18.404 Recitation 11

Nov 20, 2020

Today's Topics

- Correction: NOT-STRONGLY-CONNECTED $\in$ NL
- Prove: $EQ_{REX} \in$ PSPACE
- $P^{TQBF} = NP^{TQBF}$
- $P^A \neq NP^A$
- Prove: MIN-FORMULA $\in coNP^{SAT}$
- Review: BPP
- $P \subseteq BPP$, $BPP \subseteq PSPACE$

Correction: NOT-STRONGLY-CONNECTED $\in$ NL

Recall: Trying to show STRONGLY-CONNECTED $\in$ NL

(path exists from every node to every other node in directed graph)

Show: NOT-STRONGLY-CONNECTED $\in$ NL = coNL

NOT-STRONGLY-CONNECTED = “On input G,

1. nondet. guess two vertices u,v
2. Return NOT-PATH(G, u, v)”

Note: NOT-PATH is in coNL = NL, so can invoke it in NL TM.

Prove: $EQ_{REX} \in$ PSPACE

Definition: $EQ_{REX} = \{ \langle R_1, R_2 \rangle \mid$ where R_1 and R_2 are equivalent reg. exprs $\}$

Proof: Show $\neg EQ_{REX} \in$ NPSPACE $=$ PSPACE $\rightarrow$ we can negate the result

M = "On input $\langle R_1, R_2 \rangle$

1. Convert R_1 and R_2 to equivalent NFAs N_1 and N_2 having m_1 and m_2 states
2. Nondet. guess the symbols **one-by-one** of a string s of length $2^{m1 + m2}$ and simulate N_1 and N_2 on s, storing only the **current** sets of states of N_1 and N_2
3. If they ever disagree on acceptance, then *accept*
4. If they always agree on acceptance then *reject*"

$P^{TQBF} = NP^{TQBF}$

Satement: $NP^{TQBF} \subseteq NPSPACE = PSPACE \subseteq P^{TQBF}$

First: $NP^{TQBF} \subseteq NPSPACE$

- Any time TQBF oracle is invoked, NPSPACE TM can simply compute that result

Second: NPSPACE = PSPACE Savitch's Theorem

Third: $PSPACE \subseteq P^{TQBF}$

- Reduce any PSPACE language to TQBF and ask the oracle

$P^A \neq NP^A$

Idea: Force a search of the oracle’s language that is proveably not polynomial

For oracle A, define L $= \{$ strings w $\mid \exists x \in A$ s.t. $|x| = |w| \}$

Note: L $\in NP^A$

Construct A such that L $\notin P^A$

The oracle A does not return the x that works within polynomial amount of steps. This force of search is means that L must return a result after a polynomial number of steps.

If L accepts, A shall never include x. If L rejects, then in some exponential number of steps in the future, it should return x. Thus L cannot determine if w is in the language correctly in a polynomial number of steps

Implications: $P^{TQBF} = NP^{TQBF}$ *but* $P^A \neq NP^A$

If P = NP were to be proven by some procedural construction such as Savitch's Theorem showed PSPACE = NPSPACE

→ Then for every oracle X applied, $P^X = NP^X$

However, showed that an oracle A exists such that $P^A \neq NP^A$

- This means cannot show P=NP via a direct construction.
- Would need to prove via non-relivitizable methods such as arithmetization.

Currently, expectation is overwhelmingly P ≠ NP for this reason.

Prove: MIN-FORMULA $\in$ coNP^{SAT}

Definition: $EQ_{BF} = \{<\varphi_1, \varphi_2> \mid \varphi_1$ and φ_2 are equivalent boolean formulas)

$EQ_{BF} \in$ coNP because $\neg EQ_{BF} \in$ NP simply

Proof for $\neg$MIN-FORMULA $\in$ NP^{SAT}

M = "On input $<\varphi>$

1. Nondet. guess boolean formula φ' that is shorter than φ
2. Ask SAT oracle if $<\varphi,\varphi'> \in \neg EQ_{BF}$ (reduce $\neg EQ_{BF}$ problem to SAT problem)
3. If oracle answers "no", namely that φ and φ' are equivalent, so *accept*
4. Otherwise, *reject*"

