Formal Proofs and their Lengths I

Basic propositional logic

Definition 1. Let $V = \{x_1, x_2, x_3, \dots\}$ be a countable set, we will call V the set of *propositional variables* (atoms). We define a *propositional formula* (in the DeMorgan Language) to be a word defined by the following recursive conditions:

- A is a formula, if it is a propositional variable.
- A is a formula, if it is of the form $(B \wedge C)$, where B and C are formulas.
- A is a formula, if it is of the form $(B \vee C)$, where B and C are formulas.
- A is a formula, if it is of the form $\neg B$, where B is a formula.

A *subformula* of a formula A is a subword of A which is also a formula. The notation $A(p_1, \dots, p_n)$ means that the propositional variables occuring in A are among the set $\{p_1, \dots, p_n\}$.

Definition 2. Let $A(p_1, \dots, p_n)$ be a propositional formula. We call any function $h : \{p_1, \dots, p_n\} \to \{0, 1\}$ a truth assignment (of A). Any truth assignment can be extended to give a $\{0, 1\}$-value to A by the obvious recursive definition. If $h(p_i) = b_i$ for each $1 \leq i \leq n$, we denote the value $h(A)$ as $A(b_1, \dots, b_n)$.

We say A is *satisfiable* if there is a truth assignment such that $h(A) = 1$, otherwise we call it *unsatisfiable*. We say A is a *tautology* if every truth assignment h results in $h(A) = 1$.

Exercise 3. Observe that a propositional formula A is a tautology iff $\neg A$ is unsatisfiable.

Definition 4. A function of the form $f : \{0, 1\}^n \to \{0, 1\}$ is a *Boolean function*, every propositional formula $A(p_1, \dots, p_n)$ determines the *truth-table function* $\mathbf{tt}_A$ as

$$\mathbf{tt}_A : (b_1, \dots, b_n) \mapsto A(b_1, \dots, b_n).$$

Exercise 5. Show that every Boolean function is a truth-table function of some propositional formula A.

Exercise 6. Show that for every propositional Boolean formula in the De Morgan language A there exists a formula[1] A' in the language using only the connectives form the set $\{\neg, \to\}$ (interpreted as negation and implication) such that $\mathbf{tt}_A = \mathbf{tt}_{A'}$.

Definition 7. A propositional formula A is in the conjunctive normal form (CNF) if it is of the form $\bigwedge_i \bigvee_j \ell_{ij}$, where each ℓ_{ij} is either a propositional variable or a negation of one (a literal).

[1] This is not a propositional formula by our definition, but you can check an analogous definition can be made for this set of connectives.

A propositional formula A is in the disjunctive normal form (DNF) if it is of the form $\bigvee_i \bigwedge_j \ell_{ij}$, where each ℓ_{ij} is a literal.

Disjunctions of literals are called *clauses*, and conjunctions of literals are called *logical terms*.

Exercise 8. Show that every Boolean function is a truth-table function of some DNF A and some CNF B.

Exercise 9. Show there is a fast (polynomial time) algorithm deciding whether a DNF A is satisfiable.

Exercise 10. Show there is a fast (polynomial time) algorithm deciding whether a CNF A is a tautology.

Exercise 11. Show that there is a Boolean function such that its smallest DNF representation is exponentially smaller than its CNF representation (or vice-versa).

Exercise 12 (bonus)**.** Show that for each polynomial $p(x)$ there is a Boolean function with n inputs, which is not a truth-table function of any propositional formual A with less than $p(n)$ symbols.

Propositional Proof Systems

Definition 13. Let A be a finite set of symbols. We define $A^{\leq n} := \bigcup_{i=0}^{n} A^i$ and $A^* := \bigcup_{i\geq 0} A^i$.

Definition 14. A predicate $f : \{0,1\}^* \to \{0,1\}$ is in $\mathbf{P}$ if there is a Turing machine M computing f in polynomial time[2].

Definition 15 (Cook-Reckhow)**.** A *propositional proof system* (or a PPS) P is determined by a predicate $f(x,y)$ in $\mathbf{P}$ such that for every propositional formula A:

$$A \text{ is a tautology} \iff (\exists y \in \{0,1\}^*)\, f(A,y),$$

here we interpret f to be a predicate checking that y is a valid "proof" of A. That is, if $f(A,y) = 1$, then we say y is a P-proof of A.

Example 16. The truth-table proof system is a system determined by a predicate

$$f(A,y) = \begin{cases} 1 & y \text{ is the truth-table of } A,\ (\forall \overline{x})\mathbf{tt}_A(\overline{x}) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Exercise 17. Show that the truth-table proof system is a propositional proof system by the definition of Cook-Reckhow.

Exercise 18 (First lower bound!)**.** Show that every truth-table proof of a tautology is exponentially long in the size of that tautology.

[2]The precise definition of a Turing machine in fact does not matter. If you have never encountered the definition of a Turing machine, it is enough to consider the intuitive idea of an algorithm, whose number of steps does not exceed a specific polynomial in the length of the input and this itself just means, that the algorithm is somehow feasible — does not run too long. For example, such an algorithm cannot look at every truth assignment of a formula it receives as an input.

A Little Bit of Complexity

Definition 19 (*)**.** A predicate $f : \{0,1\}^* \to \{0,1\}$ is in **NP** if there is a function $g(x,y)$ in **P** and a polynomial p such that for every $x \in \{0,1\}^n$:

$$f(x) = 1 \iff (\exists y \in \{0,1\}^{\leq p(n)})\, g(x,y) = 1,$$

if such a y exists it is called the *witness*.

Definition 20 (*)**.** A predicate $f : \{0,1\}^* \to \{0,1\}$ is in **coNP** if there is a function $g(x,y)$ in **P** and a polynomial p such that for every $x \in \{0,1\}^n$:

$$f(x) = 0 \iff (\exists y \in \{0,1\}^{\leq p(n)})\, g(x,y) = 0.$$

Exercise 21 (*)**.** Show that $f(x) \in$ **NP** if and only if $\neg f(x) \in$ **coNP**.

Theorem 22 (Cook-Reckhow)**.** **NP** $=$ **coNP** if and only if there is a propositional proof system P which has polynomial sized P-proofs of every tautology.

Exercise 23 (*)**.** Prove the Cook-Reckhow theorem.

Frege systems I

Definition 24. The textbook Frege proof system is determined by the proofs of the following form:

The connectives in every formula in the system are just $\{\neg, \to\}$. A proof of a formula A is a sequence of propositional formulas $(B_1, \ldots, B_k)$, where $B_k = A$ and for each $1 \leq i \leq k$ one of the following is true:

- B_i has any of the forms
  1. $p \to (q \to p)$
  2. $(p \to (q \to r)) \to ((p \to q) \to (p \to r))$
  3. $(\neg p \to \neg q) \to (q \to p)$,

  where p, q and r are arbitrary formulas. Such a B_i is called an axiom (in the textbook Frege system).
- There are $1 \leq j_1, j_2 < i$ such that $B_{j_1} = p$, $B_{j_2} = (p \to q)$ and $B_i = q$. Such a B_i is said to be introduced by the *modus ponens* rule:

  $$\frac{p, p \to q}{q}$$

Example 25. Prove $(a \to a) \to (a \to (a \to a))$ in the textbook Frege system.

Example 26. Prove $(a \to b) \to (a \to a)$ in the textbook Frege system.

Example 27 (Bonus)**.** Prove $a \to a$ in the textbook Frege system.

Open problem 28. Does every tautology have a polynomial sized proof in the textbook Frege system?