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:

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

$$\text{(a)} \qquad\qquad\qquad \text{(b)} \qquad\qquad\qquad \text{(c)}$$

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

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

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 ⌜~*p* & ~*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 ⊤⊤ is either ⊤ or ⊥, giving us two cases, and in each of these cases, the output for the second pair of inputs ⊤⊥ is either ⊤ or ⊥, 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

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

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 '$(A \rightarrow (B \lor C)) \rightarrow (A \rightarrow (B \& C))$' 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 I of truth-values to sentence-letters $\pi_1,...,\pi_n$ if and only if that formula is true on I and on no other assignment to $\pi_1,...,\pi_n$. Then given an assignment I to $\pi_1,...,\pi_n$, one can use $\pi_1,...,\pi_n$ to construct a formula in '&' and '~' which defines I as follows: take each sentence-letter which is assigned $\top$ and the negation of each which is assigned $\bot$ and form the conjunction of these letters and negated letters. So, for example, the interpretation consisting in the assignment of $\bot$ to 'C', $\top$ to 'D', $\bot$ to 'B', $\bot$ to 'A' and $\top$ 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 $\top$ or all to $\bot$. Otherwise, to construct a formula for the table, we choose sentence-letters $\pi_1,...,\pi_n$ and use them to construct the formulae which define the interpretations where there is a $\top$ 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 $\top$ 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 $\top$. 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 $\top$s in the final column, let the formula be 'A & ~A'; if there are no $\bot$s, let it be 'A $\lor$ ~A'.
- Otherwise, using the appropriate number of sentence-letters 'A', 'B' and so on, for each interpretation which gives a $\top$ 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:

A B	
⊤ ⊤	⊥
⊤ ⊥	⊤
⊥ ⊤	⊥
⊥ ⊥	⊤

⊤ ⊤ ⇒ ⊥
⊤ ⊥ ⇒ ⊤
⊥ ⊤ ⇒ ⊥
⊥ ⊥ ⇒ ⊤

    Function   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:

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

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

    Function   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

$$\sim[\sim(A \mathbin{\&} \sim B) \mathbin{\&} \sim(\sim A \mathbin{\&} \sim B)]$$

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 '↓' 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 ⊤ or ⊥, giving two cases, and in each of these cases the second element may be ⊤ or ⊥, 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 nth 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 ⊤ or ⊥, giving two cases, and in each of these, the output for the second input may be ⊤ or ⊥, 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)		(2)		(3)	
⊤⊤⊤ ⇒ ⊤		⊤⊤⊤ ⇒ ⊥		⊤⊤⊤ ⇒ ⊤	
⊤⊤⊥ ⇒ ⊥		⊤⊤⊥ ⇒ ⊥		⊤⊤⊥ ⇒ ⊤	
⊤⊥⊤ ⇒ ⊥		⊤⊥⊤ ⇒ ⊤		⊤⊥⊤ ⇒ ⊤	
⊤⊥⊥ ⇒ ⊥		⊤⊥⊥ ⇒ ⊤		⊤⊥⊥ ⇒ ⊤	
⊥⊤⊤ ⇒ ⊤		⊥⊤⊤ ⇒ ⊥		⊥⊤⊤ ⇒ ⊥	
⊥⊤⊥ ⇒ ⊥		⊥⊤⊥ ⇒ ⊥		⊥⊤⊥ ⇒ ⊥	
⊥⊥⊤ ⇒ ⊥		⊥⊥⊤ ⇒ ⊤		⊥⊥⊤ ⇒ ⊥	
⊥⊥⊥ ⇒ ⊥		⊥⊥⊥ ⇒ ⊤		⊥⊥⊥ ⇒ ⊤	

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), '←' means 'if', so ⌜$p \leftarrow q$⌝ is read ⌜p if q⌝.

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

*III {~,↔} is expressively incomplete. Can you think of a general pattern of distribution of ⊤s and ⊥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 {&,∨}).