for different relation symbols. If any overlap, you don't have a formula.

If no overlap, underline or circle or whatever each of these relation symbols (including $\approx$), together with its attached collection of terms. These will be the occurrences of atomic subformulae coming from relation symbols, if in the end it turns out that you do have a formula.

Temporarily replace each occurrence of an atomic formula by a propositional variable, all different, keeping a record, so that the atomic formulae can be restored later, if needed.

Using the previous two sections, check whether you now have a term in the following language:

constant symbols corresponding to the propositional variables used;

unary function symbols corresponding to the unary connectives $\neg$ and the infinite sequence of $\forall_{x_i}$'s ;

one infix binary function symbol corresponding to conjunction.

If you do have a term, then what you started with was a puritanical formula. Otherwise, it wasn't.

That takes care of (i).

Now what we'd really like is to be able to automatically check, for formulahood, a string written in our previous more relaxed style for $1^{\underline{\text{st}}}$order formulae, including all the possible abbreviations. So let's deal with (ii) above.

First replace all abbreviated names for variables, such as z, by their real names, such as x_3.

Next test the brackets and make sure they come in pairs. (I assume you know how to do this from high school algebra.)

Next search for any $\forall$. It must be followed by a variable x_i. Replace the two of them by $\forall_{x_i}$.

Next search for any $\exists$. It must be followed by a variable x_i. Replace the two of them by $\neg\forall_{x_i}\neg$.

In both of the last two cases, remove any pair of brackets, where the left one occurred just before the original quantifier.

Now apply the first part of the algorithm for (i) just above. That is, single out all the potential atomic subformulae by searching out any symbols $\approx$ or r, a relation symbol, and finding the terms to which they apply.

Also as in (i), temporarily replace each occurrence of an atomic formula

by a propositional variable, keeping a record, so that the atomic formulae can be restored later, if needed.

Restore any bracket pairs that were removed to produce abbreviations as discussed in Chapters 1 and 2.

Now, rather than getting rid of the abbreviations $\vee, \rightarrow$ and $\leftrightarrow$, just proceed as in the latter part of the algorithm for (i), except that now we are checking the string for being a term in the language with 4 infix binary function symbols rather than just $\wedge$, together with the infinity of unaries, and with propositional variables as constant symbols.

Example. Here is an example of applying this procedure. If the very long abbreviated formula from $1^{\underline{\text{st}}}$order set theory which defined the short form "$X = Y \times Z$" in Section **5.6** had been obtained using $\{y, \{y, z\}\}$ for (y, z), it would have been the following, with all brackets around subformulae built by quantifiers removed, and a pair of associativity brackets missing :

$$\forall x(x \in X \leftrightarrow \exists y \exists z(y \in Y \wedge z \in Z \wedge \forall u(u \in x \leftrightarrow (u \approx y \vee \forall v(v \in u \leftrightarrow (v \approx y \vee v \approx z))))))$$

Now making the quantifiers look more like unary function symbols/connectives, it becomes

$$\forall_x(x \in X \leftrightarrow \exists_y \exists_z(y \in Y \wedge z \in Z \wedge \forall_u(u \in x \leftrightarrow (u \approx y \vee \forall_v(v \in u \leftrightarrow (v \approx y \vee v \approx z))))))$$

First replace the existential quantifiers, and use N as a unary function symbol/connective in place of $\neg$. Then, being a bit flexible and deviating from the previous prescription by replacing those symbols $\forall_x \exists_y \cdots \forall_v$ by $Q_1, Q_2, \cdots, Q_5$ respectively, we get

$$Q_1(\underline{x \in X} \leftrightarrow NQ_2NNQ_3N(\underline{y \in Y} \wedge \underline{z \in Z} \wedge Q_4(\underline{u \in x} \leftrightarrow (\underline{u \approx y} \vee Q_5(\underline{v \in u} \leftrightarrow (\underline{v \approx y} \vee \underline{v \approx z}))))))$$

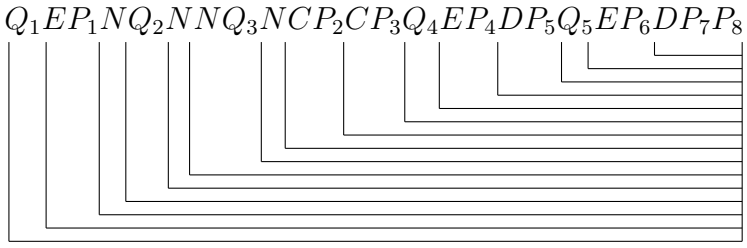
Above we underlined the atomic subformulae, and now we replace them by propositional variables $P_1, \cdots, P_8$. We also reinstall the associativity pair of brackets which hadn't been used—this involves a choice.

$$Q_1(P_1 \leftrightarrow NQ_2NNQ_3N(P_2 \wedge (P_3 \wedge Q_4(P_4 \leftrightarrow (P_5 \vee Q_5(P_6 \leftrightarrow (P_7 \vee P_8)))))))$$

Going to the first right bracket and replacing the infix notation $\vee$ subformula (which ends with that bracket) by a prefix notation D gives

$$Q_1(P_1 \leftrightarrow NQ_2NNQ_3N(P_2 \wedge (P_3 \wedge Q_4(P_4 \leftrightarrow (P_5 \vee Q_5(P_6 \leftrightarrow DP_7P_8))))))$$

Doing steps similar to the one immediately above several times, we finally arrive at the Polish version, and, then immediately apply the tub algorithm, to get



This formula is somewhat atypical—every tub ends at the right-hand end; that is, every occurrence of a subformula, except for atomic subformulae, ends at the right-hand end of the formula.

Remark. Having a formulahood checking (Type D) algorithm allows us to produce a (Type L) algorithm which lists all formulae. You take an algorithm which lists all strings, but divert the output to the formula checking algorithm, which discards the non-formulae, and only outputs the formulae. Using this listing, a more involved procedure then lists all derivations from a given premiss set, as long as that premiss set is decidable. See the beginning of this chapter.

9. ANCIENT AND ROMANTIC LOGIC

In this chapter we'll have some fun with arguments from ordinary life, first analyzing the arguments from the introduction of this book, then looking at the logic invented in ancient Greece, and its generalization and mathematicization by Boole about 150 years ago. All these turn out to be convertible into quite simple examples of 1$^{\underline{\text{st}}}$order logic. To finish the book proper with something more fundamental, there is a short section on Russell's famous paradox of the 'set of all sets which are not members of themselves'. It gives rise to an interesting example of logical validity.

9.1 The arguments in the introduction of the book.

For the penultimate one, about rattling cars, the word "some" in the second premiss, namely My car is some car, has nothing to do with the existential quantifier, which is what the same word in the first premiss means, so the argument is nonsense because of this. The second premiss can't really be translated into any 1$^{\underline{\text{st}}}$order language satisfactorily. Of course, it could all be a misunderstanding: Perhaps in France there exists the "Somé Auto Manufacturing Company", and the person transcribing Smullyan's argument had twice omitted the accent and capital letter, and tended to drop articles. Then the argument would be valid.

For first three arguments, we are thinking of an interpretation whose set is that of all humans: smoking, running, voting and making political speeches. But, when attempting to show that an argument is invalid, *any* interpretation can be used.

For the first argument, the language should have three unary relation (or 'predicate') symbols:

s, where sx is interpreted as "x is a smoker";

r, where rx is interpreted as "x is a marathon runner";

h, where hx is interpreted as "x is healthy, except possibly for conditions unrelated to smoking".

The argument is

$$\exists x\ (sx \wedge rx) \quad , \quad \forall x\ (rx \to hx)\ \ ;\ \ \text{therefore}\ \ \forall x\ (sx \to hx)\ .$$

Actually, the second premiss stated in the original verbal argument is probably stronger than the above, but the argument is evidently still invalid without the "conditions$\cdots$" phrase in that premiss.

In the exercises below (9.1 and 9.5), you show that the argument is invalid, but would be valid if $\forall$ is changed to $\exists$ in the conclusion. *Wishful thinking* (e.g. theology) *often takes this form*—using a universal quantifier when only an existential quantifier is warranted.

A stronger valid conclusion is $\exists x\ (sx \wedge hx)$. (See Exercise 9.3).

In any case, a subject concerning health as it relates to environment is always far too statistically complicated to be explained with classical logic alone and no probabilistic considerations. Of course the apologists for the tobacco industry thrive on the simplistic, but even then employ logically faulty arguments.

. . . curse Sir Walter Raleigh
he was such a stupid GIT!
J. Lennon/P. McCartney

Exercises

9.1 By giving a small finite interpretation, show that the above argument is not valid.

9.2 Let F and G be formulae in any $1^{\underline{\text{st}}}$order language.

(i) Show that $\{F \to G\} \models \forall xF \to \forall xG$. (It follows from the completeness theorem that $\{F \to G\} \vdash \forall xF \to \forall xG$.)

(ii) Find a derivation to show $\{F \to G\ ,\ \forall xF\} \vdash \forall xG$. Deduce (for a second time) that $\{F \to G\} \vdash \forall xF \to \forall xG$, using a slightly strengthened deduction lemma.

(iii) Can you find a derivation showing $\{F \to G\} \vdash \forall xF \to \forall xG$, perhaps by working through the proof of the deduction lemma with this example?

9.3 Show that the argument of 9.1 would become valid if we changed the conclusion to $\exists x\ (sx \wedge hx)$. (Find a derivation, or at least prove that one exists.)

Hints : Replace $\exists$ by its definition. Get from $rx \to hx$ to $sx \wedge rx \to sx \wedge hx$; and also from $\neg(sx \wedge hx) \to \neg(sx \wedge rx)$ to $(\forall x\neg(sx \wedge hx)) \to \forall x\neg(sx \wedge rx)$, by using 9.2. An alternative route would be to alter the previous exercise to showing $\{F \to G\} \vdash \exists xF \to \exists xG$.

9.4 Possibly an easier way to do the previous exercise, and closer to how we argue in everyday life, consider what must happen in an arbitrary model and show that

$$\{\exists x\ (sx \wedge rx)\ ,\ \forall x(rx \to hx)\} \quad \models \quad \exists x\ (sx \wedge hx)\ .$$

Of course, 9.3 with soundness proves 9.4 indirectly, and 9.4 with completeness proves 9.3 indirectly.

9.5 Show that the argument of 9.1 would become valid if we changed the $\forall$ in the conclusion to $\exists$.

For the second argument, the language should have a constant symbol b whose interpretation will be Bushworth, and three unary relation symbols:
v, where vx is interpreted as "x gets your vote";
m, where mx is interpreted as "x is a man";
h, where hx is interpreted as "x is honest".
The argument is

$$hb \wedge mb \quad , \quad \forall x\ (vx \to hx \wedge mx) \ \ ; \ \ \text{therefore} \ \ (\exists x\ vx) \to vb\ .$$

Remarks. (i) The word "should" occurring in the original verbal argument has been replaced here. That word has many shades of meaning, some of which make a sentence into more of a *command* than an *assertion*, and so, into an inappropriate sentence for this kind of logic.

(ii) The conclusion says : "If you vote, it is for Bushworth." Too strong would be vb, unless you add $\exists x\ vx$ i.e."you actually do vote", as a premiss. Another possible hidden premiss is $\forall x \forall y\ (vx \wedge vy \to x \approx y)$ i.e. "you can't vote twice!" But the argument remains invalid with it added.

Exercise 9.6 Show that this argument is not valid.

For the last one, the language can be almost the same as the previous, but we could dispense with the predicate m; i.e. the irrelevant sex (NOT gender!) of the candidate does not appear in the argument. Also we need the predicate c, interpreted as "is a candidate".
The argument is

$(hb \wedge cb) \wedge \forall x\ (hx \wedge cx\ \to x \approx b)\ ,\ \forall x\ (vx \to hx \wedge cx)\ ;$ therefore $(\exists x\ vx) \to vb\ .$

Exercise 9.7 Show that this argument is valid, perhaps by finding a derivation (or otherwise proving that a derivation exists) and applying the soundness theorem.
Hints: Since 'being honest' and 'being a candidate' never appear except conjuncted, it cleans things up a bit to use a single relation symbol j, 'is an honest candidate'. One can replace $\exists$ by its definition, and argue to get rid of universal quantifiers that it suffices to derive $vx \to vb$ from the premiss set $\{jb, jx \to x \approx b, vx \to jx\}$. And really the premiss jb is a red herring—the assumption that Bushworth is an honest candidate is beside the point—it is the assumption that *nobody else is* which is needed. Finally—and this is a 'rabbit out of a hat'—to derive a conclusion $F \to G$, it suffices to derive $F \to (F \to G)$, since $(P \to (P \to Q)) \to (P \to Q)$ is a tautology.

Exercise 9.8 Possibly easier than the previous exercise and closer to how we argue in everyday life, consider what must happen in an arbitrary model and show that

$$\{(hb \wedge cb) \wedge \forall x\ (hx \wedge cx\ \to x \approx b)\ ,\ \forall x\ (vx \to hx \wedge cx)\}\ \models\ (\exists x\ vx) \to vb\ .$$

As above, note that 9.7 and 9.8 are indirectly deducible from each other by Gödel's completeness theorem.

9.2 The syllogisms of Aristotle and his two millennia of followers.

Syllogisms are certain forms of argument in natural language in which there are always two premisses (and a conclusion). In English, each of these three must have one of the following forms.

All X is Y.

No X is Y.

Some X is Y.

Some X is not Y.

Of course, "some" means "at least one", not "at least two"!

The simplest example of a valid syllogism is the following.

(A) All P is M , all M is S ; therefore all P is S.

As a specific instance of this, which also illustrates that some of S, P and M may just consist of a single individual, one gets the famous argument below.

Socrates is a man. All men are mortal. Therefore, Socrates is mortal.

This is the initial example of an argument presented in the introduction to some logic books, guaranteed to impress many readers falsely that they won't be learning anything of practical importance.

An example of an invalid syllogism is the following.

(B) No P is M , some S is M ; therefore no P is S.

Invalidity seems pretty obvious here, but see the second Venn diagram in the next section if you need more convincing.

By always using P and S in that order in the conclusion, always using M and P in the first premiss (in one order or the other), and always using M and S in the second premiss (in one order or the other), it is not hard to count and see that the total number of possible syllogisms is

$$256 \;=\; (\text{“8” first premisses}) \times (\text{“8” second premisses}) \times (\text{“4” conclusions}) \,.$$

After Aristotle's introduction and masterful analysis of syllogisms, a great deal of blood, sweat and tears was expended over the following 2,000 or so years in memorizing this material, and devising schemes and nomenclature to help in the task. The much admired philosopher Kant expressed the opinion that nothing more than what Aristotle had done was possible in logic. [Presumably he is much admired for something other than being dead wrong in this opinion (and in the opinion that Euclidean geometry was the only possible geometry for physical space—see **Appendix G**). Once again I am undoubtedly being unfair to the philosophers, who will point out how much more subtle were Kant's views than the above characterization.]

> *Secondly, apart from two other rather short passages, the book gives no information about traditional Aristotelian logic, and contains no material drawn from it. But I believe that the space here devoted to traditional logic corresponds well enough to the small role to which this logic has been reduced in modern science; and I also believe that this opinion will be shared by most contemporary logicians.*
>
> Alfred Tarski (who didn't teach at St. Jerome's)

In the next section, we give a very simple method, Venn diagrams, for deciding the validity or otherwise of a syllogism, with little need for memorization.

In Aristotelian syllogistics, there was an additional assumption that each of S, P and M contained at least one individual. In terms of the Boolean analysis of the section after next, *'all basic sets are non-empty'* was the assumption. With this assumption, it turns out that exactly 24 of the 256 syllogisms are valid.

A Venn diagram method is possible for doing Aristotelian syllogistics, but from now on, the non-emptiness assumption will NOT be made. In this modern syllogistics, it turns out that only 15 of the 256 syllogisms are valid. A table on page 8 of **Burris** gives a neat way of actually listing which syllogisms are valid, in both of the senses.

The point that needs to be made here is that syllogistics can be regarded as a small corner of $1^{\underline{\text{st}}}$order logic. Here is a simple $1^{\underline{\text{st}}}$order language for doing that.

We take three unary relation symbols, s, p and m, (these are often called *predicate symbols* or just *predicates*) and no constant or function symbols. The four little sentences at the beginning of this section are now rewritten in this $1^{\underline{\text{st}}}$order language as follows.

$$\text{All S is P} \;:\; \forall x\,(sx \to px) \;=\; \forall x\,\neg(sx \wedge \neg px) \quad {}^{\text{treq}} \quad \neg\exists x(sx \wedge \neg px)\,.$$

$$\text{No S is P} \;:\; \forall x\,(sx \to \neg px) \quad {}^{\text{treq}} \quad \forall x\,\neg(sx \wedge px) \quad {}^{\text{treq}} \quad \neg\exists x(sx \wedge px)\,.$$

$$\text{Some S is P} \;:\; \exists x\,(sx \wedge px) \quad = \quad \neg\forall x\,\neg(sx \wedge px) \quad = \quad \exists x\,(sx \wedge px)\,.$$

$$\text{Some S is not P} :\; \exists x\,(sx \wedge \neg px) \;=\; \neg\forall x\,\neg(sx \wedge \neg px) \;=\; \exists x\,(sx \wedge \neg px)\,.$$

Note the symmetry and relative exhaustiveness of the (non-abbreviated) formulae in the middle, and particularly of the right-hand sides.

With enough elbow-grease, one can write down every possible syllogism in this $1^{\underline{\text{st}}}$order language and verify that the ones which are valid $1^{\underline{\text{st}}}$order arguments are exactly the 15 valid syllogisms in the modern sense. See also the small print after Exercise 9.12.

By adding the premisses $\exists x\ sx$, $\exists x\ px$, and $\exists x\ mx$, the Aristotelian sense can also be dealt with.

So the 20th century logic of this book is not just "another in a long line"; it pretty well subsumes anything of value that went before. The main figures in this transformation are Boole, Frege, Peano, Russell, Hilbert and Gödel.

9.3 Venn diagrams.

Forget about $1^{\underline{\text{st}}}$order for this section.

To quickly check validity of a syllogism, draw three discs so that every possible intersection contains some points. Label them P, S and M. With X and Y naming two of those three, now do shading and dotting as follows. (Shading tells you that there is nothing in the part shaded. Dotting tells you that there is at least one thing in the part dotted.)

For any premiss of the form 'All X are Y', shade completely that part of X not in Y.

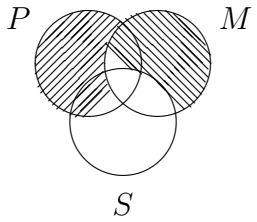
For any premiss of the form 'No X are Y', shade completely the intersection of X with Y.

For any premiss of the form 'Some X are Y', place a dot in the intersection of X with Y.

For any premiss of the form 'Some X are not Y', place a dot in that part of X not in Y.

Now to decide the validity of the argument, check whether the appropriate shading or dotting exists with respect to the conclusion.

Example (A) from the last section would produce the following Venn diagram.

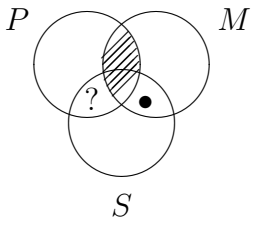


The shading 

comes from the premiss 'all P is M'.
The shading

comes from the premiss 'all M is S'.
Because all the part of P which is not part of S has now been shaded, we conclude that that syllogism, with conclusion 'all P is S', is indeed valid.

Example (B) from the last section would produce the following Venn diagram.

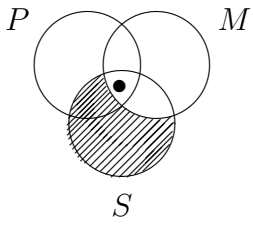


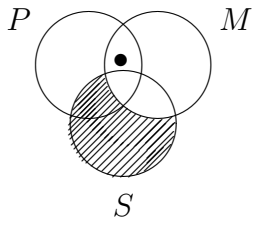
The shading comes from the premiss 'no P is M'. The dot comes from the premiss 'some S is M'. (Always do the shading, if any, before the dot(s), or you might waste time drawing an irrelevant diagram.) The question mark, inserted after dotting and shading, indicates why the conclusion, 'no P is S', does not follow. And so, this syllogism is not valid.

Note that when dotting is done, the region where the dot goes is divided in two, so sometimes you must be careful to consider both possibilities. This is illustrated in the following example.

(C) Some P is M , all S is M ; therefore some P is not S.

Here are the two corresponding Venn diagrams.





The diagram on the right shows the case where the conclusion does hold, but the other case, given by the diagram on the left, shows that this syllogism

is not valid—there is no dot in either part of 'P outside S'.

Note that we have no information one way or the other about any region which is neither shaded nor contains a dot.

It is all too easy for one to get impressed by how much humans have learned over the centuries, and to become a "know-it-all". A good antidote is to consider historical examples, like the introduction of Venn diagrams, which did not occur till the late 1800's. This was after 2,000 years of "blood, sweat and tears". It shows how a very simple possibility can exist, and be overlooked by generations of the best minds in a culture. What simple things are we overlooking that you might discover? So let's admire Aristotle, and go easy on Kant (though not necessarily on his less discriminating admirers).

Exercises

9.9 Use Venn diagrams to check the following arguments for validity.

(i) Some celebrities are not absent-minded.
Some celebrities are not professors.
Therefore, some professors are not absent-minded.

(ii) Some professors are not celebrities.
No celebrities are absent-minded.
Therefore, some professors are absent-minded.

(iii) Absent-minded people are not celebrities.
Some celebrities are professors.
Therefore, no professor is absent-minded.

(iv) All professors are celebrities.
No celebrities are absent-minded.
Therefore, no professors are absent-minded.

9.10 Convert each of the arguments in 9.9 into a $1^{\underline{st}}$order argument, and then use $1^{\underline{st}}$order methods to re-verify your decision about validity in the previous exercise.

9.4 Boole's reduction to algebra of 'universal supersyllogisms'.

Evidently there are lots of possibilities for having more than the three categories (P, M and S), but once you get up to five or more, Venn diagrams become pretty useless. If we stick to the first two types of statements, "all $\cdots$" and "no $\cdots$" (i.e. don't allow the two kinds of "some $\cdots$" statements), there is a method of Boole which is, in principle, easily used, but in fact, inefficient for large inputs.