Review: BPP

BPP = { A | exists a poly-time Probabilistic TM that decides A with error ϵ = 1/3 }

or $\epsilon < 1/2$

Amplification Lemma: If M_1 is a poly-time PTM with error ϵ_1 = 1/3 then,

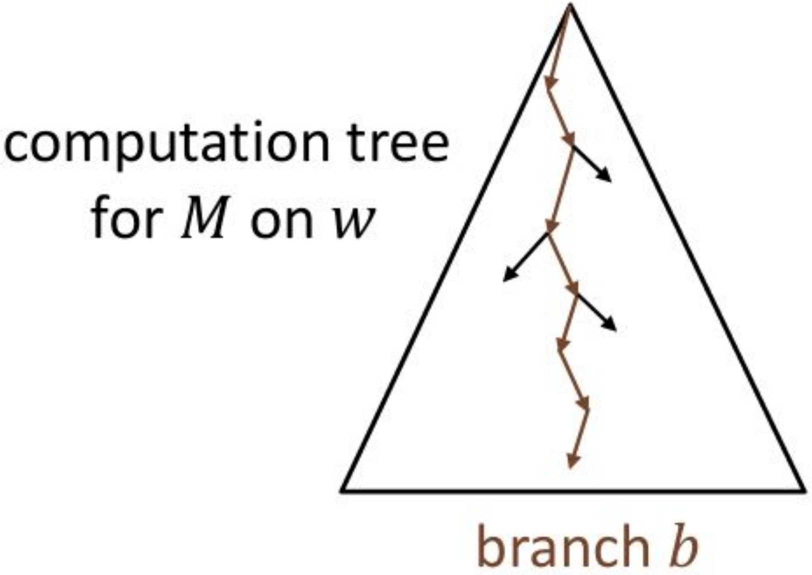
for any 0 < ϵ_2 < 1/2, there is an equivalent poly-time PTM M_2 with error ϵ_2

Can strengthen to make $\epsilon_2 < 2^{-1 * \text{poly}(n)}$

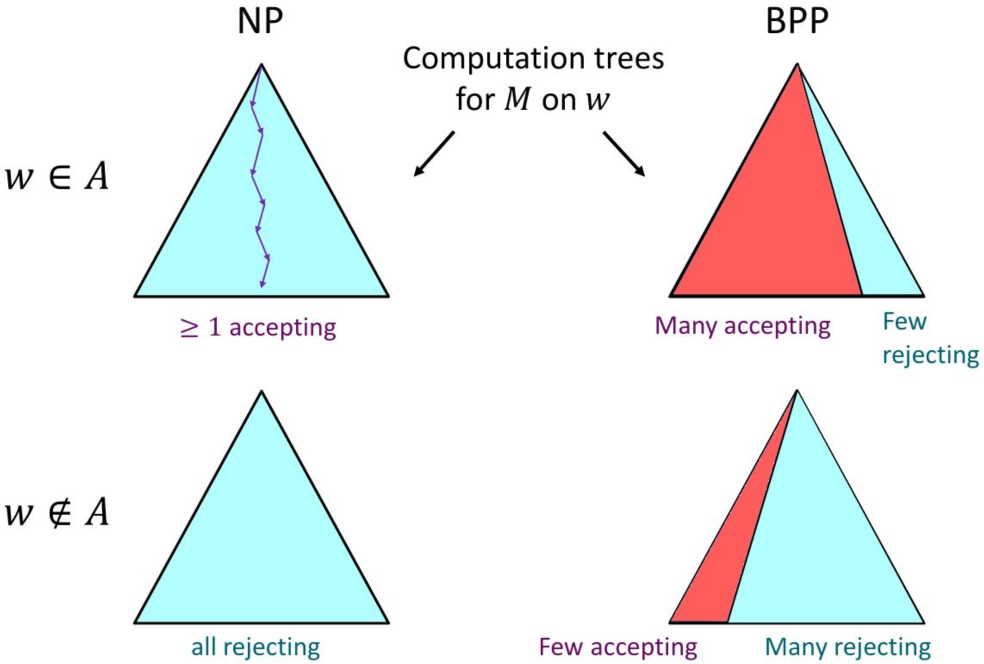
Run M_1 k times and return majority result which reduces error probability

Significance: Can make the error probability arbitrarily small (never 0 however!)

Review: BPP



Review: BPP


NP
BPP
Computation trees for M on w
$w \in A$
≥ 1 accepting
Many accepting
Few rejecting
$w \notin A$
all rejecting
Few accepting
Many rejecting

$P \subseteq BPP$, $BPP \subseteq PSPACE$

$P \subseteq BPP$

- Statement: a BPP TM can decide all languages in P

$BPP \subseteq PSPACE$

- Statement: a PSPACE TM can decide all languages in BPP