"What we cannot speak about we must pass over in silence." (Wittgenstein)

A puzzle
Anybody with just one pound is a poor man.
Anybody with just one pound more than a poor man is poor themselves.

Is it impossible not to be poor? If not why not?

This week's lecture continues on from using Natural Deduction to prove formulae in propositional logic and covers a number of topics which should be dealt with before we look at predicate calculus. General strategies for proving in Natural Deduction. By now you should be reasonably good at proving simple theorems in Natural Deduction and probably have a set of "rules of thumb" for when to try to solve a problem in a particular way. Tomassi (Logic pp 96-117, 1999) offers a "Golden rule" which is works most of the time:

1. Is the main connective in the conclusion a conditional? If so, assume the antecedent of the conditional and then try and prove the consequent using the premises given.

2. Is the main connective of any of the premises a disjunction? If so try to use the vE rule i.e. assume the first disjunct and try to prove the conclusion and then assume the second disjunct and try to prove the conclusion. Finally draw the conclusion from the original disjunctive premise using vE.

3. Try RAA (reducto ad absurdium) i.e. assume the opposite of what you're trying to prove and then prove that this leads to a contradiction. The double negation rule will then allow you to finish the proof.

Let's try these out.
1. \( P \rightarrow Q, \neg Q : \neg P \)
   1. \( P \) H {1}
   2. \( P \rightarrow Q \) Premise {1,2}
   3. \( Q \rightarrow e \) {1,2}
   4. \( \neg Q \) Premise {1,4}
   5. \( \neg P \) RAA {2,4}

2. \( P \land Q : P \rightarrow Q \)
   1. \( P \) H {1}
   2. \( P \land Q \) Premise {1,2}
   3. \( Q \land \text{elim} \) {1,2}
   4. \( P \rightarrow Q \rightarrow i \) {2}
   5. 

3. \( (P \land Z) \lor Q, Q \rightarrow Z : Z \)
   1. \( (P \land Z) \lor Q \) Premise {1}
   2. \( P \land Z \) H {2}
   3. \( Z \land \text{elim} \) {2}
   4. \( Q \) H {4}
   5. \( Q \rightarrow Z \) Premise {4,5}
   6. \( Z \rightarrow e \) {4,5}
   7. \( Z \lor \text{elim} \) 1,3,6 {1,5}
4. : ((P → Q) ∧ ¬Q) → ¬P
   1. | (P → Q) ∧ ¬Q H {1}
   2. | P → Q ∧_elim {1}
   3. || P H (1,3)
   4. || P → Q i {1}
   5. || Q → elim {1,3}
   6. || ¬Q ∧elim {1,3}
   7. || ¬P RAA {1}
   8. (P → Q) ∧ ¬Q → ¬P →I {}

**Primitive versus derived rules**

Consider an arbitrary set of inference rules for PL. It is reasonable to ask if the set is sufficient to prove every valid theorem in a language. Clearly if we restrict our set of natural deduction rules (for example by removing every rule involving conjunctions) then there will be valid theorems in PL which we cannot prove.

Because of this, we can distinguish between two classes of rule. The first set are primitive rules, that is to say the introduction of each rule allows something new to be proved. Derived rules by definition do not add any further theorems which can be proved.

A set of inference rules is "syntactically complete" if no further primitive rules exist which can be added without introducing inconsistancy.

A good example of this distinction is RAA versus modus tollens and double negative elimination.

Any proof involving RAA can also be proved by modus tollens and double negative elimination and therefore if we have either rule(s) in our set of primitive inference rules then we gain no addition to what we can prove by adding the other.

This might make derived rules seem rather useless. In actual fact, it is often the case that the use of some derived rule (e.g. modus tollens) might make for an elegant (or shorter) proof. We can introduce rules to our existing proof theory to handle this.

**Theorem Introduction rule (TI)**

The theorem introduction rule allows us to add any known theorem at any stage of a proof with an empty set of dependency numbers. For example:

\[ \neg P \rightarrow Z, P \rightarrow Q, \neg Q: Z \]
1. P → Q Premise {1}
2. ¬Q Premise {2}
3. (P → Q) ∧ ¬Q ∧_intro {1,2}
4. (P → Q) ∧ ¬Q → ¬P TI {} (modus tollens above)
5. ¬P ¬_elim {1,2}
6. ¬P → Z Premise {6}
7. Z → elim {1,2,6}

Suppose we know the theorem : P → P then we can introduce it at any stage in a proof
\[ n P → P TI {} \]

We can also introduce
\[ n ((P ∧ Q) v Z) → ((P ∧ Q) v Z) TI {} \]

Since it is the same theorem except the various parts of the theorem are uniformly replaced. This process applies to all theorem introduction and is called uniform substitution. This is a very simple
idea but is crucial to formal logic and we'll need to do more of this later.
(A warning about TI - before you use it in any exam paper, check to make sure it's allowed!)

Soundness
It's worth revisiting our concept of validity. At the start of the module we defined validity as follows:

**Semantic validity**
An inference is valid iff it cannot be the case that all the premises are true and the conclusion false at the same time.
The obvious way to check this is to construct a truth table and show that under all possible combinations of truth that the inference is valid.
We also developed a different approach to proof involving the use of inference rules. Here, our notion of validity was slightly different:

**Syntactic validity**
An inference is syntactically valid iff the conclusion can be derived from the premises by means of stipulated rules of inference.
This notion of syntactic validity appears weaker than semantic validity. In fact we use a different notation for signalling semantic validity:

\[ P \rightarrow Q, P \vdash Q \text{ Syntactically valid theorem} \]
\[ P \rightarrow Q, P \not\vDash Q \text{ Semantically valid theorem} \]

Given any logic it's necessary to ask whether every syntactically proveable sequent is also semantically valid. It turns out that PL has this property i.e.

**Soundness**
\[ \vdash S \Rightarrow \vDash S \]
i.e. each and every rule of inference in PL is truth preserving. The proof of this is beyond an introductory module on logic but you should be expected to understand the issue and the meaning of soundness (and also not to confuse "soundness" with the idea of a "sound argument").

Completeness
Soundness guarantees that any syntactically valid proof will also mean that the theorem is semantically valid. Completeness is the complementary concept: Given a semantically valid theorem, can we construct a syntactic proof using inference rules? Happily in PL the answer is yes. i.e.

**Completeness**
\[ \vDash S \Rightarrow \vdash S \]
Again the proof of this is beyond this module.