(He also had a successful, but very hard to explain, method that dealt with "some $\cdots$". Recent scholars of the history of this who are also mathematicians (**Burris and Hailperin**) have shown how to make sense of this method. But we won't get into it.)

Boole's method is the algebra of sets. Each statement among the premisses and conclusion is rewritten in the form $F = \emptyset$, where F is a set theoretic expression involving union, intersection, and complementation. Then F is transformed into a new, equivalent expression. This transformation will be exactly analogous to the process of producing the disjunctive normal form in propositional logic. (This is more-or-less because we are only dealing with what would be universal prenex sentences, if converted to $1^{\underline{\text{st}}}$order. So by considerations similar to those in the penultimate section of Chapter 8 on clauses in $1^{\underline{\text{st}}}$order, the matter is effectively reduced to propositional logic.)

But let's stop this mysterious talk, and get on with some examples of Boole's method.

The simplest cases are the 16 syllogisms not involving "some $\cdots$" statements. Venn diagrams are better here, but this does make the point that Boole generalized Aristotle, in a sense.

Example (A) from before would be written by Boole as

$$PM' = \emptyset \quad , \quad MS' = \emptyset \quad ; \quad \text{therefore} \quad PS' = \emptyset \; .$$

To explain: the first equation is saying that the intersection of P with the complement of M is empty—clearly the same as saying P is a subset of M, or in the syllogism form, "All P is M". Note that we write PM' rather than $P \cap M'$ to economize.

Now Boole's method of algebraically checking that this argument is valid can be done, but we'll save its introduction for a more interesting example below.

Here is another of those 16 syllogisms.

No M is P. All M is S. Therefore no P is S.

Written in Boole's form, it becomes

$$PM = \emptyset \ , \ \ MS' = \emptyset \ ; \ \text{ therefore } \ PS = \emptyset \ .$$

Now for a more interesting (at least mathematically!) example.

Suppose given a group of people who are one or more of the following: actors, bores, clowns and/or dopes. Suppose also that we know the following three things:

All boring actors are either clowns or dopes.
No boring clowns are dopey non-actors nor acting non-dopes.
No dopey clowns are bores nor actors.

Then, I claim,

every boring actor is a dopey non-clown.

First let's translate this into set theoretic equations, letting A, B, C and D be, respectively, the sets of all actors, bores, clowns and dopes.

The first premiss gives $AB \subset C \cup D$ [which may be written

$$AB(C \cup D)' = \emptyset \ ] \ .$$

The second premiss gives $BC(AD' \cup A'D) = \emptyset$.

The third premiss gives $CD(A \cup B) = \emptyset$.

The conclusion gives $AB \subset C'D$ [which may be written $AB(C'D)' = \emptyset$].

So we now have each premiss plus the conclusion written in the form that Boole wants us to. Note that the "no $\cdots$" statements translate directly; whereas the "every $\cdots$" statements give a set inclusion $X \subset Y$, which can then be converted as $XY' = \emptyset$.

Before illustrating Boole's method for automatically checking the validity of such an argument, it is worth remarking that these (just like syllogisms) can be regarded as very simple $1^{\underline{\text{st}}}$order arguments. This one could be converted as follows, again with three premisses and a conclusion (here ax translates the formula "x is an actor", so a is a predicate, i.e. unary relation symbol, as are b, c, and d):

$$\forall x(ax \wedge bx \rightarrow cx \vee dx) \quad ,$$

[which is the same as $\forall x\ \neg(ax \wedge bx \wedge \neg(cx \vee dx))$] ;

$$\forall x\ \neg(bx \wedge cx \wedge ((ax \wedge \neg dx) \vee (\neg ax \wedge dx))) \quad ;$$

$$\forall x\ \neg(cx \wedge dx \wedge (ax \vee bx)) \quad .$$

Therefore

$$\forall x\ (ax \wedge bx \rightarrow \neg cx \wedge dx) \quad ,$$

[which is the same as $\forall x\ \neg(ax \wedge bx \wedge \neg(\neg cx \wedge dx))$] .

In fact, because they all use just a single universal quantifier whose scope is the entire formula, they can even be regarded as arguments in propositional logic, as follows :

$$P \wedge Q \rightarrow R \vee S \quad ,$$

[which is the same as $\neg(P \wedge Q \wedge \neg(R \vee S))$] ;

$$\neg(Q \wedge R \wedge ((P \wedge \neg S) \vee (\neg P \wedge S))) \quad ;$$

$$\neg(R \wedge S \wedge (P \vee Q)) \quad .$$

Therefore

$$P \wedge Q \rightarrow \neg R \wedge S \quad ,$$

[which is the same as $\neg(P \wedge Q \wedge \neg(\neg R \wedge S))$] .

Notice how intersection, union, complement and containment translate as conjunction, disjunction, negation and implication, respectively.

See the comments and exercises after Exercise 9.12 for more details about converting earlier logic into 20th century logic, as well as the equivalence of propositional logic to fragments of 1$^{\underline{\text{st}}}$order logic.

The above 1$^{\underline{\text{st}}}$order/propositional argument is valid, but it's perhaps easiest to use Boole's method to show that the set theory form of the argument is valid.

For that, you expand each of the left-hand sides as a union of constituents, where each such constituent is an intersection of four sets: A or A', B or B', C or C', and D or D'. The set theory identities used to do this are exactly

analogous to the ones used to convert a formula to disjunctive normal form, so we won't even list them, but just do the manipulations:

$\emptyset = AB(C \cup D)' = ABC'D'$, a union of (only one) constituent.

$\emptyset = BC(AD' \cup A'D) = ABCD' \cup A'BCD$.

$\emptyset = CD(A \cup B) = ACD \cup BCD =$
$A(B \cup B')CD \cup (A \cup A')BCD = ABCD \cup AB'CD \cup A'BCD$.

Therefore
$\emptyset = AB(C'D)' = AB(C'' \cup D') = ABC \cup ABD' =$
$ABC(D \cup D') \cup AB(C \cup C')D' = ABCD \cup ABCD' \cup ABC'D'$.

Now Boole's method finishes with a straightforward inspection: just check that each of the (three, in this case) constituents in the conclusion appears as a constituent in at least one of the premisses (which *does* happen in this case, so the argument *is* valid.) It is very easy to see that this condition on constituents is necessary and sufficient for an argument (of this set-theoretic kind) to be valid.

It is also interesting to note that, in this example, we can't omit any of the premisses, or the argument would become invalid. No two of the premisses contain all three of the constituents in the conclusion.

As explained below the next two exercises, this method has a very close analogue for checking validity of propositional arguments, when the premisses and conclusion are written as 'negated disjunctive forms'.

Exercises

9.11 The 109 year old nordic skier was asked about his diet. He replied as follows.

If I don't drink beer at supper, I always eat fish.

Whenever I have both beer and fish, I don't eat ice cream.

If I have ice cream but not beer, then I won't have fish.

During the next season, a competitor in his age category was finally able to beat him in a race, which was attributed to the competitor's new diet. That competitor was knowledgeable about logic, but had a poor memory, which is understandable at that age. (Don't remind me!)

Can you also simplify the first skier's diet, so that it becomes easy to remember?

9.12 In Exercise 2.22, take each of the four English arguments produced as an answer, invent an appropriate $1^{\underline{\text{st}}}$order language, translate that argument into the language, and then determine whether the argument is valid or not (presumably it will be). Possibly you may want to translate some of them also into Boolean form and use Boole's method.

Remarks on Boole vs. propositional vs. $1^{\underline{\text{st}}}$order logic. To follow up the remarks illustrating how Boole's method can be regarded as part of 20th century logic, here are some comments and exercises to show how it, and propositional logic, and a special class of formulae in a quite simple $1^{\underline{\text{st}}}$order language, are essentially all the same theory. This also relates to those syllogisms which use the "all" and "no" statement types, but not the latter two of the four (though the discussion could be expanded to do so).

We didn't specify Boole's logic in any really exact way. Let's assume that the part with which we're dealing only concerns itself with arguments expressed as a set of equations (premisses, plus one conclusion equation), where the equations use symbols $A_1, A_2, \cdots$. And that these equations are written as

$$\emptyset = [\text{ a finite union of finite intersections (written as juxtaposition) }] \ ,$$

where the sets intersected are some of the A_i and A_i'. Even the extreme case of equations like $\emptyset = A$ or $\emptyset = A'$ are allowed now. If A were the set of all ants, they would say, respectively, "Nothing is an ant" and "Everything is an ant". (I suppose the first is true from the viewpoint of a whale, and the second from the viewpoint of mitochondria inside an ant.)

Although we should be careful not to identify too closely these informal sets with the terms in the $1^{\underline{\text{st}}}$order language of set theory, one of the main features of Boole's method is that it is in fact purely formal, not requiring one to assign any properties to the symbols A_i, just to operate with them formally.

So to mimic Boole in $1^{\underline{\text{st}}}$order set theory, take the language of $1^{\underline{\text{st}}}$order set theory, but supplement it with extra constant symbols b and $B_1, B_2, \cdots$. Consider only formulae from this language which don't involve any quantifiers, any symbols $\approx$, nor symbols $\emptyset$, if it is part of the language. Also, in atomic formulae, we shall not allow b to occur immediately after the $\in$ symbol, nor any B_i immediately before it. So these formulae will be built using $\neg$ and $\wedge$ entirely from atomic formulae of the form $b \in B_i$. The idea for matching this with Boole is fairly obvious. For example, Boole's

$$\emptyset \ = \ A_1' A_3 \cup A_2 A_4' A_3$$

would correspond to the formula

$$\neg((\neg b \in B_1 \wedge b \in B_3) \vee (b \in B_2 \wedge \neg b \in B_4 \wedge b \in B_3)) \ .$$

Basically because of the following exercise,

Exercise 9.13 Prove that *every formula in propositional logic is truth equivalent to the negation of a formula in disjunctive form.*

we won't really be adding anything extra by allowing *any* 1st-order formula built as above, including abbreviations. For example, where Boole would convert

$$A_3 \subset A_1 \quad \text{to} \quad A_1' A_3 \;=\; \emptyset \;\;,$$

we could just leave it as

$$b \in B_3 \rightarrow b \in B_1 \;\;, \quad \text{rather than converting it to} \quad \neg((\neg b \in B_1 \wedge b \in B_3) \;\;.$$

The reader is probably already realizing that our 1st-order language is more complicated than necessary. We could simply change it to the language with a single constant symbol b, and an infinite sequence of unary relation symbols $p_1, p_2, \cdots$. (that is, *predicates*). For example the formula in the third display above would be changed to

$$\neg((\neg p_1 b \wedge p_3 b) \vee (p_2 b \wedge \neg p_4 b \wedge p_3 b)) \;\;.$$

So here the appropriate set of formulae consists of those without quantifiers or equality symbols.

But now, why not just go all the way back to propositional logic? After all, without quantifiers, we're hardly doing 'real' 1st-order logic above. To illustrate what we have in mind, we'd just replace the formula in the display immediately above by

$$\neg((\neg P_1 \wedge P_3) \vee (P_2 \wedge \neg P_4 \wedge P_3)) \;\;.$$

All this is fine, but if you look back at the examples in the text in both this section and that on syllogisms, where we attempt to illustrate the fact that all this older logic is really subsumed by 1st-order logic, the example being used just above would have been written

$$\forall x \neg((\neg p_1 x \wedge p_3 x) \vee (p_2 x \wedge \neg p_4 x \wedge p_3 x)) \;\;.$$

It is clear that, leaving aside this last class of formulae for the moment, there is a natural 1-1 correspondence between the set of all formulae in propositional logic, and each of the two sets of formulae as described above without quantifiers in the two 1st-order languages.

Furthermore, there is a 1-1 correspondence between the set of propositional logic formulae which are negated disjunctive forms, and the set of all Boole equations as described above. To finish dealing with the Boole language, do the following.

Exercise 9.14 Show that a propositional argument, in which the conclusion and all the premisses are negated disjunctive *normal* forms, is valid if and only if each of the DNF constituents in the conclusion (that is, in the formula which follows the negation sign in the conclusion) occurs in at least one of the premisses.

A very simple example is the obvious fact that

$$\{\; P \rightarrow Q \;,\; Q \rightarrow R \;,\; P \;\} \;\models\; R \;\;,$$

which in negated disjunctive, but not negated DNF, form may be written

$$\{\; \neg(P \wedge \neg Q) \;,\; \neg(Q \wedge \neg R) \;,\; \neg\neg P \;\} \;\models\; \neg\neg R \;\;.$$

The corresponding Boolean argument might be written

$$\{\ AB' = \emptyset\ ,\ BC' = \emptyset\ ,\ A' = \emptyset\ \}\ \models\ C' = \emptyset\ \ .$$

It might be arguing : Every ant is a bug. Every bug is a crustacean. Everything is an ant. Therefore, everything is a crustacean. (This argument should give biologists a good laugh!)

This ties in 20th century logic with Boole's actual method for checking validity. We'll stick to the former for the rest of this mini-section. Notice that, in agreement with the $\mathcal{P} \neq \mathcal{NP}$ conjecture, Boole's method does often produce a combinatorial explosion when the number of propositional variables P_i involved (or rather sets A_i) gets large.

Exercise 9.15 Show that both of the 1-1 correspondences (between the set of all propositional logic formulae and the two sets of quantifier-free formulae described above) preserve truth equivalence. That is, show that a pair of propositional formulae are truth equivalent in the sense of truth assignments if and only if the corresponding pair of quantifier-free $1^{\underline{st}}$order formulae are truth equivalent 'in the sense of interpretations'. (You might wish to organize your answer by mimicking the way in which 9.16 below is divided up, though both of these are easier than 9.16.)

It follows immediately that arguments in these three languages which correspond under the 1-1 correspondences of formula sets behave the same as each other with respect to validity. An invalid argument in one language can't correspond to a valid one in another.

To deal with the $1^{\underline{st}}$order formulae with a single universal quantifier at the front which were used in the main part of the text (but not the ones with a single *existential* quantifier in front which were needed for the latter two of the four syllogism statements), we should get a bit more explicit than above, as follows.

Firstly, let $\mathcal{F}$ denote the set of all formulae in propositional logic. And let $\mathcal{U}$ denote following the set of formulae in $1^{\underline{st}}$order logic from that earlier language with an infinite sequence $p_1, p_2, \cdots$ of predicates, and with no constant, function or 2-ary or higher relation symbols. Call the elements of $\mathcal{U}$ usents, and define them inductively, using a single variable x as follows:

$\forall x p_i x$ is a usent for each i;

If $\forall x F$ is a usent, then $\forall x \neg F$ is also;

If $\forall x F$ and $\forall x G$ are both usents, then $\forall x(F \wedge G)$ is also.

All usents arise from applying the above three rules finitely often.

Note that, as in the third rule, all of $\forall x(F \to G)$, $\forall x(F \vee G)$, and $\forall x(F \leftrightarrow G)$ will be abbreviated usents, as long as $\forall x F$ and $\forall x G$ are.

Note also that a usent is always a sentence, containing only the variable x, having no equality symbols, and only one (universal) quantifier (at the left-hand end), whose scope is the whole sentence.

The "u" in usent can stand for universally quantified, unique variable sentence with unary relation symbols only.

Now define maps back-and-forth as follows:

$$\mathcal{F} \to \mathcal{U} \quad \text{with notation} \quad G\ \mapsto\ G^{@}\ \ ;$$

$$\mathcal{U} \to \mathcal{F} \quad \text{with notation} \quad \forall x F \ \mapsto \ (\forall x F)' \ .$$

These are the maps where, for any 'expression', $\mathcal{E}$, built using $\neg$ and $\wedge$, the propositional formula $\mathcal{E}(P_1, P_2, \cdots)$ corresponds to $\forall x \ \mathcal{E}(p_1 x, p_2 x, \cdots) \in \mathcal{U}$. Thus they are evidently inverses of each other. But we are being fussy here, and learning good careful tedious style in logic, so define them inductively as follows:

$$P_i^{@} \ := \ \forall x \ p_i x \ ;$$
$$(\neg G)^{@} \ := \ \forall x \neg F \quad \text{if} \quad G^{@} = \forall x F \ ;$$
$$(G_1 \wedge G_2)^{@} \ := \ \forall x (F_1 \wedge F_2) \quad \text{if} \quad G_1^{@} = \forall x F_1 \text{ and } G_2^{@} = \forall x F_2 \ .$$

$$(\forall x \ p_i x)' \ := \ P_i \ ;$$
$$(\forall x \neg F)' \ := \ \neg (\forall x F)' \ ;$$
$$(\forall x (F_1 \wedge F_2))' \ := \ (\forall x F_1)' \wedge (\forall x F_2)' \ .$$

Exercise 9.16 Prove by induction that
(i) $G^{@\prime} = G$ for all $G \in \mathcal{F}$; and that
(ii) $(\forall x F)'^{@} = \forall x F$ for all $(\forall x F) \in \mathcal{U}$.
(This establishes the 1-1 correspondence claimed just above.)

We also want to show that the semantics correspond exactly.

(iii) Possibly using the hints below (vii), prove the following, where $\overset{\text{treq}_P}{}$ denotes truth equivalence in the sense of propositional logic, and $\overset{\text{treq}_1}{}$ denotes truth equivalence in the sense of $1^{\underline{\text{st}}}$order logic.
(a) $G_1 \ \overset{\text{treq}_P}{} \ G_2 \iff G_1^{@} \ \overset{\text{treq}_1}{} \ G_2^{@}$;
(b) $\forall x F_1 \ \overset{\text{treq}_1}{} \ \forall x F_2 \iff (\forall x F_1)' \ \overset{\text{treq}_P}{} \ (\forall x F_2)'$.
(iv) Deduce from (iii) that G is a propositional tautology if and only if the corresponding usent $G^{@}$ is logically valid; and then, $G^{@} = \forall x F$ for a $1^{\underline{\text{st}}}$order tautology F.
(vi) Show that an argument in propositional logic is valid if and only if the corresponding argument involving usents is valid (in either of the senses $\models$ or $\models^*$, there being no difference for sentences).
(vii) Show that a set of propositional formulae is satisfiable if and only if the corresponding set of usents is $\models$-satisfiable (or $\models^*$-satisfiable, since it makes no difference for sentences).

Here are some suggestions for doing (iii).
Firstly, show easily, using (i) and (ii), that (a)$\Rightarrow$ implies (b)$\Leftarrow$, and (b)$\Rightarrow$ implies (a)$\Leftarrow$. So we need only prove the $\Rightarrow$-halves.
For (b)$\Rightarrow$, given a truth assignment e, define an interpretation $\mathcal{V}_{\mathsf{e}}$ to be just the singleton set $\{v\}$, with the predicate symbols interpreted by saying that p_i is true at v (that is, for any variable x, $p_i x$ is true at the unique $\underline{v}$) if and only if $\mathsf{e}(P_i) = \mathsf{Tr}$. Then prove by induction that for all usents $\forall x F$, that usent is true at $\underline{v}$ if and only if $\mathsf{e}((\forall x F)') = \mathsf{Tr}$. Now use this last fact to quickly finish.
For (a)$\Rightarrow$, recall that $G_1 \ \overset{\text{treq}_P}{} \ G_2$ if and only if $G_1 \leftrightarrow G_2$ is a propositional tautology. Referring to Section **6.6** on $1^{\underline{\text{st}}}$order tautologies, it follows that $F_1 \leftrightarrow F_2$ is a $1^{\underline{\text{st}}}$order

tautology, where $\forall x F_i = G_i^{@}$. Now quite generally in 1$^{\underline{\text{st}}}$order, show that $F_1 \leftrightarrow F_2$ being logically valid implies that $\forall x F_1 \leftrightarrow \forall x F_2$ is logically valid. (The converse is false, as it happens—see the next exercise.) The required result now follows immediately from Proposition **8.1**.

Exercise 9.17 In both (i) and (ii) below, show that the formulae G and H are such that $\forall x G \leftrightarrow \forall x H$ is logically valid, but $G \leftrightarrow H$ is not. Equivalently, show that $\forall x G \ ^{\text{treq}_1} \ \forall x H$, but G and H are not truth equivalent. See the last section of Chapter 6 for related material.

(i) Choose any 1$^{\underline{\text{st}}}$order language and formula F from it for which $F \to \forall x F$ is not logically valid. Then take $G := F$ and $H := \forall x F$.

(ii) Take the language of ring theory. Let R be a sentence which gives all the axioms defining the word *ring*—for example, R could be the conjunction of all the usual axioms, suitably expressed in 1$^{\underline{\text{st}}}$order. Let $G := R \wedge x \times x \approx x$ and $H := R \wedge x \times x \approx x \wedge x + x \approx 0$. (Formulae such as this last one give ammunition to authors who believe in using brackets around every atomic formula!)

9.5 Russell's paradox.

Frege, in the late 1800's, constructed a logical system which was supposed to reduce all of mathematics to pure logic, in some sense. This was a magnificent attempt, and gave rise to a great deal of modern logic. It would have been a system much grander than 1$^{\underline{\text{st}}}$order logic, including both it and a series of higher order logics which we won't attempt to describe.

Around 1900, Bertrand Russell discovered that Frege's system was inconsistent. It allowed one to construct 'the set of all sets which are not members of themselves'. Informally, the contradiction, or so-called paradox, arises as follows. Name that 'set' R, so it's supposed to be a set with the property

$$\text{for all } x, \ [\ x \in R \iff x \notin x\ ].$$

Now either R is a member of itself, or it isn't a member of itself. However,

$$R \in R \implies R \notin R\ ,$$

and

$$R \notin R \implies R \in R\ .$$

So both possibilities lead to a contradiction.

