

Problem

Show that if $P = NP$, a polynomial time algorithm exists that produces a satisfying assignment when given a satisfiable Boolean formula. (Note: The algorithm you are asked to provide computes a function; but NP contains languages, not functions. The $P = NP$ assumption implies that SAT is in P, so testing satisfiability is solvable in polynomial time. But the assumption doesn't say how this test is done, and the test may not reveal satisfying assignments. You must show that you can find them anyway. Hint: Use the satisfiability tester repeatedly to find the assignment bit-by-bit.)

Step-by-step solution

Step 1 of 2

- SAT or SATISFIABILITY problem probes whether there exists an interpretation of the fed-in input that satisfies (or returns TRUE) a given Boolean formula.
- $P=NP$ implies the existence of a polynomial time algorithm to test the decidability of SAT. By decidable, we mean that a Turing Machine (TM) exists for our SAT problem that either outputs ACCEPT or REJECT.
- Consider the below described algorithm:

For a TM U :

U = "on input α , where α is a Boolean formula of variables x_1, x_2, \dots, x_n ".

1. Run U on α . If our Boolean input α is not satisfiable, reject. This means that no matter what interpretations we might try to achieve for input α , our TM would never return TRUE.
2. For i from 1 to n :
3. Replace all the x_i s in α with 1 and simulate our TM U on that. For example: say, for $\alpha = 0_{x_1} 1_{x_2} 1_{x_3} 0_{x_4} 1_{x_5} 1_{x_6}$, we shall first replace $x_1(0)$ with 1 & then in the next iteration x_2 with 1, & so on.
4. If U accepts, we perform a permanent overwrite for x_i with 1, otherwise write x_i as 0.

This algorithm is deterministically in P, since both the "for loop" & "replacement of α in for loop" have polynomial running times.

[Comment](#)

Step 2 of 2

Since, $\text{polynomial} \times \text{polynomial} = \text{polynomial}$ devised algorithm is polynomial time.

[Comment](#)