Problem

Recall that NP^{SAT} is the class of languages that are decided by nondeterministic polynomial time Turing machines with an oracle for the satisfiability problem. Show that $NP^{SAT} = \Sigma_2 P$.

Step-by-step solution

Step 1 of 1

It is required to show $NP^{SAT} = \Sigma_2 P$.

1. Firstly, to show that $\Sigma_2 P \subseteq NP^{SAT}$.

• Suppose a language L is decided by Σ_2 alternating Turing machine M. Then that Turing machine performs some number of existential branching, followed by some number of universal branching.

• Consider the sub-trees of the computation path whose roots are the first universal step along the path. For each such sub-tree, M is (by definition) performing a Π_2 computation. Thus, it is deciding a language in $Co\text{-}NP$ (Note: it may possibly be a different language for each sub-tree, but it does not matter).

• Now consider a nondeterministic machine S that behaves just as M does, but replaces each of the computation sub-trees discussed above with a deterministic computation by a machine in P^{SAT} (this is feasible because $Co-NP \subseteq P^{SAT}$). Note that this machine only contains one run of existential branches, each extended with a deterministic computation that uses an SAT oracle and runs in polynomial time.

• If assume $a(n)$ be the maximum number of steps taken by the alternating machine before the universal branches start, and $p(n)$ be the maximum number of steps taken by any of the P^{SAT} machines which have substituted for the universal branching, then the running time of S is bounded by $a(n)+p(a(n))$. Note that the $p(a(n))$ term is a composition of functions, because the P^{SAT} sub procedures are computing with inputs that may be longer than n (but must be smaller than or equal to $a(n)$, since only $a(n)$ steps have been executed at the time the sub-procedures are used).

• Since a and p are both polynomials, so is their composition. Therefore, S is in NP^{SAT}.

2. Next, to show that $NP^{SAT} = \Sigma_2 P$.

• Suppose a language L is decided by a NP machine M with an SAT oracle.

• Consider the following modification of M. At each step of $M's$ computation that depends on the result of an oracle query, replace the step with a nondeterministic "split" that guesses the answer to the query. Each branch of the "split" also writes down the queried formula and answer obtained. Thus, at every "leaf" node of the computation tree (whenever the computation terminates), the tape contains a record of all queries and answers which lead to that leaf. Finally, in order to preserve the proper accepting behavior of M, replace each accepting "leaf" of M with a computation that checks that all the oracle.

• This last computation can be done purely with a run of existential branches followed by a run of universal branches. First, all the positively answered queries are checked by running an NP machine that decides SAT. Second, all of the negatively answered queries are checked by running a $Co\text{-}NP$ machine that decides $UNSAT$.

• Finally, we check that the machine we have produced is indeed in $\Sigma_2 P$. Clearly, the steps have been ordered so that all of the existential branches precede all of the universal branches. Now it is just required to check that the maximum branch length is polynomial with respect to the input. There are at most a polynomial number of oracle queries and replacing each of them with a branch that writes the formula and query result only lengthens the branch by a polynomial amount. Thus the total branch lengthening caused by this replacement is only a polynomial times a polynomial. Since the "checking" step at the end is performed by a polynomial number of applications of NP and $Co\text{-}NP$ machines (each run on a formula that is at most polynomial larger than the original input), the total time taken for this "checking" step is polynomial as well. Therefore, the machine is in $\Sigma_2 P$.

Comment