3 Semantics for Sentential Logic

1 Truth-functions

Now that we know how to recover the sentential logical form of an English argument from the argument itself, the next step is to develop a technique for testing argument-forms for validity. The examples we have already considered indicate that whether or not a logical form is valid depends at bottom on the meanings of the sentential connectives which occur in it. For example, in Chapter 1 we considered the two cases

A: P & Q
$\therefore$ P

and

B: P $\vee$ Q
$\therefore$ P

the first of which is valid and the second invalid. Since the only difference between the two is that one has '&' where the other has '$\vee$', it must be the difference in meaning between these two connectives that explains the difference in validity status between the two argument-forms. So a technique for testing argument-forms for validity must be based upon a precise specification of the meanings of the connectives.

First, some terminology. If a sentence is true, then it is said to *have the truth-value* TRUE, written '$\top$'. If a sentence is false, then it is said to *have the truth-value* FALSE, written '$\bot$'. We make the following assumption, often called the *Principle of Bivalence:*

> There are exactly two truth-values, $\top$ and $\bot$. Every meaningful sentence, simple or compound, has one or other, but not both, of these truth-values.

We already remarked that classical sentential logic is so called in part because it is the logic of the *sentential* connectives. What makes it *classical* is the fact that the Principle of Bivalence is embodied in the procedure for giving meaning to sentences of LSL. (By implication, therefore, there are other kinds of sentential logic based on different assumptions.) Granted the Principle of Bivalence, we can precisely specify the meaning, or *semantics*, of a sentential connective in the following way. A connective attaches to one or more sentences to form a new sentence. By the principle, the sentence(s) to which it attaches already have a truth-value, either $\top$ or $\bot$. The compound sentence which is formed must also have a truth-value, either $\top$ or $\bot$, and which it is depends both on the truth-values of the simpler sentence(s) being connected and on the connective being used. A connective's semantics are precisely specified by saying what will be the truth-value of the compound sentence it forms, given all the truth-values of the constituent sentences.

Negation. The case of negation affords the easiest illustration of this procedure. Suppose p is some sentence of English whose truth-value is $\top$ ('2 + 2 = 4'). Then the truth-value of ⌜it is not the case that p⌝ is $\bot$. In the same way, if p is some sentence of English whose truth-value is $\bot$ ('2 + 2 = 5'), the truth-value of ⌜it is not the case that p⌝ is $\top$. Hence the effect of prefixing 'it is not the case that' to a sentence is to *reverse* that sentence's truth-value. This fact exactly captures the meaning of 'it is not the case that', at least as far as logic is concerned, and we want to define our symbol '~' so that it has this meaning. One way of doing so is by what is called a *truth-table*. The truth-table for negation is displayed below.

p	$\sim p$
$\top$	$\bot$
$\bot$	$\top$

The symbol '~' forms a compound wff by being prefixed to some wff p. p has either the truth-value $\top$ or the truth-value $\bot$, and these are listed in the column headed by p on the left. On the right, we enter on each row the truth-value which the compound formula ⌜$\sim p$⌝ has, given the truth-value of p on that row.

Any sentential connective whose meaning can be captured in a truth-table is called a *truth-functional* connective and is said to *express a truth-function*. The general idea of a function is familiar from mathematics: a function is something which takes some object or objects as input and yields some object as output. Thus in the arithmetic of natural numbers, the function of squaring is the function which, given a single number as input, produces its square as output. The function of adding is the function which, given two numbers as input, produces their sum as output. A truth-function, then, is a function which takes a truth-value or truth-values as input and produces a truth-value as output. We can display a truth-function simply in terms of its effect on truth-values, abstracting from sentences. Thus the truth-function expressed by '~' could be written: $\top \Rightarrow \bot$, $\bot \Rightarrow \top$, which says that when the input is a truth the output is a falsehood ($\top \Rightarrow \bot$), and when the input is a falsehood, the output is a truth

($\bot \Rightarrow \top$). Notice that the input/output arrow '$\Rightarrow$' we use here is different from the arrow '$\rightarrow$' we use for the conditional. '~' expresses a *one-place* or *unary* truth-function, because the function's input is always a single truth-value. In the same way, squaring is a one-place function of numbers, since it takes a single number as input.

Conjunction. The truth-table for conjunction is slightly more complicated than that for negation. '&' is a two-place connective, so we need to display two formulae, p and q, each of which can be $\top$ or $\bot$, making four possibilities, as in (a) below. The order in which the possibilities are listed is by convention the standard one. (a) is empty, since we still have to decide what entries to make on each row of the table. To do this, we simply consider some sample English conjunctions. The assertion 'salt is a solid and water is a liquid' is one where both conjuncts are true, and this is enough to make the whole conjunction true. Since this would hold of any other conjunction as well, $\top$ should go in the top row of the table. But if one of the conjuncts of a conjunction is false, be it the first or the second, that is enough to make the whole conjunction false: consider 'salt is a gas and water is a liquid' and 'water is a liquid and salt is a gas'. Finally, if both conjuncts are false, the result is false as well. So we get the table in (b). We can write the resulting truth-function in the arrow notation, as in (c),

p	q	p & q
$\top$	$\top$	
$\top$	$\bot$	
$\bot$	$\top$	
$\bot$	$\bot$	

(a)

p	q	p & q
$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$

(b)

&

$\top\top \Rightarrow \top$
$\top\bot \Rightarrow \bot$
$\bot\top \Rightarrow \bot$
$\bot\bot \Rightarrow \bot$

(c)

&	$\top$	$\bot$
$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$

(d)

though this time the function is *two-place*, since the appropriate input is a *pair* of truth-values (addition is an example of a two-place function on numbers, since it takes two numbers as input). There is also a third way of exhibiting the meaning of '&', which is by a matrix, as in (d). The values in the side column represent the first of the two inputs while the values in the top row represent the second of the two inputs. A choice of one value from the side column and one from the top row determines a position in the matrix, where we find the value which conjunction yields for the chosen inputs. (b), (c) and (d), therefore, all convey the same information.

Disjunction. To exhibit the semantics of a connective, we have to write out the truth-function which the connective expresses in one of the three formats just illustrated. What truth-function does disjunction express? Here matters are not as straightforward as with negation and conjunction, since there tends to be some disagreement about what to enter in the top row of the truth-table for '$\vee$', as we noted in connection with Example 2.2.10 on page 17. Recall also

(1) Either I will watch television this evening or read a good book

on the supposition that I do *both*, so that both disjuncts are true. But we have declared the policy of always treating disjunction as meaning inclusive disjunction, so (1) is true if I do both, and therefore the top row of the table for '$\vee$' contains $\top$.

The remaining rows of the truth-table are unproblematic. If I only watch television (row 2 of the table below) or only read a good book (row 3), then (1) is clearly true, while if I do neither (row 4), it is false. So in the inclusive sense, a disjunction is true in every case except where both disjuncts are false. This leads us to the following representations of the meaning of '$\vee$':

p	q	$p \vee q$
$\top$	$\top$	$\top$
$\top$	$\bot$	$\top$
$\bot$	$\top$	$\top$
$\bot$	$\bot$	$\bot$

(a)

$$\begin{array}{ccc} & \vee & \\ \top\top & \Rightarrow & \top \\ \top\bot & \Rightarrow & \top \\ \bot\top & \Rightarrow & \top \\ \bot\bot & \Rightarrow & \bot \end{array}$$

(b)

$\vee$	$\top$	$\bot$
$\top$	$\top$	$\top$
$\bot$	$\top$	$\bot$

(c)

The difference between the actual table (a) for '$\vee$' and the one we would have written had we chosen to use the symbol for exclusive disjunction (one or the other but not both) would simply be that the top row would contain $\bot$ instead of $\top$.

The Conditional. This leaves us still to discuss the conditional and the biconditional. Since the biconditional was defined as a conjunction of conditionals (page 24), we will be able to *calculate* its truth-table once we have the table for '$\rightarrow$', since we already know how to deal with conjunctions. However, the table for '$\rightarrow$' turns out to be a little problematic. There is one row where the entry is clear. The statement

(2) If Smith bribes the instructor then Smith will get an A

is clearly false if Smith bribes the instructor but does not get an A. (2) says that bribing the instructor is sufficient for getting an A, or will lead to getting an A, so if a bribe is given and an A does not result, what (2) says is false. So we can enter a $\bot$ in the second row of the table for '$\rightarrow$', as in (a) below.

p	q	$p \rightarrow q$
$\top$	$\top$	
$\top$	$\bot$	$\bot$
$\bot$	$\top$	
$\bot$	$\bot$	

(a)

p	q	$p \rightarrow q$
$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\bot$	$\top$	$\top$
$\bot$	$\bot$	$\top$

(b)

$$\begin{array}{ccc} \top\top & \Rightarrow & \top \\ \top\bot & \Rightarrow & \bot \\ \bot\top & \Rightarrow & \top \\ \bot\bot & \Rightarrow & \top \end{array}$$

(c)

$\rightarrow$	$\top$	$\bot$
$\top$	$\top$	$\bot$
$\bot$	$\top$	$\top$

(d)

But what of the other three rows? Here are three relevant conditionals:

(3) If Nixon was U.S. president then Nixon lived in the White House.

(4) If Agnew was British prime minister then Agnew was elected.

(5) If Agnew was Canadian prime minister then Agnew lived in Ottawa.

(3) has a true antecedent and a true consequent, (4) a false antecedent and a true consequent, and (5) a false antecedent and a false consequent, but all three of the conditionals are true (only elected members of Parliament can be British prime minister, but unelected officials can become U.S. president). Relying just on these examples, we would complete the table for '→' as in (b) of the previous figure, with equivalent representations (c) and (d).

The trouble with (b), (c) and (d) is that they commit us to saying that *every* conditional with a true antecedent and consequent is true and that *every* conditional with a false antecedent is true. But it is by no means clear that this is faithful to our intuitions about ordinary indicative conditionals. For example,

(6) If Moses wrote the Pentateuch then water is H_2O

has an antecedent which is either true or false—most biblical scholars would say it is false—and a consequent which is true, and so (6) is true according to our matrix for '→'. But many people would deny that (6) is true, on the grounds that there is no relationship between the antecedent and the consequent: there is no sense in which the nature of the chemical composition of water is a *consequence* of the identity of the author of the first five books of the Bible, and if (6) asserts that it *is* a consequence, then (6) is false, not true.

As in our discussion of 'or', there are two responses one might have to this objection to our table (b) for 'if…then…'. One response is to distinguish two senses of 'if…then…'. According to this response, there is a sense of 'if…then…' which the table correctly encapsulates, and a sense which it does not. The encapsulated sense is usually called the *material* sense, and '→' is said to express the *material conditional*. Indeed, even if it were held that in English, 'if…then…' never expresses the material conditional, we could regard the table (b) above as simply a *definitional introduction* of this conditional. There would then be no arguing with table (b); the question would be whether the definitionally introduced meaning for the symbol '→' which is to be used in translating English indicative conditionals is adequate for the purposes of sentential logic. And it turns out that the answer to this question is 'yes', since it is the second row of the table which is crucial, and the second row is unproblematic. An alternative response to the objection is to say that the objector is confusing the question of whether (6) is literally true with the question of whether it would be appropriate to assert (6) in various circumstances. Perhaps it would be inappropriate for one who knows the chemical composition of water to assert (6), but such inappropriateness is still consistent with (6)'s being literally true, according to this account.

The parallel with the discussion of 'or' is not exact, and these are issues we will return to in §8 of this chapter. But whatever position one takes about the meaning of 'if…then…' in English, the reader should be assured that it is adequate for the purposes of sentential logic to translate English indicative conditionals into LSL using the material conditional '$\rightarrow$', even if one does regard this conditional as somewhat artificial. The artificiality will not lead to intuitively valid arguments being assessed as invalid, or conversely.

