Mathematical concepts for Computer Science

January 3, 2023

Contents

Chapter 1

Propositional Logic

1.1 Introduction

1.1.1 Propositions and predicates

Definition 1.1.1 *A proposition p is a statement that is either True or False*

Notation 1.1.1 *We'll indicate False as* 0 *and True as* 1

Definition 1.1.2 *A predicate, or formula, is a proposition whose truth depends on the value of its variables.*

Notation 1.1.2 *We'll usually use capital letters such as P to denote predicates.* $P(n) \in \{0, 1\}$

1.1.2 Axioms

Definition 1.1.3 *An axiom is a proposition that we consider true by default*

Axioms are at the base of mathematics. A well-known set of axioms are the one established by Euclid, or the one currently used for mathematics, the ZFC axioms.
We use axioms to establish if a proposition is true or false.

Definition 1.1.4 (Completeness) *A set of axioms is said to be* ***complete*** *if $\forall P$ we can establish whether $P(n) = 0$ or $P(n) = 1$*

Definition 1.1.5 (Consistency) *A set of axioms is said to be* ***consistent*** *if $\forall P$, $P(n)$ isn't both true and false (free from contradictions)*

Gödel incompleteness theorems

With its two incompleteness theorems, in 1930 Gödel proved that $\forall$ complete set of axioms S, $\exists P(n)$ such that $P(n) = 0 \land P(n) = 1$, basically proving that S **can't be both complete and consistent**

1.1.3 Theorems, Corollaries, Lemmas and Conjectures

Definition 1.1.6 *A theorem is an important proposition which is proved to be true*

Definition 1.1.7 *A corollary is a true proposition derived with simple steps from the theorem*

Definition 1.1.8 *A lemma is a preliminary proposition used for proving later propositions, such as other lemmas and the theorem*

Definition 1.1.9 *A conjecture is a proposition which is tought to be true but for which there isn't a proof*

1.1.4 Boolean variables and assignments

Every proposition can be transformed into a set of variables that can be either true or false, 0 or 1.
The algebra that studies the mathematical expressions with these so-called boolean variables is called boolean Algebra.
Given a proposition P, an assignment α of P is one of the unique possible combination of values of its variables within the set of all possible assignments A.
The notation of an assignment is $\alpha(P)$, whose result is $\in \{0, 1\}$

1.2 Notation

In order to make formal logic, well, formal, we need to introduce some symbols. We'll begin by discussing about truth tables, a way to show graphically for the results of $\alpha(P)\ \forall\alpha \in A$.

1.2.1 NOT

Symbol: $\neg$. Translates the clauses *not* A. Truth table:

A	$\neg A$
0	1
1	0

1.2.2 OR

Symbol: $\vee$. Translates the clauses *this or that* in a non-exclusive way.
Truth table:

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

1.2.3 AND

Symbol: $\wedge$. Translates the clauses *this and that*.

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

1.2.4 IMPLIES

Symbol: $\Rightarrow$. Translates the clauses *if this, than that*.
Expresses a **sufficient** condition for B.
Truth table:

A	B	$A \Rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

$$A \Rightarrow B = \neg B \Rightarrow \neg A = \neg A \vee B$$

1.2.5 IF AND ONLY IF

Symbol: $\iff$. Translates the clauses *if and only if this than that.* Express a **sufficient and necessary** condition for Bs Truth table:

A	B	$A \iff B$
0	0	1
0	1	0
1	0	0
1	1	1

$$A \iff B = (A \Rightarrow B \wedge B \Rightarrow A)$$

1.2.6 SAT, UNSAT e TAUT

There exists three big sets for propositions in logic.
SAT, the set of satisfiable propositions, UNSAT, the set of unsatisfiable proposition, and TAUT, the set of all tautologies (also called *valid propositions*).
A proposition P is

- $\in$ SAT $\iff \exists \alpha \in A \mid \alpha(P) = 1$
- $\in$ TAUT $\iff \forall \alpha \in A \mid \alpha(P) = 1$. TAUT $\subset$ SAT.
- $\in$ UNSAT $\iff \nexists \alpha \in A \mid \alpha(P) = 1 \iff \neg P \in$ TAUT

1.2.7 Boolean Algebra and Logic

Due to the fact that in boolean algebra variables may only have 2 states, and given how boolean's operations are defined, this algebra model is equivalent to formal logic.
We are basically stating that boolean algebra is equivalent to logic algebra

1.2.8 CNF and DNF

Knowing this, and knowing that every boolean formula can be transformed into an Sum-of-products or Product-of-sums form, we know that every proposition P can be transformed into a CNF or DNF.

