

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 \perp 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` evaluates to either \top or \perp

Syntax

The formal language of logic consists of three ingredients;

1. Propositional atoms (propositional variables), evaluate to a truth value of either \top or \perp . These are represented with letters; $p, p', p_0, p_1, p_2, p_n, q, r, s, \dots$
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 $p \rightarrow q$ 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 \perp 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) $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ (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 \rightarrow . Due to the ambiguity, brackets should always be used.

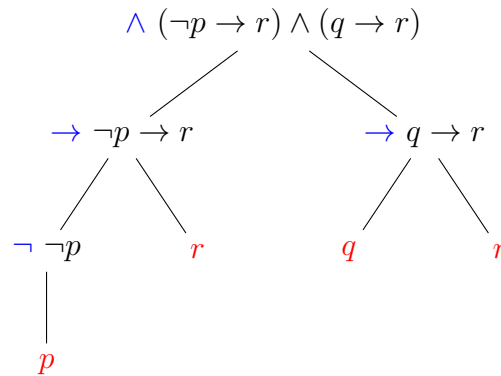
There are also exceptions to the rule, for example with $p \rightarrow r \wedge q \rightarrow r$ - this should be $p \rightarrow (r \wedge q) \rightarrow r$ according to our binding conventions, but brackets should be used to ensure the correct interpretation.

Formulas

Something is a **well-formed formula** only if it is built from the following rules (the brackets are required);

1. a propositional atom ($p, p', p_0, p_1, p_2, p_n, q, r, s, \dots$) 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**

Semantics

The connectives covered above have a rough English translation. However a natural language has ambiguity, and as engineers, we need precise meanings for formulas. This is the truth table for every connective that will be used in this course (?):

p	q	\top	\perp	$p \wedge q$	$p \vee q$	$\neg p$	$p \rightarrow q$	$p \leftrightarrow q$	$p \uparrow q$
0	0	1	0	0	0	1	1	1	1
0	1	1	0	0	1	1	1	0	1
1	0	1	0	0	1	0	0	0	1
1	1	1	0	1	1	0	1	1	0

Note how we can also define new connectives (see how $A \uparrow B$ was defined in the last column); this is a NAND connective - equivalent to $\neg(A \wedge B)$.

Translation

English to Logic

- **but** means **and**

"I will go out, but it is raining"

$(i \text{ will go out}) \wedge (it \text{ is raining})$

- **unless** generally means **or**

"I will go out unless it rains"

$(i \text{ will go out}) \vee (it \text{ will rain})$ (note the will)

$\neg(it \text{ will rain}) \rightarrow i \text{ will go out}$

There is also the strong form of **unless**, but in we generally use the weak form in computing

$(i \text{ will go out}) \leftrightarrow \neg(it \text{ will rain})$

- **or** generally refers to exclusive or (strong reading) in English, but it can also refer to inclusive or (weak reading). However, we always take the weak reading in computing.

Modality

I don't know what this means, so I'm just ignoring it for now

Logic to English

While the others are slightly more straightforward, \rightarrow is a pain to translate.

For example, $(i \text{ am the pope}) \rightarrow (i \text{ am an atheist})$ is true, as falsity implies anything, however if we were to translate it into English, "If I am the Pope, then I am an atheist" is (most likely) untrue.