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
∴ P

and

B: P ∨ Q
∴ 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 '∨', 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 '⊤'. If a sentence is false, then it is said to *have the truth-value* FALSE, written '⊥'. We make the following assumption, often called the *Principle of Bivalence:*

> There are exactly two truth-values, ⊤ and ⊥. 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)

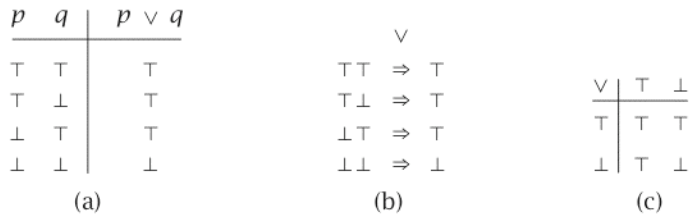
though this time the function is *two-place*, since the appropriate input is a *pair* of truth-values (addition is an example of a two-place function on numbers, since it takes two numbers as input). There is also a third way of exhibiting the meaning of '&', which is by a matrix, as in (d). The values 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 '$\vee$':

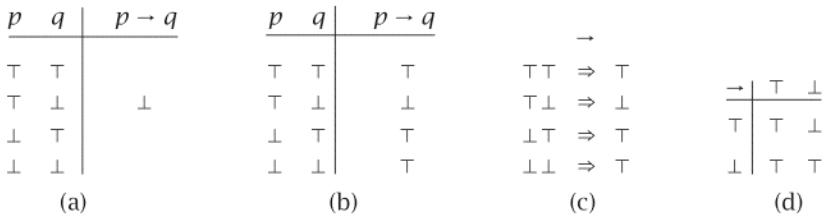


The difference between the actual table (a) for '$\vee$' and the one we would have written had we chosen to use the symbol for exclusive disjunction (one or the other but not both) would simply be that the top row would contain $\bot$ instead of $\top$.

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.



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 '→', 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 '→', we will arrive at the truth-function expressed by '↔'. 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	$(p \rightarrow q) \,\&\, (q \rightarrow p)$
⊤	⊤	
⊤	⊥	
⊥	⊤	
⊥	⊥	

(a)

p	q	$(p \rightarrow q) \,\&\, (q \rightarrow p)$		
⊤	⊤	⊤		⊤
⊤	⊥	⊥		⊤
⊥	⊤	⊤		⊥
⊥	⊥	⊤		⊤

(b)

p	q	$(p \rightarrow q) \,\&\, (q \rightarrow p)$		
⊤	⊤	⊤	**⊤**	⊤
⊤	⊥	⊥	**⊥**	⊤
⊥	⊤	⊤	**⊥**	⊥
⊥	⊥	⊤	**⊤**	⊤

(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 '→' 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 ⊥ on row 3 and ⊤ 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 '↔' 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)

$\leftrightarrow$

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

(b)

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

(c)

biconditional displayed above. Note that, by contrast with $\ulcorner p \to q \urcorner$ and $\ulcorner q \to p \urcorner$, $\ulcorner p \leftrightarrow q \urcorner$ 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:

		$\&$	$\vee$	$\to$	$\leftrightarrow$
		$\top\top \Rightarrow \top$	$\top\top \Rightarrow \top$	$\top\top \Rightarrow \top$	$\top\top \Rightarrow \top$
$\sim$		$\top\bot \Rightarrow \bot$	$\top\bot \Rightarrow \top$	$\top\bot \Rightarrow \bot$	$\top\bot \Rightarrow \bot$
$\top \Rightarrow \bot$		$\bot\top \Rightarrow \bot$	$\bot\top \Rightarrow \top$	$\bot\top \Rightarrow \top$	$\bot\top \Rightarrow \bot$
$\bot \Rightarrow \top$		$\bot\bot \Rightarrow \bot$	$\bot\bot \Rightarrow \bot$	$\bot\bot \Rightarrow \top$	$\bot\bot \Rightarrow \top$

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?

