4.B: the ***Philosophyldeas*** overview of

Propositional Logic

If you reason either 'she's in the kitchen' or 'she's in the study', and you observe 'she's not in the kitchen', you can infer 'she's in the study'. In general, if p and q are two sentences affirming facts, you can reason 'either p or q, and not p, so q'. Syllogistic logic does not cover such reasoning, which is Propositional Logic (PL). Early stoic philosophers studied such logic, but a formal system for PL was developed in the 19th century. The modern logic starts with a symbolic language, and we introduce letters **p**, **q** and **r** for propositions, and a symbol for 'not' (**¬**), and a symbol for 'and' (**∧**). If we then write ¬(¬p ∧ ¬q), this means you can't deny both propositions together, so you are allowed to affirm at least one of them. This gives us a definition of 'or' in terms of our two symbols, so we introduce a third symbol for 'or' – '**v**'. If we then write ¬(p ∧ ¬q), this means that whenever you can affirm p, q is undeniable, so affirming one makes the other affirmable. This gives us a definition of 'if…then' or 'material implication', so we introduce a fourth symbol '→'. If we permit the brackets we have been using, we now have a working language, which can describe the steps of inference from one sentence to another. The four symbols **¬**, **∧**, **v**, and → are the main 'connectives' of the language. You should now be able to translate this into English: ((p→q) ∧ (q→¬r)) → (p→¬r).

PL can be specified by picking out a few basic truths as axioms (which can be whittled down to three), and then showing their implications. Nowadays it is most commonly presented as a set of **ten rules**, which are the foundations of all modern logic. The most basic rules tell you how to introduce a connective into an argument, and how to eliminate it. You then have a clear but powerful system for symbolic reasoning. Here (briefly) are the ten rules:

1. *Assumptions* (A): any proposition may be introduced as an assumption at any stage of a proof
2. *Modus Ponens* (MP): given p and p→q, we may derive q
3. *Modus Tollens* (MT) or *Contraposition*: given p→q and ¬q, we may derive ¬p
4. *Double Negation* (DN): given p, we may derive ¬¬p, and vice versa
5. *Conditional Proof* (CP): if we assume p and then prove q, we may introduce p→q
6. *And-introduction* (∧I): given p and given q, we may derive p∧q
7. *And-elimination* (∧E): given p∧q, we may derive p, and we may derive q
8. *Or-introduction* (vI): given p, we may derive pvq, and given q we may also derive pvq
9. *Or-elimination* (vE): given pvq, and that both p and q will separately prove r, we may derive r
10. *Reductio ad Absurdum* (RAA): if from an assumption of p we can prove both q and ¬q, we can derive ¬p

For all of these rules, the results will depend on any assumptions you have made during the proof, and the eventual aim is to reach a conclusion which does *not* depend on any assumptions (which must therefore somehow be 'discharged'). Some of the rules have been questioned; for example, if you think 'I am not unhappy' is not quite the same as 'I am happy', you might wonder about double negation. The standard procedure for proving any p is to assume ¬p, and then reduce the assumption to absurdity; this proves ¬¬p (by RAA), which proves p (by DN).

So far we have said we can 'assume' or 'write' or 'derive' p or q. This is neutral language for describing the system of procedures, but we have not said what the system means. That is, our rules give us the 'syntax', but we now need a '**semantics**'. You can interpret a formal system in any way you like. We could try interpreting it as a board game (if you like!), or we could interpret assertion and denial as 'on' and 'off' for switches in an electronic machine, but philosophers love truth (and hate falsehood), so the semantics usually uses 'T' and 'F'.

The semantics of '¬' says when p is T, ¬p is F, and vice versa. For any pair of propositions, they can be both true, or first true and second false, or first false and second true, or both false (i.e. TT, TF, FT, FF). If there are more than two propositions, many more combinations will be possible. For two propositions, we can list the four possible inputs to three simple propositions, and show the output in each case:

p	q	p∧q ('and')	pvq ('or')	p→q ('if…then')
T	T	T	T	T
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

This gives the basis for the semantics of PL. It can be proved that the syntax and semantics match perfectly. PL is 'sound' because all proofs from the syntax are true, and 'complete' because all truths expressible in the language are provable. The column for implication raises interesting questions, studied under the heading of 'conditionals'.

In PL it is normal to start a proof by making assumptions, but sentences provable with no assumptions are called 'theorems', which are the basic sentences of the logic. Some of these are very useful, for rewriting a complex sentence in a more convenient form. The definitions of 'v' and '→' are examples of theorems, and you can, for example, replace 'pvq' with '¬(¬p∧¬q)', because they say the same thing (so their equivalence is a 'tautology').

Two background assumptions are of interest in this very successful system of logic. One is that if a premise implies a conclusion, adding further premises will make no difference; the argument can survive 'Thinning'. This makes the logic 'monotonic', meaning that what is proved stays proved (which may not be the case in science, or in law). The other principle is that material implications ('→') can work in chains, so that you can leave out the middle steps and jump from the start to the finish, which is called 'Cutting'. All of these features gradually combine (with the additions of Predicate Calculus' (PC)), to form what is now referred to as 'Classical Logic'.

Propositional Logic fulfils all the conditions we would ask of a really reliable system, but it says no more about the propositions themselves than that 'p' and 'p' are the same proposition, and 'p' and 'q' are different propositions.