Symbolic Logic 1

Quotations are from Howson, unless otherwise indicated.

Preliminaries

Declarative sentences = sentences that are used to say things capable of being *true* or *false*. [Questions, commands, nonsense sentences, … are not declarative sentences.]

Inference involves *premise(s)* and *conclusion*. For our purposes the premises and conclusions are declarative sentences. We are also restricting ourselves to inferences that purport to be *deductively valid*.

Deductive validity

For an inference to be deductively valid it is *impossible for the conclusion to be false if the premises are true.*

NB: 'impossible' - not just actually true when premises are true.
NB: what is not being said here. Nothing said about the actual truth or falsity of premises and conclusion in any particular argument or inference; validity is a conditional matter, what happens *if* the premises were true.

Since we are not interested in how things actually are, but only in what would happen if, we are not tied down to particular cases; what we are looking at is how declarative sentences behave in certain structures or forms (whence 'formal logic'). We construct a mini-language to capture the very few things we are interested in.[1]

The syntax of a truth-functional logical language, principle of composition

The simplest mini-language is a matter of putting declarative sentences together into compound sentences, without looking at what is going on inside the basic (or *atomic*) declarative sentences. The items that are used to build up compound sentences are called *connectives*. There are various alternative ways of arriving at a mini-language that can do what we want with respect to these compounds; the most natural way is to use four connectives (corresponding roughly to what in English is said by 'not', 'and', 'or', and 'if … then ...').

So our mini-language has:

> an unlimited set of declarative sentences, which we can abbreviate by using *P*, *Q*, …
>
> a set of connectives:

1 A sad fact about logic compared to mathematics is that there are any number of alternative ways of writing the simple languages. Hodges uses *P, Q*, etc. as abbreviations for particular declarative sentences, φ, ψ, etc. for sentence variables; Howson uses A, B, etc. for particular sentences, X, Y, for arbitrary sentences. There are similar variations for the connectives across different writers. I will try to follow Hodges here, except when quoting Howson.

Rough translation	Connective	Name of connective
not	¬	negation
and	∧	conjunction
or	∨	disjunction
if … then …	→	conditional

The first connective in the table, corresponding to 'not', operates on a single sentence to create a compound; it is called a *unary* or *1-plac*e connective. We write the connective in front of the sentence upon which it operates. The other connectives take two sentences to form a compound and are therefore called *binary* or *2-place* connectives. We write them between the two sentences they operate on.

There is no limit to the compounding of sentences out of other sentences. Given the way we are writing them, it is necessary to use brackets to disambiguate complex compounds. So, for instance, $P \wedge (Q \vee R)$ is a different compound from $(P \wedge Q) \vee R$.[2]

"Denoting arbitrary sentences … by the letters X, Y, Z, etc., we can give a compact statement of the principle of composition in which bracketing is automatically taken care of. The statement has two clauses, one unconditional, the other conditional: *A, B, C, etc. are sentences, and if X and Y are sentences, then so are* $\neg X$, $\neg Y$, $(X \wedge Y)$, $(X \vee Y)$, $(X \rightarrow Y)$ (in informal discussion the outer brackets will generally be dropped)." (p. 6)

We can use this syntactic rule to determine whether or not any arbitrary string made up of our vocabulary elements (letters, connectives, brackets) is or is not a well-formed formula (often abbreviated to *wff*) of the mini-language.

The semantics

In our mini-language we associate with each atomic sentence one of two *truth-values* 'true' and 'false', which we will write as T and F.

We are not going to explain what truth or falsity are; they are taken to be well-enough understood.

For our mini-language, any particular sentence has one and only one of the two values.

The crucial assumption in our mini-language is that the truth-value of compound sentences depends on the truth-values of their atomic components and the rules associated with each connective, and on nothing else. This is what it is for our logic and the connectives in our mini-language to be *truth-functional*.

Given that the truth-value of a compound sentence depends on its atomic sentences and the

2 Given footnote 1 you will not be surprised to find different conventions for bracketing as well; more significantly there is a notation that permits one to do without bracketing altogether (so-called 'Polish' notation), but it is easier for machines to read than for humans.

connectives, we can evaluate the truth-value of any compound by considering all the possible combinations of truth and falsity of the atomic sentences. This results in a *truth table* for the compound.