The Biconditional. For any LSL wffs p and q, the biconditional $\ulcorner p \leftrightarrow q \urcorner$ simply abbreviates the corresponding conjunction of conditionals $\ulcorner (p \rightarrow q) \,\&\, (q \rightarrow p) \urcorner$, according to our discussion in §3 of Chapter 2. It follows that if we can work out the truth-table for that conjunction, given the matrices for '&' and '$\rightarrow$', we will arrive at the truth-function expressed by '$\leftrightarrow$'. What is involved in working out the table for a formula with more than one connective? In a formula of the form $\ulcorner (p \rightarrow q) \,\&\, (q \rightarrow p) \urcorner$, p and q may each be either true or false, leading to the usual four possibilities. The truth-table for $\ulcorner (p \rightarrow q) \,\&\, (q \rightarrow p) \urcorner$ should tell us what the truth-value of the formula is for each of these possibilities. So we begin by writing out a table with the formula along the top, as in (a) below:

p	q	$(p \rightarrow q)$	&	$(q \rightarrow p)$
$\top$	$\top$			
$\top$	$\bot$			
$\bot$	$\top$			
$\bot$	$\bot$			

(a)

p	q	$(p \rightarrow q)$	&	$(q \rightarrow p)$
$\top$	$\top$	$\top$		$\top$
$\top$	$\bot$	$\bot$		$\top$
$\bot$	$\top$	$\top$		$\bot$
$\bot$	$\bot$	$\top$		$\top$

(b)

p	q	$(p \rightarrow q)$	&	$(q \rightarrow p)$
$\top$	$\top$	$\top$	$\boxed{\top}$	$\top$
$\top$	$\bot$	$\bot$	$\boxed{\bot}$	$\top$
$\bot$	$\top$	$\top$	$\boxed{\bot}$	$\bot$
$\bot$	$\bot$	$\top$	$\boxed{\top}$	$\top$

(c)

The formula is a conjunction, so its truth-value in each of the four cases will depend upon the truth-value of its conjuncts in each case. The next step is therefore to work out the truth-values of the conjuncts on each row. The first conjunct is the (material) conditional $\ulcorner (p \rightarrow q) \urcorner$ whose truth-table we have already given, so we can just write those values in. The second conjunct is $\ulcorner q \rightarrow p \urcorner$. We know from our discussion of '$\rightarrow$' above that the only case where a material conditional is false is when it has a true antecedent and false consequent, and this combination for $\ulcorner q \rightarrow p \urcorner$ occurs on row 3 (*not* row 2); so under $\ulcorner q \rightarrow p \urcorner$ we want $\bot$ on row 3 and $\top$ elsewhere. This gives us table (b) above. We have now calculated the truth-value of each conjunct of $\ulcorner (p \rightarrow q) \,\&\, (q \rightarrow p) \urcorner$ on each row, so it remains only to calculate the truth-value of the whole conjunction on each row. Referring to the tables for conjunction, we see that a conjunction is true in just one case, that is, when both conjuncts are true. In our table for $\ulcorner (p \rightarrow q) \,\&\, (q \rightarrow p) \urcorner$, both conjuncts are true on rows 1 and 4, so we can complete the table as in (c) above. Notice how we highlight the column of entries under the main connective of the formula. The point of doing this is to distinguish the final answer from the other columns of entries written in as intermediate steps.

The example of $\ulcorner (p \rightarrow q) \,\&\, (q \rightarrow p) \urcorner$ illustrates the technique for arriving at the truth-table of a formula with more than one occurrence of a connective in it, and it also settles the question of what truth-function '$\leftrightarrow$' expresses. We complete our account of the meanings of the connectives with the tables for the

p	q	$p \leftrightarrow q$
$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\top$

(a)

$$\begin{array}{c}
\leftrightarrow \\
\top\top \Rightarrow \top \\
\top\bot \Rightarrow \bot \\
\bot\top \Rightarrow \bot \\
\bot\bot \Rightarrow \top
\end{array}$$

(b)

$\leftrightarrow$	$\top$	$\bot$
$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\top$

(c)

biconditional displayed above. Note that, by contrast with $\ulcorner p \rightarrow q \urcorner$ and $\ulcorner q \rightarrow p \urcorner$, $\ulcorner p \leftrightarrow q \urcorner$ and $\ulcorner q \leftrightarrow p \urcorner$ have the same truth-table; this bears out our discussion of Examples 2.4.6 and 2.4.7 on page 32, where we argued that the order in which the two sides of a biconditional are written is irrelevant from the logical point of view.

In order to acquire some facility with the techniques which we are going to introduce next, it is necessary that the meanings of the connectives be memorized. Perhaps the most useful form in which to remember them is in the form of their function-tables, so here are the truth-functions expressed by all five connectives:

$$\begin{array}{c}
\sim \\
\top \Rightarrow \bot \\
\bot \Rightarrow \top
\end{array}
\qquad
\begin{array}{c}
\& \\
\top\top \Rightarrow \top \\
\top\bot \Rightarrow \bot \\
\bot\top \Rightarrow \bot \\
\bot\bot \Rightarrow \bot
\end{array}
\qquad
\begin{array}{c}
\vee \\
\top\top \Rightarrow \top \\
\top\bot \Rightarrow \top \\
\bot\top \Rightarrow \top \\
\bot\bot \Rightarrow \bot
\end{array}
\qquad
\begin{array}{c}
\rightarrow \\
\top\top \Rightarrow \top \\
\top\bot \Rightarrow \bot \\
\bot\top \Rightarrow \top \\
\bot\bot \Rightarrow \top
\end{array}
\qquad
\begin{array}{c}
\leftrightarrow \\
\top\top \Rightarrow \top \\
\top\bot \Rightarrow \bot \\
\bot\top \Rightarrow \bot \\
\bot\bot \Rightarrow \top
\end{array}$$

The information represented here can be summarized as follows:

- Negation reverses truth-value.
- A conjunction is true when and only when both conjuncts are true.
- A disjunction is false when and only when both disjuncts are false.
- A conditional is false when and only when its antecedent is true and its consequent is false.
- A biconditional is true when and only when both its sides have the same truth-value.

These summaries should also be memorized.

There are some entertaining puzzles originated by Raymond Smullyan which involve manipulating the notions of truth and falsity in accordance with the tables for the connectives. In a typical Smullyan setup, you are on an island where there are three kinds of inhabitants, Knights, Knaves and Normals. Knights always tell the truth and Knaves always lie, while a Normal may sometimes lie and sometimes tell the truth. You encounter some people who make certain statements, and from the statements you have to categorize each of the people as a Knight, a Knave, or a Normal. Here is an example, from Smullyan:

> You meet two people, A and B, each of whom is either a Knight or a Knave. Suppose A says: ‘Either I am a Knave or B is a Knight.’ What are A and B?

We reason to the solution as follows. There are two possibilities for A, either Knight or Knave. Suppose that A is a Knave. Then what he says is false. What he says is a disjunction, so by any of the tables for ‘$\vee$’, both disjuncts of his statement must be false. This would mean that A is a Knight and B is a Knave. But A cannot be a Knight if he is a Knave (our starting supposition). Thus it follows that he is not a Knave. So by the conditions of the problem, he is a Knight and what he says is true. Since he is a Knight, the first disjunct of his statement is false, so the second disjunct must be true. Hence B is a Knight as well.

❑ Exercises

The following problems are from Smullyan.[1] In each case explain the reasoning that leads you to your answer in the way just illustrated.

(1) There are two people, A and B, each of whom is either a Knight or a Knave. A says: ‘At least one of us is a Knave.’ What are A and B?

(2) With the same conditions as (1), suppose instead A says: ‘If B is a Knight then I am a Knave.’ What are A and B? [Refer to the truth-table for ‘$\rightarrow$’.]

(3) There are three people, A, B and C, each of whom is either a Knight or a Knave. A and B make the following statements:

A: ‘All of us are Knaves.’
B: ‘Exactly one of us is a Knight.’

What are A, B and C?

*(4) Two people are said to be of the *same type* if and only if they are both Knights or both Knaves. A and B make the following statements:

A: ‘B is a knave.’
B: ‘A and C are of the same type.’

On the assumption that none of A, B and C is Normal, can it be determined what C is? If so, what is he? If not, why not?

(5) Suppose A, B and C are being tried for a crime. It is known that the crime was committed by only one of them, that the perpetrator was a Knight, and the only Knight among them. The other two are either both Knaves, both Normals, or one of each. The three defendants make the statements below. Which one is guilty?

A: 'I am innocent.'
B: 'That is true.'
C: 'B is not Normal.'

(6) A, who is either a Knight or a Knave, makes the following statement:

A: 'There is buried treasure on this island if and only if I am a Knight.'

(i) Can it be determined whether A is a Knight or a Knave?

(ii) Can it be determined whether there is buried treasure on the island?

2 Classifying formulae

Any formula of LSL has a truth-table, for every formula is constructed from a certain number of sentence-letters and each sentence-letter can be either ⊤ or ⊥. So there are various possible combinations of truth-values for the sentence-letters in the formula, and for each of those possible combinations, the formula has its own truth-value. Here are truth-tables for three very simple formulae, 'A → A', 'A → ~A' and '~(A → A)':

A	A → A
⊤	⊤
⊥	⊤

(a)

A	A → ~A
⊤	⊥
⊥	⊤

(b)

A	~	(A → A)
⊤	$\boxed{\bot}$	⊤
⊥	$\boxed{\bot}$	⊤

(c)

In table (a), on the first row, 'A → A' is ⊤ → ⊤, which by the function-table or matrix for '→' is ⊤, and on the second row, ⊥ → ⊥, which is also ⊤. In table (b) we have ⊤ → ⊥ on the top row and ⊥ → ⊤ on the bottom, which gives ⊥ and ⊤ respectively. Finally, in table (c) we have the negation of the formula of table (a), so in its final answer column, table (c) should have ⊥ where table (a) has ⊤, and ⊤ where (a) has ⊥. We enter the column for the subformula 'A → A' first, directly under the main connective of this subformula, and then apply the truth-function expressed by '~' to that column. This gives the final answer, *which we always display under the main connective of the whole formula.*

These three formulae exhibit the three possibilities for any formula: that in its truth-table the final answer column contains nothing but ⊤s, or a mixture of ⊤s and ⊥s, or nothing but ⊥s. There is a technical term for each of these three kinds of formula:

- A formula whose truth-table's final answer column contains only ⊤s is called a *tautology*.
- A formula whose truth-table's final answer column contains only ⊥s is called a *contradiction*.
- A formula whose truth-table's final answer column contains both ⊤s and ⊥s is said to be *contingent*.

Thus 'A → A' is a tautology, '~(A → A)' is a contradiction, and 'A → ~A' is a contingent formula.

When a formula only has a single sentence-letter in it, as in the three formulae just exhibited, there are only two possibilities to consider: the sentence-letter is either ⊤ or ⊥. And as we have seen in giving the truth-tables for the binary connectives, when there are two sentence-letters there are four possibilities, since each sentence-letter can be either ⊤ or ⊥. The number of possibilities to consider is determined by the number of sentence-letters in the formula, not by the complexity of the formula. Thus a truth-table for '(A ↔ B) → ((A & B) ∨ (~A & ~B))' will have only four rows, just like a table for 'A → B', since it contains only two sentence-letters. But it will have many more *columns:*

A	B	(A ↔ B)	→	((A & B)	∨	(~A	&	~B))
⊤	⊤	⊤	$\boxed{\top}$	⊤	⊤	⊥	⊥	⊥
⊤	⊥	⊥	$\boxed{\top}$	⊥	⊥	⊥	⊥	⊤
⊥	⊤	⊥	$\boxed{\top}$	⊥	⊥	⊤	⊥	⊥
⊥	⊥	⊤	$\boxed{\top}$	⊥	⊤	⊤	⊤	⊤
		1		2	6	3	5	4

Here the numbers indicate the order in which the columns are computed. We begin by entering the values for 'A ↔ B' and 'A & B', simply taking these from the truth-tables for '↔' and '&'; this gives us the columns numbered 1 and 2. Then we compute the entries for '~A' and '~B' by applying the negation truth-function to the entries under the two sentence-letters on the far left; this gives us columns 3 and 4. Using columns 3 and 4, we next compute the values for the conjunction '~A & ~B', which gives us column 5, since it is only on the bottom row that columns 3 and 4 both contain ⊤. We then use columns 2 and 5 to compute the entries for the disjunction '(A & B) ∨ (~A & ~B)', yielding column 6 under the disjunction symbol. Lastly, we use columns 1 and 6 to compute the final answer for the whole formula. Inspecting the final column reveals that the formula is a tautology. There is some flexibility about the order in which we compute columns—the main constraint is that before computing the column for any subformula *q* of a given formula *p*, we must first compute the columns for all *q*'s subformulae.

What of formulae with more than two sentence-letters, for example, '(A → (B ∨ C)) → (A → (B & C))'? The first question is how many different combinations of truth-values have to be considered when there are three sentence-letters. It is not difficult to see that there are eight combinations, as the following argument shows: A can be either ⊤ or ⊥, and in each of these two cases, B can be either ⊤ or ⊥, giving us four cases, and in each of these four, C can be either ⊤ or ⊥, giving eight cases in all. More generally, if there are *n* sentence-letters in a formula, then there will be 2^n cases, since each extra sentence-letter doubles the number of cases. In particular, we need an eight-row truth-table for the formula '(A → (B ∨ C)) → (A → (B & C))'.

