Lecture 2 - Propositional Logic

Vaishnavi Sundararajan

COL703/COL7203 - Logic for Computer Science

Recap: Structural Induction

- Induct over arbitrary recursive definitions (not just naturals/integers)
- Naturals, integers, trees, lists...

Consider S, defined as the smallest set satisfying the following:

- $0 \in S$
- If $a \in S$, then $(a) \in S$

Prove that every element in S has balanced left and right parentheses.

Consider the following definition of length of strings over an alphabet Σ.

- $\text{len}(\varepsilon) = 0$
- $\text{len}(sa) = 1 + \text{len}(s)$, where $a \in \Sigma, s \in \Sigma^*$

Prove that for all strings $x, y \in \Sigma^*$, $\text{len}(xy) = \text{len}(x) + \text{len}(y)$.

Recap: Structural Induction

- Induct over arbitrary recursive definitions (not just naturals/integers)
- Naturals, integers, trees, lists...

Consider S, defined as the smallest set satisfying the following:

- $0 \in S$
- If $a \in S$, then $(a) \in S$

Prove that every element in S has balanced left and right parentheses.

Consider the following definition of length of strings over an alphabet Σ.

- $\text{len}(\varepsilon) = 0$
- $\text{len}(sa) = 1 + \text{len}(s)$, where $a \in \Sigma, s \in \Sigma^*$

Prove that for all strings $x, y \in \Sigma^*$, $\text{len}(xy) = \text{len}(x) + \text{len}(y)$.

Strings in Σ^* are generated by $S \to \varepsilon \mid S \cdot a \; (a \in \Sigma)$

$len(\varepsilon) = 0 \qquad len(s \cdot a) = len(s) + 1$ for $a \in \Sigma, s \in \Sigma^*$.

To prove: $\forall x, y \in \Sigma^*$, $len(x \cdot y) = len(x) + len(y)$.

Proof: By structural induction on y.

Base case: $y = \varepsilon$: $len(x \cdot y) = len(x \cdot \varepsilon) = len(x)$

$= len(x) + 0 = len(x) + len(y)$

IH: For all $x \in \Sigma^*$ & all strings z recursively smaller than y_0,

$len(x \cdot z) = len(x) + len(z)$.

Inductive case: $y_0 = z \cdot a \qquad len(x \cdot y_0) = len(x \cdot z \cdot a) = len(x \cdot z) + 1$

by IH $len(x) + len(z) + 1 = \ldots$

Recall: Why logic?

- Logic allows us to make sense of our world
- "What constitutes a valid proof?"
- "Is my set of statements internally consistent?"
- Valid inference and internal consistency becomes paramount when we **model complex systems**
- Logic allows us to **verify** that systems work correctly...
- ...without testing each possible execution!
- Important to know when inference is sound!

~~Trust~~ Model, then verify

- A model *abstracts* away extraneous details
- Choice of model heavily tied to the verification context
- Same framework for model and properties we would like to verify
- Sometimes a very simple framework suffices, sometimes not!
- Navigate thin line between expressiveness and tractability of **syntax**
- We start with one of the simplest such: **propositional logic**

Propositional Logic

- Every statement of interest modelled as a **proposition**
- What is a proposition? A statement that can be evaluated for truth or falsehood. Examples:
  - COL703 is a core course for CS5 students
  - New Delhi is the capital of India
  - Blood is gold in colour
- What is not a proposition? Questions, exclamations, doubts...
- Statements whose truth value changes based on context

Compare

- Is there a number such that doubling it and adding two gives ten?
- $2x + 5 = 17$
- See you tomorrow!
- $2 * 4 + 5 = 17$
- $8/0 = 42$
- Hopefully quantum computers will become commonplace soon
- This is not a proposition

Propositional logic: Syntax

- When using a logic, one is bound by the rules of *syntax*
- Only "grammatically-correct" statements are "allowed"
- Start with a (countable) set *AP* of propositional **atoms**
  - "Smallest" statements of interest
  - Can build up bigger statements with these
- Combine atoms from *AP* using **operators** to form bigger propositions: AND ($\wedge$), OR ($\vee$), NOT ($\neg$), IMPLIES ($\supset$)
- Grammar for propositional logic (PL) is as follows

$$\varphi, \psi := p \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \supset \psi \qquad \text{where } p \in AP$$

- $\wedge$ and $\vee$ are left-associative; read $\varphi \wedge \psi \wedge \chi$ as $(\varphi \wedge \psi) \wedge \chi$
- $\supset$ is right-associative; read $\varphi \supset \psi \supset \chi$ as $\varphi \supset (\psi \supset \chi)$

Propositional logic: Syntax

- This grammar produces the **well-formed formulas** (wffs) of propositional logic
- Can construct abstract syntax trees (ASTs) for well-formed formulas