If there is only one atomic sentence in a compound there are only two possibilities to consider, so the truth table has two rows (not counting the heading). If there are two atomic sentences, each one may be true or false; for the truth of the first there are two possibilities for the second, and for the falsity of the first there are two for the second, so there are 4 rows to be considered. In general, for *n* atomic sentences there are 2^n rows to be calculated -- truth tables can get very big very quickly. (There are various ways of drawing up a truth table. I write the atomic propositions in the first set of columns and then the compounds to be evaluated in the next; I have the last of the atomic columns alternating T and F, the penultimate column alternating 2 Ts and 2 Fs, the antepenultimate column alternating 4 Ts and 4 Fs, and so on.)

The usual connectives (negation, conjunction, disjunction, conditional) and truth-functional equivalence

To construct a truth table for any arbitrary compound we need to know the rules for each connective, which can themselves be presented as a truth table.

Thus for the 1-place connective we shall call 'negation' we have the following assignments:

φ	¬ φ
T	F
F	T

That tells you that when negation operates on a true sentence the result is false, and when it operates on a false sentence the result is true.

(It would be nice if negation in real life, or natural languages, were so straightforward!)

Conjunction is fairly straightforward. The conjunction of two sentences is true if they are both true, false otherwise. The table looks like this:

φ	ψ	φ ∧ ψ
T	T	T
T	F	F
F	T	F
F	F	F

With disjunction and the conditional there are wider gaps between our mini-language and ordinary English. In the case of disjunction people often think that a disjunction offers an *exclusive* choice (A or B, but not both), while our mini-language finds it easier to treat disjunctions as *inclusive*, so it treats a disjunction of two sentences as true when at least one of them is true:

φ	ψ	φ ∨ ψ
T	T	T
T	F	T
F	T	T
F	F	F

In the case of what are called conditionals we have to accept the fact that our mini-language does not behave like natural languages. There are reasons to think we can get away with the distortion involved. In any case, the rule for conditionals in our mini-language is:

φ	ψ	φ → ψ
T	T	T
T	F	F
F	T	T
F	F	T

Note the terminology associated with conditionals: the sentence following 'if' is the *antecedent*, that following 'then' is the *consequent*.

We have allowed an unlimited number of compound sentences to be built up from two atomic sentences and our connectives. But there are only four sets of possible combinations of two truth-values (the four rows of our tables above) and only 16 resultant columns for the compound, so there will be very many compound expressions that have the same resultant value in each row of their truth table. We can say that they are *truth-functionally equivalent*.

To see this in action, and to see in addition why we can get away with ignoring exclusive disjunction as a primitive connective, consider these two tables:

φ	ψ	φ xor ψ
T	T	F
T	F	T
F	T	T
F	F	F

That is how an exclusive disjunction would behave: given two atomic sentences, the exclusive disjunction is true if one and only one of them (it doesn't matter which) is true. Now in the following table we have a slightly more complex compound sentence, using our existing connectives, which ends up having exactly the same final column, and is thus truth-functionally equivalent to exclusive disjunction. We can use this compound whenever we have to express the exclusive notion. It says, in effect, for some particular pair of sentences, (*either P or Q) and (not (both P and Q))*. (I have added a row at the bottom to indicate the order in which compounds are evaluated using the rules for the connectives. The final column, which tells us the values for the compound, is in bold as well.)

φ	ψ	(φ ∨ ψ)	∧	(¬	(φ ∧ ψ))
T	T	T	**F**	F	T
T	F	T	**T**	T	F
F	T	T	**T**	T	F
F	F	F	**F**	T	F
		1	4	3	2

Truth-functional equivalence is a fact about two statements in our mini-language. It is not a truth-function of the two statements. But it is an important notion and has its own symbol: ⇔.

Given that we can find many truth-functionally equivalent sentences, one can ask whether we could dispense not merely with exclusive disjunction but also with any of the other connectives we have chosen for our mini-language. The answer is in fact Yes. We can get by with negation and just one of the others. But it is more intuitive to use the four we have chosen since they have some resemblance to the natural English we are using to talk about them. In fact we can do everything we currently do using only one more unusual connective (and there are two alternative ways of doing that).

Going in the other direction, it is common to have a symbol for a connective that is easily definable in terms of those we already have: the biconditional: (if *P* then *Q*) and (if *Q* then *P*). The symbol we use is unsurprisingly ↔. As we will see next time, the second half of the expanded form means roughly what 'Q only if *P*' means, so the biconditional can be read as 'if and only if' (which is often abbreviated to *iff*).