1

Introduction

1.1. Truth

The most fundamental notion in classical logic is that of truth. Philosophers, of course, have long debated the question 'what is truth?', but that is a debate which, for the purposes of the present book, we must leave to one side. Let us assume that we know what truth is.

We are concerned with truth because we are concerned with the things that are true, and I shall call these things 'propositions'. Philosophers, again, hold differing views on what is to count as a proposition. A simple view is that a proposition is just a (declarative) sentence, but when one thinks about it for a moment, there are obvious difficulties for this suggestion. For the same sentence may be used, by different speakers or in different contexts, to say different things, some of them true and others false. So one may prefer to hold that it is not the sentences themselves that are true or false, but particular utterings of them, i.e. utterings by particular people, at particular times and places, in this or that particular situation. A more traditional view, however, is that it is neither the sentences nor the utterings of them that are true, but a more abstract kind of entity, which one can characterize as *what is said* by one who utters a sentence. Yet a further view, with a longer history, is that what one expresses by uttering a sentence is not an abstract entity but a mental entity, i.e. a judgement, or more generally a thought. Again, we must leave this debate on one side. Whatever it is that should properly be said to be true, or to be false, that is what we shall call a proposition. At least, that is the official position. But in practice I shall quite often speak loosely of sentences as being true or false. For whatever propositions are, they must be closely associated with sentences, since it is by means of sentences that we express both truths and falsehoods.

We assume, then, that there are these things called propositions, and that every one of them is either true or not. And if it is not true, we say that it is false. So there are just two truth-values, truth and falsehood, and each proposition has exactly one of them. In fact we assume more strongly that a given proposition has, *in every possible situation*, just one of these two truth-values, so that when we have considered the case in which it is true, and the case in which it is false, no *possibility* has been omitted. Since the vast majority of the propositions that we actually express in daily life suffer from vagueness in one way or another, one must admit that this assumption is something of an idealization. For with a vague proposition there are some situations in which it seems natural to say that it is neither true nor false, but classical logic makes no allowance for this. For the most part this idealization seems to do no harm, but there are occasions when it leads to trouble, i.e. when we apparently get the wrong result by applying the precise rules of classical logic to the vague propositions of everyday life.[1] But, once more, for the purposes of the present book we can only note the problem and pass by on the other side, with the excuse that our present subject is not the application of logical theory but the development of the theory itself. And that theory does depend upon the stated assumption about propositions and truth. Indeed, that assumption is what distinguishes classical logic from most of its rivals.

In developing our theory of logic we shall wish to speak generally of all propositions, and we introduce the schematic letters '*P*','*Q*','*R*',... to facilitate this. They are called sentence-letters (or, in some books, propositional letters) because they are to be understood as standing in for, or taking the place of, sentences which are or express propositions. We can therefore generalize by letting such a letter represent any proposition, arbitrarily chosen. But we shall also speak of 'interpreting' a sentence-letter, or assigning an 'interpretation' to it, and it is natural to say that here we are thinking of the letter as representing some particular and specified proposition. That is just how one does proceed when applying logical theory, for example to test actual arguments containing actual propositions. However, for our purposes in

[1] The best-known example is the so-called 'Sorites paradox'. See e.g. C. Wright, 'Language-Mastery and the Sorites Paradox', in G. Evans and J. McDowell (eds.), *Truth and Meaning* (Oxford University Press: Oxford, 1976).

this book, the *only* feature of the assigned proposition that will ever be relevant is its truth-value. So in fact we shall 'interpret' a sentence-letter just by assigning to it a truth-value, either T (for truth) or F (for falsehood). We shall not pause to specify any particular proposition which that letter represents and which has the truth-value in question.

1.2. Validity

The word 'valid' is used in a variety of ways, even within the orthodox terminology of logic. But its primary application is to arguments, so we may begin with this.

In an argument some propositions are put forward as premisses, and another proposition is claimed to follow from them as conclusion. Of course, an actual case will often involve rather more than this, for the arguer will not just *claim* that his conclusion follows from his premisses; he will also try to *show* (i.e. to prove) that it does, and this may involve the construction of long and complicated chains of reasoning. It is only in rather simple cases that a mere claim is deemed to be enough. Nevertheless, the classical definition of validity ignores this complication, and it counts an argument as valid if and only if the conclusion does in fact follow from the premisses, whether or not the argument also contains any demonstration of this fact. To say that the conclusion does follow from the premisses is the same as to say that the premisses do entail the conclusion, and on the classical account that is to be defined as meaning: it is impossible that all the premisses should be true and the conclusion false. Once more, we must simply leave on one side the philosophers' debate over the adequacy of this definition, either as a definition of validity or as a definition of entailment.

