Lecture 1

Ben Rosenberg

June 7, 2021

Mathematical background

Propositional logic

Proposition: has one property, which is its "truth value". This can take exactly one of two possible values: `true` or `false` (sometimes written alternatively as 1 and 0, or $+$ and $-$, on and off, pass and fail, anything which is recognizable to the majority of people).

Mathematics is based on axioms, which are things that we all take for granted (all assume to be true).

Everything needs to be reduced to propositions, because everything in computing is built off of truth values. But not everything *can* be modeled by propositions – one example might be a light switch, which may at times have a dimmer and therefore not be fully on or off.

Propositional operators: ways to make propositions out of propositions.

Ex. Propositions p and q.

- $p \wedge q$: p and q, operator is called "conjunction"
  - True when both p and q are `true`, and `false` otherwise
- $p \vee q$: p or q, operator is called "disjunction"
  - True when at least one of p or q is `true`
- $\neg p$: not p, operator is called "negation"
  - True when p is `false`
- $p \implies q$: p implies q, operator is called implication
  - False when p is `true` and q is `false`
- $p \Leftrightarrow q$: p iff q, or p is equal to q; operator is called equality
  - True when p is the same as q

Some propositional operators can be expressed in terms of others. For instance, negation and conjunction, or negation and disjunction, or negation and implication, can create all others from them.

Truth table for implication:

p	q	$p \implies q$
0	0	1
0	1	1
1	0	0
1	1	1

This table is a little counterintuitive on its second row. The reason that $0 \implies 1$ is always 1 is because of something called *vacuous truth*, which means that `false` $\implies q$ is a `true` proposition for all q. **Once there is a false assumption, any inference can be made – any implication can be true.**

Other relevant facts about propositional operators

- $p \wedge q = q \wedge p$ (commutativity)
- $p \vee q = q \vee p$
- DeMorgan's Laws:
  - $\neg(p \wedge q) = (\neg p) \vee (\neg q)$
  - $\neg(p \vee q) = (\neg p) \wedge (\neg q)$
- $(p \Leftrightarrow q) = (p \implies q) \wedge (q \implies p)$
- $p \implies q = \neg(p \neg q) = \neg p \vee q$ **(important)**
- $p \wedge q = \neg(p \implies \neg q) = \neg(q \implies \neg p)$

Predicate logic

Predicate: template for making propositions.

Ex. Grading

- If a student is failed, then the proposition is `true`; otherwise, it is `false`.
- There is one of these propositions for every student
- Want the conjunction of all these propositions to be true

Problem: there are a lot of conjunctions in this proposition; it's too long; it's messy.

The proposition is only `true` if every student is failed. But saying this as the conjunction of propositions is unsatisfactory, as it is uninteresting. There is a collectivizing aspect in what we want to accomplish. The plan isn't about individuals, but rather the entire class. As such, we should introduce something in order to simplify our writing and move the focus from individuals to the class as a whole.

We can think about predicate as a box, with values from a domain within it (to be explained later). The box is known as a **propositional function**; throwing a value from some domain into the box produces a proposition. As an example, the box might be a grading rule. A student's exam might be a value in a domain, which is thrown into the grading rule propositional function, which produces either "pass" or "fail".

Domain: where are argument values coming from?

Propositional function: how to calculate propositions from the argument values (input).

In order to create a predicate, we need both a domain and a propositional function. We can think of it as a machine that generates propositions.

Next time: learning the third of our basic pieces.