There is a conventional way of listing the possible combinations of truth-values for any *n* sentence-letters. Once the sentence-letters are listed at the top left of the table, we alternate ⊤ with ⊥ under the innermost (rightmost) letter until we have 2^n rows. Then under the next-to-innermost, we alternate ⊤s and ⊥s in *pairs* to fill 2^n rows. Continuing leftward we alternate ⊤s and ⊥s in fours, then eights, then sixteens, and so on, until every sentence-letter has a column of 2^n truth-values under it. This procedure guarantees that all combinations are listed and none are listed twice. For our example, '(A → (B ∨ C)) → (A → (B & C))', we will therefore have ⊤ followed by ⊥ iterated four times under 'C', two ⊤s followed by two ⊥s followed by two ⊤s followed by two ⊥s under 'B', and four ⊤s followed by four ⊥s under 'A'. We then proceed to compute the table for the formula in the usual way:

A	B	C	(A →	(B ∨ C))	→	(A →	(B & C))
⊤	⊤	⊤	⊤		⊤	⊤	⊤
⊤	⊤	⊥	⊤		⊥	⊥	⊥
⊤	⊥	⊤	⊤		⊥	⊥	⊥
⊤	⊥	⊥	⊥	⊥	⊤	⊥	⊥
⊥	⊤	⊤	⊤		⊤	⊤	
⊥	⊤	⊥	⊤		⊤	⊤	
⊥	⊥	⊤	⊤		⊤	⊤	
⊥	⊥	⊥	⊤		⊤	⊤	
			4	3		2	1

This formula is contingent, since it has a mixture of ⊤s and ⊥s. Notice also that the truth-table has not been completely filled in. When the number of rows in a truth-table is large, it is advisable to look for shortcuts in arriving at the final column. So in column 1, for example, we do not compute the bottom four rows, since we know that a material conditional with a false antecedent is true, and hence 'A → (B & C)' will be true on the bottom four rows since 'A' is false there (refer to A's column on the extreme left); the values of 'B & C' are therefore irrelevant on these rows. Similarly in column 3, we can ignore all but row 4, since the falsity of 'A → (B ∨ C)' requires the truth of 'A' and the falsity of 'B ∨ C', which in turn requires the falsity of both 'B' and 'C'. 'B' and 'C' are both ⊥ only on rows 4 and 8, and by inspection of these two rows, we see that 'A' is ⊤ only on row 4. Hence it is only on row 4 that the condition for 'A → (B ∨ C)' to be ⊥

is satisfied, and we can fill in ⊤ on all the other rows in column 4, as we have done in the displayed table.

Up to this point we have referred to combinations of truth-values listed on the left of a truth-table as 'cases' and have described a truth-table for a formula as giving the truth-value of the formula in each of the 'possible cases'. The more usual word for 'case' is *interpretation*. Thus a formula of LSL with n sentence-letters has 2^n possible interpretations. However, the term 'interpretation' is used in every kind of system of logic: an interpretation is a way of giving meaning to the sentences of the language appropriate for the kind of logic in question, and with more complex languages, this involves more than specifying truth-values for sentence-letters. So for each kind of logic, we have to say explicitly what kind of thing an interpretation of a formula of the language for that logic is. For sentential logic, we have:

> An *interpretation of a formula p of LSL* is an assignment of truth-values to the sentence-letters which occur in *p*.

So in a truth-table for *p* we find on the left a list of all the possible interpretations of *p*, that is, all the possible assignments of truth-values to the sentence-letters in *p*. A single interpretation of a formula is given by specifying the truth-values which its sentence-letters have on that interpretation. For example, the third interpretation in the table on the previous page assigns ⊤ to 'A' and 'C' and ⊥ to 'B', and this interpretation makes the formula *p* = '(A → (B ∨ C)) → (A → (B & C))' false.

Because the term 'interpretation' has application in every kind of logic, technical concepts of sentential logic defined using it will also have wider application. For example, we have previously spoken of formulae being 'equivalent' or of their 'meaning the same' in a loose sense. Exactly what this amounts to is spelled out in the following:

> Two formulae *p* and *q* are said to be *logically equivalent* if and only if, on any interpretation assigning truth-values to the sentence-letters of both, the truth-value of the first formula is the same as the truth-value of the second.

Here we have not explicitly restricted the definition to formulae of LSL, since the same notion of logical equivalence will apply to formulae of any system of logic in which formulae have truth-values or something analogous. In the special case of LSL, for formulae with exactly the same sentence letters, logical equivalence amounts to having the same truth-table. So by inspecting the table on the following page, we see that 'A & B' and '~(~A ∨ ~B)' are logically equivalent, and that '~(A ∨ B)' and '~A & ~B' are logically equivalent. For each interpretation gives the same truth-value to 'A & B' as it does to '~(~A ∨ ~B)', and each gives the same truth-value to '~(A ∨ B)' as it does to '~A & ~B'.

A	B	A & B	~(~A ∨ ~B)		~(A ∨ B)		~A & ~B
⊤	⊤	⊤	⊤	⊥	⊥	⊤	⊥
⊤	⊥	⊥	⊥	⊤	⊥	⊤	⊥
⊥	⊤	⊥	⊥	⊤	⊥	⊤	⊥
⊥	⊥	⊥	⊥	⊤	⊤	⊥	⊤

In our discussion of the connective ‘unless’ in §3 of Chapter Two, the conclusion we reached was that ⌜*p* unless *q*⌝ can be symbolized as ⌜~*q* → *p*⌝, for any formulae *p* and *q*. Suppose that *p* and *q* are sentence-letters, say ‘A’ and ‘B’. Then what we find is that ‘unless’ is just ‘or’, for we have the following table,

A	B	A ∨ B	~B → A
⊤	⊤	⊤	⊤
⊤	⊥	⊤	⊤
⊥	⊤	⊤	⊤
⊥	⊥	⊥	⊥

which shows that ‘~B → A’ is logically equivalent to ‘A ∨ B’. In testing English arguments for validity, it is an advantage to symbolize the English with formulae which are as simple as possible, so at this point we will change our policy as regards ‘unless’. Henceforth, we symbolize ‘unless’ using ‘∨’:

- ⌜*p* unless *q*⌝ is symbolized ⌜*p* ∨ *q*⌝.

Reflection on the meaning of ‘unless’ should indicate that this policy is intuitively correct: if the company will go bankrupt unless it receives a loan, that means that either the company will go bankrupt or (if it does not) then it receives (i.e. must have received) a loan.

❑ Exercises

I Construct complete truth-tables for the following formulae and classify each as tautologous, contradictory or contingent. Be sure to mark your final column clearly and place it directly under the main connective of the formula.

(1) A → (B → (A & B))
(2) ~R → (R → S)
(3) R → (S → R)
*(4) (A ↔ B) & (A & ~B)
(5) ((F & G) → H) → ((F ∨ G) → H)
(6) (A ↔ (B ∨ C)) → (~C → ~A)
(7) (A ↔ B) & ((C → ~ A) & (B → C))

II Use truth-tables to determine which formulae in the following list are logically equivalent to which. State your results.

(1) $A \vee B$	(2) $A \rightarrow B$
(3) $\sim(A \& \sim B)$	(4) $\sim(\sim A \& \sim B)$
(5) $\sim A \vee B$	(6) $A \vee \sim A$
(7) $(A \rightarrow (A \& \sim A)) \rightarrow \sim A$	

III If *p* is a sentence of LSL which is not a tautology, does it follow that ⌜$\sim p$⌝ *is* a tautology? Explain.

3 Testing for validity by exhaustive search

We are now in a position to present the first technique for testing an argument-form for validity. Recall our opening examples of a valid and an invalid English argument from §1 of Chapter 1:

A: (1) If our currency loses value then our trade deficit will narrow.
(2) Our currency will lose value.
(3) ∴ Our trade deficit will narrow.

B: (1) If our currency loses value then our trade deficit will narrow.
(2) Our trade deficit will narrow.
(3) ∴ Our currency will lose value.

Concerning argument A, we said that the truth of the conclusion (3) is 'guaranteed' by the truth of the two premises (1) and (2), but we did not explain exactly what the guarantee consists in. The (sentential) invalidity of argument B we explained in the following way: even if (1) in B is true, its truth is consistent with there being other conditions which are sufficient for a narrowing of our trade deficit, so even given the truth of (2) in B, we cannot conclude (3), since it may have been one of those other conditions which has brought about the truth of (2) without our currency having lost value at all. Hence it is incorrect to say that the truth of (1) and (2) in B *guarantees* the truth of (3) (even if in fact (1), (2) and (3) *are* all true).

Reflecting on this explanation of B's invalidity, we see that we demonstrate the lack of guarantee by describing how circumstances could arise in which both premises would be true while the conclusion is false. The point was not that the premises are in fact true and the conclusion in fact false, but merely that for all that the premises and conclusion *say*, it would be *possible* for the premises to be true and the conclusion false. And if we inspect argument A, we see that this is exactly what *cannot* happen in its case. Thus the key to the distinction between validity and invalidity in English arguments appears to have to do with whether or not there is a *possibility* of their having true premises and a false conclusion. Yet we also saw that validity or invalidity is fundamentally a property of argument-forms, not the arguments themselves. The sentential logical forms of A and B are respectively

C: $F \rightarrow N$
F
$\therefore N$

and

D: $F \rightarrow N$
N
$\therefore F$

What then would it mean to speak of the 'possibility' of C or D having true premises and a false conclusion?

We can transfer the notion of the possibility of having true premises and false conclusion from English arguments to the LSL argument-forms which exhibit the English arguments' sentential forms, by using the concept of interpretation explained on page 56. To say that it is possible for an LSL form to have true premises and a false conclusion is to say that *there is at least one interpretation* of the LSL form on which its premises are true and its conclusion false. In sentential logic, an interpretation is an assignment of truth-values, so whether or not an LSL argument-form is valid depends on whether or not some assignment of truth-values makes its premises true and its conclusion false. We render this completely precise as follows:

> An *interpretation of an LSL argument-form* is an assignment of truth-values to the sentence-letters which occur in that form.
>
> An argument-form in LSL is *valid* if there is no interpretation of it on which its premises are true and its conclusion false, and *invalid* if there is at least one interpretation of it on which its premises are true and its conclusion false.
>
> An English argument (or argument in any other natural language) is sententially valid if its translation into LSL yields a valid LSL argument-form, and is sententially invalid if its translation into LSL yields an invalid form.

Since there is no question of discerning finer structure in an LSL argument-form (as opposed to an English argument) using a more powerful system of logic, judgements of validity and invalidity for LSL forms are absolute, not relative to sentential logic. We can test an LSL form for validity by exhaustively listing all its interpretations and checking each one to see if any makes the form's premises all true while also making its conclusion false. Interpretations are listed in truth-tables, so we can use the latter for this purpose, by writing the argument-form out along the top. For example, we can test the two arguments C and D on the previous page with the following four-row table:

F	N	F → N	F	∴ N	F → N	N	∴ F
⊤	⊤	⊤	⊤	⊤	⊤	⊤	⊤
⊤	⊥	⊥	⊤	⊥	⊥	⊥	⊤
⊥	⊤	⊤	⊥	⊤	⊤	⊤	⊥
⊥	⊥	⊤	⊥	⊥	⊤	⊥	⊥

The table shows that C is valid according to our definition, since none of the four interpretations listed makes the two premises 'F → N' and 'F' true while at the same time making the conclusion 'N' false. The table also shows that argument-form D is invalid, since inspection of the entries for the third interpretation (highlighted), on which 'F' is false and 'N' is true, shows that D's premises 'F → N' and 'N' are true on this interpretation while its conclusion 'F' is false. In terms of the original English argument, the truth-values ⊥ for 'our currency will lose value' and ⊤ for 'our trade deficit will narrow' are exactly the ones which would obtain in a situation where our trade deficit narrows for some other reason while our currency stays the same or rises, which is the kind of situation whose possibility we mentioned in order to show the sentential invalidity of B. The third interpretation in the table, therefore, expresses what is common to all situations which show that a given English argument with the same form as B is sententially invalid.

To summarize, this technique of testing an LSL argument-form for validity consists in listing all its possible interpretations and exhaustively inspecting each one. If one is found which makes the premises of the argument true and its conclusion false, then the LSL argument-form is invalid; if no interpretation which does this is found, the LSL argument-form is valid.

In applying this test to more complex LSL argument-forms, with large numbers of sentence-letters, it is important to exploit as many short cuts as possible. For example, in §4 of Chapter 2, we considered the argument:

> E: If God exists, there will be no evil in the world unless God is unjust, or not omnipotent, or not omniscient. But if God exists then He is none of these, and there is evil in the world. So we have to conclude that God does not exist.

To test this English argument for sentential validity, we translate it into LSL and examine each of the interpretations of the LSL argument-form to see if any makes all the premises true and the conclusion false. If none do, the LSL argument-form is valid. This means that the English argument E is sententially valid and therefore valid absolutely. But if some interpretation does make the premises of the LSL argument-form all true and the conclusion false, then the LSL argument-form is invalid and so the English argument is sententially invalid.