Now logic is often characterized as the study of validity in argument, though in fact its scope is very much narrower than this suggests. In what is called elementary logic we study just two ways in which an argument may be valid, namely (1) when its validity is wholly due to the truth-functional structure of the propositions involved, and (2) when it is due to both truth-functional and quantificational structure working together.[2] In other areas of logic, not usually called elementary, one studies the contribution to validity of various other features of propositions, for example their tense or modality. But there is no end to the list of propositional features that *can*

[2] If the words 'truth-functional' and 'quantificational' are not familiar, then please be patient. Detailed explanations will come in the next two chapters.

contribute to validity, since *any* necessary connection between premisses and conclusion will satisfy the definition, and it would be foolish to suppose that some one subject called 'logic' should study them all. In response to this point it used to be said that logic is concerned with 'form' rather than with 'content', and accordingly that its topic can be circumscribed as 'validity in virtue of form'. My impression is that that suggestion is not looked upon with much favour these days, because of the difficulty of making any suitable sense of the notion of 'form' being invoked. In any case, I mention the point only to set it aside, along with the many other interesting problems that affect the very foundations of our subject. So far as this book is concerned, we will confine attention just to the way that truth-functional and quantificational complexity can affect validity. (But later we shall add a brief consideration of identity.)

Because our subject is so confined, we can usefully proceed by introducing what are called 'formal languages', in which the particular kind of complexity that we are studying is the *only* complexity that is allowed to occur at all. For example, to study the effects of truth-functional complexity we shall introduce a 'language' in which there are symbols for certain specified truth-functions—and these, of course, are assigned a definite meaning—but all the other symbols are merely schematic. Indeed, in this case the other symbols will be just the schematic sentence-letters already mentioned. They will occupy positions where one might write a genuine sentence, expressing a genuine proposition, but they do not themselves express any propositions. Accordingly, this so-called 'formal language' is not really a *language* at all, for the whole point of a language is that you can use it to say things, whereas in this 'formal language' nothing whatever can be said. So it is better regarded, not as a language, but as a *schema* for a language—something that would become a language if one were to replace its schematic letters by genuine expressions of the appropriate type (in this case, sentences). Let us say, then, that we shall introduce language-schemas, in which the particular kinds of complexity that we are interested in will be represented, but everything else will be left schematic.

The 'sentences' of such a language-schema are similarly not really sentences, but sentence-schemas, picking out particular patterns of sentence-construction. We shall call them 'formulae'. By taking several such formulae as our premiss-formulae, and another as a conclusion-formula, we can represent an argument-schema, which again is a pattern of argument which many particular arguments will exemplify. Then, in a new use of the word 'valid', we may say that an argument-schema is to be counted as a *valid schema* if and only if every actual argument that exemplifies it is a *valid* *argument*, in the sense defined earlier (i.e. it is impossible that all its premisses should be true and its conclusion false). It is the validity of these argument-schemas that we shall actually be concerned with. At least, that is the basic idea, though in practice we shall set up our definitions a little differently.

When any formal language is introduced, we shall specify what is to count as an 'interpretation' of it. At the moment, we have introduced just one such language, namely the language which has as its vocabulary just the sentence-letters '*P*','*Q*','*R*',..., and nothing else. In this *very* simple language, each sentence-letter is a formula, and there are no other formulae. Moreover, we have explained what is to count as interpreting a sentence-letter, namely assigning to it either T or F as its value. So this tells us how to interpret every formula of the language. We therefore know what it would be to consider all interpretations of some specified set of formulae. Suppose, then, that we take an argument-schema in this language. It will consist of some set of sentence-letters, each of which is to be counted as a premiss-formula, together with a single sentence-letter to be counted as the conclusion-formula. Then we shall say that such an argument-schema counts as a valid schema if and only if *there is no interpretation* in which each of the premiss-formulae comes out true and the conclusion-formula comes out false. (With the present very simple language, it is clear that this will be the case if and only if the conclusion-formula is itself one of the premiss-formulae.)

When the argument-schema is valid in this sense, then it will *also* be valid in the sense first suggested, i.e. every actual argument that exemplifies the schema will be a valid argument. Why so? Because when we consider 'every interpretation' of the schema, we are thereby considering 'every possibility' for the arguments that exemplify the schema, and this in turn is because—as I stressed in Section 1.1—we are assuming that a proposition must always be either true or false, and there is no third possibility for it.

The formal languages that we shall actually be concerned with in the remainder of this book are, of course, rather more complicated than the very simple example just given, but the same general principles will continue to apply. When the language is introduced, we shall specify what is to count as an interpretation of it, and the aim will be to ensure that the permitted interpretations cover all the possibilities. Provided that this is achieved, the results that we obtain for our formal or schematic languages by looking at all interpretations of them will carry with them results about what is and is not possible in the genuine languages that exemplify them. For example, if we have a formula that is not true under any interpretation, then all the propositions exemplifying that formula will be propositions that cannot possibly be true. This is the relationship required if the study of formal languages is to be a significant contribution to the study of validity in arguments, as classically conceived. But, for *most* of what follows, this relationship will simply be assumed; it will be the formal languages themselves that directly engage our attention.

