18.404 Recitation 8

Overview: 1. Relationships between classes (e.g. P, NP, PSPACE...)

2. $TQBF \in PSPACE$

3. $LADDER_{DFA} \in NPSPACE$

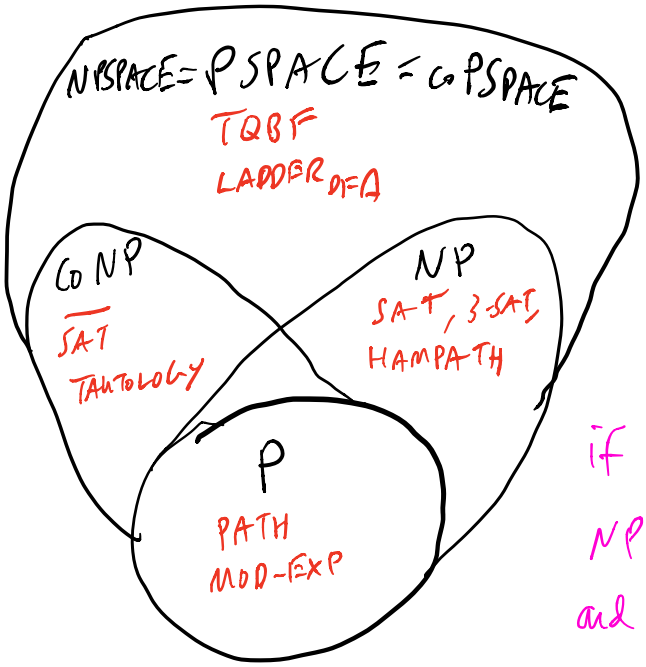
4. $LADDER_{DFA} \in PSPACE$

> Savitch's Theorem

PSPACE = NPSPACE

Relationships between P, NP, coNP, PSPACE, coPSPACE, NPSPACE

EXPSPACE $\neq$ P
EXPTIME



P = NP ?

P = PSPACE ?

if P $\neq$ NP,
NP = PSPACE ?
and so on

- PSPACE = coPSPACE.
  - PSPACE is deterministic.
  - So everything in coPSPACE is decidable w/ polynomial space deterministically.
- NPSPACE = PSPACE
  - Not obvious. Savitch's Theorem.

TQBF

Def: quantified Boolean formula (QBF) is a Boolean formula w/ quantifiers ($\exists$, $\forall$).
All variables in the formula must be quantified.

QBF is true or false.

Def: $TQBF = \{ \langle \phi \rangle \mid \phi$ is QBF that is True $\}$.

Ex: $\exists x [x]$. is this true / false?

True. Set x = true.

$\forall x [x]$ is this true / false?

False. Set x = false $\rightarrow$ false result.

$\exists x_1 \; \forall x_2 \; \exists x_3 \; \exists x_4 \ldots$

$\phi = \forall x \exists y [(x \vee y) \wedge (\bar{x} \vee \bar{y})]$

x = True, pick y = False $\rightarrow$ T

x = False, pick y = True $\rightarrow$ T

$\phi \in TQBF$

Theorem: TQBF $\in$ PSPACE.