The symbolization at which we arrived was:

F: X → [~(~J ∨ (~M ∨ ~S)) → ~V]
[X → (~~J & (~~M & ~~S))] & V
∴ ~X

F contains five sentence-letters and therefore has 2^5 interpretations. Consequently, we would appear to need a truth-table with thirty-two rows to conduct an exhaustive check of whether or not F is valid. However, we can use the following table to test it for validity:

X	V	J	M	S	X → [~(~J ∨ (~M ∨ ~S)) → ~V]	[X → (~~J & (~~M & ~~S))] & V	∴ ~X
⊤	⊤	⊤	⊤	⊤	⊥	⊤ ⊤ $\boxed{\top}$ ⊤	⊥
⊤	⊤	⊤	⊤	⊥		⊤	⊥
⊤	⊤	⊤	⊥	⊤		⊤	⊥
⊤	⊤	⊤	⊥	⊥		⊤	⊥
⊤	⊤	⊥	⊤	⊤		⊤	⊥
⊤	⊤	⊥	⊤	⊥		⊤	⊥
⊤	⊤	⊥	⊥	⊤		⊤	⊥
⊤	⊤	⊥	⊥	⊥		⊤	⊥
⊤	⊥	⊤	⊤	⊤		⊥	⊥
⊤	⊥	⊤	⊤	⊥		⊥	⊥
⊤	⊥	⊤	⊥	⊤		⊥	⊥
⊤	⊥	⊤	⊥	⊥		⊥	⊥
⊤	⊥	⊥	⊤	⊤		⊥	⊥
⊤	⊥	⊥	⊤	⊥		⊥	⊥
⊤	⊥	⊥	⊥	⊤		⊥	⊥
⊤	⊥	⊥	⊥	⊥		⊥	⊥

Two features of this table are immediately striking. The first is that it only has sixteen rows, instead of the advertised thirty-two. Where are the missing sixteen? The answer is that we are able to discount sixteen rows because we are only trying to discover whether there is an interpretation (row) on which all the premises of the argument-form are true while its conclusion is false. If we see that the conclusion is *true* on a certain interpretation, it would be a waste of effort to compute the truth-values of the premises on that interpretation, for clearly, *that* interpretation will not be one where all the premises are true and the conclusion *false*. To apply this point, observe that the conclusion of our argument-form is '~X', which is true on every interpretation which makes 'X' false, and the missing sixteen rows are exactly those on which 'X' is false. We deliberately made 'X' the first sentence-letter in the listing at the top of the table, so that the column beneath it would contain sixteen ⊤s followed by sixteen ⊥s, since this allows us to ignore the bottom sixteen rows, these being the interpretations where the conclusion is true and where we are consequently uninterested in the values ascribed to the premises.

The other striking feature of the table is that only the top row has been completed. The justification for this is comparable to that for ignoring the bottom sixteen rows. Just as we are not interested in interpretations which make the conclusion true, so we are not interested in interpretations which make one of the premises false, since those interpretations will not be ones where the conclusion is false and *all* the premises true. And it is easy to see from our table that every interpretation except the first makes premise 2 false; interpretations 9–16 make the second conjunct of the premise, 'V', false, while interpretations 2–8 make the first conjunct, 'X → (~~J & (~~M & ~~S))', false, because they make 'X' true and '(~~J & (~~M & ~~S))' false, since they make at least one of 'J' or 'M' or 'S' false. So only interpretation 1 makes premise 2 of F true, and so it is only its value for premise 1 that we are interested in computing. Hence, whether or not F is valid comes down to whether or not interpretation 1 makes premise 1 true. A simple calculation shows that in fact it makes it false. It follows that F is valid: no interpretation makes all the premises true and the conclusion false.[2] Consequently, E is sententially valid, and therefore valid absolutely. (The reader should study the reasoning of this and the previous paragraph for as long as is necessary to grasp it fully.)

This example nicely illustrates how with a little ingenuity we can save ourselves a lot of labor in testing for validity using the method of exhaustive search. But the method is still unwieldy, and completely impractical for LSL arguments which contain more than five sentence-letters. Given our definition of validity, then, the next step is to try to develop a more efficient way of testing for it.

❑ Exercises

Use the method of exhaustive search to test the arguments symbolized in the exercises for 2.4 for sentential validity. Display your work and state the result you obtain.

4 Testing for validity by constructing interpretations

A faster way of determining the validity or invalidity of an LSL argument-form is to attempt an explicit construction of an interpretation which makes the premises true and the conclusion false: if the attempt succeeds, the LSL form is invalid, and if it breaks down, then (provided it has been properly executed) the

[2] To repeat a point from Chapter 2, this does not mean that we have proved that God does not exist. We have shown merely that the original English argument is sententially valid, not that it is sound (recall that a sound argument is one which has all its premises true as well). In traditional Christian theology, the first premise would be disputed: it would be argued that the existence of evil is consistent with the existence of a just, omnipotent and omniscient God, since evil would be said to be a consequence of the free actions of human and supernatural beings, and God, it is held, is obliged not to interfere with the outcome of freely chosen actions. In other religions, or other versions of Christianity, different premises would be disputed. For instance, in some Eastern religions, it would be denied that there is evil in the world, on the grounds that all suffering is 'illusion'.

LSL form is valid. To test an English argument for sentential validity in this way, we first exhibit its form by translating it into LSL, and then we make assignments of truth-values to the sentence-letters in the conclusion of the LSL form so that the conclusion is false. We then try to assign truth-values to the remaining sentence-letters in the premises so that all the premises come out true. For example, we can test the argument G from §4 of Chapter 2, repeated here as

> A: We can be sure that Jackson will agree to the proposal. For otherwise the coalition will break down, and it is precisely in these circumstances that there would be an election; but the latter can certainly be ruled out.

We begin by translating A into LSL, which results in the LSL argument-form

B: $(\sim J \rightarrow C) \& (E \leftrightarrow C)$
$\sim E$
$\therefore J$

(reread the discussion in §4 of Chapter 2 if necessary) and then we try to find an interpretation of the three sentence-letters in B which makes the conclusion false and both premises true. To make the conclusion 'J' false, we simply stipulate that 'J' is false. Taking the simpler of the two premises first, we stipulate that 'E' is false, making '~E' true. The question now is whether there is an assignment to 'C' on which premise 1 comes out true. Since 'E' is false, we need 'C' to be false for the second conjunct of premise 1 to be true, but then '~J → C' is $\top \rightarrow \bot$, which is $\bot$, making premise 1 false. There are no other options, so we have to conclude that no interpretation makes all the premises of B true and the conclusion false. Thus B is a valid LSL argument-form, and therefore A is a sententially valid English argument, and so valid absolutely.

Here is another application of the same technique, to the LSL argument-form

C: $A \rightarrow (B \& E)$
$D \rightarrow (A \vee F)$
$\sim E$
$\therefore D \rightarrow B$

To make the conclusion false we stipulate that 'D' is true and 'B' is false. The simplest premise is the third, so next we make it true by stipulating that 'E' is false. This determines the truth-value of 'A' in the first premise if that premise is to be true: if 'E' is false then 'B & E' is false, so we require 'A' to be false for the premise to be true. So far, then, we have shown that for the conclusion to be false while the first and third premises are true, we require the assignment of truth-values $\top$ to 'D', $\bot$ to 'B', $\bot$ to 'E' and $\bot$ to 'A'. The question is whether this assignment can be extended to 'F' so that premise 2 comes out true. With 'D' being true and 'A' false, premise 2 is true when 'F' is true and false when 'F' is false. Since we are free to assign either truth-value to 'F', we obtain an interpretation which shows C to be invalid by stipulating that ‘F’ is true. In other words, the interpretation

D	B	E	A	F
$\top$	$\bot$	$\bot$	$\bot$	$\top$

makes all the premises of C true and its conclusion false, so C is invalid.

This technique is a significant improvement over drawing up the thirty-two row truth-table that would be required to test C for validity by the method of exhaustive search. However, the two examples we have just worked through contain a simplifying feature that need not be present in general, for in both B and C there is only one way of making the conclusion false. How do we proceed when there is more than one? An example in which this situation arises is the following:

D: $\sim A \vee (B \rightarrow C)$
$E \rightarrow (B \,\&\, A)$
$C \rightarrow E$
$\therefore C \leftrightarrow A$

We deal with this by distinguishing cases. There are two ways of making the conclusion false, and we investigate each case in turn to see if there is any way of extending the assignment to make all the premises true:

> *Case 1:* ‘C’ is true, ‘A’ is false. Then for premise 3 to be true, we require ‘E’ to be true, and so for premise 2 to be true, we require ‘B & A’ to be true, but we already have ‘A’ false. Consequently, there is no way of extending the assignment ‘C’ true, ‘A’ false, to the other sentence-letters so that all the premises are true. *But this does not mean that D is valid.* For we still have to consider the other way of making the conclusion false.
>
> *Case 2:* ‘C’ is false, ‘A’ is true. Then for premise 1 to be true, ‘B → C’ must be true, which requires ‘B’ to be false, since we already have ‘C’ false. (Why take premise 1 first this time? Because the assignment of $\bot$ to ‘C’ and $\top$ to ‘A’ does not determine the truth-value of ‘E’ in the simplest premise, premise 3, while it does determine the truth-value of ‘B’ in premise 1. When nothing can be deduced about the truth-values of the sentence-letters in a premise, given the assignments already made, we look for another premise where something *can* be deduced.) We now have ‘C’ false, ‘A’ true and ‘B’ false. Hence for premise 2 to be true we must have ‘E’ false, and this also makes premise 3 true. So in Case 2 we arrive at an interpretation on which D’s premises are true and its conclusion false.

Our overall conclusion, therefore, is that D is invalid, as established by the following interpretation:

E	B	C	A
⊥	⊥	⊥	⊤

Notice that we do not say that D is 'valid in Case 1' and 'invalid in Case 2'. Such locutions mean nothing. Either there is an interpretation which makes D's premises true and conclusion false or there is not, and so D is either invalid or valid *simpliciter*. The notions of validity and invalidity do not permit relativization to cases. What we find in Case 1 is not that D is 'valid in Case 1', but rather that Case 1's way of making the conclusion false does not lead to a demonstration of invalidity for D.

Example D suggests that judicious choice of order in which to consider cases can reduce the length of the discussion, for if we had taken Case 2 first there would have been no need to consider Case 1: as soon as we have found a way of making the premises true and the conclusion false we can stop, and pronounce the argument-form invalid. It is not always obvious which case is the one most likely to lead to an interpretation that shows an invalid argument-form to be invalid, but with some experience we can make intelligent guesses.

When an argument-form is valid we say that its premises *semantically entail* its conclusion, or that its conclusion is a *semantic consequence* of its premises. So in example B we have established that '(~J → C) & (E ↔ C)' and '~E' semantically entail 'J', or that 'J' is a semantic consequence of '(~J → C) & (E ↔ C)' and '~E'; and also, in example D, that '~A ∨ (B → C)', 'E → (A & B)' and 'C → E' do not semantically entail 'C ↔ A'. There are useful abbreviations for semantic entailment and non-entailment in a formal language like LSL: for entailment we use the symbol '⊨', known as the *double-turnstile*, and for non-entailment we use the symbol '⊭', known as the *cancelled double-turnstile*. The turnstiles also implicitly put quotes around formulae where they are required. So we can express the results of this section as follows:

(1) $(\sim J \rightarrow C) \,\&\, (E \leftrightarrow C), \sim E \vDash J$
(2) $A \rightarrow (B \,\&\, E), D \rightarrow (A \vee F), \sim E \nvDash D \rightarrow B$
(3) $\sim A \vee (B \rightarrow C), E \rightarrow (B \,\&\, A), C \rightarrow E \nvDash C \leftrightarrow A.$

Thus semantic entailment is essentially the same notion as validity, and semantic nonentailment essentially the same notion as invalidity. If we are asked to evaluate an expression such as (1), which says that '(~J → C) & (E ↔ C)' and '~E' semantically entail 'J', we simply use one or another technique for determining whether or not there is an interpretation which makes the premises of (1) true and the conclusion false. If there is, (1) is false, if there is not, (1) is true. Note the correct use of 'valid' versus 'true'. Arguments are valid or invalid, not true or false. But (1) is a statement *about* some premises and a conclusion: it says that the conclusion follows from the premises, and this statement itself is either true or false. Statements like (1), which contain the double-turnstile, are called *semantic sequents*.

Since entailment is essentially the same as validity, the formal definition of the symbol '$\vDash$' for LSL is just like the definition of 'valid argument-form of LSL':

> For any formulae $p_1,\ldots,p_n$ and q of LSL, $p_1,\ldots,p_n \vDash q$ if and only if there is no interpretation of the sentence-letters in $p_1,\ldots,p_n$ and q under which $p_1,\ldots,p_n$ are all true and q is false.

In the special case where there are no $p_1,\ldots,p_n$—or as it is sometimes put, where $n = 0$—we delete from the definition the phrases which concern $p_1,\ldots,p_n$. This leaves us with 'For any formula q of LSL, $\vDash q$ if and only if there is no interpretation under which q is false'. If there is no interpretation on which q is false, this means q is true on every interpretation, in other words, that q is a tautology, and so we read '$\vDash q$' as 'q is a tautology'. '$\nvDash q$', then, means that q is either contingent or a contradiction.

❑ Exercises

I Use the method of constructing interpretations to determine whether the following statements are correct. Explain your reasoning in the same way as in the worked examples, and if you claim a sequent is incorrect, exhibit an interpretation which establishes this.

