CS 228M : Logic for Computer Science Spring 2026

Practice Problem Set 2

Instructions:

- The following problems are meant for you to practice, so that your understanding of the topic improves.
- You must solve all problems to get the maximum benefit from practice problems.
- You must not submit your solutions to these problems. These are not going to be graded.
- A problem may have multiple solution techniques. Discussion among students is strongly encouraged in order to understand different perspectives.
- Questions marked * are comparably more difficult than the rest.

---

1. **2-CNF Graphs**
   A *2-CNF formula* is a Boolean formula in conjunctive normal form (CNF) in which each clause contains at most two literals.
   Let F be a 2-CNF formula over Boolean variables. The *implication graph* of F is a directed graph defined as follows:
   - For each variable x, the graph contains two vertices labeled x and $\neg x$.
   - For each clause of the form $(a \vee b)$ in F, where a and b are literals, the graph contains directed edges
   $$(\neg a \to b) \quad \text{and} \quad (\neg b \to a).$$

   (a) Consider the following 2-CNF formula:
   $$F = (x \vee y) \wedge (\neg x \vee z) \wedge (\neg y \vee \neg z).$$
   Using the definition above, construct the implication graph of F. Clearly list all vertices and all directed edges.

   (b) In a directed graph (like the graph created above), a *strongly connected component (SCC)* is a maximal set of vertices S such that for every pair of vertices $u, v \in S$, there exists a directed path from u to v and a directed path from v to u.
   Prove that if, in the implication graph of a 2-CNF formula, a variable x and its negation $\neg x$ belong to the same strongly connected component, then the formula is unsatisfiable. **Bonus:** Is the converse true, i.e. if there is no strongly connected component that contains both a literal and its complement, does it necessarily mean the 2-CNF formula is satisfiable? If so, how can a satisfying assignment be created from the graph?

2. **Exponential Blow-up***
   Define the *length* of a CNF (or DNF) formula as the total number of literals over all clauses (or cubes, respectively) in the formula. Show that there exists a family of formulas $\mathcal{F} = \{\varphi_n \mid n \in \mathbb{N}\}$ such that:
   - Each φ_n is a DNF formula over $O(n)$ propositional variables.
   - Each φ_n has length $O(n)$.

- For every n, there does not exist any semantically equivalent CNF formula ψ_n (over the same variables as φ_n) whose length is polynomial in n.

3. **Blocked Clause Elimination**
   Let φ be a CNF formula and let C be a clause in φ. We say that C is a *blocked clause* in $F\varphi$ if there exists a literal ℓ (also called *blocked literal*) in C satisfying the following condition:

   > For every other clause C' in φ that has$\neg\ell$ as a literal, there also exists another literal ℓ' in C such that $\neg\ell'$ is in C'.

   For example, if φ is the formula $(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_2 \vee x_4 \vee x_3) \wedge (x_1 \vee x_3 \vee x_4)$, then the clause $(x_1 \vee x_3 \vee \neg x_3)$ is a blocked clause with $\ell = \neg x_3$ being the blocked literal. Why? Find every other clause that has $\neg\ell$, i.e. x_3 as a literal in it. These are: $(\neg x_1 \vee x_3)$ and $(\neg x_2 \vee x_4 \vee x_3)$. For $C' = (\neg x_1 \vee x_3)$, we choose $\ell' = x_1$. Indeed $\ell' = x_1$ is present in C, and $\neg\ell' = \neg x_1$ is present in C'. Similarly, for $C' = (\neg x_3 \vee x_4 \vee x_3)$, we choose $\ell' = x_2$, since ℓ' is present in C, and $\neg\ell'$ is present in C'.

   The process of blocked clause elimination simplifies a CNF formula by removing all blocked clauses from the formula. This is an important optimization used in modern SAT solvers.

   1. Show that applying blocked clause elimination to φ repeatedly until no further blocked clauses exist, preserves satisfiability; that is, prove that the resulting formula φ' is satisfiable if and only if F is satisfiable. (First, prove that removing a single blocked clause from a formula preserves satisfiability and then, argue why repeated application of this rule until no blocked clauses remain does not affect satisfiability.)
      Hint: Divide the problem into two cases :
      1) When there is a satisfying assignment where the blocked literal is set to 1
      2) When all satisfying assignments have the blocked literal set to 0.
   2. Prove that blocked clause elimination preserves the set of satisfying assignments of φ, or give a counterexample that shows that the satisfying assignments of φ before and after blocked clause elimination can vary.

4. **Substitution Theorem**
   Two formulae A, B are said to be semantically equivalent if $A \models B$ and $B \models A$. Suppose you are told that F is semantically equivalent to G. Let H be a formula that contains F as a subformula. Let H' be the formula obtained by replacing some occurrence of sub-formula F in H with G. Then show that H is semantically equivalent to H'.

   As an example of application of the above claim, let H be the formula $(x_1 \rightarrow (x_2 \rightarrow (x_3 \wedge (x_3 \rightarrow x_4))))$, and let F be the sub-formula $x_3 \wedge (x_3 \rightarrow x_4)$. Suppose you are told that F is semantically equivalent to the formula G, given by $x_3 \wedge x_4$. The claim in the above question, applied to this example, asserts that H is semantically equivalent to the formula H' given by $(x_1 \rightarrow (x_2 \rightarrow (x_3 \wedge x_4)))$

5. **Linear Resolution***
   A resolution process is called linear if for each step, one of the clauses is the resolvent in the previous step. Prove that linear resolution is complete.