1.3. The Turnstile

Just as an argument is valid (according to the classical definition) if and only if its premisses entail its conclusion, so we may also say that an argument-schema is a valid schema if and only if its premiss-formulae entail its conclusion-formula. This uses the word 'entails' in a new way, to signify a relation between formulae, and that is how the word will be used from now on. In fact it proves more convenient to work with this notion of entailment, rather than the notion of an argument-schema being valid, so I now introduce the sign '$\models$' to abbreviate 'entails' in this sense. The sign is pronounced 'turnstile'. But before I proceed to a formal definition it will be helpful to introduce some further vocabulary, of the kind called 'metalogical'.

At the moment, our only formulae are the sentence-letters. Let us now specify these a little more precisely as the letters in the infinite series

$$P, Q, R, P_1, Q_1, R_1, P_2, \ldots$$

These are schematic letters, taking the place of sentences which are or express propositions, and used to speak generally about all propositions. More kinds of formulae will be introduced shortly. But whatever kind of formulae is under consideration at any stage, we shall wish to speak generally about all formulae of that kind, and for this purpose it will be useful to have some further schematic letters which take the place of formulae. I therefore introduce the small Greek letters

$$\varphi, \psi, \chi, \varphi_1, \psi_1, \chi_1, \varphi_2, \ldots$$

in this role.[3] Their function is like that of the sentence-letters, but at one level up. For they take the place of formulae, while formulae take the place of genuine sentences expressing propositions. I also introduce the capital Greek letters

[3] 'φ','ψ','χ' are spelled 'phi', 'psi', 'chi' respectively, and pronounced with a long 'i' in each case. The 'c' in 'chi' is hard (as in Scottish 'loch').

$$\Gamma,\Delta,\Theta,\Gamma_1,\ldots,$$

whose role is to generalize, in an analogous way, not over single formulae but over *sets* of formulae.[4] Using this vocabulary we can say that the basic notion to be defined is

$$\Gamma \vDash \varphi,$$

where φ is any formula and Γ is any set of formulae. And the definition is

> There is no interpretation in which every formula in Γ is true and the formula φ is false.

Any sentence that exemplifies the schema '$\Gamma \vDash \varphi$', with actual formulae in place of the metalogical schematic letters 'Γ' and 'φ', will be called a *sequent*. A sequent, then, makes a definite claim, that certain formulae are related in a particular way, and it is either true or false.

My introduction of the capital Greek letters 'Γ', 'Δ',... was a little curt, and indeed some further explanation is needed of how all our metalogical letters are actually used in practice. As I have said, the turnstile '$\vDash$' is to be understood as an abbreviation for 'entails'. Grammar therefore requires that what occurs to the right of this sign is an expression that refers to a formula, and what occurs to the left of it is an expression—or a sequence of expressions—referring to several formulae, or to a set of formulae, or a sequence of formulae, or something similar. But in standard practice the letter 'φ' is used to take the grammatical place, not of an expression which *refers to* a formula, but of an expression which *is* a formula. Similarly the letter 'Γ' is used to take the grammatical place, not of an expression that *refers to* one or more formulae, but of one or more expressions that *are* formulae. To illustrate this, suppose that we wish to say that if you take any set of formulae Γ, and if you form from it a (possibly) new set by adding the particular formula 'P' to its members, then the result is a set of formulae that entails the formula 'P'. Apparently the correct way of writing this would be

$$\Gamma \cup \{\text{'}P\text{'}\} \vDash \text{'}P\text{'},$$

(where '$\cup$' indicates the union of two sets, and the curly brackets round 'P' mean 'the set whose only member is "P"'). But in practice we never do use the notation in this way. Instead, we write just

$$\Gamma, P \vDash P.$$

[4] 'Γ', 'Δ', 'Θ' are spelled 'gamma', 'delta', 'theta' respectively, and the 'e' in 'theta' is long. (The corresponding lower-case Greek letters are 'γ', 'δ', 'θ'.)

Similarly, if we wish to generalize and say that the same holds for any other formula in place of 'P', then we write

$$\Gamma,\varphi \vDash \varphi.$$

Supposing, then, that '$\vDash$' really does abbreviate the verb 'entails', the notation that we actually use must be regarded as the result of the following further conventions:

(1) where an expression to the left of '$\vDash$' specifies a set by using the sign '$\cup$' of set union, this sign is always to be replaced by a comma;
(2) where an expression to the left of '$\vDash$' specifies a set by listing its members, and enclosing the list in curly brackets, the curly brackets are always to be omitted;
(3) quotation marks, needed in English to form from a formula an expression which refers to that formula, are always to be omitted.

So it comes about that in actual practice we avoid both the use of quotation marks, and the explicitly set-theoretical notation, that the explanation of '$\vDash$' as 'entails' appears to demand.