(1) $A \rightarrow B, B \rightarrow (C \vee D), {\sim}D \vDash A \rightarrow C$
(2) $(A \,\&\, B) \rightarrow C, B \rightarrow D, C \rightarrow {\sim}D \vDash {\sim}A$
(3) $A \rightarrow (C \vee E), B \rightarrow D \vDash (A \vee B) \rightarrow (C \rightarrow (D \vee E))$
*(4) $A \rightarrow (B \,\&\, C), D \rightarrow (B \vee A), C \rightarrow D \vDash A \leftrightarrow C$
(5) $A \vee (B \,\&\, C), C \vee (D \,\&\, E), (A \vee C) \rightarrow ({\sim}B \vee {\sim}D) \vDash B \,\&\, D$
(6) $A \rightarrow (B \rightarrow (C \rightarrow D)), A \,\&\, C, C \rightarrow B \vDash {\sim}B \leftrightarrow (D \,\&\, {\sim}D)$
(7) $(A \leftrightarrow B) \,\&\, (B \leftrightarrow C) \vDash (A \vee {\sim}A) \,\&\, ((B \vee {\sim}B) \,\&\, (C \vee {\sim}C))$
(8) $(A \leftrightarrow B) \vee (B \leftrightarrow C) \vDash A \leftrightarrow (B \vee C)$
(9) $({\sim}A \,\&\, {\sim}B) \vee C, (A \rightarrow D) \,\&\, (B \rightarrow F), F \rightarrow (G \vee H) \vDash {\sim}G \rightarrow (H \vee C)$

II Test the following English arguments for sentential validity by translating them into LSL and testing each of the resulting LSL arguments for validity, using the method of constructing interpretations. Give a complete dictionary for each argument and be sure not to use different sentence-letters of LSL for what is essentially the same simple sentence of English. Explain your reasoning in the same way as in the worked examples, and if you claim an argument is invalid, exhibit an interpretation which establishes this.

(1) The next president will be a woman only if the party that wins the next election has a woman leader. Since no party has a woman leader at the moment, then unless some party changes its leader or a new party comes into being, there will be no female president for a while. Therefore, unless a new party comes into being, the next president will be a man.

(2) B is a Knave, since if he is a Knight then what he says is false and in that case he is not a Knight. (In symbolizing this argument, assume everyone is either a Knight or a Knave.)

(3) A theory which has been widely accepted in the past is always refuted eventually. This being so, unless we are much more intelligent than our predecessors, the Theory of Relativity is bound to be refuted. If we are more intelligent, the human brain must have increased in size, which requires that heads be bigger. You say the human head has not got bigger in historical times. If you are right about this, the Theory of Relativity will be refuted.

*(4) If Yossarian flies his missions then he is putting himself in danger, and it is irrational to put oneself in danger. If Yossarian is rational he will ask to be grounded, and he will be grounded only if he asks. But only irrational people are grounded, and a request to be grounded is proof of rationality. Consequently, Yossarian will fly his missions whether he is rational or irrational. (F, D, R, A) (In symbolizing this argument, treat statements about people in general as if they concerned Yossarian specifically; e.g., symbolize 'only irrational people are grounded' as a statement about Yossarian.)

(5) If the safe was opened, it must have been opened by Smith, with the assistance of Brown or Robinson. None of these three could have been involved unless he was absent from the meeting. But we know that either Smith or Brown was present at the meeting. So since the safe was opened, it must have been Robinson who helped open it.

(6) If God is willing to prevent evil but is unable to do so, He is impotent. If God is able to prevent evil but unwilling to do so, He is malevolent. Evil exists if and only if God is unwilling or unable to prevent it. God exists only if He is neither impotent nor malevolent. Therefore, if God exists then evil does not exist.

(7) We don't need a space station except if we need people in orbit, and we only need people in orbit if there are going to be manned expeditions to other planets, and then only if launch technology doesn't improve. A space station is a pointless extravagance, therefore, since interplanetary exploration will all be done by machines if we don't find better ways of getting off the ground.

(8) The Mayor will win if the middle class votes for her. To prevent the latter, her rivals must credibly accuse her of corruption. But that charge won't stick if she isn't corrupt. So honesty assures the Mayor of victory.

(9) A decrease in crime requires gun control. So the Mayor's only winning strategy involves banning guns, because he won't win without the middle-class vote and he'll lose that vote unless crime goes down. (4 sentence-letters)

(10) The Mayor's three problems are crime, corruption and the environment. He can't do anything about the first and he won't do anything about the third. So he won't be reelected, since winning would require that he solve at least two of them.

5 Testing for validity with semantic tableaux

The arguments which we gave in the previous section to establish semantic consequence and failure of semantic consequence, though rigorous, are rather unstructured: at various points it is left to the reasoner to decide what to do next, and it is not always obvious what that should be. In this section we briefly describe a format for testing argument-forms for validity (semantic sequents for correctness) which imposes rather more structure on the process, a format known as a *semantic tableau*. To determine whether or not $p_1,\ldots,p_n \vDash q$ we build a *search tree* which grows downward—like parse trees, search trees are inverted—as we extend our search for an assignment of truth-values which makes $p_1,\ldots,p_n$ all true and q false. The reader may wish to review the tree terminology on page 37 of Chapter 2.5.

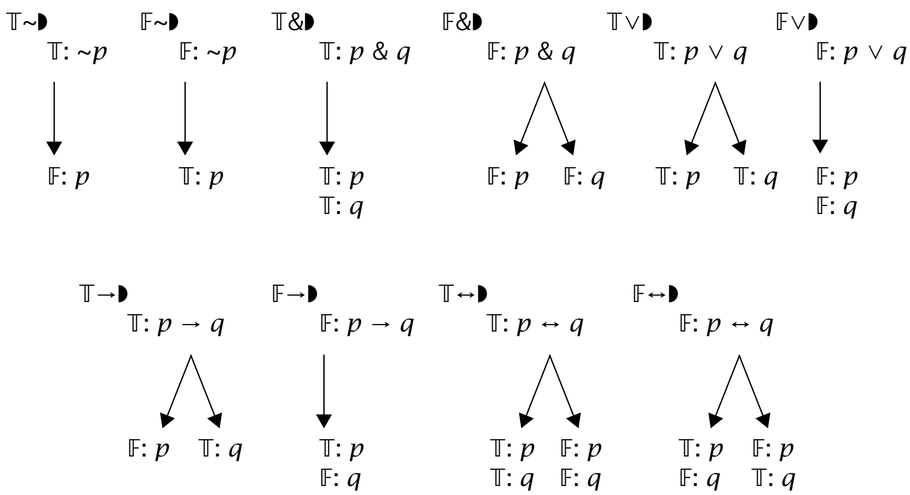
A search tree begins at the top or root node with a list of *signed formulae*. A signed formula is a formula preceded by either the characters '$\mathbb{T}$:' or the characters '$\mathbb{F}$:'. '$\mathbb{T}$:' may be thought of as abbreviating 'it is true that' and '$\mathbb{F}$:' as abbreviating 'it is false that'. There is a $\mathbb{T}$-rule and an $\mathbb{F}$-rule for each connective, and the rule reflects that connective's truth-table. The tree is extended downward by applying $\mathbb{T}$-rules and $\mathbb{F}$-rules to the main connectives of formulae at nodes on it. For example, the $\mathbb{T}$-rule for a conjunction $\ulcorner p \;\&\; q\urcorner$ allows us to extend a tree with $\ulcorner\mathbb{T}{:}p \;\&\; q\urcorner$ at a node n by adding a new node to the bottom of every path on which node n lies; the new node is labeled with the signed formulae $\ulcorner\mathbb{T}{:}p\urcorner$ and $\ulcorner\mathbb{T}{:}q\urcorner$. We call this rule $\mathbb{T}$-&. The corresponding $\mathbb{F}$-rule for a conjunction $\ulcorner p \;\&\; q\urcorner$ allows us to extend a tree which has $\ulcorner\mathbb{F}{:}p \;\&\; q\urcorner$ at a node n by splitting the tree at the bottom node of every path on which n lies, the new left node on each such path holding the signed formula $\ulcorner\mathbb{F}{:}p\urcorner$ and the new right node holding the signed formula $\ulcorner\mathbb{F}{:}q\urcorner$. We call this rule $\mathbb{F}$-&. The collection of rules we get by providing a $\mathbb{T}$-rule and an $\mathbb{F}$-rule for each connective is known as the collection of *tableau rules*. These rules reflect the requirements for a formula of the relevant sort to have the stated truth-value. Thus the rule $\mathbb{T}$-& reflects the fact that for $\ulcorner p \;\&\; q\urcorner$ to be true, p must be true and q must be true, while the rule $\mathbb{F}$-& reflects the fact that for $\ulcorner p \;\&\; q\urcorner$ to be false, either p must be false or q must be false. The disjunction here corresponds to the splitting in the search tree (and to the distinguishing of cases in some of the informal arguments in the previous section).

To search for an interpretation on which $p_1,\ldots,p_n$ are all true and q false, we construct a search tree at whose root node we list the signed formulae $\mathbb{T}{:}p_1\ldots\mathbb{T}{:}p_n$ and $\mathbb{F}{:}q$. We then extend the tree downward by applying the tableau rules. We may apply the rules to any signed formulae in any order, but whichever order we choose, one or the other of two outcomes is inevitable. To describe these outcomes, it is useful to define the notion of a path in a semantic tableau or search tree being *closed:*

> A path Π in a semantic tableau is *closed* if and only if there is a formula s and nodes n_1 and n_2 of Π such that $\ulcorner\mathbb{T}{:}\ s\urcorner$ occurs at n_1 and $\ulcorner\mathbb{F}{:}\ s\urcorner$ occurs at n_2. Π is *open* if and only if it is not closed.

The first of the two possible outcomes is that at some stage of the search, every path in the tableau closes. In this case the tableau is itself said to be closed and the search for an interpretation making $p_1,\ldots,p_n$ all true and q false has failed. So $p_1,\ldots,p_n \vDash q$. The second possible outcome is that we have eventually applied all possible rules but at least one path in the tableau is still open, that is, the path has no nodes n_1 and n_2 such that for some formula s, $\ulcorner\mathbb{T}: s\urcorner$ occurs at n_1 and $\ulcorner\mathbb{F}: s\urcorner$ occurs at n_2. In this case the search for an interpretation on which $p_1,\ldots,p_n$ are all true and q false has succeeded, for as we shall see, if we take an open path Π at this stage and assign $\top$ to every atomic sentence p such that $\mathbb{T}$: p is on Π and $\bot$ to every atomic sentence p such that $\mathbb{F}$: p is on Π, we obtain an interpretation which makes all of $p_1,\ldots,p_n$ true and q false. So in this situation, we can conclude $p_1,\ldots,p_n \nvDash q$.

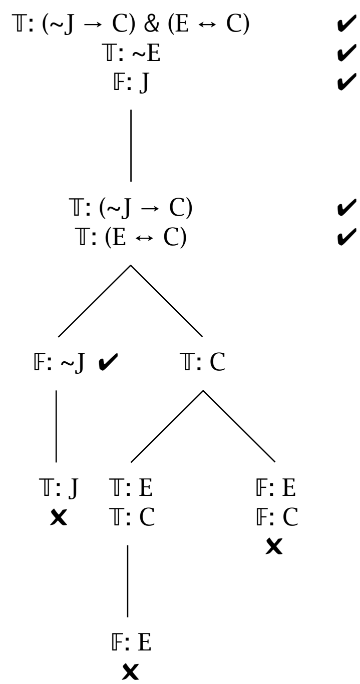
The tableau rules for the connectives of LSL are as follows:



In interpreting these rule diagrams, remember that an occurrence of a formula at a node in a tree may lie on many paths, since branching may occur below that node and a path is a total route through a tree from top to bottom. With this in mind, think of a rule with a single arrow as an instruction: to apply it to an occurrence of a formula of the form at the tail of the arrow (i.e. the upper formula), we extend all currently open paths on which this formula-occurrence lies by adding a new node to each, labeling the new node with the signed formula(e) at the head of the arrow. To apply a rule with a branching arrow to an occurrence of a formula of the form at the tail of the arrow, we extend all currently open paths containing the formula occurrence by branching to two new nodes, labeled as indicated.

We illustrate the technique by repeating two examples from §4, this time using tableaux. First, we establish the validity of the argument-form B of §4 by showing that a tableau for it closes.

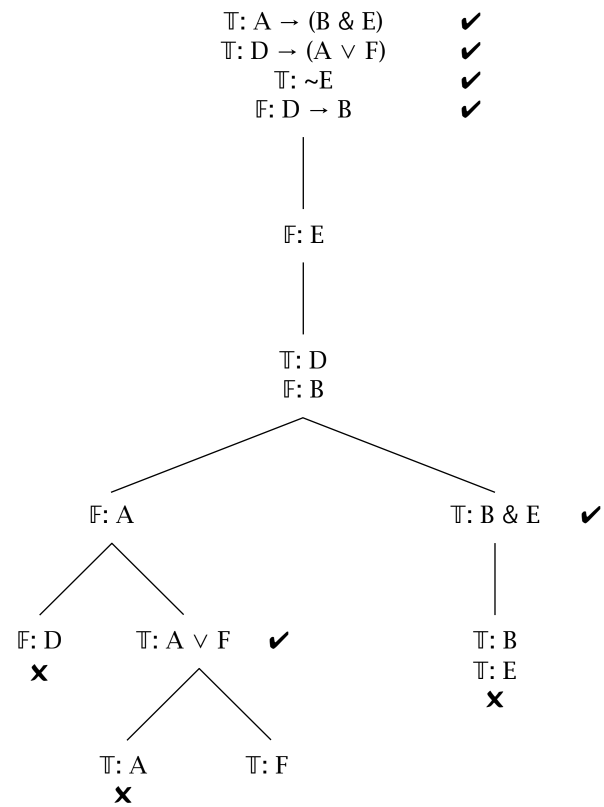
Example 1: Show (~J → C) & (E ↔ C), ~E ⊨ J.



