CS 70

Luca Manolache

January 30, 2024

CONTENTS

LECTURE 1 PROPOSITIONAL LOGIC

1.1 Propositions

Definition 1.1.1 (*Propositions*)

Statements that are true or false.

Some example propositions are $\sqrt{2}$ is irrational, $2+2=4$, and $2+2=3$. Some non examples are $4+5$ and $x+x$.

1.2 Propositional Forms

Propositions can be put together to make another proposition. Some examples of this are conjunction ($P \wedge Q$), disjunction ($P \vee Q$), and negation ($\neg P$).

Example : Sample propositional forms:

1. $\neg(2+2=4)$ is false
2. $(2+2=3) \wedge (2+2=4)$ is false

1.3 Implication

Definition 1.3.1 (*Implication*)

$P \Rightarrow Q$ can be read as "If P, then Q."
Implications are only false if P is true and Q is false. When P is false, the implication will be true, however this is a meaningless statement.
Additionally, an implication is equivelent to

$$P \Rightarrow Q \equiv \neg P \vee Q \tag{1.1}$$

Example : If you stand in the rain, then you'll get wet.
$P =$ "you stand in the rain"
$Q =$ "you get wet"
$P \Rightarrow Q$

Note. If $P \Rightarrow Q$, that does not say anything about the opposite of $Q \Rightarrow P$.

Definition 1.3.2 (*Contrapositive*)

The contrapositive of $P \Rightarrow Q$ is $\neg Q \Rightarrow \neg P$. They are logically equivelent to the implication.

Definition 1.3.3 (*Converse*)

The converse of $P \Rightarrow Q$ is $Q \Rightarrow P$. They are NOT logically equivelent to the implication.

Definition 1.3.4 (*If and only if*)

If $P \Rightarrow Q \wedge Q \Rightarrow P$, then P if and only if (iff) Q or $P \Leftrightarrow Q$.

1.4 Truth Tables

Truth tables can simplify doing calculations with propositions.

Truth tables are also able to be used to prove logical equivalence. If two statements have the same truth tables, they are logically equivelent.

1.5 Quantifiers

None of the below are propositions as they have a free variable.

- $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
- $x > 2$
- n is even and the sum of two primes

These are all called predicates. They are similar to a function which returns true or false.

To turn them into a predicate we need a quantifier.

Definition 1.5.1 (*Universe*)

A universe is the type of a variable. Some examples are:
- $N = \{0, 1, 2, \ldots\}$
- $N^+ = \{1, 2, 3, \ldots\}$

Definition 1.5.2 (*There Exists*)

$(\exists x \in S)(P(x))$ means $P(x)$ is true for some x in S.
Example: $(\exists x \in N)(x = x^2)$ True because $0 \times 0 = 0$ and is equivelent to $(0 = 0) \wedge (1 = 1) \wedge (2 = 4) \wedge \ldots$.

Definition 1.5.3 (*For all*)

$(\forall x \in S)(P(x))$ means $P(x)$ is true for all x in S.
Example: $(\forall x \in N)(x+1 > x)$ True because adding one to a number will always be larger.

Note. Quantifiers are not communitive.

1.6 DeMorgan's Law for Quantifiers

When negating a quantifier, you negate the inside and flip the quantifier. This means

$$\neg(\forall x \in S)(P(x)) \equiv (\exists x \in S)(\neg P(x)) \tag{1.2}$$

and

$$\neg(\exists x \in S)(P(x)) \equiv (\forall x \in S)(\neg P(x)) \tag{1.3}$$

LECTURE 2 PROOFS

2.1 Background and Notation

Definition 2.1.1 (*Integers*)

Integers are closed under addition.

$$a, b \in \mathbb{Z} \Rightarrow a + b \in \mathbb{Z}$$

Definition 2.1.2 (*Divisable*)

$a|b$ is read as "a divides b".
Formally, $a|b \Leftrightarrow \exists q \in \mathbb{Z}$ where $b = aq$.

2.2 Direct Proof

To prove $P \Rightarrow Q$ you assume P is true and use that to prove Q.

2.3 Proof by Contraposition

Remember 1.3.2, $(P \Rightarrow Q) \equiv (\neg Q \Rightarrow \neg P)$. Therefore, if we assume $\neg Q$ and can prove that it implies $\neg P$, then we have proved the original goal of $P \Rightarrow Q$.

2.4 Proof by Contradiction

Show that if we assume $\neg P$, then we use forward reasoning (direct proof) to show $\neg P \Rightarrow R$ and to show $\neg P \Rightarrow \neg R$. This is often found when trying to show a simple property should always not hold.

2.5 Proof by Cases

Prove all possible cases for something are true, therefore the thing you are proving must be true.

LECTURE 3 INDUCTION

3.1 Induction

Induction is used to prove statements of the form:

$$(\forall k \in \mathbb{N})(P(k))$$

The basic form for mathematical induction is:

- Prove $P(0)$ (the base case)
- Prove $P(k) \Rightarrow P(k+1)$
  - Assume $P(k)$ (induction hypothesis)
  - Prove $P(k+1)$ (induction step)

Note. We get to use $P(k)$ to prove $P(k+1)$. We only prove local things for k and $k+1$.

3.2 Two Color Theorem

Theorem 3.2.1 (Four color theorem)**.** Any map in a plane can be colored using four-colors in such a way that regions sharing a common boundary (other than a single point) do not share the same color

This theorem is way too hard to prove, so instead we focus on a simplified theorem called the two color theorem.

Theorem 3.2.2 (Two color theorem)**.** Any map formed by dividing the plane into regions by drawing straight lines can be properly colored with two colors.

Proof: Start with one line dividing the plane. This can be clearly divided in two colors.

Add a line. Each added line will get the previous one and then fixed conflicts by switching on one side. ■

LECTURE 4 STABLE MATCHING

4.1 Propose and Reject

- Job proposes to its favorite candidate
- Candidate holds favorite job on a string and rejects all others
- If no jobs are rejected, algorithm halts

4.2 Properties

Stable matching gets better every day for candidates

Lemma 4.2.1 (Improvement Lemma)**.** If on day t a candidate c has a job j on a string, any job, j' on candidate c's string for any day $t' > t$ is at least as good as j.

Proof: $P(k)$ - "Job on c's string is at least as good as j on day $t+k$" By induction, base case: $P(0)$ - true, candidate has j on string. Assume $P(k)$. Let j' be job on string on day $t+k$. On day $t+k+1$, job j' still on string. Candidate c can choose j' or do with a better job j''. That is $j' \geq j$ by inductive hypothesis. ■

Lemma 4.2.2. There is no rogue couple for the matching formed by stable matching algorithm.

Definition 4.2.1 (*x-optimal matching*)

x's partner is its best partner in any stable pairing.

Definition 4.2.2 (*x-pessimal matching*)

x's partner is its worst partner in any stable pairing.

Definition 4.2.3 (*job-optimal matching*)

It is x optimal for all jobs.