This famous observation can be formulated as a particularly interesting logically valid formula existing in any 1$^{\underline{\text{st}}}$order language which has a constant symbol R, and a binary relation symbol, which we'll call b (rather than $\in$)

and write with infix notation. We just mimic Russell's definition to produce a sentence which is nowhere true. But since we don't want to be "nattering nabobs of negativity" (*Spiro T. Agnew—vice-presidential jailbird*), and are unsatisfied with always talking about the unsatisfiable, we then negate that Russell sentence to produce a logically valid sentence. Here it is:

$$\neg\forall x(xbR \leftrightarrow \neg xbx) \ .$$

It is a good exercise, with Tarski's definitions in Chapter 6, to check that

$$\underline{v} \ \in \ (\neg\forall x(xbR \leftrightarrow \neg xbx))^V$$

if and only if, for some $u \in V$, either

$$(u, R^V) \in b^V \quad \text{and} \quad (u, u) \in b^V$$

or

$$(u, R^V) \notin b^V \quad \text{and} \quad (u, u) \notin b^V \ .$$

But, by letting $u = R^V$, the condition in the double display just above clearly holds. This shows, as required, that the negated Russell sentence above is logically valid.

The reason I decided to change the formal $1^{\underline{\text{st}}}$order symbol $\in$ to b above was that we needed to use $\in$ in the sense of intuitive set theory in the display immediately above. If that had been written to include something like

$$(u, u) \ \in \ \in^V$$

[using $\in$ in the two different ways right next to each other], readers might start thinking about that notational point, rather than concentrating on the real issue here!

It is probably fair to say that one of the surprising things about $1^{\underline{\text{st}}}$order languages is our ability, with a bit of help from our ingeneous ancestors, to produce short simple sentences like this which are logically valid. And yet they are very far from obviously tautological in the informal sense of that word.

Here is a challenge for you. By Gödel's completeness theorem, there is a derivation of the negated Russell sentence above, using no premisses. Find one!

The introduction of *defined* (or *short form*) constants, functions and relations, as opposed to the basic special symbols of the language, is a prominent theme in more advanced treatments of $1^{\underline{st}}$order logic aimed at foundational studies. For example, in set theory in Chapter 5, we mentioned introducing union, intersection, containment, cartesian product, (and even the empty set as a constant, if $\in$ is taken as the only special symbol).

In the case of constants and functions, these definitions make use of existence statements. Russell's example shows that this is not an entirely straightforward matter. From the point of view of 'pure' set theory, going back to $\in$ rather than b, it's probably better to change the negated Russell sentence to

$$\neg\exists y\forall x(x \in y \leftrightarrow \neg x \in x) \ .$$

Exercises

9.18 Show that the formula just above is also logically valid, and find a specific derivation.

9.19 Do the exercise suggested several paragraphs above with Tarski's definitions.

Appendix G : 'NAIVE' AXIOM SYSTEMS IN GEOMETRY

The main point of this appendix is to give you some appreciation of what is involved in *non-Euclidean geometry*, a discovery which occurred during the 1800's, and which has had considerable influence on logic and mathematics, and on physics and philosophy as well. This is written so as to be completely independent of the rest of the book, and in particular, written from a 'naive' point-of-view, which pays no attention to formalizing the language. What we have called a set of premisses earlier occurs here under the name "axiom system". These are statements specific to the subject being studied, geometry or numbers, and must not be confused with the logical axioms from the proof systems earlier in the book.

First we'll need to discuss by example some general ideas about axiom systems. Most of these systems will be from synthetic geometry, but a system for the real numbers will also come up. A good reference for the geometry is **M.J. Greenberg**: *Euclidean and Non-Euclidean Geometries*. Another masterful one, containing the actual axioms we use, plus a lot more geometry, is **H.S.M. Coxeter**: *Introduction to Geometry*.

1. Incidence Geometry, illustrating the concepts of Consistency, Independence and Categoricity.

A *system* or *theory* is given axiomatically by first naming some objects and some relations between the objects, and then listing some axioms which the relations must obey. An example of a specific theory is *Incidence Geometry*, as follows.

Incidence Geometry

Two types of objects: points and lines .

One relation: incidence, relating some points to some lines. The sentence "*the point P is incident with the line ℓ*" is usually shortened to "*P is on ℓ* " or to "*ℓ passes through P* ".

Four axioms:

Axiom 1. *If P and Q are points, then there is a line through both P and Q.*
Axiom 2. *Every line passes through at least two points.*
Axiom 3. *If $\ell_1 \neq \ell_2$, then there is at most one point on both ℓ_1 and ℓ_2 .*
Axiom 4. *There exist ℓ and P such that P is not on ℓ.*

Simple Theorems of Incidence Geometry

1.1 *There are at least three points in an incidence geometry.*

Proof. By Axiom 4, choose ℓ, and P not on ℓ. By Axiom 2, there are at least two points Q and R on ℓ. Then P, Q and R are three distinct points.

1.2 *For every line, there is at least one point not on the line.*

Proof. Let m be any line. Let ℓ and P be as in Axiom 4. If $m = \ell$, then P is a point not on m. If $m \neq \ell$, then ℓ and m have at most one point in common by Axiom 3. But there are at least two points on ℓ, so one of them is not on m.

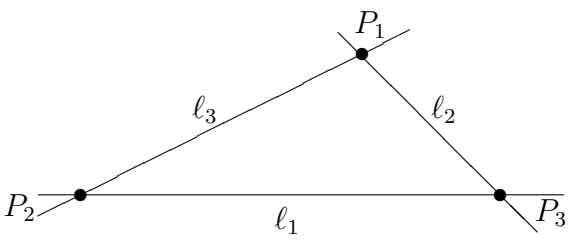
1.3 *For every point, there is at least one line not passing through it.*

1.4 *For every point P, there are at least two lines passing through P.*

1.5 *There exist three lines such that no point is on all three.*

Exercise G1.1 Prove **1.3**, **1.4** and **1.5** (using only the axioms !)

Models for a system. A model for a given axiom system is any "specific" or "concrete" system which satisfies the axioms; i.e. the objects and relations are specified in some particular way so that the axioms hold. For example, the picture below gives a *finite* model for incidence geometry. There are three points P_1, P_2 and P_3, and three lines ℓ_1, ℓ_2 and ℓ_3. Incidence is defined by having P_i on ℓ_j when and only when $i \neq j$. This is certainly the smallest of all models, by the above theorems. *Note that there are only three points; the other ink molecules or whatever on the lines in the picture are not supposed to denote points in the model.*



Consistency of a system. An undesirable possibility with respect to an axiom system is the possibility of being able to deduce some statement as a theorem and also to deduce the negation of that statement. Such a system is called *inconsistent*, and is seldom of interest. (A logician could show you that, within such a system, *every* sentence is a deducible theorem. See Theorem **3.11** earlier in the book.)

How might we show that incidence geometry is consistent? Note that each deducible theorem will be a property of every model. Since it is clear that our previous model cannot both have some property and have the negation of that property, incidence geometry must be consistent. In general, **to show that a system is consistent, one would like to produce at least one model constructed out of entities in whose 'consistency' one already believes.**

Axioms and models using the language of sets

This gets rid of some of the vagueness in the previous discussion. Much of axiomatic mathematics can be put in this form. In general, the objects will lie in sets and the relations will be subsets of Cartesian products. For example, an incidence geometry is now defined to be an ordered triple $(\mathcal{P},\ \mathcal{L},\ \mathcal{I}\ )$, where $\mathcal{P}$ and $\mathcal{L}$ are sets (whose elements it is convenient to call "points" and "lines" respectively), and $\mathcal{I}$ is a subset of $\mathcal{P} \times \mathcal{L}$. We often say " P is on ℓ " or "ℓ passes through P " in place of "$(P, \ell) \in \mathcal{I}$ ". Of course we also require the axioms to hold.

For example, Axiom 1 in set language is:
$\forall P \in \mathcal{P} \quad \& \quad \forall Q \in \mathcal{P}, \quad \exists \ell \in \mathcal{L} \quad \textit{such that} \quad (P, \ell) \in \mathcal{I} \quad \& \quad (Q, \ell) \in \mathcal{I}$.

Exercise G1.2 Translate the other three axioms to set language.

In set language, our previous model is given by taking any two sets

$$\mathcal{P} = \{\ P_1\ ,\ P_2\ ,\ P_3\ \} \quad \text{and} \quad \mathcal{L} = \{\ \ell_1\ ,\ \ell_2\ ,\ \ell_3\ \}\ ,$$

each having exactly three members, and letting

$$\mathcal{I} = \{\ \ (P_1, \ell_2)\ \ ,\ \ (P_1, \ell_3)\ \ ,\ \ (P_2, \ell_1)\ \ ,\ \ (P_2, \ell_3)\ \ ,\ \ (P_3, \ell_1)\ \ ,\ \ (P_3,\ \ell_2)\ \ \}.$$

or, more compactly

$$\mathcal{I} = \{\ \ (P_i, \ell_j)\ \ :\ \ i \neq j\ \ \}.$$

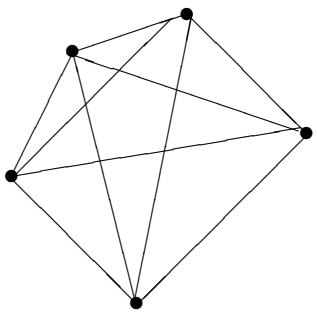
Since people who distrust the consistency of finite set theory tend to end up in lunatic asylums rather than universities, you will no doubt be convinced that the axioms of incidence geometry cannot lead to a contradiction.

Independence. In general, a statement is *independent* of an axiom system if it can't be deduced from the axioms. Consider the following statement:

Possible Axiom 5. *If the point P is not on the line ℓ, then there is* **at most** *one line m such that P is on m but no point is on both m and ℓ;*

i.e. $\cdots$ *there is at most one parallel, m, to ℓ through P*

How could we show P.Ax.5 to be independent of the axioms of incidence geometry? **By exhibiting a model of incidence geometry where** P.Ax.5 **is false!** Such a model is pictured as follows :



i.e. 5 points and 10 lines, with exactly 2 points on each line.

Exercise G1.3 Express this model in set language and verify carefully the four axioms.

Exercise G1.4 Formulate P.Ax.5 and the negation of P.Ax.5 in set language, and show that the above model satisfies the negation of P.Ax.5 .

Exercise G1.5 (i) Generalize our two models by finding a model of incidence geometry with exactly " n " points for any $n \geq 3$.

(ii) What is the maximum number of lines in a model with " n " points?

The fact that the parallel postulate is independent of the axioms of incidence geometry is hardly exciting. But the general set up, when we come to Euclidean geometry in the section after next, will be the same: we'll show that *the parallel postulate is independent of the system consisting of all the other axioms for Euclidean geometry* (axioms which are more numerous and complicated than those for incidence geometry). The models which achieve this proof are concrete versions of what is called *hyperbolic geometry.*

Given any system, it is desirable to know that each axiom is independent of all the others. If some axiom is not, that axiom is clearly redundant, and can be dropped from the list of axioms without changing the theory, since that axiom can be deduced from the others.

In general, an axiom is shown independent of the others in some system as follows: **Exhibit a model for that new system which consists of all the other axioms plus the negation of the given axiom.** For example,

pictures a model for the system [Ax.1 , Ax.2 , Ax.3 , NOT Ax.4], a model which has one line and two points only. This shows that Ax.4 is an independent axiom in incidence geometry.

Exercise G1.6 Show that each of the other three axioms is independent (use pictures if you wish; set language is tedious and not really necessary in such a simple situation).

Categoricity You'll have noticed that incidence geometry has several "essentially different" models. In fancy language, it is *non-categorical.* An axiom system is *categorical* if and only if any two models can be put into one–to–one correspondence so that the relations are preserved i.e. if and only if any two models are *isomorphic.* This is easily expressed in set language. For example, in incidence geometry, two models $(\mathcal{P}_1,\ \mathcal{L}_1,\ \mathcal{I}_1\ )$ and $(\mathcal{P}_2,\ \mathcal{L}_2,\ \mathcal{I}_2\ )$ are isomorphic iff there are bijective maps

$$\alpha\ :\ \mathcal{P}_1 \to \mathcal{P}_2$$

and

$$\beta \ : \ \mathcal{L}_1 \to \mathcal{L}_2$$

such that $(P, \ell) \in \mathcal{I}_1$ if and only if $(\alpha(P), \beta(\ell)) \in \mathcal{I}_2$ Since models where $\mathcal{P}_1$ and $\mathcal{P}_2$ have different numbers of points clearly can't be isomorphic, incidence geometry is certainly *not* categorical.

Exercise G1.7 (a) Find two non-isomorphic models for incidence geometry where there are four points in both cases.

(b) Are there non-isomorphic models where both the number of points and the number of lines is the same in both models?

The usual model of the plane as the set of all pairs (x, y) of real numbers (and with lines defined as usual by linear equations as in elementary analytic geometry) is an example of an *infinite* model for incidence geometry.

Exercise G1.8 Find a model for incidence geometry in which both $\mathcal{P}$ and $\mathcal{L}$ are countable sets, i.e. infinite sets which can be written down as sequences (indexed by 1, 2, 3, $\cdots$ in the usual notation for sequences).

Exercise G1.9 Is there a model where $\mathcal{P}$ is infinite but $\mathcal{L}$ is finite?

Categoricity brings out two distinct aspects of axiomatic mathematics:

Categorical systems, such as Peano's axioms for the natural numbers and the axioms for the real numbers (both in the section after next), and the axioms for Euclidean geometry (the section after that), tend to put our knowledge of these very important systems in a small package from which a great deal of mathematics follows by purely logical means, i.e., by deduction. It is, however, a serious question in logic whether every true statement about the model can necessarily be deduced from the axiom system. See also the "Subtle Question" a few pages ahead in the subsection "**Consistency of Euclidean Geometry**."

On the other hand, *non-categorical* systems with lots of models, such as incidence geometry, are a "good thing" because each theorem which is deducible in the system holds for every model. So in this case the axiomatic method exhibits its power for generality. Modern algebra particularly is involved mainly with this latter type of system.

An example is **group theory**: A group is a set G together with a multipli-

cation

$$\mu \ : \ G \times G \longrightarrow G$$

[where we write $\mu(a, b)$ as $a \star b$], such that the following axioms hold:

G1. *$(a \star b) \star c = a \star (b \star c)$ for all a, b, c in G.*

G2. *There is an $e \in G$ such that $e \star a = a \star e = a$ for all $a \in G$.*

G3. *For all $a \in G$, there is a $b \in G$ such that $a \star b = b \star a = e$.*

Other important systems occurring in algebra have names like rings, fields, vector spaces, lattices, modules, etc.

2. Order Geometry

In intuitive Euclidean geometry (and in everyday physical life), we certainly have a notion of "in-betweenness". Intuitively, point B is between points A and C, if and only if B lies on the interior of the line segment joining A to C. Physically, B would block light shining from a point source at A from the view of a point observer at C. One of the later discovered gaps in Euclid's original treatment of geometry (which was the first instance historically of the axiomatic method) was that he used this "in-betweenness" notion without ever mentioning explicitly exactly what basic properties (i.e. axioms) he was using concerning it.

We shall write this relation using the notation $A : B : C$.

Order geometry is the theory in which

(i) there is only one kind of undefined object, called point (and we'll denote points by A, B, C etc. . . , lines will be **defined** below);

(ii) there is one kind of relation, denoted $A : B : C$ (notice that the relation is *ternary*, not binary); and

(iii) the following seven axioms are satisfied.

Axiom O1. *If $A \neq B$, then there exists C with $A : B : C$.*

Axiom O2. $A : B : C \implies A \neq C$.

Axiom O3. $A : B : C \implies C : B : A$.

Axiom O4. $A : B : C \implies$ *not* $B : C : A$.

Definition. If $A \neq B$, then the line joining A to B is the set of points

$$\{C \ | \ C : A : B \ \text{ or } \ C = A \ \text{ or } \ A : C : B \ \text{ or } \ C = B \ \text{ or } \ A : B : C \ \} \ ,$$

and is denoted $\ell(A,\ B)$.
Note that if we took $A = B$ in the definition of $\ell(A,\ B)$, we'd get a singleton set $\ell(A,\ A) = \{A\}$ which will not be called a line for obvious reasons. A line will be any set which has the form $\ell(A,\ B)$ for a pair of **distinct** points A and B.

Axiom O5. *If* $A \neq B$, $B : C : D$ *and* $C : E : A$, *then*

$$\exists\ F \in \ell(D,\ E) \quad \text{satisfying} \quad A : F : B.$$

Axiom O6. *There are at least two distinct lines.*

Axiom O7. $A : B : C \quad \Rightarrow \quad \ell(A,\ B) = \ell(A,\ C)$

Exercise G2.1 Show how a model for order geometry gives a model for incidence geometry.

Exercise G2.2 Show that any model for the above axiom system has infinitely many points.

Exercise G2.3 Can you find a model (for order geometry) quite different from (in particular, not isomorphic to) the usual plane, $\mathbf{R}^2$, used in high school analytic geometry?

3. Axioms for Number Systems

We shall be very brief here.

Peano based the *natural numbers* $\{1, 2, 3, \cdots\}$ (for convenience, here we use the "high school teacher's natural numbers"—we don't include 0) on an axiomatic system consisting of a set $\mathbf{N}$ and a function $s : \mathbf{N} \longrightarrow \mathbf{N}$. The latter is called the *successor function*, and is the only relation needed. The intuitive idea is that s is the function which 'adds 1'.

Peano's Axioms

P1. *The function s is injective.*
P2. *The function s is not surjective.*
P3. *There exists an element $l \in \mathbf{N}$ such that if A is a subset of $\mathbf{N}$ containing l and containing the successor of each of its members, then $A = \mathbf{N}$.*

Axiom P3 is the *axiom of induction.*

The main point here is that it is possible to start with a model for the above axioms (which surely exists), and build up a model for the following axiom system for the set, $\mathbf{R}_{>0}$, of *positive real numbers.* Many books on real analysis begin by doing something similar. The relations in this system are $+$, $\cdot$, and $<$.

Axioms for the Positive Reals

A_+ $\quad (a+b)+c = a+(b+c)$

$\mathsf{A}_\bullet$ $\quad (a\cdot b)\cdot c = a\cdot(b\cdot c)$

C_+ $\quad a+b = b+a$

$\mathsf{C}_\bullet$ $\quad a\cdot b = b\cdot a$

D $\quad a\cdot(b+c) = (a\cdot b)+(a\cdot c)$

U $\quad 1\cdot a = a$

I *Given a and b, exactly one of $a<b$, $a=b$, $b<a$ holds.*

II *If $a<b$ and $b<c$, then $a<c$.*

III $a<b \iff \exists d$ *with* $a+d=b$.

IV *For each a, there exists b such that $ab=1$.*

V (Completeness). *If $a_1<a_2<a_3<\cdots$ is an increasing sequence, and $\exists$ u with $a_i<u$ for all i, then $\exists$ ℓ such that:*

i) $a_i<\ell$ for all i ; and

ii) if $a_i<m$ for all i, then $\ell\le m$.

In other words: **an increasing sequence with an upper bound has a least upper bound.**

This construction of $\mathbf{R}_{>0}$ from $\mathbf{N}$ can be done in two steps. First introduce the positive rationals $\mathbf{Q}_{>0}$ as fractions, define the order relation and arithmetic operations in $\mathbf{Q}_{>0}$, and prove some basic properties. Then define a positive real to be an equivalence class of bounded, strictly increasing sequences of positive rationals. The equivalence relation is easy to describe: two sequences are equivalent if and only if they keep passing each other. It is actually quite easy to prove the completeness property, by using the diagonal in a sequence of sequences.

Going on to produce the usual set of all real numbers is now just a matter of producing a second copy to serve as the negatives, and of introducing 0.

In the next section we shall build a model for non-Euclidean geometry, using a certain set of ordered pairs of real numbers. This will show that

the parallel postulate is an independent axiom of Euclidean geometry. Then there will be no doubt that **any claimed 'proof' of the parallel postulate amounts to no less than the scarcely believable claim that Peano's postulates have no model, so the natural numbers don't exist!**

The last axiom above, completeness, has an analogue in geometry, namely Axiom O8 in the next section, which is an important addition to our understanding of geometry since the time of Euclid.

4. Independence of the Parallel Postulate, which is equivalent to the Consistency of Hyperbolic Geometry

A rough idea of what we shall do may have occurred to you by now. We'll give a list of axioms for Euclidean geometry. One axiom is the parallel postulate. To prove its independence, we find models (actually several apparently different, but not really different, models) which satisfy all the other axioms, as well as the negation of the parallel postulate. This is due to Beltrami, Klein and Poincaré. The non-Euclidean geometry for which these are models is called *hyperbolic geometry*. Briefly at the end is some explanation about other non-Euclidean geometries; axioms other than the parallel postulate fail for them. Gauss, Bolyai and Lobachevski were the first to study hyperbolic geometry, and to 'believe in' its consistency. Others had, in effect, been studying it earlier, when trying in vain to give a proof by contradiction for the parallel postulate.

The term *non-Euclidean geometry* is ambiguous. Sometimes it is used to mean strictly hyperbolic geometry; sometimes either elliptic or hyperbolic geometry; and sometimes it is used more literally to mean any geometry except possibly Euclidean geometry.

These geometries are richer than incidence geometry, in that they talk about basic relations both of in-betweenness (or "order") and of congruence (or "comparative length"), as well as incidence. **Hilbert's** book "Grundlagen der Geometrie" gave the first modern thorough treatment of these geometries, including analysis of independence and categoricity. He (and **Greenberg**) take these three relations as undefined, with points and lines as undefined objects (also planes, but he did 3-dimensional geometry and we'll stick to 2-dimensional.) We'll follow **Coxeter**'s book. By making the axioms a bit more complicated, we need only two relations:

the order relation, $A : B : C$, which is read "B is between A and C";

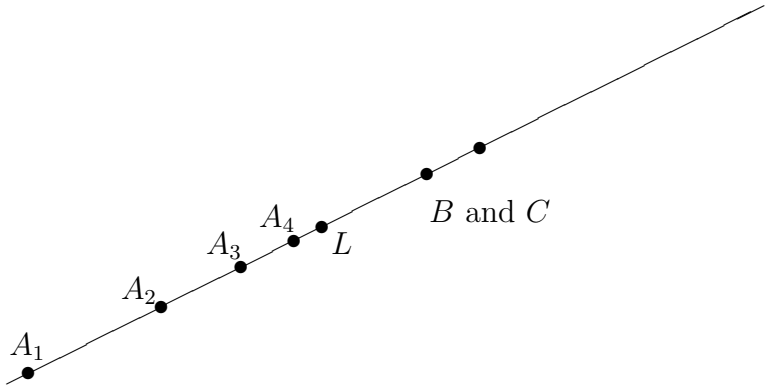
and the congruence relation, $AB \equiv CD$, which reads " segment AB is congruent to segment CD ";

with only points as undefined objects.