Each node in this tree is generated by applying one of the tableau rules to a signed formula at a previous node, though not necessarily the node immediately above. Once we have applied a tableau rule to a signed formula *s*, we check *s* with the dingbat '✔'. We mark a closed path by positioning the dingbat '✘' under its leaf. In this example there are three paths (because there are three bottom nodes), and all three are marked as closed. Path 1, the path down the tree which terminates in the node labeled '𝕋: J', is closed because it also has '𝔽: J' on it; path 2, the path down the tree which terminates in the node labeled '𝔽: E', is closed because it also has '𝕋: E' on it (the last rule used on this path is 𝕋~, which is applied to the middle formula at the root node, producing the occurrence of '𝔽: E' which closes the path); and path 3, the path down the tree which terminates in the node labeled '𝔽: C', is closed because it also has '𝕋: C' on it. It is worth noting that in general, the size of a tree can be kept to a minimum by applying rules like 𝕋~ which do not cause branching before applying rules like 𝕋↔ which do cause branching. However, in Example 1 only one path is still open at the point at which 𝕋~ is applied to the root node, and so only one copy of a node labeled with '𝔽: E' has to be added to the tree.

The next example is argument-form C in §4, which we already know to be invalid. So a tree search for an interpretation establishing invalidity should succeed, that is, the tree should have at least one open path. Here is the tree.

Example 2: Determine whether A → (B & E), D → (A ∨ F), ~E ⊨ D → B.



Paths 1, 2 and 4 are closed. However, path 3 does not satisfy the condition for being closed, and all nonatomic signed formulae above its leaf are checked, so there is nothing more we can do to attempt to close the path. Thus our search ends with the path still open, which shows that A → (B & E), D → (A ∨ F), ~E ⊭ D → B. As we already indicated, we can derive an interpretation which establishes this failure of semantic consequence by looking at the signed atomic formulae on the open path, in this case '𝔽: E', '𝕋: D', '𝔽: B', '𝔽: A' and '𝕋: F'. Reading truth-value assignments off the signatures, we obtain exactly the interpretation demonstrating invalidity that we arrived at through applying the method of constructing interpretations to argument-form C in §4, as exhibited on page 64.

❑ Exercises

Repeat the exercises of I of §4 using semantic tableaux rather than the method of constructing interpretations.

6 Properties of semantic consequence

One advantage of the double-turnstile notation is that it enables us to raise in a convenient form certain questions about relationships among argument-forms. For example, in symbolizing A of §4 as B (page 63), we treated all of the first sentence after 'for' as a single premise, a conjunction. Would it have made any difference if we had symbolized that piece of English as two separate premises? This question is a general one, and we can put it in the following way: for any formulae p, q and r of LSL,

Example 1: If $(p \mathbin{\&} q) \vDash r$, does it follow that $p, q \vDash r$?

Example 2: If $p, q \vDash r$, does it follow that $(p \mathbin{\&} q) \vDash r$?

In other words, as far as LSL validity is concerned, is there any significant difference between multiple premises versus a conjunction with multiple conjuncts?

It is not too difficult to see that the answers in both examples are yes. But there is a general strategy for answering questions like these which it is useful to be able to illustrate with simple examples. Let us begin with Example 1. We reason as follows:

> *Example 1:* If $(p \mathbin{\&} q) \vDash r$ then by definition of '$\vDash$' this means that no interpretation makes $\ulcorner p \mathbin{\&} q \urcorner$ true and r false. So by the table for '&', no interpretation makes p true, q true and r false. But if $p, q \nvDash r$ then some interpretation makes p true, q true and r false, which is what we have just ruled out. Consequently, if $(p \mathbin{\&} q) \vDash r$ it cannot be that $p, q \nvDash r$, so it follows that $p, q \vDash r$.
>
> *Example 2:* If $p, q \vDash r$ then this means that no interpretation makes p true, q true and r false. On the other hand, if $(p \mathbin{\&} q) \nvDash r$ then some interpretation makes $\ulcorner p \mathbin{\&} q \urcorner$ true and r false, which means that it makes p true, q true and r false. But that is what we have just ruled out. Hence it cannot be that $(p \mathbin{\&} q) \nvDash r$, so it follows that $(p \mathbin{\&} q) \vDash r$.

In problems like these, we are using the metavariables 'p', 'q' and 'r' to abstract from sequents with LSL formulae as their premises and conclusions, and are considering instead sequents with metalanguage formulae that describe patterns which sequents with LSL formulae may instantiate. One such abstract sequent is *given*, and another is *queried*. The strategy for solving the problem is to begin by using the definition of '$\vDash$' to determine which kinds of interpretation are ruled out by the given sequent. To determine whether the queried sequent is correct, we then ask what kinds of interpretation would show that it is *not* correct. If all interpretations of these kinds have been ruled out by the given sequent, the queried sequent does follow from the given one. But if they have not all been ruled out, the queried sequent does not follow from the given one. For any particular formulae p, q and r, the resulting queried sequent may of course be semantically correct. The issue is whether the correctness of the given sequent guarantees this for all formulae p, q and r.

Here are two further examples, where p, q and r are any sentences of LSL:

Example 3: If $(p \leftrightarrow q) \vDash (p \,\&\, q)$, does it follow that $(p \vee q) \vDash (p \,\&\, q)$?

> *Answer:* If $(p \leftrightarrow q) \vDash (p \,\&\, q)$ then no interpretation makes $\ulcorner p \leftrightarrow q \urcorner$ true and $\ulcorner p \,\&\, q \urcorner$ false. Thus the interpretations which are ruled out are those which make p false and q false. (Remember that p and q are any LSL wffs, not necessarily atomic ones, so there might be more than one interpretation which results in p false, q false.) If $(p \vee q) \nvDash (p \,\&\, q)$ then some interpretation makes $\ulcorner p \vee q \urcorner$ true and $\ulcorner p \,\&\, q \urcorner$ false. Any such interpretation must either have p true and q false or vice-versa. In fact, neither of these combinations has been ruled out, so it does not follow from $(p \leftrightarrow q) \vDash (p \,\&\, q)$ that $(p \vee q) \vDash (p \,\&\, q)$.

Example 4: If $\vDash (p \rightarrow (q \rightarrow r))$, does it follow that $\sim r \vDash (\sim p \rightarrow \sim q)$?

> *Answer:* If $\vDash (p \rightarrow (q \rightarrow r))$ then no interpretation makes p true, q true and r false. If $\sim r \nvDash (\sim p \rightarrow \sim q)$ then some interpretation makes $\ulcorner \sim r \urcorner$ true and $\ulcorner \sim p \rightarrow \sim q \urcorner$ false, that is, it makes r false, p false and q true. Such an interpretation has not been ruled out, so from $\vDash (p \rightarrow (q \rightarrow r))$ it does not follow that $\sim r \vDash (\sim p \rightarrow \sim q)$.

The semantic consequence relation is the fundamental concept of modern logic. It is just as important to be able to reason about it, as in Examples 1–4 above, as it is to be able to execute a technique for detecting when it holds and when it fails, like the ones developed in §3–§5 of this chapter.

❑ Exercises

In the following, p, q, r and s are any sentences of LSL. In (1)–(6), explain your reasoning in the manner illustrated by the examples of this section.

(1) If $\vDash p \rightarrow (q \rightarrow r)$, does it follow that $p, q \vDash r$? Does it follow that $r \vDash (p \rightarrow q)$?
(2) If $(p \rightarrow q) \vDash r$, does it follow that $p \vDash (q \rightarrow r)$?
*(3) If $p \vDash (q \,\&\, r)$, does it follow that $(p \leftrightarrow q) \vDash (p \leftrightarrow r)$?
(4) If $(p \leftrightarrow q) \vDash (r \vee s)$, does it follow that $(p \,\&\, q) \vDash r$?
(5) If $(p \rightarrow q) \vDash (\sim r \vee s)$, does it follow that $(p \,\&\, r) \vDash \sim q$?
(6) If $(p \,\&\, q) \vee (r \,\&\, s) \vDash (p \rightarrow r) \leftrightarrow (q \rightarrow s)$, does it follow that $(p \,\&\, \sim r), \sim q \vDash \sim s$?
(7) Although there is no sentence-letter in common between premise and conclusion, $(\text{A} \,\&\, \sim\text{A}) \vDash \text{B}$. Briefly explain why.
*(8) Although there is no sentence-letter in common between premise and conclusion, $\text{B} \vDash (\text{A} \vee \sim\text{A})$. Briefly explain why.

7 Expressive completeness

At the end of §1 in Chapter 2 we claimed that our five sentential connectives '~', '∨', '&', '→' and '↔' are all we need in sentential logic, since other sentential connectives are either *definable* in terms of these five or else *beyond the scope* of sentential logic. When we say that a connective is beyond the scope of classical sentential logic, what we mean is that it is *non-truth-functional;* in other words, there is no truth-function that it expresses (see §1 of this chapter for a discussion of expressing a truth-function). In the next section we will consider various connectives of this sort. Meanwhile, we will concern ourselves with the definability of other truth-functional connectives.

An example of a truth-functional connective which is definable in terms of our five is 'neither…nor…', since for any English sentences p and q, ⌜neither p nor q⌝ is correctly paraphrased as ⌜not p and not q⌝ (see (11) on page 18). But this is just one example. How can we be confident that *every* truth-functional connective can be defined in terms of '~', '∨', '&', '→' and '↔'? Our confidence is based in the fact that our collection of connectives has a property called *expressive completeness*, which we now explain.

At the end of §1 of this chapter, we listed the function-tables for the one-place, or *unary*, function expressed by '~', and the four two-place, or *binary*, functions expressed by the other connectives. However, there are many more unary and binary truth-functions than are expressed by the five connectives individually. For example, there are three other unary truth-functions:

$\top \Rightarrow \top$	$\top \Rightarrow \top$	$\top \Rightarrow \bot$
$\bot \Rightarrow \top$	$\bot \Rightarrow \bot$	$\bot \Rightarrow \bot$
(a)	(b)	(c)

To show that all unary truth-functional connectives are definable in terms of our five basic connectives, we establish the stronger result that all unary truth-functions are definable, whether or not they are expressed by some English connective. (While (b) is expressed by 'it is true that', neither (a) nor (c) has an uncontrived rendering.) Our question is therefore whether we can express all of (a), (b) and (c) in terms of our five chosen connectives. And in this case it is easy to see that (a) is captured by '…∨ ~…', (b) by '~~…', and (c) by '…& ~…', where in (a) and (c) the same formula fills both ellipses.

What about the other binary truth-functions? We have connectives for four, and we know how to define a fifth, the truth-function

$$\begin{array}{l} \top\top \Rightarrow \bot \\ \top\bot \Rightarrow \bot \\ \bot\top \Rightarrow \bot \\ \bot\bot \Rightarrow \top \end{array}$$

which is expressed by 'neither…nor…' (to repeat, ⌜neither p nor q⌝ is true just in case both p and q are false, so we express it with ⌜$\sim p \ \& \sim q$⌝). But there are many more binary truth-functions, and again we are concerned to define all of them, not merely those which correspond to some idiomatic phrase like 'neither…nor…'. First, how many other binary functions are there? There are as many as there are different possible combinations of outputs for the four pairs of truth-values which are the inputs to binary truth-functions. Hence there are sixteen different binary truth-functions: the output for the first pair of input truth-values $\top\top$ is either $\top$ or $\bot$, giving us two cases, and in each of these cases, the output for the second pair of inputs $\top\bot$ is either $\top$ or $\bot$, giving a total of four cases so far, and so on, doubling the number of cases at each step for a total of sixteen. So apart from examining each of the remaining eleven binary truth-functions one by one, is there a *general* reason to assert that they can all be defined by our five connectives?

Even supposing that we can give a general reason why all binary truth-functions should be definable in terms of our five, that would not be the end of the matter, since for every n, there are truth-functions of n places, though when $n > 2$ they rarely have a 'dedicated' English phrase which expresses them.[3] The claim that our five chosen connectives suffice for sentential logic is the claim that for any n, every truth-function of n places can be expressed by our five connectives. This is our explanation of the notion of expressive completeness of a collection of connectives, which we embody in a definition:

> A set of connectives S is *expressively complete* if and only if for every n, all n-place truth-functions can be expressed using only connectives in S.

