Reminder

- Oracles: $\text{pSAT}$ (resp. $\text{NP}_{\text{SAT}}$) is the class containing all languages decidable by a poly-time TM (resp. NTM) with access to a SAT oracle, that given a formula $\phi$, answers if $\phi \in \text{SAT}$.
- $\text{pSAT} \equiv \text{NP}_{\text{SAT}}$, $\text{NP}_{\text{SAT}} \equiv \text{coNP}_{\text{SAT}}$.
- $\text{pTQBF} = \text{NP}_{\text{TQBF}} (= \text{PSPACE})$.
- $\text{BPP}$ is the class of languages decidable by a probabilistic TM with error at most $1/3$. If $L \in \text{BPP}$, then there exists a probabilistic TM $M$ such that the following holds for every $x \in \Sigma^*$.
  - $x \in L \implies \Pr[M \text{ accepts } w] \geq 2/3$.
  - $x \notin L \implies \Pr[M \text{ accepts } w] \leq 1/3$.

The above probabilities are over the coin tosses of the machine $M$.

Example: MIN-FORMULA is in $\text{coNP}_{\text{SAT}}$

Recall the a Boolean formula is minimal if no shorter Boolean formula is equivalent to it. We defined

$$\text{MIN-FORMULA} = \{ \langle \phi \rangle \mid \phi \text{ is a minimal Boolean formula} \}.$$ 

Show that $\text{MIN-FORMULA} \in \text{coNP}_{\text{SAT}}$. 
Solution

We will show that $\text{MIN-FORMULA} \in \text{NP}^\text{SAT}$. Let

$$\text{EQ}_{\text{BF}} = \{ \langle \phi_1, \phi_2 \rangle \mid \phi_1 \text{ and } \phi_2 \text{ are equivalent Boolean formulas} \}.$$.

It holds that $\text{EQ}_{\text{BF}} \in \text{coNP}$: nondeterministically, guess an assignment on which the two formulas have different values. So, $\text{EQ}_{\text{BF}} \in \text{NP}$.

Here is a $\text{NP}^\text{SAT}$ algorithm for $\text{MIN-FORMULA}$:

“On input $\langle \phi \rangle$:

1. Nondeterministically, guess a Boolean formula $\phi'$ that is shorter than $\phi$.
2. Ask the SAT oracle if $\langle \phi, \phi' \rangle \in \text{EQ}_{\text{BF}}$.
3. If the oracle answers “no”, namely that $\phi$ and $\phi'$ are equivalent, accept.
4. Otherwise, reject.”

Example: Randomized Algorithm for 3SAT

Assume you are given a 3CNF Boolean formula with $m$ variables and want to find a satisfying assignment. The naïve brute-force algorithm will go over all possible $2^m$ assignments until it finds a satisfying one. Its running time is about $O(2^m)$ (ignoring the time for testing if an assignment actually satisfies the formula).

We will show a randomized algorithm, due to Schöning, that runs in time roughly $O(1.78^m)$ and finds a satisfying assignment, if exists, with high probability, say 2/3.

“On input $\langle \phi \rangle$:

1. Repeat for $O(1.78^m)$ times:
   (a) Choose at random an assignment $A$.
   (b) Repeat for $m/2$ times:
      i. If $A$ satisfy $\phi$, output $A$.
      ii. Otherwise, choose a clause that is not satisfiable by $A$.
         Choose at random one literal $\ell$ in that clause.
         Change $A$ so that $A(\ell)$ is the opposite to before.”

2. Answer “$\phi$ not satisfiable”.

Analysis. If $\phi$ is not satisfiable, the algorithm will answer correctly. Assume that $\phi$ is satisfiable and let $B$ be a satisfying assignment for it.

Consider the inner loop in which the algorithm chooses a random assignment $A$. How far is $B$ from $A$ when $A$ was first chosen? about $m/2$-far. Let assume that is exactly $m/2$-far, namely that $A$ and $B$ assign different values to $m/2$ variables. In order for the algorithm to transform $A$ into $B$ it needs to change all of those variables. When the algorithm chooses an unsatisfiable clause, $A$ and $B$ disagree on at least one of its variables. So, with probability at least $1/3$ — the probability that the algorithm changes $A$ on that variable — the new assignment $A$ is closer to $B$.

In picture, it looks like this:
The algorithm runs the inner loop for $m/2$ steps, and in each step, it wants $A$ to get closer to $B$. This happens with probability of at least $(1/3)^{m/2}$ and in this case the algorithm finds $B$.

If we run the loop for roughly

$$\frac{1}{(1/3)^{m/2}} = \sqrt{3}^m \approx 1.78^m$$

times, the algorithm finds $B$ with high probability (see Remark #1 below).

Recall that we assumed that $A$ is $m/2$-far from $B$. This might not be the case, but from chernoff bound we know that it is very close to being $m/2$-far. To handle this we might want to run the inner loop for say $3m/4$ times.

**Remark #1.** If there is an event that happens with probability $p$, then when (independently) repeating the experiment for $1/p$ times, the event is likely to happen (will happen with constant probability). If we repeat it for say $4/p$ times, it is very likely to happen.

Here is an example: Say you have an honest die (or dice) and you are interested in the event that the number “1” is shown on the top after you roll it. The probability that this event happens is $1/6$. What is the probability that this event happens if you roll the die for 6 times?

$$\Pr[\text{“1” is shown on the top in one of the 6 rolls}] = 1 - \Pr[\text{“1” is not shown on the top in all of the 6 rolls}] = 1 - (5/6)^6 \approx 0.665.$$ 

So, we improved the probability to see “1” on the top to almost $2/3$. What if we repeat for $4 \cdot 6 = 24$ times?

$$\Pr[\text{“1” is shown on the top in one of the 24 rolls}] = 1 - \Pr[\text{“1” is not shown on the top in all of the 24 rolls}] = 1 - (5/6)^{24} \approx 0.987.$$ 

By repeating the experiment for $4 \cdot 6$ times, we get probability that is very close to one.

**Remark #2.** The above algorithm can actually be derandomized to a deterministic one with the same running time: instead of one assignment the algorithm will start with the all true assignment $A$ and the all false assignment $A'$. One of them is definitely $m/2$-far from $B$.

**Remark #3.** The above is a simplified version of Schöning’s actual algorithm whose running time is roughly $O((4/3)^m)$. 