It may seem more natural, then, to adopt a different explanation of '$\vDash$', not as abbreviating the verb 'entails', but simply as representing the word 'therefore'. What grammar requires of an ordinary use of the word 'therefore' is that it be preceded by one or more whole sentences, stating the premisses of the argument, and followed by another whole sentence, stating its conclusion. Of course it would be quite wrong to enclose each of these sentences in its own quotation marks. So when we abstract from this an argument-schema, which many different arguments may exemplify, we shall naturally do this just by writing formulae in place of the original sentences, again without adding any quotation marks. And similarly when we wish to generalize about our argument-schemas, we shall do this by using 'φ' to take the place of any formula, and 'Γ' to take the place of any sequence of formulae. So the grammar that is actually used with the turnstile, not only in this book but (so far as I am aware) in every other, is very much more natural if we take it to mean 'therefore' rather than 'entails'.

There is of course a difference between the two interpretations. On the first approach, whereby '$\vDash$' means 'entails', the schema '$\Gamma \vDash \varphi$' is a schema whose instances are sentences which make a definite claim, true or false. On the second, whereby '$\vDash$' means 'therefore', the schema '$\Gamma \vDash \varphi$' is a schema whose instances are argument-schemas, such as 'P; not both P and Q; therefore not Q'. An argument-schema does not itself make any claim at all;

rather, we may make claims about that schema, e.g. the claim that it is valid. So, on this second approach, if one wishes to claim that the formulae Γ entail the formula φ one writes not

$$\Gamma \vDash \varphi$$

but

'$\Gamma \vDash \varphi$' is valid.

In practice, it makes very little difference which interpretation is adopted. Some books use the one, others use the other, and in several cases the sign appears to be being used in both ways at once. But no serious confusion results.

In this book I shall adopt the first interpretation, and what is written to the left of '$\vDash$' will be taken as indicating a set of formulae, even though that may not be what the notation naturally suggests.

One reason for this—not a very important one—is that the order in which the premiss-formulae are listed, and the number of times that any formula occurs in the list, evidently make no difference to the correctness of an entailment claim. This is automatically catered for if we say that what is in question is the *set* of all the premiss-formulae, since it will still be the same set whichever way we choose to list its members, so long as it is the same members that are listed. (But of course we could obtain this result in other ways too, as we shall do in Chapter 7.) The more significant reason is that the notion of a *set* of premiss-formulae very naturally includes two cases which we shall want to include, but which would be unnatural as cases of arguments or argument-schemas. These are the case when we have infinitely many premisses, and the case when we have none at all. The idea of an argument with no premisses—an 'argument' which begins with the word 'therefore' (i.e. 'for that reason') referring back to no statement previously given as a reason—is certainly strange; so too is the idea of an argument with so many premisses that one could *never* finish stating them, and so could never reach the stage of drawing the conclusion. But if we are speaking simply of what is entailed by this or that set of propositions (or formulae), then these two cases are less strange. In any case I stipulate that they are to be included: the set of formulae Γ may be infinite, and it may be empty. Both cases are automatically covered by the definition already given.

It may be noted that, in accordance with our convention for omitting curly brackets to the left of the turnstile, we shall write simply

$$\vDash \varphi$$

to say that the formula φ is entailed by the empty set of formulae, and its definition can of course be simplified to

> There is no interpretation in which φ is false.

At a later stage in the book (Chapter 7) I shall generalize the definition of the turnstile so that what is to the right of it may also be a set of formulae, and not just a single formula. I do not introduce that generalization now, since in the earlier chapters there would be no use for it. But it is convenient to introduce now what is, in effect, one special case of the generalization to come later: we shall allow that what is to the right of the turnstile may be either a single formula or no formula, and consequently a new definition is needed now for the case where there is no formula to the right. It is easy to see what this definition should be, namely

$$\Gamma \models$$

is to mean

> There is no interpretation in which every formula in Γ is true.

Any instance of the schema '$\Gamma \models$', with actual formulae in place of 'Γ', will also be called a sequent.

It is worth noting at once that our definition includes the special case in which Γ is empty, so that in the notation we actually use there are no formulae either to the right or to the left of the turnstile, and we are faced with just this claim:

$$\models.$$

This is a false claim. It says that there is no interpretation in which every formula in the empty set is true. But there is such an interpretation, indeed *any* interpretation whatever will suffice, including the interpretation in which every sentence-letter is assigned F. For since there are no formulae in the empty set anyway, it follows that there are none which are not true, in this interpretation and in any other. (As always in logic, we understand 'Every *A* is *B*' to mean the same as 'There is no *A* which is not *B*', and so it is true if there is no *A* at all.) Here we have reached our first result about '$\models$', namely that when it stands by itself to make a claim about the empty set of formulae, it is false. It is convenient to write '$\not\models$' in place of '$\models$' to express the negation of what '$\models$' expresses. Using this convention, we can set down our result in this way:

$$\not\models.$$

But perhaps it is less confusing to express the point more long-windedly in English: the empty sequent is false.

