# OLD CS 473: Fundamental Algorithms, Spring 2015

# Discussion 12

**April 16, 2015**

**12.1.** Building 3CNF formulas.

CNF formula (***conjunctive normal form***) is a boolean formula that is the 'and' of clauses, where every clause is the 'or' of literals, where every literal is either a variable or its negation. For example, a CNF formula is

$$\left(\overline{x_1} \vee x_2\right) \wedge \left(\overline{x_2} \vee x_3 \vee x_4 \vee \overline{x_5}\right) \wedge \left(\overline{x_2} \vee \overline{x_3} \vee x_4 \vee \overline{x_5}\right).$$

A formula is 3CNF if every clause contains exactly 3 literals that are of three distinct variables. An example of a 3CNF formula:

$$\left(\overline{x_1} \vee x_2 \vee \overline{x_5}\right) \wedge \left(\overline{x_2} \vee x_4 \vee \overline{x_5}\right) \wedge \left(\overline{x_1} \vee x_4 \vee \overline{x_5}\right) \wedge \left(\overline{x_3} \vee x_4 \vee \overline{x_5}\right).$$

(A) Consider the following boolean function $f$ and $g$ defined by a truth table. Generate a 3CNF formulas that computes these two functions.

| $x$ | $y$ | $z$ | $f(x,y,z)$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

(i)

| $x$ | $y$ | $z$ | $g(x,y,z)$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

(ii)

(B) Given an arbitrary boolean formula $f(x,y,z)$, describe how to convert it into an equivalent 3CNF formula.

(C) Argue that any boolean formula with $n$ variables can be converted into a $n$CNF formula (i.e., CNF formula where every clause has at most $n$ variables).

(D) Show how to convert an $n$CNF to a 3CNF.
(As such, any boolean formula has an equivalent 3CNF formula.)

(E) Why one can not convert an $n$CNF to a 2CNF?

**12.2.** From Set Cover to Monotone SAT.

Consider an instance $I$ of a CNF formula specified by clauses $C_1, C_2, \ldots, C_k$ over a set of boolean variables $x_1, x_2, \ldots, x_n$. We say that $I$ is ***monotone*** if each term in each clause consists of a nonnegated variable i.e. each term is equal to $x_i$, for some $i$, rather than $\overline{x_i}$

(i.e., no negations are allowed). They could be easily satisfied by setting each variable to 1. For example, suppose we have three clauses $(x_1 \vee x_2), (x_1 \vee x_3), (x_3 \vee x_2)$. These could be satisfied by setting all three variables to 1, or by setting $x_1$ and $x_2$ to 1 and $x_3$ to 0. Given a monotone instance of CNF formula, together with a number $k$, the problem Monotone Satisfiability asks whether there is a satisfying assignment for the instance in which at most $k$ variables are set to 1.

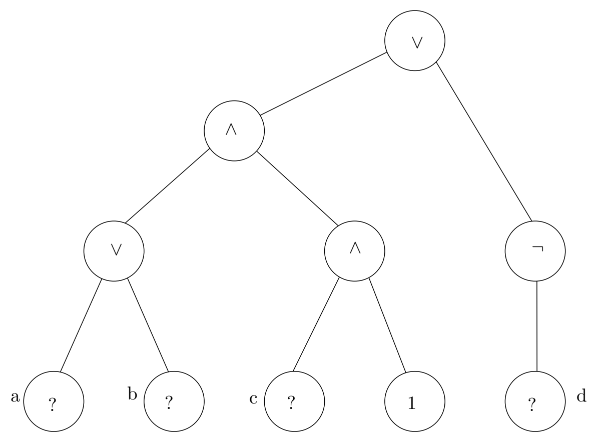
The Set Cover problem asks, given a collection $\mathcal{F}$ of subsets $S_1, S_2, \ldots, S_m$ of a ground set $U = \{1, \ldots, n\}$, what is the minimum number of sets of $\mathcal{F}$ whose union is $U$?

(A) Given a decision instance of Set Cover (i.e., given $S$, $\mathcal{F}$, and a $k$ – is there a cover of $U$ by $k$ subsets?), show a Karp reduction to Monotone Satisfiability.

(B) Show how to solve the optimization version of Set Cover (i.e., you are given $U, \mathcal{F}$, and you have to compute the minimum number of sets of $\mathcal{F}$ that cover the ground set) by an algorithm performing a polynomial number of calls to a solver of Monotone Satisfiability.

## 12.3. FROM CIRCUIT-SAT TO SAT.

Convert the following Circuit-SAT instance into a SAT formula such that the resulting formula is satisfiable if and only if the curcuit sat instance is satisfiable. Use $x_a, x_b, x_c, x_d$ as the variable names for the four unknowns shown in the figure. You may need additional variables.




## 12.4. REDUCING FROM 3-COLORING TO SAT.

SAT is a decision problem that asks whether a given boolean formula in conjunctive normal form (CNF) has an assignment that makes the formula true. The 3-Coloring problem is a decision problem that asks given an undirected graph $G$, can its vertices be colored with three colors, so that every edge touches vertices with two different colors? Give a polynomial time reduction from 3-coloring to 3SAT.

**Comment:** We will show an intricate reduction in lecture from 3SAT to 3-coloring which shows that the latter problem is hard.