Definition 1.2.1 (Clause) *A clause is a formula formed from a finite set of variables V.*
In CNF, $C = \bigvee_{i=0}^{n} L_i$, and $C = \emptyset = \square = 0$.
In DNF, $C = \bigwedge_{i=0}^{n} L_i$, and $C = \emptyset = \square = 1$

Definition 1.2.2 (CNF) *A CNF (Conjuctive normal form) is a way to express a proposition P as an AND of ORs.*

$$P = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{\#C_i} L_{i,j}$$

Definition 1.2.3 (DNF) *A DNF (Disjunctive normal form) is a way to express a proposition P as an OR of ANDs.*

$$P = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{\#C_i} L_{i,j}$$

Set notation for normal forms

After stating whether we are working in DNF or CNF, we can write a normal form of n clauses as

$$\{C_i \, \forall i \in [1, n]\} = \{\{L_{ij}\} \, \forall i \in [1, n] \, \forall j \in [1, \#C_i]\}$$

1.3 Determine if $P \in$ SAT

1.3.1 The $P = NP$ problem

This problem is equivalent to $P = NP$, the holy grail of computer science.
That is because to check if a proposition is satisfiable, we may have to check all combinations of variables, so, for a proposition F with n variables, we may have to check 2^n combinations.
Currently there exists some method of proving whether a normal form is SAT in polynomial time, but in order to do it we must first convert F into a NF, which may cause the formula to grow exponentially.

1.3.2 The resolution method

Here we are gonna state the resolution method to resolve any CNF.
The algorithm works as follow:

1. Check if there $\exists C_1, C_2 \in F \mid C_1 \neq C_2 \land p \in C_1 \land \neg p \in C_2$
2. If there exists such couple, remove p from both clauses. If there isn't such couple, $F \in$ SAT
3. If $C_1 = \Box \lor C_2 = \Box$ then $F \in$ UNSAT, else repeat

This method is complete and sound

Definition 1.3.1 (Completeness of RIS) *If $F \in$ UNSAT then the resolution method will yield UNSAT as the result*

Definition 1.3.2 (Soundness of RIS) *If the resolution method yields UNSAT as the result then $F \in$ UNSAT*

Chapter 2

Proofs

2.1 Inference rules

Inference rules are the way we prove new propositions based on propositions previously proved.

- Modus ponens: $P == 1 \land (P \Rightarrow Q) \Rightarrow Q$
- Transitivity: $P \Rightarrow Q \land Q \Rightarrow R \Rightarrow P \Rightarrow R$
- Contrapositive: $\neg Q \Rightarrow \neg P \Rightarrow P \Rightarrow Q$

2.2 How to: prove an implication

There are multiple ways of showing an implication is true.
They are all correct, but some work better than others in certain cases.

- Direct proof: assuming $P == 1$, show that it implies $Q == 1$
- Proof by contraposition: use the direct proof on the contrapositive $\neg Q \Rightarrow \neg P$

2.3 How to: prove an iff

- Direct 2-way proof: prove $P \Rightarrow Q$ and prove $Q \Rightarrow P$
- Chain proof: Construct a chain of iff clauses such that $P \iff ... \iff Q$

2.4 Proof by cases

A proof by cases is an *exhaustive* proof. It demonstrates that for all the possible cases that there may be and prove all sub-cases.

2.5 Proof by contradiction

In a proof by contradiction we show that if some proposition P in our proof was to be false, then some other proposition T known to be true would have to be false

2.6 How to: write a good proof

- State the thesis and how you're going to demonstrate it
- Keep a linear flow: it's a proof, not a movie
- It's not a math test: don't put too many calculations in the proof
- Avoid using too many symbols: where and when you can, use words
- But don't use no symbols at all: don't be afraid to introduce new symbols as notations, variables, ecc..., if it will make the proof easier to read
- Use lemmas and corollaries: write the proof as a one-step-at-a-time process, making use of lemmas and corollaries when usefull and making each part of the proof clear
- Keep an eye towards the obvious: remember that everything that seems obvious for you isn't necessarily obvious for the reader, so explain and demonstrate more advanced stuff

2.7 Well-ordering principle and Induction

2.7.1 Well-ordering principle

The well-ordering principle is an axiom of the ZFC axiom set.

Axiom 2.7.1 (Well-ordering principle)

$$\forall S \subseteq \mathbf{N}, S \neq \emptyset \Rightarrow \exists n \in S \mid n \leq m \, \forall m \in S$$

Proofs with the WOP

This principle is usefull to construct proofs, usually by contradiction.
The usual scheme is:

- Define the set $C = \{n \in \mathbf{N} \mid \neg P(n)\}$
- Assume $C \neq \emptyset$
- Reach a contradiction
- Conclude C must be empty by contradiction

2.7.2 Induction

Induction is another proof method equivalent to contradiction, states the following:

Axiom 2.7.2 (Induction principle)

$$(P(c \in \mathbf{N}) \wedge P(n) \Rightarrow P(n+1)\ \forall n \geq c) \Rightarrow P(n)\ \forall n \geq c$$

Proofs using Induction

The usual schema is:

- Prove that $\exists c \in \mathbf{N} \mid P(c) == 1$
- Prove that $P(n) \Rightarrow P(n+1)\ \forall n \geq c$

Strong Induction

Strong induction is equivalent to the proof by induction with the addition that, other from $P(n) == 1$, we also assume $P(m) == 1\, \forall c \leq m \leq n$ in order to prove $P(n+1)$