Further results about '$\models$' are best postponed until we have introduced the formulae to which it will relate. Meanwhile, let us summarize what has been said so far. In logic we study sequents, which have the turnstile '$\models$' as their main verb. In the standard case, a sequent

$$\Gamma \models \varphi$$

will have several formulae to the left of the turnstile, and one formula to the right, and in this case the turnstile abbreviates 'entails'. But we also allow for a sequent of the form

$$\Gamma \models$$

with no formula on the right. In this case the turnstile can be read as 'is inconsistent'. And we allow too for a sequent of the form

$$\models \varphi$$

with no formula on the left. In this case we shall say that the sequent claims that the formula φ is valid. Note that this is yet a third use of the word 'valid', in which it is applied not to an argument, nor to an argument-schema, but to a single formula. This is the only way in which the word will be used henceforth. Despite these different ways of reading the turnstile in English, depending on whether one or other side of the sequent is empty, nevertheless it is recognizably the same notion in each case. For every sequent claims:

> There is no interpretation in which everything on the left is true and everything on the right is false.

2

Truth-Functors

The most elementary part of logic is often called 'propositional logic' (or 'sentential logic'), but a better title for it is 'the logic of truth-functors'. Roughly speaking, a truth-functor is a sign that expresses a truth-function, so it is the idea of a truth-function that first needs attention.

2.1. Truth-Functions

A truth-function is a special kind of function, namely a function from truth-values to truth-values.

Functions in general may be regarded as rules correlating one item with another. A function will be 'defined on' items of some definite kind (e.g. numbers), and these items are the possible inputs to the function. To each such item as input, the function assigns another item (or possibly the same item) as its output for that input. The outputs may be items of the same kind as the inputs, or they may be items of a different kind. For example, the expression 'the square of...' expresses a function defined on numbers; given any number x as input to the function, the function yields another number, namely x^2, as its output for that input. Similarly, the expression 'the father of...' expresses a function defined on people; given any person x as input, the function yields another person, namely the father of x, as its output for that input. So the first is a function from numbers to numbers, and the second a function from people to people. In each of these cases the outputs are items of the same kind as the inputs, but a function does not have to be like this. For example, 'the number of...'s children' expresses a function from people to numbers. The important thing about a function is just that it does always have one and only one output for each input of the specified kind. We call the input to the function an 'argument' to the function, and its output for that input is called its 'value' for that argument. Thus the function expressed by 'the square of...' has the value 4 for the argument 2, the value 9 for the argument 3, the value 16 for the argument 4, and so on.

A truth-function is a function which takes truth-values as arguments and which yields truth-values as values; that is to say, it is a function from truth-values to truth-values. A nice simple truth-function is the one which yields F as value for T as argument and T as value for F as argument. It is briefly specified by this truth-table:

Argument	Value
T	F
F	T

This is an example of a *one-place* truth-function (also called a *unary*, or *monadic*, truth-function). There are not many one-place truth-functions. (In fact there are only three others. Write down their truth-tables.) But there are also *two-place* truth-functions (also called binary, or dyadic), and *three-place* truth-functions (also called ternary, or triadic), and so on indefinitely. It is natural to think of a two-place function as taking two arguments simultaneously, and this is perfectly all right, so long as one distinguishes them as the *first* argument and the *second*. Alternatively, one can think of a two-place function as taking just one argument, where that one argument is an ordered *pair* of items. In that case, the truth-functions should be described as functions which take as arguments either single truth-values, or ordered pairs of truth-values, or ordered trios of truth-values, and so on. (To express the point generally, it is usual to speak of ordered n-tuples.) But if we speak in the first way, which is perhaps more natural, then the truth-functions are functions which take as arguments one or more truth-values, in a specified order. The values of a truth-function are always single truth-values.

For example, among the two-place truth-functions there is one specified by the following truth-table:

First argument	Second argument	Value
T	T	T
T	F	F
F	T	T
F	F	T

Among the three-place truth-functions there is one specified by the following truth-table:

First argument	Second argument	Third argument	Value
T	T	T	T
T	T	F	F
T	F	T	F
T	F	F	F
F	T	T	F
F	T	F	F
F	F	T	F
F	F	F	T

It is clear that with the two-place function just specified the order of the arguments does make a difference, for the function takes the value F only when the *first* argument takes the value T, and the *second* takes the value F. But, as it happens, the order of the arguments is irrelevant to the three-place function just specified: it takes the value T when and only when all its three arguments take the *same* value, and this does not depend upon which order they are taken in.

