Tutorial 2

1. A *literal* is a propositional variable or its negation. A *clause* is a disjunction of literals such that a literal and its negation are not both in the same clause, for any literal. Similarly, a *cube* is a conjunction of literals such that a literal and its negation are not both in the same cube, for any literal. A propositional formula is said to be in *Conjunctive Normal Form (CNF)* if it is a conjunction of clauses. A formula is said to be in *Disjunctive Normal Form (DNF)* if it is a disjunction of cubes.

   Let P, Q, R, S, T be propositional variables. An example DNF formula is $(P \wedge \neg Q) \vee (\neg P \wedge Q)$, and an example CNF formula is $(P \vee Q) \wedge (\neg P \vee \neg Q)$. Are they semantically equivalent?

   For the propositional formula $(P \vee T) \rightarrow (Q \vee \neg R) \vee \neg(S \vee T)$, find semantically equivalent formulas in CNF and DNF. Show all intermediate steps in arriving at the final result.

   Hint: Use the following semantic equivalences, where we use $\varphi \Leftrightarrow \psi$ to denote semantic equivalence of φ and ψ:

   - $\varphi_1 \rightarrow \varphi_2 \;\Leftrightarrow\; (\neg\varphi_1 \vee \varphi_2)$
   - *Distributive laws:*
     - $\varphi_1 \wedge (\varphi_2 \vee \varphi_3) \;\Leftrightarrow\; (\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \wedge \varphi_3)$
     - $\varphi_1 \vee (\varphi_2 \wedge \varphi_3) \;\Leftrightarrow\; (\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3)$
   - *De Morgan's laws*
     - $\neg(\varphi_1 \wedge \varphi_2) \;\Leftrightarrow\; (\neg\varphi_1 \vee \neg\varphi_2)$
     - $\neg(\varphi_1 \vee \varphi_2) \;\Leftrightarrow\; (\neg\varphi_1 \wedge \neg\varphi_2)$

2. Consider the parity function, PARITY : $\{0,1\}^n \mapsto \{0,1\}$, where PARITY evaluates to 1 if and only if an odd number of inputs is 1.

   1. Prove that any CNF representation of PARITY must have n literals (from distinct variables) in every clause.

      [Hint: Take a clause, and suppose the variable v is missing from it. What happens when you flip v?]

   2. Prove that any CNF representation of PARITY must have 2^{n-1} clauses (all the clauses are assumed to be distinct).

      [Hint: What is the relation between CNF/DNF and truth tables?]

   PARITY is ubiquitous across Computer Science; for example, in Coding Theory (see Hadamard codes), Cryptography (see the Goldreich-Levin theorem), and discrete Fourier Analysis (yes, that is a thing)!

3. We have already defined CNF and DNF formulas in Question 1. Recall the definition of *equisatisfiable* formulas taught in class. A **stronger notion of equisatisfiability** is as follows: A formula F with variables $\{f_1, f_2 \cdots f_n\}$ and a formula G with variables $\{f_1, f_2 \cdots f_n, g_1, g_2 \cdots g_m\}$ are **strongly-equisatisfiable** if:

   - For any assignment to the f_is which makes F evaluate to true there exists an assignment to the g_is such that G evaluates to true under this assignment with the same assignment to the f_is.

     $$AND$$

   - For any assignment setting G to true, the assignment when restricted to f_is makes F evaluate to true.

   Why does this help us? A satisfying assignment for G tells us a satisfying assignment for F, and if we somehow discover that G has no satisfying assignments, then we know that F also has no satisfying assignments. In other words, the satisfiability of F can be judged using the satisfiability of G. Satisifiability is a central problem in computer science (for reasons you will see in CS218 later) and such a reduction is very valuable. In particular, for a formula F in k-CNF we look for a formula G that has a small number of literals in each clause, and the total number of clauses itself is not too large - ideally a linear factor of k as compared to F.

   Now consider an r-CNF formula, i.e a CNF formula with r literals per clause. To keep things simple, suppose $r = 2^k$, for some $k > 0$. We will start with the simplest case: a 2^k-CNF formula containing a single clause $C_k = p_0 \vee p_1 \vee p_2 \cdots p_{2^k-1}$ with 2^k literals.

   1. Let's try using 'selector variables' to construct another formula which is strongly-equisatisfiable with C_k. The idea is to use these selector variables to 'select' which literals in our clause are allowed to be false. Use k selector variables $s_1, s_2, s_3 \cdots s_k$ and write a CNF formula that is **strongly-equisatisfiable with** C_k, such that each clause has size $O(k)$.

      [Hint: If you consider the disjunction of $\tilde{s}_1 \vee \tilde{s}_2 \vee \tilde{s}_3 \cdots \tilde{s}_k$, where $\tilde{s}_i$ denotes either s_i or $\neg s_i$, it is true in all but one of the 2^k possible assignments to the s_i variables]

   2. What is the number of clauses in the CNF constructed in the above sub-question? How small can you make the clauses by repeatedly applying this technique?

   3. Consider a k-CNF formula with n clauses. When minimizing the clause size using the above technique repeatedly, what is the (asymptotic) blowup factor in the number of clauses? Observe that the factor is almost linear, upto some logarithmic factors.

      [Hint: Apply the above reduction repeatedly and see how much net blowup occurs]

   4. Can you think of a smarter way to do the reduction that only requires linear blowup?

      [Hint: Consider using different and more auxiliary variables (like the 'selector variables' above) to reduce the size of clauses.]

4. **[Take-away question – solve in your rooms]**

   In this question we will view the set of assignments satisfying a set of propositional formulae as a language and examine some properties of such languages.

   Let $\mathbf{P}$ denote a countably infinite set of propositional variables $p_0, p_1, p_2, \ldots$. Let us call these variables positional variables. Let Σ be a countable set of formulae over these positional variables. Every assignment $\alpha : \mathbf{P} \to \{0, 1\}$ to the positional variable can be uniquely associated with an infinite bitstring w, where $w_i = \alpha(p_i)$. The language defined by Σ - denoted by $L(\Sigma)$ - is the set of bitstrings w for which the corresponding assignment α, that has $\alpha(p_i) = w_i$ for each natural i, satisfies Σ, that is, for each formula $F \in \Sigma$, $\alpha \models F$. In this case, we say that $\alpha \models \Sigma$. Let us call the languages definable this way PL-definable languages.

   (a) Show that PL-definable languages are closed under countable intersection, ie if $\mathcal{L}$ is a countable set of PL-languages, then $\bigcap_{L \in \mathcal{L}} L$ is also PL-definable.

   (b) Show that PL-definable languages are closed under finite union, ie if $\mathcal{L}$ is a finite set of PL-languages, then $\bigcup_{L \in \mathcal{L}} L$ is also PL-definable.

   [Hint: Try proving that the union of two PL-definable languages is PL-definable. The general case can then be proven via induction. If $\mathbf{F} = \{F_1, F_2, \ldots\}$ and $\mathbf{G} = \{G_1, G_2, \ldots\}$ are two countable sets of formulae, then an infinite bitstring $w \in L(\mathbf{F}) \cup L(\mathbf{G})$ if and only if either $w \models$ every F_i or $w \models$ every G_i.]