A line is defined to be a set of points as in order geometry, and our axioms of order are its axioms, O1 to O7, plus O8 and O9 below.

Axiom O8. *Suppose that the sequence* $A_1,\ A_2,\ A_3,\ \cdots$ *of points satisfies:*
(i) $A_1 : A_i : A_{i+1}\ \ \forall i > 1$ *, and*
(ii) $\exists B$ *such that* $A_1 : A_i : B\ \ \forall i > 1$.

Then $\exists L$ *such that :*
(a) $A_1 : A_i : L\ \ \forall i > 1$ *, and*
(b) $\forall C\ \ (\ A_1 : A_i : C\ \ \forall i > 1 \quad \Longrightarrow \quad$ either $C = L$ or $A_1 : L : C\ )$.



Note. Axiom O8 is analogous to the completeness axiom V for $\mathbf{R}_{>0}$. One needs it, for example, to show that a line segment, which joins a point inside a given ellipse to a point outside, must intersect the ellipse.

Definition. If $A,\ B,\ C$ are not all on one line, the *plane* defined by them is the union of all lines $\ell(D, E)$, as D and E range over the lines $\ell(A, B),\ \ell(B, C)$ and $\ell(A, C)$.

Axiom O9. *All points are in one plane.*

This is the axiom which makes sure that we are studying 2-dimensional geometry, not a higher dimension.

Axioms of Congruence.

C1. *$AB \equiv BA$ for all points A and B .*

C2. *$AB \equiv CD$ and $CD \equiv EF \implies EF \equiv AB$.*

C3. *If $A \neq B$ and $C \neq D$, then $\exists! E$ with*
(i) $AB \equiv CE$ and
(ii) $C : E : D$ or $E = D$ or $C : D : E$.
(i.e. given $A \neq B$, on any *ray* from C there is a unique E with $CE \equiv AB$.)

C4. *If $(A : B : C\ ,\ A' : B' : C'\ , AB \equiv A'B'$, and $BC \equiv B'C')$,*
then $AC \equiv A'C'$.
(i.e. 'adding' congruent intervals to congruent intervals yields congruent intervals.)

C5. *The consequence $AD \equiv A'D'$ follows from the following set of six assumptions: $B : C : D\ ,\ B' : C' : D'$,*
$AB \equiv A'B'\ ,\ BC \equiv B'C'\ ,\ AC \equiv A'C'$ and $BD \equiv B'D'$.

Exercises G4.1 Use these to show
(i) $AA \equiv BB \quad \forall A\ ,\ B$.
(ii) $AA \equiv BC \implies B = C \quad \forall A\ ,\ B\ ,\ C$.

G4.2 Draw diagrams illustrating O5, O8, C3, C4, C5 (all cases!).

G4.3 Using only C1 and C2, show that :
(i) $AB \equiv AB$ for all A, B ;
(ii) $AB \equiv CD \implies CD \equiv AB$.

G4.4 Using only O1 to O4, show that, for three points A, B, C, either none or exactly two of the six possible ordering relations hold for them.

G4.5 Using O1 to O7, show that $A : B : C \implies B \neq C$.

G4.6 Using only O1 to O7 try to prove

Sylvester's Theorem: *Any finite set of points which intersects no line in exactly two points is a subset of some line. (i.e. If $n \geq 2$ points are not collinear, some line contains exactly two of them).*

The system determined by O1 to O9 plus C1 to C5 is called **neutral geometry** or **absolute geometry**. Its theorems are exactly those theorems common to Euclidean and hyperbolic geometry. In particular one can :

define perpendicularity of two lines ;

prove that perpendiculars exist ; and

use the perpendicular to a perpendicular to a given line to show the existence of at least one parallel to a line through a point.

Euclidean geometry (2-dimensional) is neutral geometry plus one extra axiom:

YESPP. *For all points C and lines ℓ, if $C \notin \ell$, there is at most one line m such that $C \in m$, but m and ℓ have no points in common.*

(Hence there is *exactly* one such line m).

Hyperbolic geometry (2-dimensional) is neutral geometry plus the extra axiom:

NOPP. *There exists a point C and lines ℓ , m_1 and m_2 such that*

$$m_1 \neq m_2 \ , \ C \notin \ell \ , \ C \in m_1 \ , \ C \in m_2 \ ,$$

but ℓ has no points in common with either m_1 or *m_2.*

Note. YESPP *is* the parallel postulate (essentially the same as P.Ax.5 after incidence geometry), and NOPP is the *negation* of YESPP. It can be proved in hyperbolic geometry that whenever C is not on ℓ there are at least two (and in fact infinitely many) parallels to ℓ through C. The axiom NOPP asserts only the existence of one such pair $(C \ , \ \ell)$ and the existence of two parallels m_1 and m_2 (from which infinitely many can be deduced).

Consistency of Euclidean geometry. Just take the usual model, with $\mathbf{R}^2$ as the set of points. (So maybe this should be called Descartes' theorem.) The basic relations are defined as usual:

$$(x_1,\ y_1) : (x_2,\ y_2) : (x_3,\ y_3) :$$

$$\Updownarrow$$

$$\exists\ \ t \in \mathbf{R} \ \text{ with } \ 0 < t < 1$$

$$\text{and } \ x_2 = tx_1 + (1-t)x_3 \quad ; \quad y_2 = ty_1 + (1-t)y_3 \ ;$$

and

$$(x_1,\ y_1)(x_2,\ y_2) \equiv (x_3,\ y_3)(x_4,\ y_4)$$

$$\Updownarrow$$

$$(x_1 - x_2)^2 + (y_1 - y_2)^2 = (x_3 - x_4)^2 + (y_3 - y_4)^2$$

The verifications of the axioms (whenever not totally trivial) are uninteresting computations with coordinates.

Let us denote this model as $\mathcal{C}$.

Subtle question: If something happens to be true in this model $\mathcal{C}$ (i.e. in "analytic" geometry), does it necessarily follow from the axioms? (i.e. is it a theorem of "synthetic" geometry?)

Categoricity of Euclidean geometry. Any two models are isomorphic. This follows by showing that any model is isomorphic to the Cartesian model $\mathcal{C}$. A more precise statement can be made (see the next theorem), which justifies the common procedure in concrete problems of geometry and physics, where great simplification often results from a judicious choice of "origin", "x-axis" and "unit distance", related to the details of the particular problem.

Theorem. *Given $\mathcal{E}$, a model for Euclidean geometry, and distinct points O and A in $\mathcal{E}$, there is an isomorphism*

$$\mathcal{E} \ \rightarrow \ \mathcal{C}$$

mapping O to $(0,0)$ and A to (1,0). In fact there are exactly two such isomorphisms. (They are "mirror images of each other".)

Remarks. This is a theorem of metageometry because it is not a theorem *within* geometry but rather one *about* geometries. An isomorphism is any bijective map θ between the sets of points which *preserves both of the relations.* That is, besides being one-to-one and onto, θ must satisfy the following:

$$A : B : C \text{ in } \mathcal{E} \quad \Longleftrightarrow \quad \theta(A) : \theta(B) : \theta(C) \text{ in } \mathcal{C} \text{ ; and}$$

$$AB \equiv CD \text{ in } \mathcal{E} \quad \Longleftrightarrow \quad \theta(A)\theta(B) \equiv \theta(C)\theta(D) \text{ in } \mathcal{C} \text{ .}$$

How do you get the isomorphism? First map the line $\ell(O, A)$ onto the set $\{(x, 0) \mid x \in \mathbf{R}\}$ as follows: $O \mapsto (0, 0); A \mapsto (1, 0); B \mapsto (2, 0)$, where B is the unique point with $O : A : B$ and $OA \equiv AB$; $C \mapsto (3, 0)$ where C is the unique point with $A : B : C$ and $AB \equiv BC$; etc... This gives a set of points mapping bijectively onto the set $\{(x, 0) \mid x \in \mathbf{N}\}$. Now let $D \mapsto (-1, 0)$ where $D : O : A$ and $DO \equiv OA$; $E \mapsto (-2, 0)$ where $E : D : O$ and $ED \equiv DO$; etc.. giving points mapping to $\{(-x, 0) \mid x \in \mathbf{N}\}$. Next let $F \mapsto (1/2, 0)$ where $O : F : A$ and $OF \equiv FA$; $G \mapsto (3/2, 0)$ where $A : G : B$ and $AG \equiv GB$ etc..; so we now have points for all coordinates $(x/2, 0)$ as x ranges over integers. These last steps depend on first proving the existence and uniqueness of a point bisecting a given interval. Now repeat the process again and again, getting points for coordinates $(x/4, 0)$, then $(x/8, 0)$ etc... for integers x.

In the limit (i.e. by induction) we get a subset of $\ell(O, A)$ mapping onto

$$\{\ (m/2^n,\ 0)\ \mid\ \ n \in \mathbf{N}\ \ ,\ \ m\ \text{ or }\ -m \in \mathbf{N}\ \ \}\ .$$

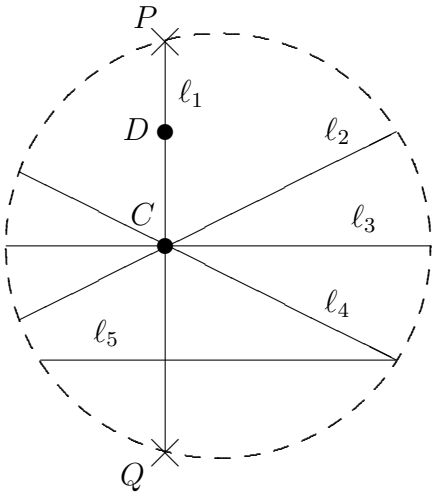
To finish mapping this line, $\ell(O, A)$, we use completeness. Each real x is the least upper bound of a sequence $m_1/2^{n_1} < m_2/2^{n_2} < \cdots$ of rationals with denominators equal to a power of 2. If $A_1, A_2, \cdots$ are the points on $\ell(O, A)$ with corresponding first coordinates, let L be their least upper bound point given by O8 and let $L \mapsto (x, 0)$. One can then prove that we've defined a bijective map $\ell(O, A) \to \{\ (x, 0)\ \mid\ x \in \mathbf{R}\ \}$ which preserves order and congruence.

Construct now the line through O perpendicular to $\ell(O, A)$. Here is where the choice arises: Choose one of the two points H on this line for which $OH \equiv OA$ and let $H \mapsto (0, 1)$ [The other choice is its mirror image in $\ell(O, A)$ and will map to $(0, -1)$]. Now the line $\ell(O, H)$ can be coordinatized by $\{\ (0, y)\ \mid\ y \in \mathbf{R}\ \}$, just as we did for $\ell(O, A)$. Finally, for any arbitrary point C, let the perpendiculars from C to $\ell(O, A)$ and $\ell(O, H)$ meet those lines

respectively in points coordinatized by $(x, 0)$ and $(0, y)$. Let $C \mapsto (x, y)$. This completes the construction. It is now possible to show the map constructed is bijective and preserves $:\ :$ and $\equiv$.

Beltrami's model $\mathcal{B}$ for hyperbolic geometry: This is the main point in this entire appendix. It shows hyperbolic geometry to be consistent, at least relative to the axioms for the reals and hence to Peano's axioms for **N**. Another way of phrasing it is that a person, claiming to have a deduction of the parallel postulate from the other axioms which we've given for Euclidean geometry, would in fact be claiming to have deduced a contradiction from Peano's postulates!

The points in the model are specified to be the points which are strictly inside some given circle in $\mathbf{R}^2$.



"In-betweenness" in the model is defined exactly as usual in Cartesian geometry, except that we only talk about points strictly within the given open disc.

Congruence is defined using a "funny distance function",

$$f(C, D) \ = \ \log \left[\ \frac{d(C, P) \cdot d(D, Q)}{d(C, Q) \cdot d(D, P)} \ \right] \ ,$$

where d is ordinary Euclidean distance.

N.B. The points P and Q on the circular boundary in the picture are *not* points in the model; they are named for the given C and D so that Q is closer to C and P closer to D, when $C \neq D$. Thus $f(C, D)$ is positive.

Define

$$C_1D_1 \equiv C_2D_2 \iff f(C_1,\ D_1) = f(C_2,\ D_2).$$

Checking the axioms:

From the definition, lines are just the chords (not including end points). In particular, the lines ℓ_2, ℓ_3, ℓ_4 in the picture are all through C and parallel to ℓ_5. This shows NOPP to hold, so we certainly do *not* have Euclidean geometry.

The argument to verify that O1 to O9 hold for this proposed model is pretty obvious. They hold simply because the disc is open and convex (i.e. it contains the Euclidean straight line interval between any two of its points), and because those axioms hold in the $\mathbf{R}^2$-model of Euclidean geometry. Notice that O1 would definitely fail if we allowed any points on the circular boundary to be in the model.

Thus we are left to check C1 to C5.

For C5, see Greenberg, pp. 205-216.

C1: We need only check $f(C, D) = f(D, C)$, which is immediate from the definition of f.

C2: $[\ f(A, B) = f(C, D)\ \&\ f(C, D) = f(E, F)\ ] \ \Rightarrow\ f(A, B) = f(E, F).$

C3: Given a "partial chord", starting at C and passing through D on its way to the disc boundary, we must show that $f(C, E)$ takes each real positive value exactly once [in particular the value $f(A, B)$], as E varies on the partial chord. But (with C fixed), $f(C, E)$ is a continuous function of E (since log and ordinary distance d are continuous). Furthermore $f(C, E) \to 0$ as $E \to C$, and $f(C, E) \to \infty$ as $E \to$ the boundary point P. The intermediate value theorem then applies to show that $f(C, E)$ takes all values from 0 to "∞", as required. Also $f(C,\ E_1) \neq f(C,\ E_2)$ if $E_1 \neq\ E_1$, since $f(C,\ X)$ strictly increases as X moves continuously away from C to the boundary, on the ray through D.

C4: It suffices to show that

$$A : B : C \implies f(A, B) + f(B, C) = f(A, C)$$

(which seems reasonable for any function which is supposed to measure some kind of distance!) Using this, we would get $AC \equiv A'C'$ since

$$f(A, C) = f(A, B) + f(B, C) = f(A', B') + f(B', C') = f(A', C').$$

To prove the above "additivity "is an easy computation with logs:

$$f(A,B)+f(B,C) \;=\; \log\,[\,\frac{d(A,P)\cdot d(B,Q)}{d(A,Q)\cdot d(B,P)}\,] \quad + \;\log\,[\,\frac{d(B,P)\cdot d(C,Q)}{d(B,Q)\cdot d(C,P)}\,]$$

$$=\; \log\,[\,\frac{d(A,P)\cdot d(C,Q)}{d(A,Q)\cdot d(C,P)}\,] \quad = \quad f(A,C)\;.$$

Comments. Below I'll give four other models, $\mathcal{P}$ (for Poincaré), $\mathcal{K}$ (for Klein), $\mathcal{M}$ (for Minkowski), and $\mathcal{H}$ (for hemisphere). They have *not* been named for their inventors, since, for example, I think that Poincaré invented two of them! These additional models have some geometric advantages over $\mathcal{B}$ (for Beltrami) above. Also $\mathcal{M}$ has an interesting relation to special relativity. **Hyperbolic geometry is, in fact, categorical**. We'll give some interesting projection maps which are isomorphisms between pairs of these models.

Exercise G4.7 Show that rotating the disc by some given angle about its centre gives an isomorphism from $\mathcal{B}$ to itself. Can you think of any isomorphism which doesn't map the centre to itself? (They do exist; the centre just happens to look special in the model; just as in Euclidean geometry, all points in hyperbolic geometry are 'created equal'; one uses group theory to make this precise: the group of all self-isomorphisms under the operation of composition has at least one element which maps any chosen point to any other chosen point.)

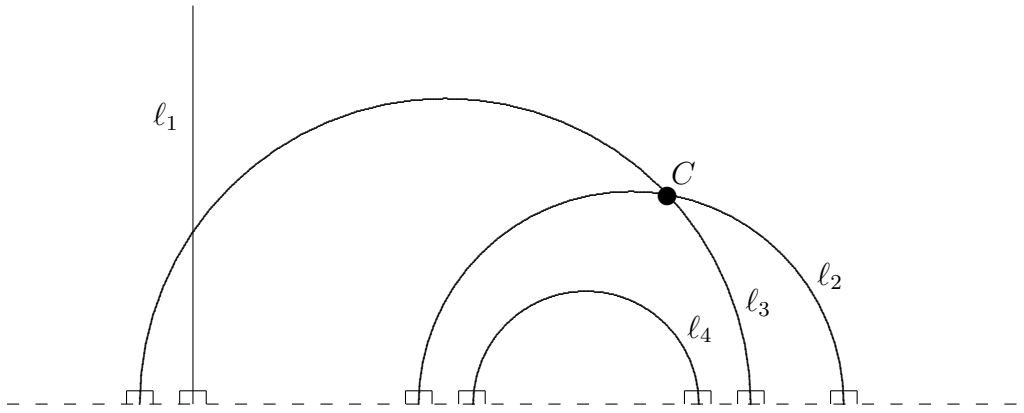
Poincaré's half-plane model $\mathcal{P}$. Here the points are all the points in the upper half-plane i.e. the set

$$\{(x,y)\in\mathbf{R}^2 \;\mid\; y>0\;\}\;.$$

Lines are

i) vertical half lines, and

ii) semi-circles centred on the x-axis.



In the picture, ℓ_2 and ℓ_3 are distinct lines through C parallel to ℓ_4. In-betweenness is the usual notion for curves in $\mathbf{R}^2$. Congruence is again defined by a "funny distance", as follows. Given a curve $[x(t), y(t)]$ in the upper half plane with continuous derivatives dx/dt and dy/dt , one can define the "funny arc length" along the curve from $[x(t_0), y(t_0)]$ to $[x(t_1), y(t_1)]$ to be

$$\int_{t_0}^{t_1} \frac{1}{y(t)} \sqrt{(dx/dt)^2 + (dy/dt)^2}\, dt \quad .$$

Note that $y(t) > 0$, so the integrand is positive and continuous, and most importantly, note that a change from parameter t to a different parameter will not alter the answer. It turns out that the curve for which the answer is minimized between two given points in $\mathcal{P}$ is a semi-circular arc, centred on the x-axis, except when one point is vertically above the other. In the latter case, the vertical line segment gives the shortest "funny arc length". (In "Calculus of Variations", one learns how to "minimize functionals" as above. This leads among other places to a beautiful treatment of classical mechanics via Lagrangians, Hamiltonians, etc.)

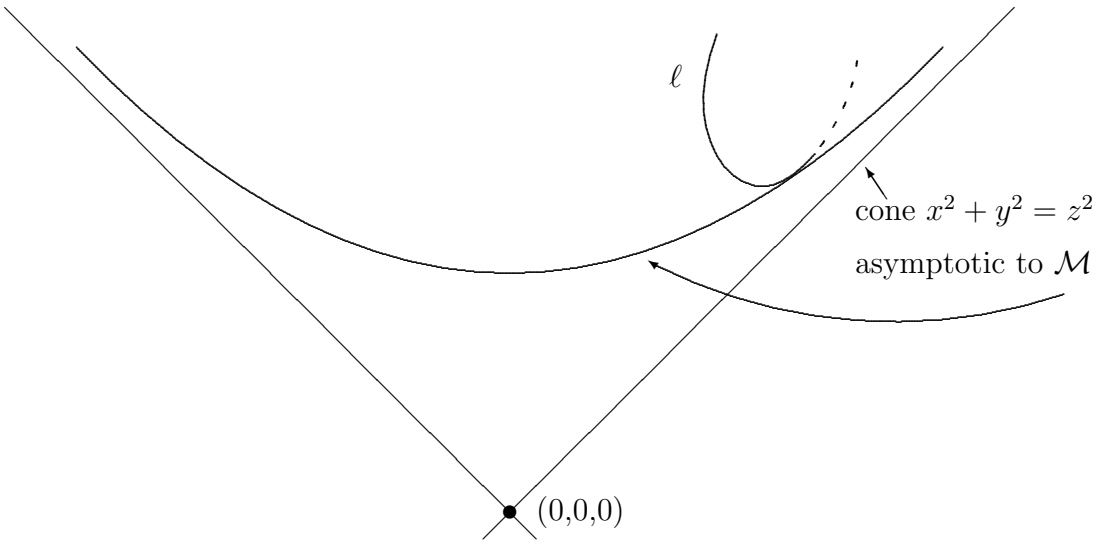
It goes without saying that $\mathcal{P}$ here has little to do with the same notation which we used a few sections earlier in discussing the use of set notation for incidence geometry (although any model for any of the types of geometry which have occurred since then can also be regarded as a model for incidence geometry).

Exercise G4.8 Show that two points in $\mathcal{P}$ with different 1st coordinates

are on one and only one semicircle centred on the x-axis.
N.B. Semicircle means with respect to the Euclidean geometry of $\mathbf{R}^2$ here; a "circle" using the hyperbolic distance would look quite strange.