In general, the number of n-place truth-functions is 2^{2^n}. Thus, as already mentioned, there are 4 one-place functions. We can add that there are 16 two-place functions, 256 three-place functions, and so on. In the other direction there are 2 *zero-place* functions. Admittedly it is stretching the notion of a function somewhat to suppose that there could be such a thing as a zero-place function. Such a 'function' is not in any natural sense a 'correlation', but it can be regarded as something that has an 'output' for a 'zero input'. For example, among functions from numbers to numbers one might regard a particular numeral, say '2', as expressing a zero-place function; it requires no number as input, and it yields the number 2 as output. Similarly among functions from truth-values to truth-values one may regard as a sign for a zero-place function any symbol that always takes the value T as its value, without requiring any argument to enable it to do so, and similarly any symbol that always takes the value F as its value. I shall shortly introduce signs that work just like this.

EXERCISES

2.1.1. Write out the truth-tables of the following truth-functions:

(*a*) The three-place function which takes the value T if and only if just one of its arguments takes the value T.

(*b*) The three-place function which takes the value T if and only if at least two of its arguments take the value T.

(*c*) The three-place function that takes the value T if and only if its first argument takes the value F.

2.1.2. Describe the relation between the three-place function in 2.1.1(*c*) and the two-place function that takes the value T if and only if its first argument takes the value F. Can a two-place function be the *same* function as a three-place function?

2.1.3. Write out the truth-tables of all the two-place truth-functions. Estimate how long it would take you to do the same for all the three-place truth-functions.

2.2. Truth-Functors

Given any proposition 'P' one can form from it another proposition which is its negation. In English this is usually done by inserting the word 'not' in some appropriate place in the sentence expressing it, though ambiguity is better avoided by tediously writing out 'It is not the case that' at the front of the sentence. In this book we shall use '$\neg$' as our negation sign, written in front of what it negates, as in '$\neg P$'. (Some other books use '$-$' or '$\sim$' instead.) Similarly, given any propositions 'P' and 'Q' one can form from them another proposition which is their conjunction. In English this is usually done by writing the word 'and' between the two sentences in question, though again one can prevent some possible misunderstanding by using instead the long-winded construction 'It is the case both that . . . and that . . .'. In this book we shall use '$\wedge$' as our conjunction sign, written between what it conjoins, as in '$P \wedge Q$'. (Some other books use '&' or '.' instead.)

A word such as 'not' or 'and', when used in this way, may be regarded as expressing a function from sentences to sentences; for you supply it with a sentence, or a pair of sentences, as input, and it forms from them a new sentence as its output for that input. So we may call it a sentence-functor. Clearly there is no end to the different ways of forming new sentences from given sentences, but we shall at once confine our attention to those, such as inserting 'not' or 'and', that are truth-functional. This means simply that the sentence-functor which gives a sentence as output for one or more sentences as input corresponds to a truth-function, namely the truth-function which yields the truth-value of the output sentence as its output when it is given the truth-value(s) of the input sentence(s) as its input. A sentence-functor which corresponds in this way to a truth-function will be called, simply, a truth-functor.

Just as a truth-function is given by a truth-table, so too a corresponding truth-functor is also characterized by the same truth-table. For example, the negation sign '$\neg$' has this truth-table:

P	$\neg P$
T	F
F	T

And the conjunction sign '$\wedge$' has this truth-table.

P	Q	$P \wedge Q$
T	T	T
T	F	F
F	T	F
F	F	F

This tells us that the negation '$\neg P$' is false when 'P' is true and true when 'P' is false; and that the conjunction '$P \wedge Q$' is true if both of its conjuncts are true, and false otherwise. To put this kind of information succinctly, let us write '$|P|$' as an abbreviation for 'the truth-value of "P"', and similarly for any other letter in place of 'P'. Let us also abbreviate 'if and only if' simply to 'iff'. Then the information contained in these tables can also be put briefly like this:

$$
\begin{array}{lll}
|\neg P| = \mathrm{T} & \text{iff} & |P| = \mathrm{F} \\
|P \wedge Q| = \mathrm{T} & \text{iff} & |P| = \mathrm{T} \text{ and } |Q| = \mathrm{T}.
\end{array}
$$

Notice that it is enough if we just spell out the conditions under which a proposition is true, for it then follows that in all other conditions it will be false, in view of our assumption that a proposition always is either true or false.

In addition to the truth-functors '$\neg$' and '$\wedge$' we shall also use '$\vee$', '$\rightarrow$', and '$\leftrightarrow$' as truth-functors which correspond, in a rough and ready way, to the English words 'or', 'if... then...', and 'if and only if'. Their truth-tables are

P	Q	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
T	T	T	T	T
T	F	T	F	F
F	T	T	T	F
F	F	F	T	T

In view of the correspondence with English just noted, the same information can also be given in this way:

$$
\begin{array}{lll}
|P \vee Q| = \mathrm{T} & \text{iff} & |P| = \mathrm{T} \text{ or } |Q| = \mathrm{T} \\
|P \rightarrow Q| = \mathrm{T} & \text{iff} & \text{if } |P| = \mathrm{T} \text{ then } |Q| = \mathrm{T} \\
|P \leftrightarrow Q| = \mathrm{T} & \text{iff} & |P| = \mathrm{T} \text{ iff } |Q| = \mathrm{T}.
\end{array}
$$

(In some other books one finds '$\supset$' in place of '$\rightarrow$', and '$\equiv$' in place of '$\leftrightarrow$'.) When two sentences are joined by '$\vee$' we call the whole a disjunction and the two sentences are its disjuncts; when they are joined by '$\rightarrow$' we call the whole a conditional, the first sentence being its antecedent and the second its consequent; when they are joined by '$\leftrightarrow$' we call the whole a biconditional (and there is no special name for its parts).

