10.4.4

Preliminaries

Assume a set ϕ of clauses (a formula in CNF). Consider two clauses $C = (x \vee C')$ and $D = (\neg x \vee D')$ in ϕ, where C' and D' represent the rest of the clauses C and D, respectively. The clause $(C' \vee D')$ is called a resolvent of C and D. If C and D belong to a set of clauses ϕ, then a new set of clauses ϕ' can be obtained by adding a resolvent of C and D to ϕ. (this is called a resolution step). Clearly ϕ' is satisfiable if ϕ is. Note that the correctness of each resolution step, i.e. whether ϕ' was obtained from ϕ by adding a resolvent, can be verified in time polynomial in $|\phi'|$.

We know that a set of clauses ϕ is unsatisfiable iff it has a finite resolution sequence $(\phi_0, \ldots, \phi_n)$ such that (a) $\phi_0 = \phi$, (b) ϕ_{i+1} is obtained from ϕ_i by adding to ϕ_i a resolvent of ϕ_i for $i = 0, \ldots, n-1$; and (c) ϕ_n contains the empty clause.

The polynomial resolution conjecture

We say that a set of clauses ϕ has resolution of depth n if it has a resolution sequence $(\phi_0, \ldots, \phi_n)$. The polynomial resolution conjecture states that any unsatisfiable expression has resolution depth polynomial in the size of the expression.

The problem

Show that the polynomial resolution conjecture implies **NP** = **coNP**.

A solution

Take an expression (a set of clauses) ϕ of length n. Then, in case ϕ is unsatisfiable, ϕ has resolution sequence $(\phi_0, \ldots, \phi_{n^k})$ for some fixed k. Construct a nondeterministic polynomial time Turing machine M that, given ϕ as input, (i) first guesses $(\phi_0, \ldots, \phi_{n^k})$, (ii) then (deterministically) checks that $(\phi_0, \ldots, \phi_{n^k})$ is a valid resolution sequence, and (iii) finally says "yes" if this was the case ("no" otherwise). Clearly M says "yes" iff ϕ is unsatisfiable. Therefore the complement of SAT, $\overline{\text{SAT}}$, is in **NP**.

The fact that $\overline{\text{SAT}} \in$ **NP** means that **coNP** $\subseteq$ **NP** since $\overline{\text{SAT}}$ is complete for **coNP** (and both **NP** and **coNP** are closed under reductions).

Now take any language $L \in$ **NP**. We can reduce L to SAT since SAT is **NP**-complete. Since $\overline{\text{SAT}} \in$ **NP**, SAT $\in$ **coNP** by the definition of complement classes and therefore $L \in$ **coNP**. Thus **NP** $\subseteq$ **coNP**.

To sum up, if the polynomial resolution conjecture were true, **NP** = **coNP**.

A note

The polynomial resolution conjecture was disproved by Haken in Theoretical Computer Science 39:297–308, 1985.

10.4.6

Preliminaries

A strong non-deterministic Turing machine (SNTM) is one that has three possible outcomes (halting states): "yes", "no" and "maybe". We say that such a machine decides a language L if this is true: If $x \in L$, then all computations end up with "yes" or "maybe", and at least one with "yes". If $x \notin L$, then all computations end up with "no" or "maybe", and at least one with "no".

The problem

Show that L is decided by a strong non-deterministic machine in polynomial time if and only if $L \in \mathbf{NP} \cap \mathbf{coNP}$.

A solution

(The "only if" direction) Let L be a language that is decided in polynomial time by an SNTM M. Now L belongs to **NP** because we can build a normal non-deterministic Turing machine (NTM) $M_{\mathbf{NP}}$ by replacing each "maybe" halting state by "no". Now $M_{\mathbf{NP}}$ clearly decides L polynomial time i.e. $L \in \mathbf{NP}$. Similarly, L belongs to **coNP** since we can build a normal NTM $M_{\mathbf{coNP}}$ by (i) swapping the halting states "no" and "yes" and (ii) replacing each "maybe" halting state by "no". Now clearly $M_{\mathbf{coNP}}$ decides the complement of L in polynomial time meaning that $L \in \mathbf{coNP}$. Therefore if L is decided by an SNTM, then L belongs to $\mathbf{NP} \cap \mathbf{coNP}$.

(The "if" direction) Let $L \in \mathbf{NP} \cap \mathbf{coNP}$. Let $M_{\mathbf{NP}}$ be a normal NTM that decides L in polynomial time and $M_{\mathbf{coNP}}$ a normal NTM that decides the complement of L in polynomial time. Construct M that is the "cross product" of $M_{\mathbf{NP}}$ and $M_{\mathbf{coNP}}$, executing both machines synchronously in parallel. Define the halting states of M in the following way: (i) if the $M_{\mathbf{NP}}$-part of M says "yes", then M says "yes" (ii) if the $M_{\mathbf{coNP}}$-part of M says "yes", then M says "no" and (ii) both parts say "no", then M says "maybe". Note that items (i), (ii) and (iii) are disjoint when a single computation path of M is considered. M clearly decides L since if $x \in L$, then there is at least one path in M in which the $M_{\mathbf{NP}}$ part says "yes" and therefore M also says "yes"; and furthermore, M cannot say "no" because $M_{\mathbf{coNP}}$ cannot say "yes" for $x \in L$. Symmetric argumentation holds for the case $x \notin L$. Therefore, if $L \in \mathbf{NP} \cap \mathbf{coNP}$, then there is an SNTM that decides L.

As conclusion, L is decided by a SNTM iff $L \in \mathbf{NP} \cap \mathbf{coNP}$.