Modelling Propositional Logic

Propositions

We model propositions as having five basic types:

- `AtomicProposition`: Meant to model the atomic propositions of the language, $p, q, r \ldots$.
- `ConjunctiveProposition`: Represents the conjuction $p_1 \wedge \ldots \wedge p_n$ of n propositions.
- `DisjunctiveProposition`: Similarly $p_1 \vee \ldots \vee p_n$.
- `ComplementProposition`: Represents the proposition $\neg p$.
- `ImplicativeProposition`: Represents the proposition $p \Rightarrow q$.

Deduction Rules

A deduction rule is modelled as one of the introduction or elimination rules for Natural Deduction.

$$\frac{A \qquad B}{A \wedge B} \wedge\text{-int}$$

Deduction rules have a weird property - they are universally quantified over propositions, but nobody ever writes them this way. When we write the above rule, it is simply understood that the propositions A and B may be replaced by any other definite proposition we like. But then we risk running into ambiguity if we *do* have definite propositions called A and B.

This can probably just be modelled by the fact that the particular deduction is wrapped up inside of a rule.

Proof

A proof is a sequence of propositions, each of which is a consequence of the earlier hypotheses, according to one of the rules of deduction. In order to simplify the check for validity, we implement a `ProofStep` object, which is supposed to model a single application of a deduction rule. A `Proof` is then a sequence of `ProofStep`s, and the proof is valid if and only if all of its steps are valid.

There is an interesting