idea: use recursion

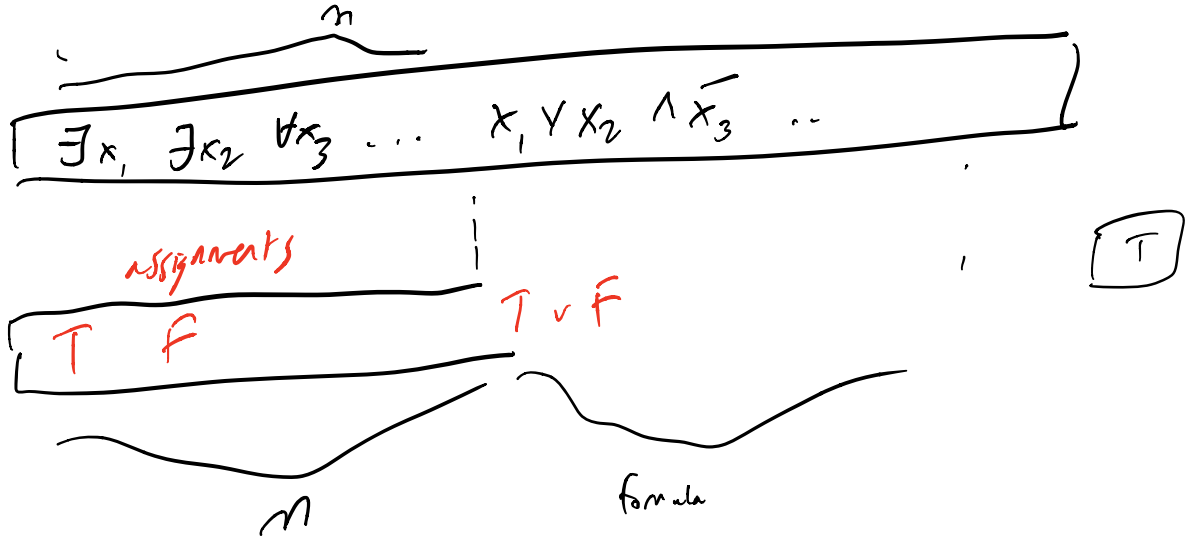
- Suppose $\phi = \exists x \, \psi$
  - just try setting x = True in ψ and x = False in ψ. If either evaluates to true, then ϕ is true, so accept. Otherwise reject.
- Suppose $\phi = \forall x \, \psi$.
  - try setting x = True, x = False. If both evaluate to true, accept, otherwise, reject.

- Recurse on ψ.
  - we set x to a boolean value.
  - ψ is just another QBF, but now it has x set.
  - So we can pass in ψ (with x set) back into our formula.

- If ϕ has no quantifiers (so it has no variables), then it's just a boolean value so output that value.

Space complexity analysis:

When we do the recursion, we write on the same space (so don't copy the formula to new space). |input| = n.



$LADDER_{DFA} \in NPSPACE$

Def: $LADDER_{DFA} = \{\langle B, u, v \rangle \mid B \text{ is DFA}, L(B) \text{ has the ladder } y_1 \cdots y_k, \text{ where } y_1 = u, y_k = v\}$.

Proof: Here's an algorithm:

1. Starting string $y = u$. Let $n = |u|$.
2. Repeat at most $|\Sigma|^n$ times.
   - 2a. Nondeterministically change y at 1 character.
   - 2b. Check if $y = v$. If so, accept.
   - 2c. Check if $y \notin L(B)$. Reject if $y \notin L(B)$.
3. Reject.

This uses linear space. $LADDER_{DFA} \in NPSPACE$.

$LADDER_{DFA} \in PSPACE$

Proof idea: Do recursion, but less recursion than a naive way by using binary search.

Subproblem: $BOUNDED\text{-}LADDER_{DFA}$

$= \{ \langle B, u, v, b \rangle \mid B$ is a DFA, there is a word ladder of length $\leq b$ from u to $v \}$.

Claim: B-L can be solved w/ binary search.

Recursion depth: $O(\log b)$.

At every level of recursion, we use $O(n)$

Decide $LADDER_{DFA}$, for a given input $\langle B, u, v \rangle$, by passing in $\langle B, u, v, |\Sigma|^n \rangle$ into B-L. This uses $O(n) \cdot O(\log b)$ space

$= O(n) \cdot O(\log |\Sigma|^n)$

$= O(n) \cdot O(n)$

$= O(n) \cdot O(n)$

$= O(n^2)$.

So. $LADDER_{DFA} \in PSPACE$. ∎

$NPSPACE = \bigcup_k$ languages that can be decided in $O(n^k)$ space, nondeterministically