
12

In CS, we usually use BA 2, the boolean algebra with {0,1} only.

Eg: given a BA, convert it into sum of Products

x_1	x_2	....	x_n	$f(x_1 .. x_n)$
0	0	...	0	
1	1	...	1	

Extract the rows from rows which contain 1 as the value of f.

Always require 2^n complexity

In what follows, the course will also restrict to two element BA.

Def²:

A fu: ϕ is satisfiable, iff there is a valuation

$$v : [\![\cdot]\!]_2 : var \to \{0,1\}$$

s.t,

$$v(\phi) = [\![\phi]\!]_2 = 1$$

Observation:

ϕ is unsatisfiable iff $\neg\phi$ is valid.

In general, we do not know better method than brute force checking a formula for every valuation, for solving a satisfiability problems.

Satisfiability checking problem is of wider interest from CS perspective.

There are many other problems, which can be solved efficiently iff SAT can be solved eff^{ly}.

eg:

i) Hamiltonian cycle Problem

↳ There is a cycle in the input graph which goes through each vertex exactly once.

ii) K - Clique Problem

(G, K), if G has a complete subgraph of size K

decisions problem → answers → yes / no.

efficiently solvable

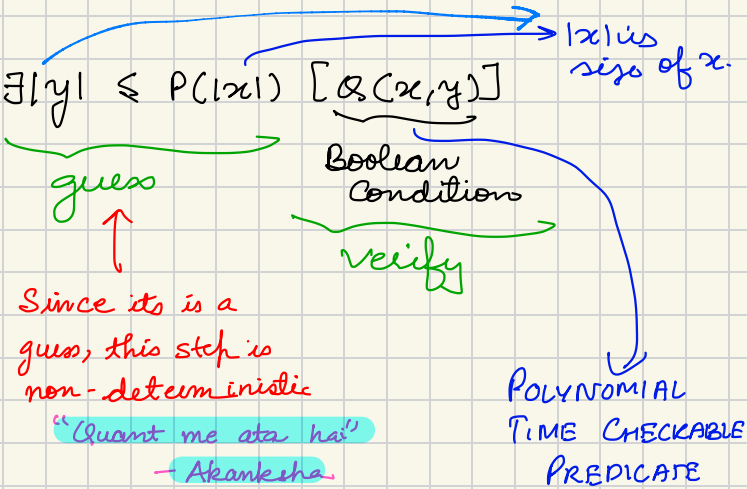
⇒ worst case complexity is polynomial in size of input.

Class NP → Polynomials verifiable in polynomial time

A problem is class NP.

If there is a polynomial P, such that

Input is x
Output is yes iff $\exists |y| \leq P(|x|)\ [Q(x,y)]$



SAT, HAM, Clique are all hardest problems in NP, in a sense.

NP complete Problem.

$$P \stackrel{?}{=} NP$$

Special cases of SAT, can be solved efficiently.

eg: Input ϕ is a DNF.

ϕ is unsatisfiable iff every disjunct D, there is a variable P, s.t.

$$P, \neg P \in D.$$

$\Rightarrow$ (proof by contrapositive)

If there is disjunct D_i, such that for no P, it has, $P, \neg P \in D_i$.

Define a valuation v, st,

$$v(i) = \begin{cases} 1 & \text{if } q \in D_i \\ 0 & \text{if } \neg q \in D_i. \end{cases}$$

v satisfies D_i, or ϕ. $\square$.

CNF Satisfiability is NP complete

3-CNF, (each clause can have at most 3 literals) is also NP complete.

CNF SAT, 3-CNF SAT are NP-Complete

DNF SAT is linear time

Horn Clause SAT is $O(n^2)$