We shall also use '$\top$' and '$\bot$' as 'zero-place truth-functors', i.e. as sentences which take a constant truth-value, the first being true in every possible situation and the second false. So their 'truth-tables' amount just to this:

$$|\top| = \mathrm{T} \qquad |\bot| = \mathrm{F}$$

If you wish, you may think of '$\top$' and '$\bot$' as abbreviating some entirely familiar propositions, the first necessarily true and the second necessarily false, for example '0 = 0' and '0 = 1'. That is an approach which will give entirely the right results for the purposes of this book. But from a more philosophical perspective one might well wish to quarrel with it. For it is very often held that our other truth-functors are *defined* by their truth-tables, and so have no other meaning than the truth-table gives to them. If that is so, then presumably '⊤' and '⊥' should equally be regarded as defined by their truth-tables, so that '⊥' is a sentence with no other meaning than that what it says is, in all possible situations, false. In that case, '⊥' is a wholly unfamiliar sentence. (And so is '⊤'.)

Setting aside the rather odd case of '⊤' and '⊥', the other truth-functors just listed are chosen partly because it proves convenient to have a short way of expressing the truth-functions in question, and partly because they have a rough correspondence (as noted) with familiar English expressions. No doubt these two reasons are connected with one another, though one may well debate just how this connection should be understood. (Does one of the reasons given explain the other? If so, which way round does the explanation go?) One may also debate upon how close the correspondence is between these truth-functors and their English counterparts, and why it is not perfect. But, as usual, we shall forgo the pleasures of such a debate, since our concern is with the logical theory itself and not with its application to English. From this perspective, there is certainly some arbitrariness in choosing to introduce simple signs for just these truth-functions but not others. In Sections 2.7 and 2.9 we shall explore some consequences that would flow from selecting one set of truth-functors rather than another, but although these introduce some constraints, they still leave a great deal of freedom. So I do not in fact specify any definite list of truth-functors as *the* ones to be employed. Instead, the treatment will be general enough to allow for any choice of truth-functors, though the ones just listed will be the ones most commonly employed in illustrations.

EXERCISES

2.2.1. Determine whether the following sentence-functors are truth-functors. (Method: see whether it is possible to construct a complete truth-table for them.)

(*a*) *P* because *Q*.
(*b*) Even if *P*, still *Q*.
(*c*) John believes that *P*.
(*d*) Either John believes that *P* or he does not.

2.2.2. Discuss the following proposals:

(*a*) that '¬' and 'not' mean the same.
(*b*) that '∨' and 'or' mean the same.
(*c*) that '→' and 'only if' mean the same.

2.3. Languages for Truth-Functors

We shall now introduce suitable formal languages for studying the effects that truth-functors have on entailment. As already noted (p. 6), these are not really languages, in the ordinary sense of the word, but rather language-schemas. For they will be built from a vocabulary which includes some truth-functors—it does not matter which—and otherwise only schematic sentence-letters, together with brackets to show punctuation. The full list of sentence-letters is the infinite list

$$P,Q,R,P_1,Q_1,R_1,P_2,...$$

A formal language for truth-functors may contain all of these letters in its vocabulary, or it may contain only some. If we take as an example the language which contains them all, and which contains all the truth-functors of the previous section, and nothing else, then this language is specified by the following *formation rules*:

(1) Each sentence-letter is a formula.
(2) '$\top$' and '$\bot$' are formulae.
(3) If φ is a formula, so is $\neg\varphi$.
(4) If φ and ψ are formulae, so are $(\varphi\wedge\psi)$, $(\varphi\vee\psi)$, $(\varphi\rightarrow\psi)$, $(\varphi\leftrightarrow\psi)$.
(5) Nothing else is a formula.

It is easy to see how the rules are to be varied to accommodate different choices of the initial vocabulary. For example, our rule (1) might just say: 'The letters "*P*", "*Q*", and "*R*" are formulae', and then in view of clause (5) no other letters would be included in the language. Any or all of rules (2)–(4) might be omitted, and some other truth-functor might be added. For example, one might wish to consider a language with just the one truth-functor '$\uparrow$' (to be introduced later, p. 58), so that in place of all of rules (2)–(4) we should just have

If φ and ψ are formulae, so is $(\varphi\uparrow\psi)$.

