CO140 - Logic

Introduction

A logic system consists of 3 things:

1. Syntax - formal language used to express concepts
2. Semantics - meaning for the syntax
3. Proof theory - syntactic way of identifying valid statements of language

Considering the basic example in a program, we can then see the features;

```
if count > 0 and not found then
    decrement count;
    look for next entry;
end if
```

1. basic (**atomic**) statements (**propositions**) are either $\top$ or $\bot$ depending on circumstance;
   i. `count > 0`
   ii. `found`
2. **boolean operations**, such as `and`, `or`, `not`, etc. are used to build complex statements from **atomic propositions**
3. the final statement `count > 0 and not found` evalulates to either $\top$ or $\bot$

Syntax

The formal language of logic consists of three ingredients;

1. Propositional atoms (propositional variables), evaluate to a truth value of either $\top$ or $\bot$. These are represented with letters; $p, p', p_0, p_1, p_2, p_n, q, r, s, ...$
2. Boolean connectives;
   - `and` is written as $p \wedge q$ — p and q both hold
   - `or` is written as $p \vee q$ — p or q holds (or both)
   - `not` is written as $\neg p$ — p does not hold
   - `if-then / implies` is written as $p \to q$ — if p holds, then so does q
   - `if-and-only-if` is written as $p \leftrightarrow q$ — p holds if and only if q holds
   - `truth`, and `falsity` are written as $\top$, and $\bot$ respectively. — logical constants
3. Punctuation. Similar to arithmetic, the lack of brackets can make an expression ambiguous. For example, $p_0 \vee p_1 \wedge p_2$ can be read as either $(p_0 \vee p_1) \wedge p_2$ or $p_0 \vee (p_1 \wedge p_2)$, which are different. The latter is the correct interpretation due to binding conventions.

   We can order the boolean connectives by decreasing binding strength;

   (strongest) $\neg$, $\wedge$, $\vee$, $\to$, $\leftrightarrow$ (weakest)

   While repeated disjunctions ($\vee$), and conjunctions ($\wedge$) are fine, as $p \wedge q \wedge r$ is equivalent to $p \wedge (q \wedge r)$, and the same for $\vee$, due to associativity, the same isn't true for $\to$. Due to the ambiguity, brackets should always be used.

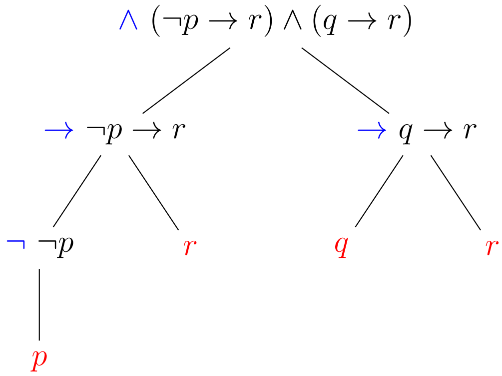
   There are also exceptions to the rule, for example with $p \to r \wedge q \to r$ - this should be $p \to (r \wedge q) \to r$ according to our binding conventions, but brackets should be used to ensure the correct interpretation.

Formulas

Something is a **well-formed formula** only if it is built from the following rules;

1. a propositional atom $(p, p', p_0, p_1, p_2, p_n, q, r, s, ...)$ is a propositional formula
2. $\top$, and $\perp$ are both formulas
3. if A is a formula, then $(\neg A)$ is also a formula
4. if A, and B are both formulas, then $(A \wedge B), (A \vee B), (A \rightarrow B), (A \leftrightarrow B)$ are also formulas

We can also create a tree to parse a logical formula, for example; $(\neg p \rightarrow r) \wedge (q \rightarrow r)$



Note that this tree shows the principal connective in blue, and the propositional atoms in red. Note that $\wedge$ is the principal connective in the top layer, and it therefore has the general form $A \wedge B$, and so on going down.

Technical terms

- A formula is a **negated formula** when it is in the form $\neg A$, negated atoms are sometimes called **negated-atomic**.
- $A \wedge B$, and $A \vee B$ are **conjunctions**, and **disjunctions**. A, and B, are **conjuncts**, and **disjuncts**, respectively.
- $A \rightarrow B$ is an implication. A is the **antecedent**, and B is the **consequent**