The set of connectives which we wish to prove expressively complete is $\{\sim,\&,\vee,\rightarrow,\leftrightarrow\}$ (the curly parentheses, or *braces*, are used for sets, with all the members of the set exhibited between them). But what does it mean to say that a truth-function is *expressed* using connectives in this set? This means that there is a *formula* which expresses the truth-function and which is built up from sentence-letters and connectives in the set. And this in turn is explained using truth-tables. We observe that every truth-function corresponds to a truth-table (and conversely). For example, the three-place function

$$\begin{array}{l}
\top\top\top \Rightarrow \top\\
\top\top\bot \Rightarrow \bot\\
\top\bot\top \Rightarrow \bot\\
\top\bot\bot \Rightarrow \top\\
\bot\top\top \Rightarrow \top\\
\bot\top\bot \Rightarrow \top\\
\bot\bot\top \Rightarrow \top\\
\bot\bot\bot \Rightarrow \top
\end{array}$$

corresponds to the truth-table laid out on page 55. In general, given a function-

[3] The phrase 'if…then…, otherwise…' is an example of a locution expressing a three-place sentential connective. However, ⌜if p then q, otherwise r⌝ can be paraphrased as ⌜if p then q and if not-p then r⌝.

table, the corresponding truth-table is the table with the output of the function as its final column. We say that a function is *expressed* by a formula, or a formula *expresses* a function, if that formula's truth-table is the table corresponding to the function. So the three-place function just exhibited is expressed by the formula '(A → (B ∨ C)) → (A → (B & C))' from page 55.

Consequently, to show that every truth-function is expressible in terms of the five connectives of LSL, it suffices that we show how, given any truth-table, we can recover a formula which contains only LSL connectives and whose truth-table it is. In other words, we have to develop a technique that is the reverse of the one we have for constructing truth-tables, given formulae; the problem now is to construct formulae, given truth-tables.

There is a systematic way of doing this. A truth-table lists various possible interpretations, that is, assignments of truth-values to certain sentence-letters. Say that a formula *defines* an assignment $\mathcal{I}$ of truth-values to sentence-letters $\pi_1,\ldots,\pi_n$ if and only if that formula is true on $\mathcal{I}$ and on no other assignment to $\pi_1,\ldots,\pi_n$. Then given an assignment $\mathcal{I}$ to $\pi_1,\ldots,\pi_n$, one can use $\pi_1,\ldots,\pi_n$ to construct a formula in '&' and '~' which defines $\mathcal{I}$ as follows: take each sentence-letter which is assigned ⊤ and the negation of each which is assigned ⊥ and form the conjunction of these letters and negated letters. So, for example, the interpretation consisting in the assignment of ⊥ to 'C', ⊤ to 'D', ⊥ to 'B', ⊥ to 'A' and ⊤ to 'F' is defined by '~C & D & ~B & ~A & F', since this formula is true on that assignment, and only that one, to those sentence-letters. Now suppose we are given a randomly chosen truth-table with 2^n rows and a final column of entries, but no formula and no sentence-letters are specified. It is easy to find a formula for the table in the two special cases in which all interpretations lead to ⊤ or all to ⊥. Otherwise, to construct a formula for the table, we choose sentence-letters $\pi_1,\ldots,\pi_n$ and use them to construct the formulae which define the interpretations where there is a ⊤ in the final column of the table, and disjoin these interpretation-defining formulae together. This produces a disjunction such that each disjunct is true on exactly one row of the table (the one it defines), making the whole disjunction true at that row. For each row where there is a ⊤ there is a disjunct in the constructed formula with this effect. And the formula has no other components. Therefore it is true on exactly the rows in the table where there is a ⊤. Consequently, this disjunction expresses the truth-function given by the table.

In sum, we have the following three-step procedure for constructing a formula for any truth-table:

- If there are no ⊤s in the final column, let the formula be 'A & ~A'; if there are no ⊥s, let it be 'A ∨ ~A'.
- Otherwise, using the appropriate number of sentence-letters 'A', 'B' and so on, for each interpretation which gives a ⊤ in the final column of the table construct a conjunction of sentence-letters and negated sentence-letters defining that interpretation.
- Form a disjunction of the formulae from the previous step.

Here are two applications of this technique.

Example 1:

Function			
⊤⊤	⇒	⊥	
⊤⊥	⇒	⊤	
⊥⊤	⇒	⊥	
⊥⊥	⇒	⊤	

Function

A B	
⊤ ⊤	⊥
⊤ ⊥	⊤
⊥ ⊤	⊥
⊥ ⊥	⊤

Corresponding table

The second and fourth interpretations (inputs) produce ⊤s in the final column of the table. The second interpretation is: ⊤ assigned to 'A', ⊥ to 'B', so its defining formula is 'A & ~B'. The fourth interpretation is: ⊥ assigned to 'A', ⊥ to 'B', so its defining formula is '~A & ~B'. Consequently, the formula we arrive at is '(A & ~B) ∨ (~A & ~B)', and a simple calculation confirms that this formula does indeed have the displayed truth-table. Of course, there are many other (in fact, infinitely many other) formulae which have this table. For example, the reader may have quickly noticed that the formula '(A ∨ ~A) & ~B' also has the table in Example 1. But to show that the truth-function is expressible, all we have to find is *at least one* formula whose table is the table corresponding to the function, and our step-by-step procedure will always produce one. Moreover, when we consider functions of three places or more it is no longer so easy to come up with formulae for their corresponding tables simply by inspecting the entries and experimenting a little. So it is best to follow the step-by-step procedure consistently, as in our next example.

Example 2:

⊤⊤⊤	⇒	⊤
⊤⊤⊥	⇒	⊥
⊤⊥⊤	⇒	⊥
⊤⊥⊥	⇒	⊤
⊥⊤⊤	⇒	⊥
⊥⊤⊥	⇒	⊤
⊥⊥⊤	⇒	⊥
⊥⊥⊥	⇒	⊤

Function

A B C	
⊤ ⊤ ⊤	⊤
⊤ ⊤ ⊥	⊥
⊤ ⊥ ⊤	⊥
⊤ ⊥ ⊥	⊤
⊥ ⊤ ⊤	⊥
⊥ ⊤ ⊥	⊤
⊥ ⊥ ⊤	⊥
⊥ ⊥ ⊥	⊤

Corresponding table

Here it is interpretations 1, 4, 6 and 8 which produce a ⊤. The four formulae defining these interpretations are respectively: 'A & B & C', 'A & ~B & ~C', '~A & B & ~C' and '~A & ~B & ~C'. Therefore a formula for the table, grouping disjuncts conveniently, is:

[(A & (B & C)) ∨ (A & (~B & ~C))] ∨ [(~A & (B & ~C)) ∨ (~A & (~B & ~C))].

It should be clear from this method that we are proving something stronger than that the set of LSL connectives {~,&,∨,→,↔} is expressively complete, since our procedure for finding a formula for an arbitrary table involves only the connectives in the subset {~,&,∨}. What we are showing, therefore, is that {~,&,∨} is expressively complete, from which the expressive completeness of {~,&,∨,→,↔} follows trivially: if we can express any truth-function by some formula in {~,&,∨}, then the same formula is a formula in {~,&,∨,→,↔} which expresses the truth-function in question (the point is that *p*'s being a formula in {~,&,∨,→,↔} requires that *p* contain no *other* connectives, but not that it contain occurrences of *all* members of {~,&,∨,→,↔}). But can we do better than this? That is, is there an even smaller subset of {~,&,∨,→,↔} which is expressively complete? Given the expressive completeness of {~,&,∨}, a simple way to show that some other set of connectives is expressively complete is to show that *the connectives of the other set can define those of* {~,&,∨}.

What is it for one or more connectives to define another connective? By this we mean that there is a rule which allows us to replace every occurrence of the connective *c* to be defined by some expression involving the defining connectives. More precisely, if *p* is a formula in which there are occurrences of *c*, then we want to select every subformula *q* of *p* of which *c* is the main connective, and replace each such *q* with a formula *q′* logically equivalent to *q* but containing only the defining connectives. For instance, we already know that we can define '↔' using the set of connectives {&,→}, since every occurrence of '↔' in a formula *p* is as the main connective of a subformula ⌜(*r* ↔ *s*)⌝, and we have the following substitution rule:

- Replace each subformula *q* of *p* of the form ⌜(*r* ↔ *s*)⌝ with ⌜((*r* → *s*) & (*s* → *r*))⌝.

Applying this substitution rule throughout *p* yields a logically equivalent formula *p′* which contains no occurrence of '↔'. For example, we eliminate every occurrence of '↔' from 'A ↔ (B ↔ C)' in two steps (it does not matter which '↔' we take first):

Step 1: [A → (B ↔ C)] & [(B ↔ C) → A]
Step 2: [A → ((B → C) & (C → B))] & [((B → C) & (C → B)) → A]

Using a similar approach, we can show that the set of connectives {~,&} is expressively complete. Given that {~,&,∨} is expressively complete, the problem can be reduced to defining '∨' in terms of '~' and '&'. For a disjunction to be true, at least one disjunct must be true, which means that it is not the case that the disjuncts are both false. So the substitution rule should be:

- Replace every subformula of the form ⌜(*r* ∨ *s*)⌝ with ⌜~(~*r* & ~s)⌝.

Since each substitution produces a logically equivalent formula, or as we say, since *substitution preserves logical equivalence*, then if we begin with a formula in {~,&,∨} for a given truth-table, and replace every occurrence of '∨' using the substitution rule, we end up with a formula in {~,&} for that same truth-table. Since every table has a formula in {~,&,∨}, it follows that every table has a formula in {~,&}, hence {~,&} is expressively complete. For example, we have already seen that '(A & ~B) ∨ (~A & ~B)' is a formula for the table of Example 1 above. Applying the substitution rule to the one occurrence of '∨' in this formula yields the logically equivalent formula

~[~(A & ~B) & ~(~A & ~B)]

which is therefore also a formula for the table in Example 1.

By the same technique we can show that {~,∨} and {~,→} are expressively complete (these are exercises). However, not every pair of LSL connectives is expressively complete; for example, {&,∨} and {~,↔} are not. This can be proved rigorously by a technique known as mathematical induction, but for our purposes it is enough to understand one example, {&,∨}, intuitively. The point is that negation cannot be expressed in terms of {&,∨}, since no formula built out of 'A' and '&' and '∨' can be false when 'A' is true. Since '~A' is false when 'A' is true, this means no formula built out of 'A' and '&' and '∨' has the same truth-table as '~A'. Thus {&,∨} is expressively *in*complete.

There are two connectives which are expressively complete *individually*, though neither belongs to LSL. One is a symbol for 'neither…nor…', '↓', and the other is called Sheffer's Stroke, after its discoverer, and written '|'. Their function-tables are:

↓	|
⊤⊤ ⇒ ⊥	⊤⊤ ⇒ ⊥
⊤⊥ ⇒ ⊥	⊤⊥ ⇒ ⊤
⊥⊤ ⇒ ⊥	⊥⊤ ⇒ ⊤
⊥⊥ ⇒ ⊤	⊥⊥ ⇒ ⊤

To see that '↓' is expressively complete, we use the already established expressive completeness of {~,&}. Since every table has a formula whose only connectives are '~' and '&', we can derive a formula for any table by finding the formula in '~' and '&' for it and then using substitution rules to eliminate all occurrences of '~' and '&', replacing them with formulae containing only '↓'. What should the substitution rules be in this case? One can play trial and error with truth-tables, but it is not hard to see that ⌜not-*p*⌝ can be paraphrased, if awkwardly, as ⌜neither *p* nor *p*⌝. We also note that ⌜neither not-*p* nor not-*q*⌝ is true exactly when *p* and *q* are both true, which makes it equivalent to that conjunction; and we already know how to eliminate 'not' in ⌜not-*p*⌝ and ⌜not-*q*⌝. So we get the following substitution rules:

- Replace every subformula of the form ⌜~*r*⌝ with ⌜(*r*↓*r*)⌝.
- Replace every subformula of the form ⌜(*r* & *s*)⌝ with ⌜(*r*↓*r*)↓(*s*↓*s*)⌝.

It is easy to check the correctness of these rules using truth-tables. Hence '$\downarrow$' is expressively complete by itself, and a similar argument shows Sheffer's Stroke to be expressively complete by itself (this is an exercise).

A final comment. We saw earlier that there are four unary truth-functions and sixteen binary ones. But for an arbitrary *n*, how many *n*-place truth-functions are there? To answer this we generalize the reasoning which gave us the answer 'sixteen' in the binary case. If a truth-function takes a sequence of *n* truth-values as input, there are 2^n different possible such input sequences for it: the first element of an input sequence may be $\top$ or $\bot$, giving two cases, and in each of these cases the second element may be $\top$ or $\bot$, giving a total of four cases, and so on, doubling the number of cases at each step, giving a total of 2^n cases at the *n*th element of the sequence. And if there are 2^n different possible input sequences to a truth-function, there are 2^{2^n} different possible combinations of truth-values which can be the function's output: the output for the first input may be either $\top$ or $\bot$, giving two cases, and in each of these, the output for the second input may be $\top$ or $\bot$, and so on, doubling the number of cases with each of the 2^n inputs, for a total of 2^{2^n} different possible outputs. Hence there are 2^{2^n} different *n*-place truth-functions.