6. **k-SAT to 3-SAT**
   A CNF formula is called a *k-SAT formula* if every clause contains at most k literals. The *k-SAT problem* asks whether a given k-SAT formula is satisfiable. It is known that for every $k \geq 3$, the k-SAT problem is computationally no harder than the 3-SAT problem. In other words, if you could find a way to solve the 3-SAT problem efficiently, you could also find a way to solve the k-SAT problem efficiently for every $k \geq 4$. A key step in establishing this fact is showing that any k-SAT instance can be *reduced* or transformed to an equisatisfiable 3-SAT instance using only a linear increase in size.

Let

$$G = (\ell_1 \vee \ell_2 \vee \cdots \vee \ell_k)$$

be a clause with $k \geq 4$, where each ℓ_i is a literal.

Using a *Tseitin-style encoding*, construct an equisatisfiable 3-CNF formula G' for G by introducing fresh auxiliary variables. Your construction should use only a linear (in k) number of new variables and clauses.

7. **Polytime Satisfiability**
Let F be a Boolean formula in conjunctive normal form (CNF) over a set of propositional variables $V = \{x_1, x_2, \ldots, x_n\}$. Assume that each propositional variable occurs atmost twice in F counting both positive and negative occurrences. Show using resolution there exists a polynomial time algorithm to decide whether F is satisfiable. Because every variable appears at most twice, think about eliminating variables using resolution or by setting them when they occur only once, and argue that these eliminations preserve satisfiability and terminate in polynomial time.

8. **K true satisfiability***
Given a CNF formula F on variables $X = \{x_1, \ldots, x_n\}$ and an integer k. Construct a CNF formula Ψ (size poly in n, k) such that

$$\Psi \text{ is satisfiable } \leftrightarrow F \text{ has a satisfying assignment with exactly } k \text{ variables true.}$$

You are allowed to use auxiliary variables as long as cnf size is polynomial. In particular, you may use variables of the form

$$s_{i,j} \qquad (1 \leq i \leq n-1,\ 1 \leq j \leq t)$$

for suitable values of t, where $s_{i,j}$ means among first i variables $y_1, \ldots, y_i$ atleast j are true.
Hint: Try enforcing "exactly k true variables" by combining a constraint that ensures at least k of the x_i are true with another constraint that ensures at least $n - k$ of the x_i are false.

9. **Positive Resolution***
Define Positive resolution as a restriction of ordinary resolution as follows: derive a resolvent from clauses C1 and C2 only if C1 is a positive clause, i.e., it consists only of positive literals. Prove or disprove : If F is an unsatisfiable CNF formula then one can derive the empty clause from F using only positive resolution.

10. **HornSAT**
A *Horn clause* is a clause that contains at most one positive literal. A *Horn formula* is a Boolean formula in conjunctive normal form (CNF) in which every clause is a Horn clause.

(a) Consider the following Horn formula:

$$F = (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_3 \vee x_4) \wedge (\neg x_4) \wedge (\neg x_5 \vee x_3) \wedge (\neg x_6 \vee x_5) \wedge (\neg x_7 \vee x_6).$$

Apply the *Horn-SAT algorithm* discussed in class to determine whether the formula F is satisfiable. Clearly show the sequence of implications and the final truth assignment produced by the algorithm.

(b) The Horn-SAT algorithm produces the *minimal model* of a Horn formula by repeatedly setting variables to 1 only when forced by implications.
Suppose that after the algorithm terminates, we assign the value 1 (true) arbitrarily to some variables that are still unassigned.

(i) Give an example of a Horn formula and a variable assignment where assigning additional variables to 1 after the algorithm terminates still results in a satisfying assignment.

(ii) Give an example of a Horn formula and a variable assignment where assigning additional variables to 1 after the algorithm terminates causes the formula to become unsatisfied.

In each case, justify your answer.

11. **Hyper Resolution** We studied propositional resolution where in given two clauses C_1, C_2 where an atom p appears exactly as a literal p in one of the clauses and $\neg p$ literal in the other, we can derive a new clause C_3 such that it contains union of all the literals in C_1, C_2 except $\{p, \neg p\}$. That is,

$$\frac{A \vee p, \quad B \vee \neg p}{A \vee B}$$

Based on this, one can define a new rule called Hyper Resolution as follows:

$$\frac{l \vee x_1 \vee x_2 \vee \ldots x_n \quad \neg x_1 \vee A_1 \quad \neg x_2 \vee A_2 \quad \neg x_n \vee A_n}{l \vee A_1 \vee A_2 \vee \ldots \vee A_n}$$

Notice that a special interesting case of this rule could be when $A_1 = A_2 = \ldots A_n = A$ and the resulting clause essentially becomes a binary clause i.e. $l \vee A$. (Infact, if l was false, then it could very well be a unit clause.)

Using this insight, prove that the following set of clauses are unsatisfiable by first applying only Propositional Resolution and then by a combination of Hyper Resolution and Propositional resolution and compare which of the two strategies reaches UNSAT faster. Comment on the same why is this the case.

$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_5) \wedge (x_6 \vee x_8) \wedge (\neg x_2 \vee x_5) \wedge (\neg x_3 \vee x_5) \wedge (\neg x_6 \vee \neg x_5) \wedge (\neg x_8 \vee \neg x_5)$$