Minkowski model $\mathcal{M}$: Let the point set be the upper sheet of a hyperboloid of two sheets in $\mathbf{R}^3$; specifically, the set

$$\{(x, y, z) \mid x^2 + y^2 + 1 = z^2 \text{ and } z > 0 \} .$$



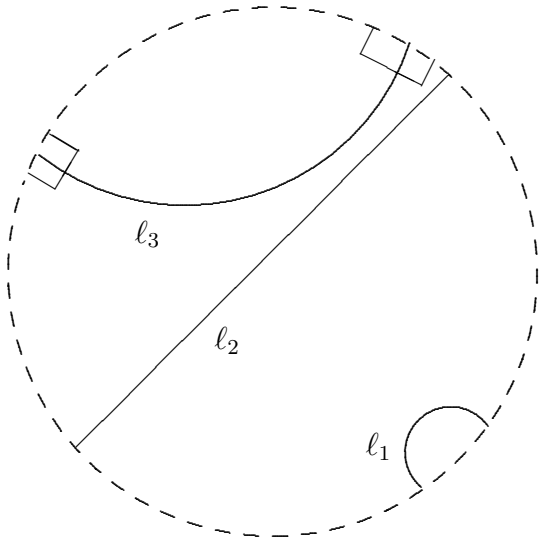
A plane through the origin which cuts the cone (i.e. a plane whose angle with the plane "$z = 0$" is larger than 45°) will cut our sheet in one branch of a hyperbola. Each pair of distinct points on our sheet is on exactly one such hyperbola branch, since the two points together with the origin determine a unique plane. These branches are the lines, and inbetweenness is the usual notion (i.e. $A : B : C$ holds iff traveling along the hyperbola from A to C requires passing through B). Congruence is given by another "funny arc length"; this time for a curve ($x(t)$, $y(t)$, $z(t)$) on the sheet, it is

$$\int_{t_0}^{t_1} \sqrt{(dx/dt)^2 + (dy/dt)^2 - (dz/dt)^2}\, dt \quad .$$

Notice that the function inside the square root is never negative for a curve which stays on the sheet. Just as with $\mathcal{P}$, our hyperbolic line segments in this model again give curves of shortest funny arc length.

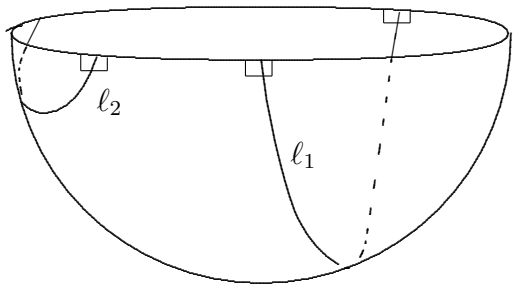
The connection with Einstein's special relativity is that the space of 'observers' or 'inertial frames' is hyperbolic geometry. The cone is the 'light cone'. But in reality, this connection holds for the 3-dimensional version of hyperbolic geometry, not the 2-dimensional version which we have been studying here. Congruence is defined as "equal relative velocities". The definition of the relation $A : B : C$ is that A and C are observed by B to be moving away from her in exactly opposite directions, at constant velocity of course.

The model $\mathcal{K}$: As with $\mathcal{B}$, take the interior of a disc as the set of points. But this time the lines are to be all the circular arcs perpendicular to the boundary circle plus all the "diameters" (chords through the centre). This is explained in detail in Greenberg. It is better than $\mathcal{B}$ for many purposes because the angle (in the sense of Euclidean geometry), which two such "lines" make, turns out to coincide with their angle as measured in hyperbolic geometry. (Hence $\mathcal{K}$ is called a *conformal* model.)



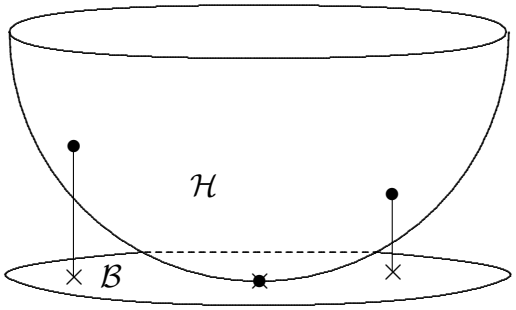
Hemisphere model $\mathcal{H}$. Take the southern hemisphere on a sphere (just the surface and not including the equator). The lines are all the semicircles

orthogonal to the equator. This model will be useful to connect up the other models by isomorphisms (from which you could figure out funny distance functions for both $\mathcal{H}$ and $\mathcal{K}$, if you wished).



Pictures of isomorphisms:

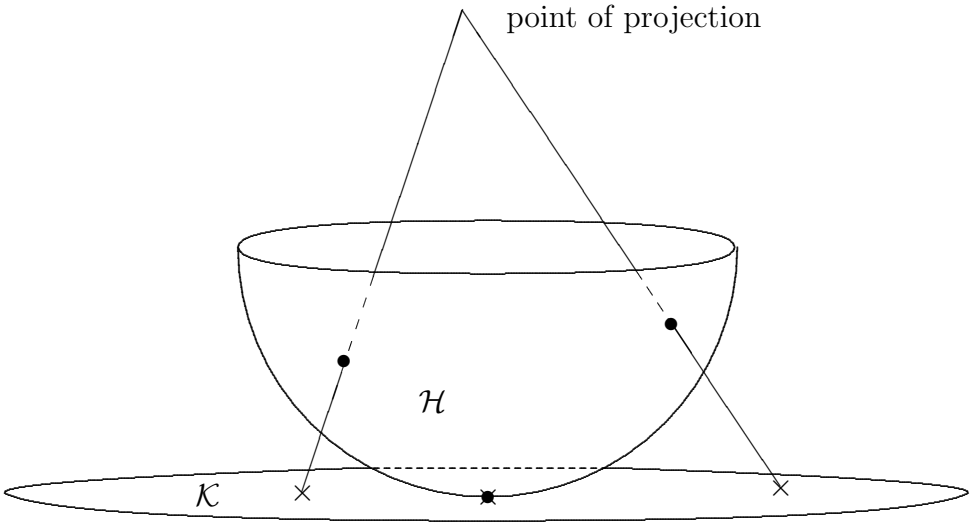
$$\mathcal{H} \cong \mathcal{B}$$



Let the disc be tangent to the hemisphere, with centres touching, and use vertical projection to get a map $\mathcal{H} \to \mathcal{B}$. Note that lines get mapped to

lines.

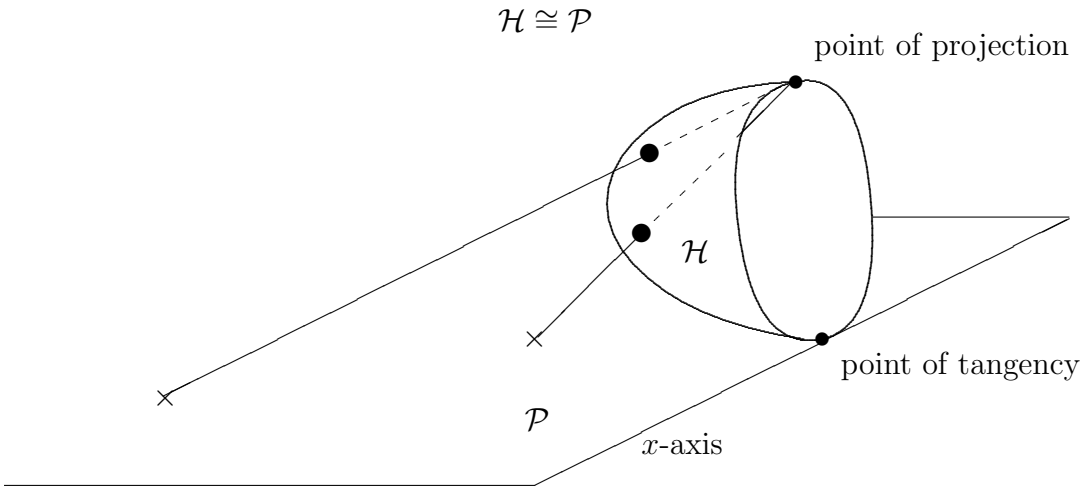
$$\mathcal{H} \cong \mathcal{K}$$



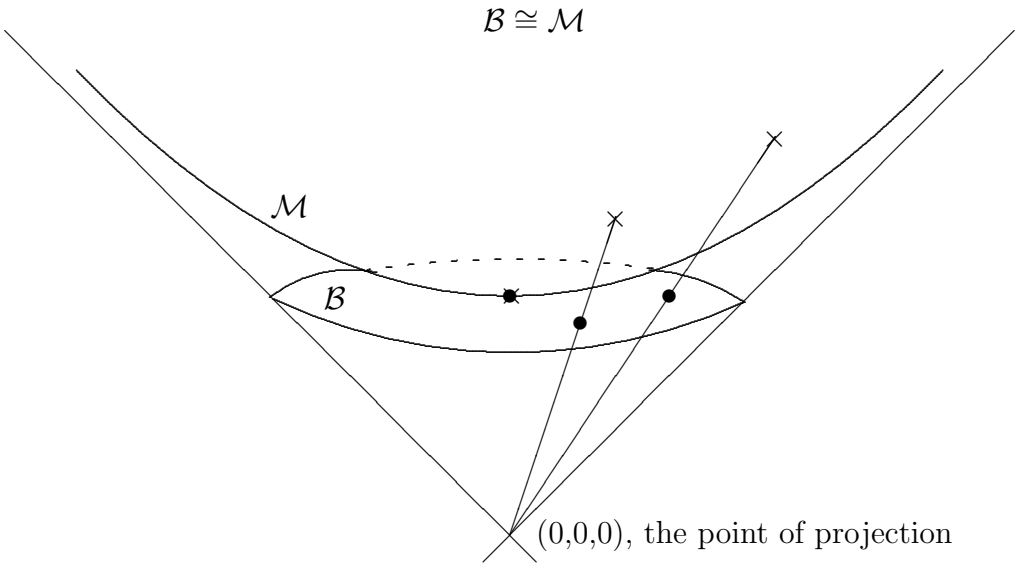
.

Again $\mathcal{K}$ is tangent to $\mathcal{H}$, but this time the map projects from the north pole of the sphere of which $\mathcal{H}$ is the southern hemisphere. $\mathcal{K}$ is therefore a somewhat larger disc than $\mathcal{B}$ was in the previous.

The picture on the cover of Greenberg combines the last two isomorphisms.



This time, place the sphere tangent to the plane so that the equator boundary of $\mathcal{H}$ is tangent to the x-axis boundary of $\mathcal{P}$. Let the map $\mathcal{H} \to \mathcal{P}$ be projection from the point which is antipodal to the point of tangency.



In this case, regard $\mathcal{B}$ as the disc

$$x^2 + y^2 = 1 \quad ; \quad z = 1$$

in 3-space. It is then tangent to the sheet $\mathcal{M}$. The map is just projection from the vertex of the asymptotic cone.
Note Again, lines map to lines.

Exercise G4.9 By choosing convenient analytic equations for the spaces involved, work out an *explicit formula* for each of these isomorphisms. For example, if we take

$$\mathcal{B} = \{\ (x, y) \in \mathbf{R}^2 \ \mid \ x^2 + y^2 < 1\ \}$$

and take

$$\mathcal{H} = \{\ (x, y, z) \in \mathbf{R}^3 \ \mid \ x^2 + y^2 + z^2 = 1 \ \text{ and } \ z > 0\ \}$$

the isomorphism $\mathcal{H} \to \mathcal{B}$ just sends (x, y, z) to (x, y). (In these choices, $\mathcal{B}$ isn't tangent to $\mathcal{H}$, but that is immaterial.)

What is Differential Geometry? Gauss initiated the study of curves and surfaces (and Riemann, the study of higher dimensional *manifolds*), in which the geometry is defined by a metric which tells one how to compute arc length along smooth curves. The models $\mathcal{M}$ and $\mathcal{P}$ are examples, with metrics ds as given. However the hyperboloid sheet *as pictured* has the metric

$$ds^2 = dx^2 + dy^2 + dz^2 \ ,$$

inherited from the 'ordinary' $\mathbf{R}^3$ metric. Its curves of shortest length in this metric (called *geodesics*) are *not* the hyperbola-branches which are the lines of $\mathcal{M}$. The latter 'lines' are the geodesics with respect to

$$ds^2 = dx^2 + dy^2 - dz^2 \ .$$

(You could get the geodesics with respect to the Euclidean metric "physically" by stretching a rubber band over the surface between the two points.)

Taking as another example the surface of a sphere $x^2 + y^2 + z^2 = 1$ in $\mathbf{R}^3$ with ordinary metric $ds^2 = dx^2 + dy^2 + dz^2$, the geodesics turn out to be

great circles (i.e. intersections of the sphere with planes through its centre). They play the role of 'straight lines' in this geometry, which is called *spherical geometry*, and has applications to, for example, saving jet fuel.

By 'gluing together' each pair of antipodal points to form a single point, one obtains a geometry in which any two lines intersect in exactly one point—*no parallels!!* (since great circles intersect in a pair of antipodal points). This is a model for the other classical non-Euclidean geometry called *elliptic geometry*. The study of just its incidence properties (no congruence) is *projective geometry*. In these geometries there is no 3-ary *inbetweenness* relation, only a 4-ary relation which speaks about whether or not one pair of points on a line separates the other pair from each other—think of how inbetweenness doesn't make sense for three points on a circle, but separation of pairs does make sense for four such points. The lines in projective and elliptic geometry are closed curves.

Hilbert showed that neither the elliptic plane nor the hyperbolic plane in their entirety could be defined as surfaces in $\mathbf{R}^3$ with the ordinary metric.

Using the unit disc, $\{(x, y) \ : \ x^2 + y^2 < 1 \}$, a metric for $\mathcal{K}$ is

$$ds^2 = \frac{dx^2 + dy^2}{(1 - x^2 - y^2)^2} \quad ,$$

and for $\mathcal{B}$ is

$$ds^2 = \frac{dx^2 + dy^2 - (xdy - ydx)^2}{(1 - x^2 - y^2)^2} \quad .$$

Appendix L :

LESS THAN EVERYTHING

YOU NEED TO KNOW ABOUT LOGIC

This appendix also is written so as to be independent of the rest of the book. And it contains a number of statements whose proofs are given informally at best, sometimes incompletely, and sometimes not at all. It is intended to motivate you to go on to study more advanced books on logic, particularly to learn the careful details concerning algorithms, Turing machines, recursive functions, Gödel numbers, etc. See [**CM**]. However, modulo that, we give another (see also Appendix VQ to Chapter 5) detailed proof of Gödel's incompleteness theorem near the end, as well as of fundamental theorems due to Church and Tarski. In contrast to **Appendix G**, this appendix proceeds fairly swiftly and requires more sophistication of the reader.

Further down, when dealing with one version (syntactic) of Gödel's first incompleteness theorem, we won't even need to assume that the natural number system exists, as an 'actual infinity'. But, to begin, let's agree (as normal sane people do) that there's no doubt about the existence of the infinite set $\mathbf{N} = \{0, 1, 2, 3, \cdots\}$ of natural numbers, with the following standard additional structure on $\mathbf{N}$: its addition, $+$, multiplication, denoted $\times$ (both mapping $\mathbf{N}\times\mathbf{N}$ into $\mathbf{N}$), and successor ($S : \mathbf{N} \to \mathbf{N},\ S(a) = a+1$) functions; and its natural ordering, denoted as usual by $<$.

This structure, $(\mathbf{N}, S, +, \times, <, 0)$, satisfies various $1^{\underline{\text{st}}}$*order sentences*, such as the following nine examples of such sentences, where the logical symbols used are explained just below.

$$\forall x_1\ \neg Sx_1 \approx 0$$
$$\forall x_1 \forall x_2\ (Sx_1 \approx Sx_2\ \rightarrow\ x_1 \approx x_2)$$
$$\forall x_1\ x_1 + 0 \approx x_1$$
$$\forall x_1 \forall x_2\ S(x_1 + x_2) \approx x_1 + Sx_2$$
$$\forall x_1\ x_1 \times 0 \approx 0$$
$$\forall x_1 \forall x_2\ x_1 \times Sx_2 \approx (x_1 \times x_2) + x_1$$
$$\forall x_1\ \neg x_1 < 0$$
$$\forall x_1 \forall x_2\ (x_1 < Sx_2\ \rightarrow\ (x_1 < x_2\ \vee\ x_1 \approx x_2))$$
$$\forall x_1 \forall x_2\ ((x_1 < x_2\ \vee\ x_1 \approx x_2)\ \vee\ x_2 < x_1)$$

Of course, there are lots more such sentences which are true in $\mathbf{N}$. We explain below how to get well-formed sentences. (Alternatively, read Chapter 5.) Plenty of these sentences will be false in $\mathbf{N}$, and part of what is assumed here is that every such sentence is either true or false in $\mathbf{N}$, but not both. This is entirely reasonable, and should not be confused with the much stronger (and false!) assumption that there exists a method for deciding which (true or false) for any given sentence. The particular nine true sentences above will also play a basic role in Gödel's incompleteness theorems (see **L4** ahead, and also just before the **Addendum**).

In addition to $(S, +, \times, <, 0)$, these formulae use only parentheses, variables and the following logical symbols :

$\forall$ "for all"; $\neg$ "not"; $\vee$ "or"; $\rightarrow$ "implies"; and also $\approx$ for "equals". The last one is a formal symbol distinct from $=$, which here means "is the same as" (and does *not* appear in formulae). So, for example, "$(x \approx y) = (x \approx y)$" is meaningful, as is "$\approx = <$", the first being true and the second false. But neither is a $1^{\underline{\text{st}}}$order formula, because "$=$" is not an allowed symbol. The non-logical symbols $\{S, +, \times, <, 0\}$ appearing in the formulae are just that—symbols; and, despite the overlap in notation, are distinct from the functions etc. which interpret them in $\mathbf{N}$. Actually, in the main body of the book, there is no overlap, because the actual functions, relations, etc. in $\mathbf{N}$ would there be denoted $\{S^{\mathbf{N}}, +^{\mathbf{N}}, \times^{\mathbf{N}}, <^{\mathbf{N}}, 0^{\mathbf{N}}\}$, as indicated in the general definition of *interpretation* in Section **6.1**.

To define how one generally gets the strings of symbols which are formulae, such as those nine listed above:

(1) *Terms* are built up : by first applying one of S, $+$ and $\times$ to 0 and/or to *variables* $x_1, x_2, x_3, \cdots$ (these variables and 0 are the smallest terms); and then applying the above function symbols successively to larger terms.

(2) *Atomic formulae* are all of the form $term_1 \approx term_2$ or $term_1 < term_2$.

(3) *Bigger formulae* are built up inductively from smaller ones, $F, G, \cdots$, by forming

(i) $\neg F$;

(ii) $(F \vee G)$ [and possibly $(F \rightarrow G)$, $(F \wedge G)$ "and", $(F \leftrightarrow G)$ "iff", though the latter three can be defined from $\neg$ and $\vee$]; and also forming

(iii) $\forall x F$ [and possibly $\exists x F$, meaning "there exists x, F" , though again $\exists x$ is definable as $\neg \forall x \neg$]. Here x is any one of the variables.

(4) *Sentences* are formulae in which every appearance of every variable is *bound* (or *quantified*, or *within the scope of some* $\forall$ *or some* $\exists$); that is, no variables are *free*. To be precise, all occurrences of variables in atomic

formulae are free, and, for larger formulae, (i) and (ii) have no effect on freeness, but (iii) converts all free occurrences of x in F into bound occurrences in $\forall x F$, but has no effect on free occurrences of variables different from x.

The set of all formulae built this way is an example of a $1^{\underline{\text{st}}}$order language. Other examples are obtained by changing the set of *non-logical symbols* $\{S, +, \times, <, 0\}$ to something else: some function symbols, relation symbols, constant symbols. Of particular interest is the language of $1^{\underline{\text{st}}}$*order set theory*, where the relation symbol $\in$ is the only non-logical symbol needed. (It is amusing that a single binary relation symbol also gives the language of $1^{\underline{\text{st}}}$order directed graph theory.) In $1^{\underline{\text{st}}}$order set theory, the union and the intersection are examples of function symbols which can be introduced later. They are not necessarily taken to be part of the basic language. Often, however, the symbol $\emptyset$ is used as a basic constant symbol (interpreted as the empty set), though $\emptyset$ can also be introduced later as a defined symbol.

Going back to number theory, let us now agree that there is a well-defined set $\mathcal{L}$ of all formulae in this language, and subsets $\mathcal{T}_{\mathbf{N}}$ (resp. $\mathcal{F}_{\mathbf{N}}$) of those formulae which are true (resp. false) in the structure $(\mathbf{N}, S, +, \times, <, 0)$, even though the human species is still largely ignorant of what is in these two subsets! By definition, the truth of a formula reduces to that of a sentence, by putting enough $\forall x_i$'s in front of the formula to convert it into a sentence (and similarly for false, using $\exists x_i$'s). For example, the formula $x \approx 0 \ \vee \ 0 < x$ is in $\mathcal{T}_{\mathbf{N}}$ because $\forall x \ (x \approx 0 \ \vee \ 0 < x)$ is a sentence which is true in $\mathbf{N}$. Some non-sentence formulae, such as $x \approx 0$, are neither true nor false. Furthermore, as indicated earlier, we accept that, with $\mathcal{S}$ as the subset of sentences: (1) $\mathcal{S}$ is the disjoint union of $\mathcal{T}_{\mathbf{N}} \cap \mathcal{S}$ and $\mathcal{F}_{\mathbf{N}} \cap \mathcal{S}$; i.e. every sentence is either *true in* $\mathbf{N}$ or *false in* $\mathbf{N}$ but *not both*; and (2) $G \in \mathcal{T}_{\mathbf{N}}$ if and only if $\neg G \in \mathcal{F}_{\mathbf{N}}$. So, in some sense, the set of sentences is divided neatly in half by truth and falsity in the *interpretation* $(\mathbf{N}, S, +, \times, <, 0)$ of the language.

We wish to consider the possibility of a set $\mathcal{A}$ (contained in $\mathcal{T}_{\mathbf{N}}$, i.e. a set of formulae true in $\mathbf{N}$) which might be a candidate for "a set, $\mathcal{A}$, of axioms for $1^{\underline{\text{st}}}$order number theory". (In the main part of this book, non-logical axioms like this are often called *premisses*.) It would be nice if there were a *finite* such set $\mathcal{A}$ from which every formula in $\mathcal{T}_{\mathbf{N}}$ could be shown to be "formally deducible", even if determining whether a given sentence was in $\mathcal{T}_{\mathbf{N}}$ remained

a hard problem in most cases. But *this is very far from being possible*, as we'll see below.

But what does "to formally deduce" mean? Basically it means to use a proof system for $1^{\underline{\text{st}}}$order logic. Mainly, this consists of specifying some rules of inference. An example is $\frac{A\ ,\ (A \to B)}{B}$, known as *modus ponens* or (MP), which occurs among the basic parts of many such systems. It would be used in a formal proof as follows:

(1) A *formal proof* is by definition a finite sequence of formulae constructed by using these rules of inference plus axioms. The last formula in the sequence is "what is proved" by that particular formal proof.