But so long as at least one expression is given outright as a formula by rule (1) (or rule (2)), and so long as at least one truth-functor is introduced by a rule with the form of rules (3) or (4), saying that that truth-functor may be applied to any formulae to yield new formulae, then we shall have a language with infinitely many formulae in it. For there is no upper bound on the length of a formula, and indeed the rules will not allow of there being any such bound.

The only limit on the length of a formula is that every formula must be of *finite* length, and rule (5) is intended to be so understood that it has this consequence. One should think of this rule as saying that there are no formulae other than the ones that there *have* to be in order to satisfy the other rules. It comes to the same thing to say that the set of formulae is the *smallest* set that satisfies the other rules, because it is a subset of every set that satisfies them. So, since we do not need formulae of infinite length in order to satisfy those rules, there are none. On the contrary, every formula is built up by starting with some *atomic formulae*, given by rules (1) and (2), and then applying rules (3) and (4) to bind these together into successively longer and longer formulae, until, after some finite number of applications, the last truth-functor is added and the whole formula is completed.

The formulae that are formed along the way are called the *subformulae* of the whole formula (and the whole formula is trivially counted as a subformula of itself). The subformulae of a given formula are just those parts of it that are themselves formulae, except that for this purpose we do not count a sentence-letter as having any parts smaller than itself. (For example, 'P' is not a subformula of 'P_2', and 'P_2' is not a subformula of 'P_{22}'.) We may add that for each occurrence of a truth-functor in our whole formula there will be a definite stage in the process of building up the whole at which it was first incorporated, and that will be the stage when the shortest subformula containing that occurrence was formed. This shortest subformula is called the *scope* of the given occurrence, and the truth-functor concerned is said to be the *main* functor of that subformula. It is easily seen that the punctuation supplied by the brackets that figure in rule (4) ensures that each formula does have a unique decomposition into subformulae, so that there is never any ambiguity over the scope of an occurrence of a truth-functor.

Nevertheless, all these brackets are rather tedious in practice, and it is convenient to have some conventions for omitting them. Without any ambiguity we may always omit the outer pair of brackets in any formula that begins and ends with a bracket. Where we have a continued conjunction, as in

$$((\varphi \wedge \psi) \wedge \chi) \quad \text{or} \quad (\varphi \wedge (\psi \wedge \chi))$$

we may also omit the inner pair of brackets and write simply

$$(\varphi \wedge \psi \wedge \chi).$$

This increases readability, and should not be misleading, since it will make no difference which way the inner brackets are restored. The same convention applies to a continued disjunction

$$((\varphi \vee \psi) \vee \chi) \quad \text{or} \quad (\varphi \vee (\psi \vee \chi)).$$

Finally we may regard the functors $\rightarrow$ and $\leftrightarrow$ as 'outranking' $\wedge$ and $\vee$ in the sense that, where brackets are not shown, they should be restored in a way that gives a larger scope to $\rightarrow$ or $\leftrightarrow$, and a smaller scope to $\wedge$ and $\vee$, rather than vice versa. Thus

$$\varphi \vee \psi \rightarrow \chi$$

is to be understood as an abbreviation for

$$(\varphi \vee \psi) \rightarrow \chi$$

and not for

$$\varphi \vee (\psi \rightarrow \chi).$$

Similarly,

$$\varphi \vee (\psi \wedge \chi) \quad \leftrightarrow \quad (\varphi \vee \psi) \wedge (\varphi \vee \chi)$$

is short for

$$(\varphi \vee (\psi \wedge \chi)) \quad \leftrightarrow \quad ((\varphi \vee \psi) \wedge (\varphi \vee \chi))$$

and not for any of the many other ways in which brackets might be restored.

It would be possible to avoid brackets altogether by a change in the notation for two-place truth-functors, i.e. by writing the functor *before* its two arguments, rather than *between* them. That is, one writes '$\vee \varphi \psi$' rather than '$\varphi \vee \psi$', and similarly for any other two-place functor. In this notation (which is known as Polish notation), the potential ambiguity in

$$\varphi \vee \psi \rightarrow \chi$$

cannot be reproduced. For on one way of construing it (i.e. the correct way), it is written as

$$\rightarrow \vee \varphi \psi \chi$$

and on the other way it is written as

$$\vee \varphi \rightarrow \psi \chi.$$

But most people find this notation difficult to read, and in any case it will not be used in this book.

EXERCISES

2.3.1.(*a*) Write formation rules for a language which contains all the sentence-letters, but just one truth-functor, namely the three-place truth-functor $\leftrightarrow(\varphi,\psi,\chi)$, which takes the value T when and only when φ, ψ, and χ each have the same value.

(*b*) Outline an argument to show that in this language no formula has an even number of sentence-letters. (A fully detailed argument for this conclusion would require the method of Section 2.8. But you should be able to give the *idea* of an argument without reading that section.)

2.3.2. How many different ways are there of restoring brackets to the formula

$$P \wedge Q \wedge Q \wedge P\,?$$

Why is it reasonable to say that it will not make any difference which way you choose to do it?