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 $p \land q$ — p and q both hold
   - `or` is written as $p \lor q$ — 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, $p_0 \lor p_1 \land p_2$ can be read as either $(p_0 \lor p_1) \land p_2$ or $p_0 \lor (p_1 \land p_2)$, which are different. The latter is the correct interpretation due to binding conventions.

   We can order the boolean connectives by decreasing binding strength;

   (strongest) $\neg$, $\land$, $\lor$, $\to$, $\leftrightarrow$ (weakest)

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

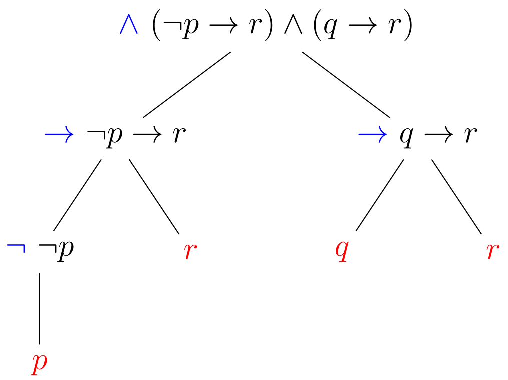
   There are also exceptions to the rule, for example with $p \to r \land q \to r$ - this should be $p \to (r \land q) \to r$ according to our binding conventions, but brackets should be used to ensure the correct interpretation.

Formulas

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

1. a propositional atom $(p, p', p_0, p_1, p_2, p_n, q, r, s, ...)$ is a propositional formula
2. $\top$, and $\bot$ 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.

Definitions

- 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	$\top$	$\bot$	$p \wedge q$	$p \vee q$	$\neg p$	$p \rightarrow q$	$p \leftrightarrow q$	$p \uparrow q$
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 will go out)` $\wedge$ `(it is raining)`

- **unless** generally means `or`

  "I will go out unless it rains" `(i will go out)` $\vee$ `(it will rain)` (note the `will`)

  $\neg$`(it will rain)` $\rightarrow$ `i will go out`

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

  `(i will go out)` $\leftrightarrow$ $\neg$`(it 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 am the pope)` $\rightarrow$ `(i am an atheist)` evaluates to 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.

Another example is the following; $p \wedge q \rightarrow r$, and $(p \rightarrow r) \vee (q \rightarrow r)$ are logically equivalent, but can be translated into different meanings. For example, let p be "event A happens", let q be "event B happens", and q be "event C happens". The former can be translated to "If both A and B happens, then C happens", whereas the latter becomes "If A happens, then C happens, or if B happens, then C also happens".

Arguments

We use the double turnstile, $\vDash$ (`\vDash` in LaTeX), to mean **therefore**. For example, the *Socrates syllogism* can be expressed as `(socrates is a man)`, `(men are mortal)` $\vDash$ `(socrates is mortal)` in logic, and in English as;

- Socrates is a man
- Men are mortal
- Therefore, Socrates is mortal

The definition of a valid argument is as follows;

Given valid formulas $A_1, A_2, ..., A_n, B$, and '$A_1, ..., A_n$ therefore B', we can write it as $A_1, ..., A_n \vDash B$, iff B is true in every situation where $A_1, ..., A_n$ are all true.

Examples

- $A, A \rightarrow B \vDash B$ **modus ponens**
- $A \rightarrow B, \neg B \vDash \neg A$ **modus tollens**
- $A \rightarrow B, B \nvDash A$ A can be false, as falsity implies anything

Definitions

- A propositional formula is logically **valid** if it's true in all situations ($\models A$), if A is **valid**
- A propositional formula is **satisfiable** if it's true in at least one situation (hence **valid** $\rightarrow$ **satisfiable**)
- Two propositonal formulas are logically **equivalent** if they are true in the same situations.

argument	validity	satisfiability	equivalence
$A \models B$	$A \rightarrow B$ valid	$A \wedge \neg B$ unsatisfiable	$(A \rightarrow B) \equiv \top$
$\top \models A$	A valid	$\neg A$ unsatisfiable	$A \equiv \top$
$A \not\models \bot$	$\neg A$ not valid	A satisfiable	
$A \models B$, and $B \models A$		$A \leftrightarrow \neg B$ unsatisfiable	$A \equiv B$

(copied directly from *Propositional Logic - Arguments and Validity.pdf*)