[1]

(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 $\top$ or $\bot$. 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
$\top$	$\top$
$\bot$	$\top$

(a)

A	A → ~A
$\top$	$\bot$
$\bot$	$\top$

(b)

A	~(A → A)	
$\top$	$\boxed{\bot}$	$\top$
$\bot$	$\boxed{\bot}$	$\top$

(c)

In table (a), on the first row, 'A → A' is $\top \rightarrow \top$, which by the function-table or matrix for '→' is $\top$, and on the second row, $\bot \rightarrow \bot$, which is also $\top$. In table (b) we have $\top \rightarrow \bot$ on the top row and $\bot \rightarrow \top$ on the bottom, which gives $\bot$ and $\top$ 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 $\bot$ where table (a) has $\top$, and $\top$ where (a) has $\bot$. 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))
⊤	⊤	⊤			⊤	⊤			⊤	⊥		⊥ ⊥
⊤	⊥	⊥			⊤	⊥			⊥	⊥		⊥ ⊤
⊥	⊤	⊥			⊤	⊥			⊥	⊤		⊥ ⊥
⊥	⊥	⊤			⊤	⊥			⊤	⊤		⊤ ⊤
		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 $\top$ 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 $\top$ to 'A' and 'C' and $\bot$ 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 ⌜$\sim q \rightarrow p$⌝, 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 ⌜$p \vee q$⌝.

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 ∨ B	(2) A → B
(3) ~(A & ~B)	(4) ~(~A & ~B)
(5) ~A ∨ B	(6) A ∨ ~A
(7) (A → (A & ~A)) → ~A	

III If p is a sentence of LSL which is not a tautology, does it follow that ⌜~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
⊤	⊤	⊤	⊤	⊤	⊥	⊤ ⊤ ⊤ ⊤ ⊤	⊥
⊤	⊤	⊤	⊤	⊥		⊤	⊥
⊤	⊤	⊤	⊥	⊤		⊤	⊥
⊤	⊤	⊤	⊥	⊥		⊤	⊥
⊤	⊤	⊥	⊤	⊤		⊤	⊥
⊤	⊤	⊥	⊤	⊥		⊤	⊥
⊤	⊤	⊥	⊥	⊤		⊤	⊥
⊤	⊤	⊥	⊥	⊥		⊤	⊥
⊤	⊥	⊤	⊤	⊤		⊥	⊥
⊤	⊥	⊤	⊤	⊥		⊥	⊥
⊤	⊥	⊤	⊥	⊤		⊥	⊥
⊤	⊥	⊤	⊥	⊥		⊥	⊥
⊤	⊥	⊥	⊤	⊤		⊥	⊥
⊤	⊥	⊥	⊤	⊥		⊥	⊥
⊤	⊥	⊥	⊥	⊤		⊥	⊥
⊤	⊥	⊥	⊥	⊥		⊥	⊥

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: (~J → C) & (E ↔ C)
> ~E
> ∴ 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 ⊤ → ⊥, which is ⊥, 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 → (B & E)
> D → (A ∨ F)
> ~E
> ∴ D → 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 ⊤ to 'D', ⊥ to 'B', ⊥ to 'E' and ⊥ 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
⊤	⊥	⊥	⊥	⊤

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 ⊥ to 'C' and ⊤ 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) (~J → C) & (E ↔ C), ~E ⊨ J
(2) A → (B & E), D → (A ∨ F), ~E ⊭ D → B
(3) ~A ∨ (B → C), E → (B & A), C → E ⊭ C ↔ 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 '$\models$' 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, $p_1,\ldots,p_n \models q$ 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, $\models q$ 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 '$\models q$' as 'q is a tautology'. '$\not\models q$', 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) $A \to B, B \to (C \vee D), \sim D \models A \to C$
(2) $(A \,\&\, B) \to C, B \to D, C \to \sim D \models \sim A$
(3) $A \to (C \vee E), B \to D \models (A \vee B) \to (C \to (D \vee E))$
*(4) $A \to (B \,\&\, C), D \to (B \vee A), C \to D \models A \leftrightarrow C$
(5) $A \vee (B \,\&\, C), C \vee (D \,\&\, E), (A \vee C) \to (\sim B \vee \sim D) \models B \,\&\, D$
(6) $A \to (B \to (C \to D)), A \,\&\, C, C \to B \models \sim B \leftrightarrow (D \,\&\, \sim D)$
(7) $(A \leftrightarrow B) \,\&\, (B \leftrightarrow C) \models (A \vee \sim A) \,\&\, ((B \vee \sim B) \,\&\, (C \vee \sim C))$
(8) $(A \leftrightarrow B) \vee (B \leftrightarrow C) \models A \leftrightarrow (B \vee C)$
(9) $(\sim A \,\&\, \sim B) \vee C, (A \to D) \,\&\, (B \to F), F \to (G \vee H) \models \sim G \to (H \vee C)$

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.