(2) Many of the formulae in the sequence will be there by using some particular rule of inference—in the case of (MP) being used, the formula, whose appearance in the proof is being justified, is B, and the appeal to (MP) must be based on both A and $(A \to B)$ having occurred earlier in the sequence. (See the (third sentence), two paragraphs down regarding the use of axioms.)

So I'll leave a lot out here, and you should either read the beginning of Chapter 7, or get out one of the references and study the system there. Safe to say that all the various systems which are worthy of consideration will yield the same results for sentences. For non-sentence formulae, there are slight variations between systems as to which are formally deducible. But all reputable systems, despite differences in which rules of inference are basic, will in the end have the same *sentences* turn out to be deducible. For example, those rules of inference in one system which don't occur as a basic part of the other can be derived within the other system. Rules of inference should have a certain *universality*—for example, use of (MP) is related to the fact that, *for all formulae A and B*, the formula

$$(A\ \wedge\ (A \to B)) \to B$$

is true in *every interpretation of every* $1^{\underline{\text{st}}}$*order language*. The same holds with other rules of inference and with the other ingredient of the system, which is the list of logical axioms. An example of a logical axiom might be $\forall x\ x \approx x$, which is again true in every interpretation of every $1^{\underline{\text{st}}}$order language. This universality is why one regards this aspect as purely 'logic', as opposed to 'mathematics'.

Once again let's return to number theory. Let $\mathcal{A}^{\vdash}$ denote the set of

all formulae F for which there is a formal proof of F from the given set $\mathcal{A}$ of (non-logical) 'axioms'. (Besides using rules of inference, the only other rule that allows formulae to occur in a formal proof is, at any point in the sequence which forms the proof, simply writing down an axiom, either logical or non-logical.)

To finally state a theorem, suppose that the axiom set consisting of some formulae true in $\mathbf{N}$ satisfies the following restriction :

$\mathcal{A}$ consists of finitely many formulae together with all the formulae obtained by substituting formulae into finitely many **axiom schemes**.

An example of such a scheme is

$$(1^{\underline{\text{st}}}\mathbf{IND}) \qquad (F^{[x\mapsto 0]} \wedge \forall x(F \to F^{[x\mapsto Sx]})) \;\to\; \forall x F \quad .$$

Here x denotes any variable and F denotes any formula. In this axiom scheme, we use (with $t = Sx$ and $t = 0$) the formula $F^{[x\mapsto t]}$. The latter is defined to mean F with all free occurrences of x substituted by t, which may be any term for which no variable in the term becomes 'quantified' after the substitution. (This latter proviso holds when x is substituted by Sx or 0. It also holds where we use this notation in the Addendum, so the proviso won't be mentioned again.) If x doesn't occur freely in F, the formula ($1^{\underline{\text{st}}}$**IND**) says nothing. If x does, then ($1^{\underline{\text{st}}}$**IND**) is a form of induction, considerably weakened by the need for F to be a $1^{\underline{\text{st}}}$*order* formula. So a scheme will in general produce infinitely many formulae (here from the infinitely many choices for F and x).

Theorem L1. (weak, but still impressive) *For any set $\mathcal{A}$ as just above, we have $\mathcal{A}^{\vdash} \neq \mathcal{T}_{\mathbf{N}}$.*

Now $\mathcal{A}^{\vdash} \subset \mathcal{T}_{\mathbf{N}}$, so this is saying that there is some G which is true in $\mathbf{N}$, but which cannot be proved from $\mathcal{A}$ (so perhaps $\mathcal{A}$ isn't much good as an axiom system!) Of course, $\neg G$ also cannot be proved since it's in $\mathcal{F}_{\mathbf{N}}$. The existence of such a pair, $\{G,\ \neg G\}$, *neither provable*, is an example of one use of the word "incompleteness". It is a striking fact that no axiom system with finitely many axioms and axiom schemes is strong enough to provide for the formal proofs of all true formulae of $1^{\underline{\text{st}}}$order arithmetic. As we'll see, Gödel's first incompleteness theorem is both somewhat different (not needing reference to the existence of a model $\mathbf{N}$ and "truth"), and much stronger.

To build **L1** up to full strength, we must talk about *procedures* or *computability* or *algorithms* or *recursive functions*. This will be done very sketchily. The first (of two kinds of) procedure here can be thought of as a 'machine' (programmed once and for all by means of a finite list of instructions, and having potentially infinite memory) which runs 'forever', gradually producing a list of objects, a single object being spat out every so often. ("After so many *steps*" is much better than "after so much *time*" as what is intended by "every so often".) Such a procedure is called a "Type L algorithm" in Chapters 4 and 8. The set of objects so produced (in this list) is called a *listable set*. There is a second way of characterizing listability, which the reader can check. In the case of our set $\mathcal{A}$, it would say that $\mathcal{A}$ is listable if and only if there is a machine which accepts any formula as an input, and produces the answer "YES" after some steps if the formula is in $\mathcal{A}$, but just runs forever, producing no output, if the input formula is not in $\mathcal{A}$.

A fundamental fact here is the following.

Theorem L2. *$\mathcal{T}_{\mathbf{N}}$ is not listable.*

It's *not* obvious, but it *is* true that $\mathcal{A}$ listable implies $\mathcal{A}^{\vdash}$ listable. So we get

Theorem L1. (industrial strength) *If the axiom set $\mathcal{A}$ is listable, then $\mathcal{A}^{\vdash} \neq \mathcal{T}_{\mathbf{N}}$.*

It is intuitively clear that any $\mathcal{A}$ as in the weak version of **L1** *is* listable, so the new version is stronger, if more technical. The weak version is still rather impressive, but suffers from the criticism that maybe someone could think of an acceptable way to supplement the axiom set $\mathcal{A}$ to $\mathcal{B} \supset \mathcal{A}$ so that $\mathcal{B}^{\vdash}$ *does* consist of all true $1^{\underline{\text{st}}}$order formulae of arithmetic. However the industrial strength version makes it hard to imagine what that supplementation might be. Clearly *some* restriction on $\mathcal{A}$ is needed to get a theorem, a restriction which gets rid of uninteresting possibilities such as taking $\mathcal{A}$ to be $\mathcal{T}_{\mathbf{N}}$ itself.

Before proceeding to the version of Gödel's theorem which is purely syntactical and stronger than **L1**, let's consider the original three Grassman-Dedekind-Peano axioms for **N**, the third of which is definitely *not* $1^{\underline{\text{st}}}$order:

$$\forall x_1 \neg Sx_1 \approx 0$$
$$\forall x_1 \forall x_2 \, (Sx_1 \approx Sx_2 \;\rightarrow\; x_1 \approx x_2)$$

Full Induction: If I is any subset of $\mathbf{N}$, $0 \in I$ and $S(I) \subset I$, then $I = \mathbf{N}$.

A well-known theorem of Dedekind says that these three axioms characterize $(\mathbf{N}, S, 0)$ up to a unique isomorphism [and that the remaining structure $(+, \times, <)$ is definable in terms of $(S, 0)$]. One might therefore expect that all of number theory follows from these three axioms, if only we were clever enough. Besides human frailty, one problem is that the full induction axiom is not $1^{\underline{\text{st}}}$order (nor is it replaceable by a $1^{\underline{\text{st}}}$order axiom scheme), and that higher order logic has neither well-behaved nor well-understood proof systems.

On the other hand, the full induction axiom can be reformulated as a $1^{\underline{\text{st}}}$order **set theory** sentence. All of $1^{\underline{\text{st}}}$order number theory can also be recast within $1^{\underline{\text{st}}}$order set theory. Roughly, the successor of any set X is defined to be $X \cup \{X\}$, and $\mathbf{N}$ is defined to be the smallest set which contains the successor of each of its members and also contains the empty set. This actually produces an $\mathbf{N}$ which contains exactly one set of each finite *cardinality* (that is, *size*), and nothing else. The three Peano axioms are then formally provable from any sensible set of $1^{\underline{\text{st}}}$order axioms for set theory.

However, **L1** can be generalized to an assertion which includes the fact that, for any listable set of $1^{\underline{\text{st}}}$order set theory axioms, either
(1) there will still be an unprovable $1^{\underline{\text{st}}}$order set theory sentence which is a reformulated $1^{\underline{\text{st}}}$order number theory sentence true in $\mathbf{N}$; or
(2) the set theory axiom set is inconsistent (and so, 'useless') in that it can prove every formula (such as $\neg 0 \approx 0$ for example).

Set theory can be replaced by any $1^{\underline{\text{st}}}$order theory which 'contains' enough number theory. The reason for interest in set theory is that all of mathematics appears to be reformulatable into it. The reason we needed (2) above rather than talking about a set of 'true' set theory formulas is that there is no model for set theory which plays the role of $\mathbf{N}$ for number theory, in that most of us speak confidently (or at least bravely) about the reality of the set of true $1^{\underline{\text{st}}}$order sentences for $\mathbf{N}$. We shall stick to number theory for the rest of this appendix.

As to the full induction axiom, one can more fully explain its unruly nature:

Theorem L3. *There exist interpretations* $(\mathbf{M}, S, +, \times, <, 0)$ *of the* $1^{\underline{\text{st}}}$*order number theory language such that*

(a) $\mathcal{T}_{\mathbf{M}} = \mathcal{T}_{\mathbf{N}}$;

(b) $\mathbf{M}$ *has an element* b (for big, $\cdots$ **very** big) *such that all* $SS\cdots S0 < b$ *;*

(c) $\mathbf{M}$ *is countable (or of any given infinite cardinality).*

A corollary is that such a model $\mathbf{M}$ is *not isomorphic* to $\mathbf{N}$ (there is no bijection $\mathbf{M} \to \mathbf{N}$ which preserves all the structure). So $\mathbf{M}$ could not satisfy the full induction axiom, by Dedekind's theorem or simply because of *(b)*. And so there is definitely no $1^{\underline{\text{st}}}$order formula or scheme equivalent to that axiom. Such models $(\mathbf{M}, S, +, \times, <, 0)$, satisfying *(a)* but not isomorphic to $\mathbf{N}$, are called *non-standard models of arithmetic*.

The proof of parts *(a)* and *(b)* is not hard using the compactness theorem for $1^{\underline{\text{st}}}$-order logic: *An infinite set,* $\mathcal{B}$*, of formulae will have a model as long as every finite subset of* $\mathcal{B}$ *does.* (See Section **7.12**, at the end of Chapter 7.)

For proving **L3** *(a)* and *(b)*, take $\mathcal{B}$ to be

$$\mathcal{T}_{\mathbf{N}} \bigcup \{\ 0 < c\ ,\ S0 < c\ ,\ SS0 < c\ ,\ \cdots\ \}\ ,$$

where we now are dealing with the $1^{\underline{\text{st}}}$order language given by the non-logical symbols $\{\ S, +, \times, <, 0, c\ \}$; i.e. there is an extra constant symbol c. Any finite subset of this $\mathcal{B}$ has $\mathbf{N}$ itself as a model—just specify c to be sufficiently large, depending on which of the extra sentences "k" $< c$ are in the finite subset.

As another aside, had we been dealing with the language of $1^{\underline{\text{st}}}$order real number theory in the previous theorem, we could show that there is no $1^{\underline{\text{st}}}$order sentence equivalent to real completeness (nothing to do with logical completeness), nor even one equivalent to the Archimedean property of the reals. The non-standard models in this case can be used to rigorously introduce infinitesmals and infinite 'reals' into calculus.

Back again to number theory, what *was* Gödel's first incompleteness theorem? Below is a version which is *weaker* than the original, in that it sticks to our language $\{S, +, \times, <, 0\}$, whereas the theorem actually holds for any language "in which enough arithmetic exists"—for example, $1^{\underline{\text{st}}}$order set theory. The version below is also slightly *stronger* (due to Rosser a few years after the 1931 publication of Gödel), in that Gödel had assumed a slightly stronger form of consistency, called ω-consistency. (See the **Addendum**.)

Let's return to the beginning and now ignore **N** or any other interpretation, just thinking about sets of formulae in the 1st-order language $\{S, +, \times, <, 0\}$. Let $\mathcal{N}$ be the set of nine sentences listed on the first page of this appendix. Consider any set $\mathcal{A}$ of formulae for which:

(1) $\mathcal{A} \cup \mathcal{N}$ is *consistent*, in that $(\mathcal{A} \cup \mathcal{N})^{\vdash}$ doesn't contain $\{F, \neg F\}$ for any formula F. This is equivalent to saying that $(\mathcal{A} \cup \mathcal{N})^{\vdash}$ is not the set of all formulae. It follows of course that $\mathcal{A}$ itself must be consistent, since $\mathcal{A} \subset \mathcal{B}$ implies that $\mathcal{A}^{\vdash} \subset \mathcal{B}^{\vdash}$.

(2) $\mathcal{A}$ is a listable set.

Theorem L4.(Gödel's first incompleteness theorem) *For any $\mathcal{A}$ satisfying (1) and (2), there is a sentence G such that neither G nor $\neg G$ is formally provable from $\mathcal{A}$; that is, $G \notin \mathcal{A}^{\vdash}$ and $\neg G \notin \mathcal{A}^{\vdash}$.*

Note that, in proving this, one might as well assume that $\mathcal{N} \subset \mathcal{A}$, since the slightly more general-sounding case as stated follows easily from that.

Theorems **L1** (industrial strength) and **L4** are evidently related, but not identical. **L4** avoids any assumption about the existence of an "actual" infinity of natural numbers with its structure; and even with that, it applies to cases where $\mathcal{A}$ might contain a sentence which is false in **N**. In **L1**, the nine particular sentences in $\mathcal{N}$ are irrelevant (below we explain to some extent what they are needed for here, but see the Addendum for more); and consistency is automatic from assuming that $\mathcal{A}$ consists only of sentences true in **N**. Thus **L1** follows from **L4** (as well as from **L2**).

There are a couple of minor differences here from some versions of Gödel's theorem, besides the fact that it applies more widely than merely to 1st-order number theory. The first is mentioned just after the theorem. More interestingly, hypothesis (2) about $\mathcal{A}$ is frequently stated in a way that appears to be stronger, but doesn't actually weaken the theorem. It is usually assumed that the set $\mathcal{A}$ is a *decidable* set. A subset of formulae being decidable means (unrigorously!) that there is a machine (i.e. procedure) which accepts as input arbitrary formulae, and outputs "yes" or "no" according as the input is a formula in the subset or not. This would be a "Type D algorithm", in the notation of Chapters 4 and 8. As before, the machine is programmed once and for all, but has no upper bound on the memory or the number of steps that can be needed. It is not unreasonable to regard a set of formulae as pretty well useless as an axiom set if there is no algorithm for deciding

whether an arbitrary formula is an axiom.

Some care is needed here in reading texts, where the word "decidable" is used in two closely related ways. We are definitely *not* assuming that $\mathcal{A}^{\vdash}$ is a decidable set. But often the phrase "$\mathcal{A}$ is decidable" is used to mean just that, thinking of $\mathcal{A}$ as defining the *theory* $\mathcal{A}^{\vdash}$. In fact, for most $1^{\underline{\text{st}}}$order languages, taking $\mathcal{A}$ to be the empty set $\emptyset$, so that only logical axioms are used in formal proofs, even the set $\emptyset^{\vdash}$ of logically provable formulae is not a decidable set. (See the fourth paragraph after **L5** ahead.) But $\emptyset^{\vdash}$ is certainly always a listable set, due to the nature of $1^{\underline{\text{st}}}$order proof systems. It is not hard to see that decidable implies listable. The failure of the converse is a very non-trivial fact, as stated above concerning $\emptyset^{\vdash}$. This is equivalent to the complement of $\emptyset^{\vdash}$ in the set of all formulae *not* being listable, since the decidable sets turn out to be the listable ones whose complements are also listable. (Give an argument for this!)

To come back to the point, a little trick due to Craig shows that, in Gödel's theorem, assuming decidability is really the same as assuming listability for the axiom set: If $\mathcal{A}$ is listable, then there is a decidable $\mathcal{B}$ for which we have $\mathcal{B}^{\vdash} = \mathcal{A}^{\vdash}$, since if $\mathcal{A}$ is listed as $F_1,\ F_2,\ F_3,\ \cdots$, one can simply take $\mathcal{B}$ to be F_1 , $F_1 \wedge F_2$, $F_1 \wedge F_2 \wedge F_3$, $\cdots$. The fact that the latter list has formulae increasing in length makes decidability easy to see. This trick also takes care of the claim before the strong version of **L1** that $\mathcal{A}$ listable implies $\mathcal{A}^{\vdash}$ listable—it is straightforward (but dependent on the details of the proof system) that $\mathcal{B}$ decidable implies $\mathcal{B}^{\vdash}$ listable. This last fact includes the degenerate case saying that $\emptyset^{\vdash}$ is listable, as claimed above. It seems that "listable according to some criterion" (e.g. listable by length, listable lexicographically) always implies decidable. Perhaps this points to part of the subtlety in the existence of a listable, but undecidable, set. See **[CM]** for much more on this.

Below we'll show how Gödel's theorem follows from a theorem of Church, so the reader will be left mainly with the task of reading and thinking enough to master rigorous definitions of decidability and listability, and the highly non-trivial proof of Church's theorem, in order to really understand what's going on here. See also the **Addendum**, where these are both proved using without proof some very important technical results of Gödel.

But the basic idea in Gödel's proof, at a fairly naive level, is too attractive

to leave without comment. To begin you should realize that part of learning 1$^{\underline{\text{st}}}$order logic consists of becoming adept at the art of translating back in forth between natural languages (including ordinary mathematical jargon) and the 1$^{\underline{\text{st}}}$order languages. And I say "art" advisedly, because there is no uniqueness in either direction.

Now Gödel's first stroke of genius here is to translate the sentence "I am a formula which is not formally provable from $\mathcal{A}$" into the 1$^{\underline{\text{st}}}$order language of arithmetic. See the **Addendum**, where a strict notion of translating, called expressing, is explained. This is where the sentences in $\mathcal{N}$ (first page of this Appendix) are needed. It involves a process which includes assigning numbers to all symbols, to all formulae, and to all formal proofs. That process, given at the start of the **Addendum**, is known as Gödel numbering.

Let G be the resulting 1$^{\underline{\text{st}}}$order sentence, translating Gödel's sentence. To conclude the 'proof'—it is not a proof, even leaving aside the lack of mathematical definitions for listable, etc. and the lack of detail for obtaining G, simply because a proof cannot rely on some vague notion of 'translation'—we suppose that G is formally provable from $\mathcal{A}$. But then G is true in every *model for* $\mathcal{A}$ (i.e. every interpretation in which all formulae in $\mathcal{A}$ are true). This includes any model where G means "I am a formula which is not formally provable from $\mathcal{A}$". But this is a contradiction. Therefore G is certainly not formally provable from $\mathcal{A}$. But now, because of what it says, G is in fact true in the latter model. Being false in at least one model for $\mathcal{A}$, the sentence $\neg G$ is also not formally provable from $\mathcal{A}$.

It is clear from previous remarks that the last paragraph is far from being a proof of **L4**, though it does give a version of the striking ideas behind the proof. But even just the *idea* of translating "I am a formula which is not formally provable from $\mathcal{A}$" becomes a bit slippery when one asks: Which interpretation of the language is this sentence supposed to be speaking about? It can't always be just **N** itself, since the theorem might apply to $\mathcal{A}$ which contain sentences false in **N**. See the Addendum, where this is made clear—it turns out that G can be correctly interpreted that way in all interpretations which are models for $\mathcal{N}$.

Note that an interesting alternative proof, due to Boolos, of essentially the **L1** version of the incompleteness theorem, is sketched in some detail in **Appendix VQ** to Chapter 5 of this book. It uses the Berry paradox rather than Gödel's adaptation of the Liar paradox.

There is a less famous but equally important theorem of Gödel (proved a few years earlier while he was a graduate student) called the *completeness theorem for* $1^{\underline{\text{st}}}$*order logic*. (This is the main content of Chapter 7 of this book.) It says that, *for any* $1^{\underline{\text{st}}}$*order language and any set,* $\mathcal{A}$*, of formulae, if a formula* F *is true in all models for* $\mathcal{A}$*, then* $F \in \mathcal{A}^{\vdash}$. The converse of this is relatively routine, since the proof system would have some 'incorrect' logical axiom or rule of inference, if the converse didn't hold. We've used the converse several times on earlier pages. The theorem itself is saying that the $1^{\underline{\text{st}}}$order proof system itself is adequately strong. There are many choices possible for an acceptable proof system. Besides completeness, the other basic criterion of acceptability is that the set of logical axioms should be decidable, as should the set of rules of inference, in a suitable sense. So the real content of Gödel's completeness theorem is that a proof system exists which is simultaneously compete and decidable. It is normally proved by exhibiting a specific proof system which is manifestly decidable, and then proceeding to prove completeness, often as sketched two paragraphs below.

Note that (despite being called completeness) the completeness theorem *doesn't* say that, for every G, either G or $\neg G$ must be in $\mathcal{A}^{\vdash}$. That would only happen if all models for $\mathcal{A}$ satisfied exactly the same $1^{\underline{\text{st}}}$order formulae.

It follows by combining completeness and incompleteness that, for the language $\{S, +, \times, <, 0\}$ and any $\mathcal{A}$ as in **L1**, and for a given $G \in \mathcal{T}_{\mathbf{N}} \setminus (\mathcal{A}^{\vdash})$, there is a model $\mathbf{P}$ for $\mathcal{A}$ such that G is not true in $\mathbf{P}$, despite being true in $\mathbf{N}$. Such a $\mathbf{P}$, for the number theory axioms considered fundamental at the time, had been called a non-standard model of number theory in decades past (see e.g. [**Kleene**; pp.327-8]). But nowadays this phrase is restricted to the use mentioned earlier, that is, to an interpretation which has some higher order properties differing from those of $\mathbf{N}$, but with the same $1^{\underline{\text{st}}}$order properties.