❑ Exercises

I Find formulae in {~,&,∨} which express the truth-functions (1), (2) and (3) below. Then give formulae in {~,&} for (1) and *(2) (use the rule on page 78).

(1)	$\top\top\top \Rightarrow \top$	(2)	$\top\top\top \Rightarrow \bot$	(3)	$\top\top\top \Rightarrow \top$
	$\top\top\bot \Rightarrow \bot$		$\top\top\bot \Rightarrow \bot$		$\top\top\bot \Rightarrow \top$
	$\top\bot\top \Rightarrow \bot$		$\top\bot\top \Rightarrow \top$		$\top\bot\top \Rightarrow \top$
	$\top\bot\bot \Rightarrow \bot$		$\top\bot\bot \Rightarrow \top$		$\top\bot\bot \Rightarrow \top$
	$\bot\top\top \Rightarrow \top$		$\bot\top\top \Rightarrow \bot$		$\bot\top\top \Rightarrow \bot$
	$\bot\top\bot \Rightarrow \bot$		$\bot\top\bot \Rightarrow \bot$		$\bot\top\bot \Rightarrow \bot$
	$\bot\bot\top \Rightarrow \bot$		$\bot\bot\top \Rightarrow \top$		$\bot\bot\top \Rightarrow \bot$
	$\bot\bot\bot \Rightarrow \bot$		$\bot\bot\bot \Rightarrow \top$		$\bot\bot\bot \Rightarrow \top$

II Granted that {~,&,∨} is expressively complete, explain carefully why each of the following sets of connectives is expressively complete (compare the explanation on page 78 for {~,&}). State your substitution rules. In (3), '$\leftarrow$' means 'if', so $\ulcorner p \leftarrow q \urcorner$ is read $\ulcorner p$ if $q \urcorner$.

(1) {~,∨} (2) {~,→} *(3) {~, ←} (4) { | }

*III {~,↔} is expressively incomplete. Can you think of a general pattern of distribution of $\top$s and $\bot$s in the final column of a truth-table which would guarantee that there is no formula in {~,↔} which has that table? Try to explain your answer (this is much harder than the expressive incompleteness of {&,∨}).

8 Non-truth-functional connectives

The five connectives of LSL have been shown to be adequate for all of truth-functional logic. What is being deliberately excluded at this point, therefore, is any treatment of non-truth-functional connectives, that is, connectives which do not express truth-functions. There are extensions of classical logic which accommodate non-truth-functional connectives, but at this point all we need to know is how to determine whether a given connective is truth-functional or non-truth-functional.

A truth-functional connective expresses a truth-function, which in turn can be written as a function-table, so a proof that a certain n-place connective is not truth-functional would consist in showing that its meaning cannot be expressed in a function-table. A function-table associates each of the possible 2^n inputs with a single output, either $\top$ or $\bot$, so what we need to prove about a connective to show that it is non-truth-functional is that for at least one input, there is no single output that could be correctly associated with the connective. For if this is so, the truth-value of a sentence formed using the connective is not a function of the truth-values of the component sentences. Here are two examples.

Example 1: Philosophers distinguish two kinds of fact, or truth, those which are *contingent* and those which are noncontingent or *necessary*. Something is contingently the case if it might not have been the case, that is, if there are ways things could have gone in which it would not have been the case. Something is necessarily the case if there is no way things could have gone in which it would not have been the case. Note that this use of 'contingent' is much broader than its use to mean 'not a tautology and not a contradiction', which was the way we employed the term in §2. Understood in this new, broad sense, we can show that the connective 'it is a matter of contingent fact that…' is non-truth-functional. 'It is a matter of contingent fact that…' is a one-place connective which can be prefixed to any complete sentence. If it were truth-functional, then it would have a function-table $\top \Rightarrow ?$, $\bot \Rightarrow ?$ like negation, in which each query is replaced by either $\top$ or $\bot$. It is easy to see that when the input is $\bot$, the output is also $\bot$. In other words, if the ellipsis in 'it is a matter of contingent fact that…' is filled by a false sentence, the result is a false sentence; for if p is false, then ⌜it is a matter of contingent fact that p⌝ is false as well, since it is not a fact at all that p. But there is a problem when the input to the connective is $\top$. If the connective is truth-functional the output must always be $\top$, or always be $\bot$, yet we can show that neither of these alternatives correctly represents the meaning of 'it is a matter of contingent fact that'.

(a) Let 'A' mean 'Gorbachev was president of the Soviet Union'. Then 'A' is true, and 'it is a matter of contingent fact that A' is true as well, since it is a contingent fact that Gorbachev was president of the Soviet Union, there being many other ways things could have gone in which he would not have achieved that office—for example, he might have been killed in World War II.

(b) Let 'A' mean 'all triangles have three angles'. Then 'A' is true, but 'it is a matter of contingent fact that A' is false, for it is not a *contingent* fact that all triangles have three angles. Having three angles is part of the meaning of '*tri*angle', so it is necessary rather than contingent that all triangles have three angles: there are no alternative ways things could have gone in which there are triangles with fewer than or more than three angles.[4]

(a) and (b) together show that there is no correct way of completing the first entry $\top \Rightarrow ?$ in a function-table for 'it is contingent that…' (a) shows that it would be wrong to put $\bot$, and (b) that it would be wrong to put $\top$. This illustrates the general technique for establishing that a connective is non-truth-functional: we find two examples which show that for a particular input there is no single correct output.

Example 2: The connective 'after' is not truth-functional. 'After' is a two-place connective, and can occur either between two complete sentences, as in $\ulcorner p$ after $q\urcorner$, or at the beginning, as in the syntactic variant $\ulcorner$After q, $p\urcorner$. To show that 'after' is not truth-functional, we show that there is no correct output for the input $\top,\top$.

(a) Let 'A' mean 'Thatcher was elected prime minister' and 'B' mean 'Nixon was elected president'. Then 'A after B' is true, since Thatcher was first elected in 1979 and Nixon last elected in 1972.

(b) Let 'A' mean 'Nixon was elected president' and 'B' mean 'Thatcher was elected prime minister'. Then 'A after B' is false.

Thus in any purported function-table for 'after' it would be impossible to complete the first entry $\top,\top \Rightarrow ?$, because (a) shows $\bot$ is incorrect and (b) shows $\top$ is incorrect.

These two examples explain the terminology 'non-truth-functional', for their morals are that the truth-value of $\ulcorner$it is a matter of contingent fact that $p\urcorner$ is not a function merely of the *truth-value* of p, and the truth-value of $\ulcorner p$ after $q\urcorner$ is not a function merely of the *truth-values* of p and q. Rather, the truth-value of $\ulcorner$it is contingent that $p\urcorner$ depends, for true p, on the nature of the *reason why* p is true, and the truth-value of $\ulcorner p$ after $q\urcorner$, for true p and q, depends on the *temporal order* of the events reported in p and q. The connective 'it is a matter of contingent fact that' belongs to an extension of classical logic called *modal* logic (see Chapter 9), and the connective 'after' to an extension of classical logic called *tense* logic (see Burgess).

[4] Do not confuse the question (i) 'Could triangles have had more than or fewer than three angles?' with the question (ii) 'Could the word "triangle" have been defined differently?' Obviously, the word 'triangle' could have been defined to mean what we actually mean by 'square' ('tri' could have been used for 'four'), but that is irrelevant to question (i), where 'triangles' is used with its normal meaning. Given what (i) means, the correct answer to it is 'no'.

The alert reader will have noticed some parallels between our discussion of 'after' and our earlier discussion of 'if…then…' in §1 of this chapter. We have demonstrated that 'after' is not truth-functional by providing two examples in both of which 'after' is flanked by true sentences and in which the resulting 'after' sentences have *different* truth-values. But a comparable pair of examples for 'if…then…' as it is used in English can apparently be constructed. We used the sentence

(1) If Nixon was president then Nixon lived in the White House

to motivate the entry $\top\top \Rightarrow \top$ in the function-table of the truth-function expressed by 'if…then…', but we also noted that many would judge that the conditional

(2) If Moses wrote the Pentateuch then water is H_2O

is not true, on the grounds that the consequent is unrelated to the antecedent. Yet on the assumption that 'Moses wrote the Pentateuch' is true, (2) has a true antecedent and true consequent. Here one example with true antecedent and true consequent gives us a true conditional while another, also with true antecedent and true consequent, does not. Why do we not just conclude that the 'if…then…' of the indicative conditional is not truth-functional?

We remarked in our discussion in §1 that on one view, there are at least two senses of 'if…then…' and our truth-table for '→' captures only one of its senses, the material sense. The senses which are not captured are the non-truth-functional ones, where the truth-value of the conditional depends not just on the truth-values of its antecedent and consequent, but also on whether there is a certain kind of connection between antecedent and consequent. Different senses of the conditional would correspond to different kinds of connection, for instance, there would be a *causal* sense in which the truth of the consequent has to be *caused* by the truth of the antecedent. Unfortunately, it seems as though *all* the natural senses of 'if…then…' are non-truth-functional, so one could reasonably worry about the reliability of translations of English conditionals into a symbolism which cannot express these natural senses. This is why the case of the conditional is unlike the case of disjunction: even though we took inclusive disjunction as basic, the exclusive sense is also truth-functional and easily defined. But we cannot define any of the non-truth-functional senses of the conditional.

However, in the case of both disjunction and the conditional, there is an alternative to postulating two or more senses of the English connective. According to the alternative view, a conditional such as 'if Moses wrote the Pentateuch then water is H_2O' really is true, but sounds strange for the following reason. It is correct to assert conditionals only when we believe it is not the case that the antecedent is true and the consequent false, and the standard grounds for such a belief are either (i) we believe that the antecedent is false, or (ii) we believe that the consequent is true, or (iii) we believe that if the antecedent is true, that will in some way *make* the consequent true. But if our grounds for the belief that it is not the case that the antecedent is true and the consequent false are (i) or (ii), then asserting the conditional violates one of the maxims we have to observe if conversation is to be an efficient way of communicating information, the maxim *to be as informative as possible*. If our grounds are (i) or (ii), we should just deny the antecedent or assert the consequent, that is, we should make the more informative statement. However, this does not mean that if we assert the conditional instead, we say something false: the conditional is still true, but it is conversationally inappropriate. The problem with 'if Moses wrote the Pentateuch then water is H_2O', therefore, is that in the absence of any mechanism tying the authorship of the Pentateuch to the chemical composition of water, the most likely ground for asserting the conditional is that we believe that water is H_2O. But then that is what we should say, not something less informative. This means that we are left with (iii) as the only grounds on which it is normally appropriate to assert a conditional. So the suggestion is that when people deny that 'if Moses wrote the Pentateuch then water is H_2O' is true, they are failing to distinguish the question of truth from the question of appropriateness. All we can really object to about (2) is that in ordinary contexts it is likely to be conversationally inappropriate; but this is consistent with its being true.

The maxim to be as informative as possible needs to be qualified, for there are circumstances in which maximum informativeness would not be appropriate in the context. For example, in giving clues to children engaged in a treasure hunt, one may say 'if it's not in the garden then it's in the bedroom' so as to leave it open which should be searched, even though one knows the treasure is in the bedroom. This is a case where the conditional seems true despite the absence of a mechanism that brings about the treasure's being in the bedroom from its not being in the garden (the treasure is in the bedroom because that is where it was put). If we distinguish the appropriateness of a conditional from its truth, therefore, we can maintain that the English 'if…then…' has only one meaning, the truth-functional one. (This approach to the conditional and related matters was developed by Paul Grice; see Grice.)

❑ Exercises

Show that the following connectives are not truth-functional:

(1) 'It is necessary that…'
(2) '…before…'
*(3) 'It is surprising that…'
(4) '…because…'
*(5) '…, which means that…'
(6) 'At noon…'
(7) 'Someone knows that…' (Note: (7) is tricky.)

9 Summary

- Negation reverses truth-value; a conjunction is $\top$ when and only when both conjuncts are $\top$; a disjunction is $\bot$ when and only when both disjuncts are $\bot$; a conditional is $\bot$ when and only when its antecedent is $\top$ and its consequent is $\bot$; a biconditional is $\top$ when and only when both its sides have the same truth-value.
- Every formula of LSL is either a tautology (true on every interpretation), a contradiction, or contingent. Equivalent formulae have the same truth-value on each interpretation.
- An English argument has a valid argument-form if its translation in LSL is a valid argument-form.
- An argument-form in LSL is valid if and only if no interpretation makes its premises true and its conclusion false. LSL validity may be determined either by exhaustive search or by constructing an interpretation.
- Classical sentential logic is the logic of truth-functional connectives, all of which can be defined by '~', '&' and '$\vee$'. Non-truth-functional sentential connectives require extensions of classical logic, such as modal logic and tense logic, to handle them.