To sketch a proof of completeness, if $\mathcal{A}$ is *not* consistent, there is nothing to prove since $\mathcal{A}^{\vdash}$ contains all formulae. If it *is* consistent, and assuming $F \notin \mathcal{A}^{\vdash}$, we get that $\mathcal{A} \cup \{\neg F\}$ is consistent. But *any consistent set,* $\mathcal{B}$*, of formulae has a model—in fact a countable (possibly finite) model.*

A first attempt at this would build the model right out of the language—take $\mathbf{M}$ to be the set of equivalence classes of terms which don't involve variables, two terms t_1 and t_2 being equivalent by definition iff $t_1 \approx t_2 \in \mathcal{B}^{\vdash}$. This attempt only works under suitable conditions. For example it cannot

work if the language has no constant symbols at all, since a model is by definition non-empty. In the case of our language of number theory, it gives the set of all "k", which looks a lot like the set of natural numbers. In this example one needs consistency to be sure these terms are all inequivalent. The proof in the general case uses this idea plus some technicalities, and is due to Henkin. (It is **7.15** earlier in this book.)

Going back to the proof of completeness, applying this to $\mathcal{B} = \mathcal{A} \cup \{\neg F\}$ gives a model for $\mathcal{A}$ in which F is false, as required.

Henkin's model existence theorem also yields the compactness theorem quite easily, since a set of formulae is certainly consistent as long as every finite subset is.

Finally we'll state an absolutely fundamental theorem of Church concerning our $1^{\underline{\text{st}}}$order language of number theory.

Theorem L5. (Church) *For any $\mathcal{A}$ for which $\mathcal{A} \cup \mathcal{N}$ is* consistent, *the set $\mathcal{A}^{\vdash}$ is not decidable.*

Going back to the beginning, and assuming that the natural numbers exist, we can take $\mathcal{A}$ in the theorem to be $\mathcal{T}_{\mathbf{N}}$, immediately yielding a proof of **L2**, since a listable consistent complete set of sentences is easily seen to be decidable (Turing). Here "consistent complete set" means a set containing exactly one of F or $\neg F$ for every sentence F.

On the other hand, Gödel's Theorem also follows, since $\mathcal{A}$ listable implies $\mathcal{A}^{\vdash}$ listable, and so the latter cannot be a complete set or else Turing would contradict Church.

However, in some fundamental sense, it is better not to think of Gödel's Theorem as 'weaker' than Church's. The point is that decidability occurs in the assumptions of the former and the conclusion of the latter. If a wider definition of *algorithm* were to someday surface, the former would likely remain true whereas the latter might not. The latter depends on Church's thesis, which basically says that the present-day definition of *computable*, and hence of *decidable*, is the only sensible one, whereas Gödel's Theorem is independent of Church's thesis.

Yet a third application takes $\mathcal{A} = \emptyset$, showing that $\emptyset^{\vdash}$ is undecidable (as long as $\mathcal{N}$ is consistent, of which one is assured by belief in the existence of the 'completed infinity', the set of all natural numbers). This very important

fact is due independently to Church and to Turing.

The second incompleteness theorem.

Surely by now the reader is wondering what Gödel's *second* incompleteness theorem might be. It led to abandonment of attempts to prove the consistency of axiom sets for arithmetic by elementary combinatorial means, because it is understood to make obvious that this can't even be done by means of the theory of numbers which the axiom set itself generates.

To begin, consider something which should lead you to realize that careful formulations are needed here. Suppose given any $1^{\underline{\text{st}}}$order language and axiom set $\mathcal{A}$ with the (vague) property that the sentence "$\mathcal{A}$ is consistent" can be translated into the language. Below we claim that this is the case for any $\mathcal{A}$ containing enough arithmetic. But this translation ability itself seems to imply that $\mathcal{A}$ *is* consistent, as follows. Call the translation C. Either $C \in \mathcal{A}^{\vdash}$ or it doesn't. If it does, we have a proof of consistency (a formal proof). If it doesn't, we have consistency since $\mathcal{A}^{\vdash}$ is not the set of all formulae (C isn't in it). But something must be wrong here, since surely we can just add a sentence to $\mathcal{A}$ to make it obviously inconsistent without destroying the ability to translate the consistency sentence.

Gödel showed that a very precise version of this "translation ability" holds in the language of arithmetic as long as $\mathcal{A}$ contains $\mathcal{N} \cup \{(1^{\underline{\text{st}}}\mathbf{IND})\}$. Recall that $\mathcal{N}$ is the set of nine sentences on the first page of this appendix, and $(1^{\underline{\text{st}}}\mathbf{IND})$ is the $1^{\underline{\text{st}}}$order axiom scheme of induction. For such $\mathcal{A}$ which are listable, **L4** gives (for Gödel's sentence G from a few pages back)

$$\text{"}\mathcal{A} \text{ consistent implies that } G \text{ is not provable from } \mathcal{A}\text{"} \ .$$

So it says

$$\text{" } \underline{\neg 0 \approx 0 \text{ is not provable from } \mathcal{A}} \text{ implies that } \underline{\underline{G \text{ is not provable from } \mathcal{A}}} \text{ " } .$$

Now the double underlined can be translated into the formal language as G, and the single underlined as C. So the entire line translates as

$$C \to G \ .$$

Furthermore, and with much elbow grease, an actual proof of Gödel's first incompleteness theorem (using $\mathcal{N} \cup \{(1^{\underline{\text{st}}}\mathbf{IND})\}$, not just $\mathcal{N}$) can be converted

into a formal proof of $C \to G$ from $\mathcal{A}$. Thus, if there were a formal proof of C from $\mathcal{A}$, modus ponens would immediately give a formal proof of G from $\mathcal{A}$. But this contradicts Gödel's first theorem, *assuming that $\mathcal{A}$ is consistent.* So we have proved

Theorem L6. (Gödel's second incompleteness theorem) *If $\mathcal{A}$ is consistent, listable, and contains $\mathcal{N} \cup \{(1^{\underline{\text{st}}}\textbf{IND})\}$, then $C \notin \mathcal{A}^{\vdash}$, where C is a sentence with interpretation "$\neg 0 \approx 0$ is not provable from $\mathcal{A}$".*

Alternatively perhaps "Any consistent listable extension of $\mathcal{N} \cup \{(1^{\underline{\text{st}}}\textbf{IND})\}$ is not strong enough to prove its own consistency". Of course, this might be vacuous—$\mathcal{N} \cup \{(1^{\underline{\text{st}}}\textbf{IND})\}$ itself is conceivably inconsistent, but don't let that stop you from wanting to do mathematics !

To ask an unfair question, what *is* wrong in the 4th previous paragraph? What it seems to be saying, more generally, is that the following 'paradoxical' state of affairs exists. Fix a $1^{\underline{\text{st}}}$order language, and a set $\mathcal{A}$ from it. An informal proof that a sentence, whose interpretation is "$\mathcal{A}$ is consistent" has *no* formal proof from $\mathcal{A}$ seems to be an informal proof that $\mathcal{A}$ is consistent. Thus discovery of a formal proof from $\mathcal{A}$ of a sentence with interpretation "$\mathcal{A}$ is consistent" possibly **decreases** one's confidence in $\mathcal{A}$'s consistency.

Hopefully, skipping around from one high point to another, as I've done, will whet your appetite, rather than spoil it, for getting down to the basic theory of $1^{\underline{\text{st}}}$order logic, the decidable-listable questions, and the proofs of Church's and Henkin's theorems (and also *model theory*, a very active area which applies formal logic to mathematics, rather than the other way round). For a start, you could have a look at the following addendum.

ADDENDUM. How the Gödel Express implies theorems of Church, Gödel and Tarski.

First let's be explicit about Gödel numbering. Any $1^{\underline{\text{st}}}$order language with finitely or countably many non-logical symbols has a countably infinite number of symbols in all (allowing for an infinite list of variables). So, list these symbols in some explicit way. For example, our number theory alphabet can be listed

$$S \quad + \quad \times \quad < \quad 0 \quad (\quad) \quad \approx \quad \neg \quad \vee \quad \wedge \quad \to \quad \leftrightarrow \quad \forall \quad \exists \quad x_1 \quad x_2 \quad x_3 \quad \cdots\cdots \quad .$$

Any finite string, $S = s_1 s_2 \cdots s_n$, of these symbols is then given the Gödel number

$$g(S) := p_1^{w(s_1)} p_2^{w(s_2)} \cdots p_n^{w(s_n)} \ ,$$

where

$$w(s_i) \text{ is the position of } s_i \text{ in the list} \quad (\text{e.g. } w(\exists) = 15)$$

and

$$p_1 < p_2 < \cdots < p_n \text{ are the first "}n\text{" primes} \ .$$

It is fairly clear that there are procedures which behave as in (1) to (4) below.

(1) input S ; output $g(S)$.

(2) input $a \in \mathbf{N}$; output $g^{-1}(a)$ (defined to be that S with $g(S) = a$) . To be thorough take the empty string as $g^{-1}(a)$ for those a not in the image of g (including $a = 1$) .

(3) input S ; output 0 or 1 according as S is or isn't a formula.

(4) input $a \in \mathbf{N}$; output 0 or 1 according as $g^{-1}(a)$ is or isn't a formula. We say that algorithm (4) "decides whether or not a is a formula#". And we write "formula#a" for $g^{-1}(a)$ when a is a formula#.

Furthermore, given a finite string, $\mathcal{S} = S_1 S_2 \cdots S_k$, of symbol strings, let

$$\hat{g}(\mathcal{S}) \ := \ p_1^{g(S_1)} p_2^{g(S_2)} \cdots p_k^{g(S_k)}$$

It is clear that there are procedures which behave as below in $(\hat{1})$, $(\hat{2})$, and also $(\hat{4})$, given $(\hat{3})$.

$(\hat{1})$ input $\mathcal{S}$; output (S).

$(\hat{2})$ input $b \in \mathbf{N}$; output $\hat{g}^{-1}(b)$.

Given a *decidable* set, $\mathcal{A}$, of formulae:

$(\hat{3})$ input $\mathcal{S}$; output 0 or 1 according as the string $\mathcal{S}$ is or isn't a formal proof using $\mathcal{A}$.

$(\hat{4})$ input $b \in \mathbf{N}$; output 0 or 1 according as $\hat{g}^{-1}(b)$ is or isn't a formal proof using $\mathcal{A}$.

The existence of $(\hat{3})$ is only clear after a careful study of the $1^{\underline{\text{st}}}$order proof system being used. One works down the list of formulae in $\mathcal{S}$. For each formula, first decide whether it is a non-logical axiom, using the decidability of the set $\mathcal{A}$. If not, check if it's a logical axiom. If not, finally try all the possibilities for using rules of inference with earlier formulae to produce it.

The functions g and $\hat{g}$ are our Gödel numberings, of formulae and proofs, respectively.

By the **Gödel express**, I mean the following very important technical theorem due to Gödel. It demonstrates the relationship of the set $\mathcal{N}$ of nine sentences on the first page of this appendix to the idea of "expressing", using our number theory language. This is a precise, syntactic version of "translating" between the formal language and ordinary mathematics, with the meaning of a $1^{\underline{\text{st}}}$order sentence being the same in all interpretations which are models for $\mathcal{N}$.

Abbreviate x_1 and x_2 to x and y respectively. Recall that $F^{[x \to t]}$ is the formula obtained from F by substituting the term t for all free occurrences of x. See the remarks before **L1**. Define the term "k" to be $SS \ldots S0$, using a total of k copies of S.

(Unary Case.) *Suppose given a procedure with input $a \in \mathbf{N}$ and output 0 or 1 [= say $f(a)$]. Then there is a formula F in our number theory language such that*

$$f(a) = 0 \quad \Longrightarrow \quad F^{[x \to \text{“}a\text{”}]} \in \mathcal{N}^{\vdash} \ ;$$
$$f(a) = 1 \quad \Longrightarrow \quad \neg F^{[x \to \text{“}a\text{”}]} \in \mathcal{N}^{\vdash} \ .$$

This says that the procedure, or the *computable function* f, or the subset $f^{-1}(0) \subset \mathbf{N}$ (also called a *predicate* or a *unary relation* with *characteristic function* f) is expressible in $1^{\underline{\text{st}}}$order number theory, being expressed by F.

(Binary Case.) *Suppose given a procedure with input $(a, b) \in \mathbf{N} \times \mathbf{N}$ and output 0 or 1 [= say $e(a, b)$]. Then there is a formula E such that*

$$e(a,b) = 0 \quad \Longrightarrow \quad E^{[x \to \text{“}a\text{”} \ , \ y \to \text{“}b\text{”}]} \in \mathcal{N}^{\vdash} \ ;$$
$$e(a,b) = 1 \quad \Longrightarrow \quad \neg E^{[x \to \text{“}a\text{”} \ , \ y \to \text{“}b\text{”}]} \in \mathcal{N}^{\vdash} \ .$$

Here the binary relation $e^{-1}(0)$ is being expressed by the number theoretic formula E. For example, taking E to be $x < y$, we find that that formula expresses the $<$ relation in $\mathbf{N}$. As a further illustration, there are proofs from $\mathcal{N}$ of $SS0 < SSS0$ and of $\neg SSS0 < S0$, corresponding to the facts that $2 < 3$ is true and that $3 < 1$ is false.

Note that if we assume the meaningfulness of $\mathcal{T}_{\mathbf{N}}$ and that $\mathcal{N} \subset \mathcal{T}_{\mathbf{N}}$ (which is hardly drastic, but unnecessary for the first two of the following three proofs), then it follows that in, for example, the binary case

$$e(a,b) = 0 \quad \Longleftrightarrow \quad E^{[x \mapsto \text{“}a\text{”},\ y \mapsto \text{“}b\text{”}]} \ \in \ \mathcal{T}_{\mathbf{N}} \ .$$

This says that e is *definable* in $1^{\underline{\text{st}}}$order number theory, a property which is weaker than being expressible. See the remarks after Tarski's Theorem at the end of this appendix.

On the other hand, the Gödel express itself is a syntactic, constructive approach, well suited to the proofs that follow. It could itself be formulated (say, in the unary case) as being a procedure: whose input is pairs (P, a), with P being a description of a procedure and a being a description of an integer, say by its digits; and whose output is the empty string if a is not an appropriate input for P, or if it is but then P outputs something other than 0 or 1; and whose output is otherwise $(F, \mathcal{F})$, where F is a formula, and $\mathcal{F}$ is a formal proof from $\mathcal{N}$ of either $F^{[x \mapsto \text{“}a\text{”}]}$ or $\neg F^{[x \mapsto \text{“}a\text{”}]}$, depending on whether P produced 0 or 1 respectively, when run with input a.

Continuing, there are ternary, 4-ary, $\cdots$ versions which we won't need here. This basic result is not totally surprising, but is certainly not trivial to prove, no matter which of the several equivalent mathematical definitions of *procedure* is adopted. Think about the converse: given an arbitrary function into $\{0\ ,\ 1\}$, assuming that a $1^{\underline{\text{st}}}$order formula exists to express it, is it true that a procedure exists to compute it? (Remember that $\mathcal{N}^{\vdash}$ is listable.) Note that with $\mathcal{N}^{\vdash}$ replaced by $\mathcal{A}^{\vdash}$ for any $\mathcal{A} \supset \mathcal{N}$, the "Gödel express" still evidently holds. And also, after replacing it, that both $\Longrightarrow$ $'s$ are strengthenable to $\Longleftrightarrow$ $'s$ when $\mathcal{A}$ is consistent. We are not even assuming that $\mathcal{N}$ is consistent, but if it isn't, there would be no content left in the Gödel express (though that would be the least of our worries) ! See the final few paragraphs of this Addendum for more discussion.

Assuming this Gödel express, below are otherwise complete proofs of the

theorem of Church, of Gödel's ω-consistency version of his first incompleteness theorem, and of a theorem of Tarski. All that is needed about "procedures" has already been stated in this appendix, so the definition to be adopted isn't now needed for these proofs, as long as you'll accept that the specific procedure used in each proof will qualify as one. Just after the three proofs we have indicated in more detail exactly what needs to be added for complete intellectual honesty.

For proving Church's Theorem, L5, first note that we may assume $\mathcal{A} \supset \mathcal{N}$, since adding a finite set of sentences to a set $\mathcal{A}$ with $\mathcal{A}^{\vdash}$ decidable will not change that decidability. This follows by adding one sentence G at a time, and noting that

$$G \in (\mathcal{A} \cup \{F\})^{\vdash} \iff F \to G \in \mathcal{A}^{\vdash} .$$

So, in effect, a procedure to decide membership in $\mathcal{A}^{\vdash}$ is used to give one for deciding whether $G \in (\mathcal{A} \cup \mathcal{N})^{\vdash}$ by applying the earlier procedure to $F_1 \wedge \cdots \wedge F_9 \to G$, where $\mathcal{N} = \{F_1, \cdots, F_9\}$. Thus $\mathcal{A}$ may be replaced by $\mathcal{A} \cup \mathcal{N}$ in proving the non-existence of a procedure.

So $\mathcal{A}$ is a consistent extension of $\mathcal{N}$, and, for a contradiction, assume that $\mathcal{A}^{\vdash}$ is decidable. Apply the unary version of the Gödel express to the following procedure:

Input $a \in \mathbf{N}$.

Check if a is a formula#.

If no, output 1.

If yes, check if $(\text{formula}\#a)^{[x \to \text{“}a\text{”}]} \in \mathcal{A}^{\vdash}$. (Use decidability.)

If yes, output 1.

If no, output 0.

For this procedure, let F be a formula guaranteed to exist by the Gödel express. Thus, using the consistency of $\mathcal{A}$ and the definition of the Gödel express,

$$(*) \quad F^{[x \to \text{“}a\text{”}]} \in \mathcal{A}^{\vdash} \iff a \text{ is a formula\# and } (\text{formula}\#a)^{[x \to \text{“}a\text{”}]} \notin \mathcal{A}^{\vdash} .$$

Let

$$(**) \qquad p := g(F) , \text{ the Gödel number of } F ; \text{ so } F \text{ is (formula } \#p) .$$

But then we get

$$F^{[x\mapsto\text{“}p\text{”}]} \in \mathcal{A}^{\vdash}$$

$$(*)\ \Updownarrow$$

$$(\text{formula}\#p)^{[x\mapsto\text{“}p\text{”}]} \notin \mathcal{A}^{\vdash}$$

$$(**)\ \Updownarrow$$

$$F^{[x\mapsto\text{“}p\text{”}]} \notin \mathcal{A}^{\vdash},$$

a contradiction. Hence $\mathcal{A}^{\vdash}$ is not decidable, as required.

For proving Gödel's Theorem, L4, first recall the remarks after its statement that we may take $\mathcal{A}$ to contain $\mathcal{N}$, and may assume that $\mathcal{A}$ is decidable, not just listable. We'll find a G with neither G nor $\neg G$ in $\mathcal{A}^{\vdash}$. The proof is arranged so as to put off to the very end the definition of ω-*consistency*. But we'll use the weaker assumption that $\mathcal{A}$ is consistent several times before that. The word "proof" below is short for "formal proof, using $\mathcal{A}$ as the set of non-logical axioms".

Apply the binary version of the Gödel express to the following procedure.

Input $(a, b) \in \mathbf{N}$.
Is a a formula#?
If no, output 1.
If yes, is b a proof# ?
If no, output 1. (This uses decidability.)
If yes, find that proof, and find (formula#a).
Is $(\text{formula}\#a)^{[x\mapsto\text{“}a\text{”}]}$ the last line of proof#b?
If no, output 1.
If yes, output 0.

For this procedure, let E be the formula from the Gödel express. We may assume that the only free variables in E are x and y by adding some $\forall$'s at the front. Using the consistency of $\mathcal{A}$ and the nature of the Gödel express,

$$(*)\ \ E^{[x\mapsto\text{“}a\text{”},\ y\mapsto\text{“}b\text{”}]} \in \mathcal{A}^{\vdash} \iff b \text{ and } a \text{ are proof and formula } \#'\text{s and}$$

$$\text{proof}\#b \text{ proves } (\text{formula}\#a)^{[x\mapsto\text{“}a\text{”}]},$$

and

$$\neg E^{[x\mapsto\text{“}a\text{”},\ y\mapsto\text{“}b\text{”}]} \in \mathcal{A}^{\vdash} \text{ for all other } (a, b) \in \mathbf{N} \times \mathbf{N} \ .$$

Let p be the Gödel number of $\forall y \neg E$, and let

$$G \ := \ \forall y \neg E^{[x \mapsto \text{“}p\text{”}]} \ = (\text{formula}\#p)^{[x \mapsto \text{“}p\text{”}]} \ .$$

For a contradiction, suppose that $G \in \mathcal{A}^{\vdash}$. Choose $b \in \mathbf{N}$ so that proof$\#b$ shows this. So proof$\#b$ proves $(\text{formula}\#p)^{[x \mapsto \text{“}p\text{”}]}$. By $(*)$, we get

$$(**) \qquad E^{[x \mapsto \text{“}p\text{”},\ y \mapsto \text{“}b\text{”}]} \ \in \ \mathcal{A}^{\vdash} \ .$$

Next write down any formal proof from $\mathcal{A}$ (proof$\#b$, if you like) which shows that $G \in \mathcal{A}^{\vdash}$. The last line is G, but now add a new last line, namely $\neg E^{[x \mapsto \text{“}p\text{”}\ ,\ y \mapsto \text{“}b\text{”}]}$. This is justified by the rule of inference which says that one can drop the first two symbols, $\forall y$, in $G = \forall y \neg E^{[x \mapsto \text{“}p\text{”}]}$, and substitute for all free y's any fixed substitutable term (such as “b”, here). This new, slightly longer proof shows that

$$\neg E^{[x \mapsto \text{“}p\text{”},\ y \mapsto \text{“}b\text{”}]} \ \in \ \mathcal{A}^{\vdash} \ .$$

Combined with $(**)$, this contradicts the consistency of $\mathcal{A}$. So we have established that $G \notin \mathcal{A}^{\vdash}$.

Finally suppose that $\neg G \in \mathcal{A}^{\vdash}$. We'll deduce that $\neg G \notin \mathcal{A}^{\vdash}$, which proves the latter by contradiction. Consistency of $\mathcal{A}$ disallows $G \in \mathcal{A}^{\vdash}$, so for no $b \in \mathbf{N}$ is b a proof$\#$ such that proof$\#b$ proves $G = (\text{formula}\#p)^{[x \mapsto \text{“}p\text{”}]}$. By $(*)$, for all b, we have that $\neg E^{[x \mapsto \text{“}p\text{”},\ y \mapsto \text{“}b\text{”}]} \in \mathcal{A}^{\vdash}$. But ω-consistency of a set, $\mathcal{B}$, of formulae is defined to say that $\mathcal{B}^{\vdash}$ cannot contain all of $F^{[z \mapsto \text{“}b\text{”}]}$ for all b as well as $\neg \forall z F$ (for a formula F and a variable z). Take $\mathcal{B} = \mathcal{A}$, $z = y$ and $F = \neg E^{[x \mapsto \text{“}p\text{”}]}$. Thus $\neg \forall y \neg E^{[x \mapsto \text{“}p\text{”}]}$ is not in $\mathcal{A}^{\vdash}$, i.e. $\neg G \notin \mathcal{A}^{\vdash}$. This completes the proof.

Remarks. (a) G will be a sentence as long as x and y are the only free variables in E. This is normally part of the conclusion of the Gödel express. Consistency follows from ω-consistency by taking z to be any variable which does not occur freely in F. Perhaps ω-consistency looks more natural if one replaces $\neg \forall z F$ by the logically equivalent formula $\exists z \neg F$.

(b) Some people would feel quite comfortable reformulating condition $(*)$ as:

$$E^{[x \mapsto \text{“}a\text{”},\ y \mapsto \text{“}b\text{”}]} \quad \text{says} \quad \text{“}b \text{ and } a \text{ are proof and formula } \#\text{'s and}$$

$$\text{proof}\#b \text{ proves } (\text{formula}\#a)^{[x \mapsto \text{“}a\text{”}]} \text{ ”} \ .$$

And from this and the fact that p *is* a formula#, they would deduce:

$$(\forall y \neg E)^{[x \mapsto \text{“}p\text{”}\ ]} \quad \text{says} \quad \text{“ for no } y \text{ is } y \text{ a proof \#}$$

$$\text{such that proof\#}y \text{ proves } (\text{formula\#}p)^{[x \mapsto \text{“}p\text{”}]} \text{ ”} \ ,$$

or simply:

$$(\forall y \neg E)^{[x \mapsto \text{“}p\text{”}\ ]} \quad \text{says} \quad \text{“(formula\#}p)^{[x \mapsto \text{“}p\text{”}]} \text{ is not provable”} \ .$$

But both of the formulae in the display just above are actually G. So they would simply assert:

G says “G is not provable” , or even just, “I am not provable” !!

(And “ provable” really means “formally provable from $\mathcal{A}$.” But those people might not bother to say what they mean by “says” !!)

Finally, here's a theorem saying *“truth is not definable”*, or more modestly:

Theorem L7.(Tarski) *Assume that $\mathcal{T}_{\mathbf{N}}$ is meaningful and that $\mathcal{N} \subset \mathcal{T}_{\mathbf{N}}$. (In particular, $\mathcal{N}$ is consistent.) Then there is no formula F in $1^{\underline{\text{st}}}$order number theory such that, for all $a \in \mathbf{N}$,*

$$[a \text{ is a formula\#} \ \text{ and } \ (\text{formula\#}a) \in \mathcal{T}_{\mathbf{N}}] \iff F^{[x \mapsto \text{“}a\text{”}]} \in \mathcal{T}_{\mathbf{N}} \ .$$

It follows that there is no F such that: (i) the right-hand side can be replaced by $F^{[x \mapsto \text{“}a\text{”}]} \in \mathcal{N}^{\ \vdash}$ and (ii) for all other a, $\neg F^{[x \mapsto \text{“}a\text{”}]} \in \mathcal{N}^{\ \vdash}$. And so, by the Gödel express, there is no procedure for deciding whether a given formula is true in $\mathbf{N}$. But that followed from Church's Theorem as well: $\mathcal{T}_{\mathbf{N}}$ is consistent, so $\mathcal{T}_{\mathbf{N}} = \mathcal{T}_{\mathbf{N}}^{\ \vdash}$ is not decidable.

Proof of Tarski's Theorem. For a contradiction, assume that F exists. Note that it suffices to find a formula G such that, for all $a \in \mathbf{N}$,

$$(*) \quad [a \text{ is a formula\# and } (\text{formula\#}a)^{[x \mapsto \text{“}a\text{”}]} \notin \mathcal{T}_{\mathbf{N}}] \iff G^{[x \mapsto \text{“}a\text{”}]} \in \mathcal{T}_{\mathbf{N}} \ .$$

For if p is the Gödel number of G, we get a contradiction by letting $a = p$:

$$(\text{formula\#}p)^{[x \mapsto \text{“}p\text{”}]} \notin \mathcal{T}_{\mathbf{N}} \iff (\text{formula\#}p)^{[x \mapsto \text{“}p\text{”}]} \in \mathcal{T}_{\mathbf{N}} \ .$$

So, using the postulated F, let's find such a G satisfying $(*)$. By the Gödel express, there is a formula H such that, for all $(a, b) \in \mathbf{N} \times \mathbf{N}$,

$$\begin{array}{c} a \text{ and } b \text{ are formula}\#'\text{s, and } (\text{formula}\#a)^{[x \to \text{“}a\text{”}]} = \text{ formula}\#b \\ \Updownarrow \\ H^{[x \to \text{“}a\text{”},\ y \to \text{“}b\text{”}]} \in \mathcal{N}^{\vdash} \\ \Updownarrow \qquad\qquad (**) \\ H^{[x \to \text{“}a\text{”},\ y \to \text{“}b\text{”}]} \in \mathcal{T}_{\mathbf{N}} \ ; \end{array}$$

and, for all other (a, b),

$$\neg H^{[x \to \text{“}a\text{”},\ y \to \text{“}b\text{”}]} \in \mathcal{N}^{\vdash} \ .$$

Let F_+ be F with all occurrences of the variable y changed to an x_j not already occurring in F, and with $j \geq 3$. (For example, use the smallest such j.) Then the condition in the theorem statement holds with F_+ in place of F. Let $F_- := \neg F_+^{[x \to y]}$. Then, for all $b \in \mathbf{N}$,

$$[b \text{ is not a formula}\# \text{ or } (\text{formula}\#b) \notin \mathcal{T}_{\mathbf{N}}] \iff F_-^{[y \to \text{“}b\text{”}]} \in \mathcal{T}_{\mathbf{N}} \qquad (***)$$

Let $G := \exists y(H \wedge F_-)$. To check $(*)$, first note that

$$G^{[x \to \text{“}a\text{”}]} \ = \ (\exists y(H \wedge F_-))^{[x \to \text{“}a\text{”}]} \ = \ \exists y(H^{[x \to \text{“}a\text{”}]} \wedge F_-) \ ,$$

partly because F_- has no free occurrences of x. So, for all a,

$$\begin{array}{c} G^{[x \to \text{“}a\text{”}]} \in \mathcal{T}_{\mathbf{N}} \\ \Updownarrow \\ \exists y(H^{[x \to \text{“}a\text{”}]} \wedge F_-) \in \mathcal{T}_{\mathbf{N}} \\ \Updownarrow \\ \text{for some } b: \quad H^{[x \to \text{“}a\text{”} \ ,\ y \to \text{“}b\text{”}]} \in \mathcal{T}_{\mathbf{N}} \ \text{ and } \ F_-^{[y \to \text{“}b\text{”}]} \in \mathcal{T}_{\mathbf{N}} \\ \Updownarrow \text{ by } (**) \text{ and } (***) \\ \text{for some } b: \ a \text{ and } b \text{ are formula}\#'\text{s and } (\text{formula}\#a)^{[x \to \text{“}a\text{”}]} = \ (\text{formula}\#b) \notin \mathcal{T}_{\mathbf{N}} \\ \Updownarrow \\ a \text{ is a formula}\# \text{ and } (\text{formula}\#a)^{[x \to \text{“}a\text{”}]} \notin \mathcal{T}_{\mathbf{N}} \ \ , \end{array}$$

as required, completing the proof.

What is the idea here? Our 'loose speaker' in remark (b) after the last proof (and from **Appendix VQ**) would probably speak as follows:

If they existed, both F and F_+ would say
"x is a formula#, and formula#x is true."
Therefore, F_- would say
"either y is not a formula#, or formula#y is not true."
On the other hand, $H^{[x\mapsto\text{“}a\text{”},\ y\mapsto\text{“}b\text{”}]}$ says
"a and b are formula#'s, and (formula#a)$^{[x\mapsto\text{“}a\text{”}]}$ = formula#b ."
Combining the last two, $G^{[x\mapsto\text{“}a\text{”}]}$ would say
"for some y: [either y is a not formula# or formula# y is not true] and [both a and y are formula#'s and (formula#a)$^{[x\mapsto\text{“}a\text{”}]}$ =formula#y]",
or, more simply,
"a is a formula#, and (formula#a)$^{[x\mapsto\text{“}a\text{”}]}$ is not true."
Therefore, $G^{[x\mapsto\text{“}p\text{”}]}$ would say
"(formula#p)$^{[x\mapsto\text{“}p\text{”}]}$ is not true."
But formula#p *is* G, so $G^{[x\mapsto\text{“}p\text{”}]}$ would say
"I am not true."

Thus, Tarski has used Gödel's technical triumph of numbering and expressing to turn the Liar Paradox into a theorem of considerable content, which possibly gives confirmation to the philosophical opinion that the paradox is due to illicit self-reference. See [**Barwise and Etchemendy**] for sophisticated discussion of the liar paradox, particularly the semantic side of it.

The arguments of Gödel and Tarski take the following parallel forms :

Gödel :
Provability **is** expressible by a formula F.
So there is a formula G_F which says *"I am not provable"*.
The formula G_F cannot be false, because of its meaning, since otherwise it would be provable, hence true.
Therefore G_F is not provable because of its meaning, and its negation is not provable because its negation is false.

Tarski :
Suppose that truth were definable by a formula F.
Then there would be a formula T_F which says *"I am not true"*.
The formula T_F cannot be false, because of its meaning.
The formula T_F cannot be true, because of its meaning.
Therefore, truth is not definable.

For the sake of intellectual honesty.

In addition to what's actually in this book, what exactly needs to be added to solidify our treatment of these three major theorems? A central need is for a precise definition of the word *algorithm* (or *procedure*). See **[CM]**, where this is discussed in very considerable detail. For these theorems, we need to define what were called "Type D" algorithms earlier in the book, and only ones in which the output set is $\{0, 1\}$, and the input set is either $\mathbf{N}$ or $\mathbf{N} \times \mathbf{N}$. Several such definitions have been made over the years, known to be all equivalent. A notable one is Turing machines. (Closely related are recursive functions, defined several paragraphs below.)

> *The search for a precise mathematical characterization of what "algorithm" and "computable function" should mean resulted half a century ago in the discovery of three well-known equivalent approaches $\cdots$:λ-definability (Church-Kleene, 1932-34); general recursiveness (Gödel-Herbrand, 1934) and Turing machines (1936). $\cdots$ Church proclaimed his famous Thesis, which identifies the notion of computable function with the formal notion of λ-definable function $\cdots$ it was only after Turing's work that Gödel accepted Church's Thesis which had then become the Church-Turing Thesis. This is the way the miracle occurred: the essence of a process which can be carried out by purely mechanical means was understood and incarnated in precise mathematical definitions.*
>
> Boris A. Trakhtenbrot

In defining the general notion of algorithm, it is always more convenient to deal with more general input/output and to include "Type semi-D" (i.e. *infinite loops* or *partial* recursive functions), equivalent to what we called "Type L" algorithms.

Having given such a definition, algorithms whose input set consists of symbol strings, or formulae, or finite lists of formulae, can be quickly reduced to algorithms just using numbers by means of Gödel numbering.

Using the adopted definition, one easily shows that the specific procedure occurring in each of the three proofs *give* computable functions in the sense of that definition, that the Gödel numbering procedures exist, and that the argument in the initial paragraph of the proof of Church's theorem is correct. Then, more challenging, one must prove what I called the *Gödel express*, connecting computable functions to syntactical questions in $1^{\underline{st}}$order number theory. We discuss this a bit more now.

In this book, we've given two incomplete proofs, of two versions of the incompleteness

theorem—one just above based on "I am not provable", and one in **Appendix VQ** based on Berry's paradox. As mentioned above, the main gap in the proofs is the assertion without proof of what I've called the Gödel express. However, for the earlier proof, the version of that is slightly different, more semantic. It is: *listable implies definable*; as opposed to: *decidable implies expressible*. Below is a table of implications to hopefully sort out any confusion about the connection between these two. It relates mainly to binary relations e [but a 3-ary relation f also appears]. All this generalizes to k-ary e [and $(k+1)$-ary f], including the case $k = 1$.

e is *decidable* : $\exists$ algorithm : input (a,b); output $e(a,b)$.	$\Longleftrightarrow$	e is a (total) *recursive* function.	$\Longleftrightarrow$	e is *expressible* : $\exists$ a formula F with $e(a,b)=0 \Rightarrow F^{[x\mapsto \text{“}a\text{”},\ y\mapsto \text{“}b\text{”}]} \in \mathcal{N}^{\vdash}$; $e(a,b)=1 \Rightarrow \neg F^{[x\mapsto \text{“}a\text{”},\ y\mapsto \text{“}b\text{”}]} \in \mathcal{N}^{\vdash}$.
$\Updownarrow$		$\Downarrow$		$\Downarrow$
e is *semi-decidable* : $\exists$ algorithm : input (a,b); output yes, if $e(a,b)=0$; runs forever, if $e(a,b)=1$.	$\Longleftrightarrow$	$e\vert_{e^{-1}(0)}$ is a *partial recursive* function.	$\Longleftrightarrow$	e is *weakly expressible* : $\exists$ a formula F with $e(a,b)=0 \iff F^{[x\mapsto \text{“}a\text{”},\ y\mapsto \text{“}b\text{”}]} \in \mathcal{N}^{\vdash}$.
$\Updownarrow$				$\Downarrow$
$e^{-1}(0)$ is *recursively enumerable* or *listable* : $\exists$ an algorithm which lists the set $\{(a,b) : e(a,b)=0\}$.				
$\Updownarrow$				
$\exists$ decidable $f : \mathbf{N}^3 \to \{0,1\}$ such that $[e(a,b)=0 \iff \exists c$ with $f(a,b,c)=0]$.				
				e is *definable* : $\exists$ a formula F with $e(a,b)=0 \iff F^{[x\mapsto \text{“}a\text{”},\ y\mapsto \text{“}b\text{”}]} \in \mathcal{T}_{\mathbf{N}}$.

Except for the implication *decidable implies expressible* (the Gödel express), and the upwards non-implication in the diagram, the rest are not really difficult; many are trivial. The non-implication is very nearly equivalent to Church's theorem.

The fact that *semi-decidable implies definable* is basically the one used in Boolos' proof of the incompleteness theorem given in **Appendix VQ**.

The Gödel express has no consistency assumptions at all, but would be trivial to any weird person who believes that $\mathcal{N}$ is not consistent. Its converse does depend on $\mathcal{N}$ being consistent.

The fact that *semi-decidable implies weakly expressible* is quickly reduced to the Gödel express applied in the 3-ary case (to f in the diagram). Roughly, if G expresses f, then $\exists z G$ weakly expresses e. This argument depends on $\mathcal{N}$ being ω-consistent. Its converse depends on $\mathcal{N}$ being consistent.

By making a list of all triples, i.e. listing $\mathbf{N}^3$, it is not difficult to see how, for the lower left $\Updownarrow$, to get f from e, and to get an algorithm for e from one for f.

The proof that *weakly expressible implies definable* depends on the fact, obvious to

the unweird, that $\mathcal{N} \subset \mathcal{T}_{\mathbf{N}}$. Then one direction is trivial. For the other direction, be sure you use $F = \exists z G$ from two paragraphs above.

As for *recursive functions*, here are the definitions:

A *partial recursive function* is any function $g : D \to N$, where $D \subset \mathbf{N}^k$ for some k, for which there is a 'recursive verification column'; that is, a finite sequence of functions $g_1, \cdots, g_m = g$ for which each g_i satisfies at least one of (I),(II) or (III) below.

(I) It can be one of the three functions $\mathbf{N} \times \mathbf{N} \to \mathbf{N}$ as follows: the addition function; or the multiplication function; or the function which maps (x, y) to 0 or 1 depending on whether $x < y$ or not. Or else it can be the kth projection function $\mathbf{N}^j \to \mathbf{N}$ which maps $(a_1, \cdots, a_j)$ to a_k, where $0 < k \leq j$.

(II) If $i > 1$, g_i can be a composition of earlier functions—that is, for some $F, H_1, \cdots, H_j$, all occurring earlier in the sequence, we have:

domain$(F) \subset \mathbf{N}^j$;

for some ℓ and all s, domain$(H_s) \subset \mathbf{N}^\ell$;

the domain of g_i is

$$\{\, (a_1, \cdots, a_\ell) \in \bigcap_{s=1}^{j} \text{domain}(H_s) \;:\; [H_1(a_1, \cdots, a_\ell),\ \ldots\ , H_j(a_1, \cdots, a_\ell)] \in \text{domain}(F) \,\}\ ; \text{ and}$$

for all $(a_1, \cdots, a_\ell)$ in the domain of g_i above,

$$g_i(a_1, \cdots, a_\ell) \;=\; F[H_1(a_1, \cdots, a_\ell),\ \ldots\ , H_j(a_1, \cdots, a_\ell)]\ .$$

(III) If $i > 1$, g_i can be a minimization of an earlier function—that is, for some F occurring earlier in the sequence, we have:

domain$(F) \subset \mathbf{N}^{j+1}$ for some $j > 0$;

domain$(g_i) \;=\; \{\, (a_1, \cdots, a_j) \;:\; \exists k$ with $F(k, a_1, \cdots, a_j) = 0 \,\}$; and

for all $(a_1, \cdots, a_j)$ in domain(g_i) above,

$$g_i(a_1, \cdots, a_j) \;=\; \min\{k : F(k, a_1, \cdots, a_j) = 0\}.$$

A *(total) recursive function* may be defined to be any partial recursive function whose domain happens to be all of $\mathbf{N}^k$ for some k.

The connections in the table of implications tieing (partial) recursivity to (semi-) decidability, in the absence of a definition of *algorithm*, may be taken as a kind of axiom. This is virtually the same as Church's thesis. For any given definition of *algorithm*, e.g. via Turing machines, the proof of that connection is fairly long and tedious, but not subtle.

The form, *recursive implies expressible*, of the Gödel express is more straightforward than trying to connect expressibility directly to any of the usual definitions of *algorithm*. This was basically the form in which Gödel first proved it.

Exercise. Show that the set of Appendix labels in this book is (mathematically) definable.

Logic, like whiskey, loses its beneficial effect when taken in too large quantities.

Lord Dunsany

References and Self-References

Barwise, Jon *An introduction to first-order logic*. pp.5–46 Handbook of Mathematical Logic, ed. Jon Barwise, North-Holland, 1977.

Barwise, J. and Etchemendy, J. *The Liar—An Essay on Truth and Circularity*. Oxford U. Press, New York and Oxford, 1987.

Bell, J.L. and Machover, M. *A Course in Mathematical Logic*. North Holland, Amsterdam, 1977.

Boolos, G. *The Logic of Provability*. Cambridge U. Press, Cambridge, 1993. (Successor to : *The Unprovability of Consistency*. Cambridge U. Press, Cambridge, 1979.)

Burris, S.N. *Logic for Mathematics and Computer Science*. Prentice Hall, New Jersey, 1998

Christensen, N.E. *Mathematical Logic and Mathematical Reasoning*. (negative reference!) pp. 175-188 in 5th Scandinavian Logic Symposium, Aalborg Univ. Press, 1979.

[CM] *Computability for the Mathematical*. this website, 2006.

Coxeter, H.S.M. *Introduction to Geometry*. Wiley, New York, 1961.

Ebbinghaus, H.D., Flum, J. and Thomas, W. *Mathematical Logic*. Undergraduate Texts in Mathematics, Springer-Verlag, 1994.

Enderton *A Mathematical Introduction to Logic*. Academic Press, New York, 1972.

Feferman, S. *In the Light of Logic*. Oxford U. Press, New York and Oxford, 1998.

Forder, H.G. *The Foundations of Euclidean Geometry*. Dover, New York, 1958.

Gordon, Michael J.C. *Programming Language Theory and its Implementation. Prentice Hall, London, 1988.*

Greenberg *Euclidean and Non-Euclidean Geometries. Freeman, San Francisco, 1980.*

Hailperin, T. *Boole's Logic and Probability. Studies in Logic and the Foundations of Mathematics*, **85**, *Elseveier, Amsterdam, 1976.*

Hilbert, D. *The Foundations of Geometry. Open Court, Chicago, 1902.*

Hilbert, D. and Ackermann, W. *Mathematical Logic. (translation of 1928 Springer original) Chelsea, New York, 1950.*

Hodel, Richard E. *An Introduction to Mathematical Logic. PWS Publishing, 1995.*

Kleene, Stephen Cole *Mathematical Logic. John Wiley & Sons, 1967.*

Lyndon, R. *Notes on Logic. Van Nostrand, Princeton, 1966.*

Manin, Yu. I. *A Course in Mathematical Logic. Graduate Texts in Mathematics, Springer-Verlag, 1977.*

Mendelson, E. *Introduction to Mathematical Logic. 3rd ed., Wadsworth & Brooks/Cole, Monterey, 1987.*

Penrose, Roger *The Emperor's New Mind. 2nd ed., Oxford University Press, Oxford, 1998.*

Penrose, Roger *Shadows of the Mind. Oxford University Press, Oxford, 1995.*

Rosser, B. *Logic for Mathematicians. Chelsea, New York, 1953.*

Russell, B. and and Whitehead, A.N. *Principia Mathematica. Cambridge, 1910-12-13.*

Schöning, U., *Logic for Computer Scientists. Birkhäuser, Berlin, 1989.*

Index

INDEX of NOTATION