Lecture 23: More PSPACE-Complete, Randomized Complexity
Final Exam Information

Who: You
On What: Everything through PSPACE (today)
With What: One sheet (double-sided) of notes are allowed
When: Friday, May 26 1:30PM - 4:30PM
Where: HERE, 04-270
Why: Because you’ll ace it
How: By studying

Practice final exam coming out tomorrow!
Definition: Language B is PSPACE-complete if:

1. $B \in \text{PSPACE}$
2. Every A in PSPACE is poly-time reducible to B (i.e. $B$ is PSPACE-hard)
Definition:
A **fully quantified Boolean formula** is a Boolean formula where *every* variable in the formula is quantified (\(\exists\) or \(\forall\)) at the beginning the formula. These formulas are either **true** or **false**

\[
\exists x_1 \ \exists x_2 \ \forall x_3 \ \exists x_4 \ \forall x_5 \ \forall x_6 \ldots [ F ]
\]

\[\text{TQBF} = \{ \phi \mid \phi \text{ is a true fully quantified Boolean formula} \} \]
TQBF = \{ \phi \mid \phi \text{ is a true fully quantified Boolean formula} \}

- SAT is the special case where all quantifiers are $\exists$

- TAUTOLOGY is the special case where all quantifiers are $\forall$

So, $\text{SAT} \leq_{P} \text{TQBF}$ and $\text{TAUTOLOGY} \leq_{P} \text{TQBF}$

Theorem (Meyer-Stockmeyer): TQBF is PSPACE-complete
PSPACE is a complexity class for two-player games of perfect information.

For formalizations of many popular two-player games, it is PSPACE-complete to decide *who* has a winning strategy on a game board.
TQBF as a Game

Played between two players, E and A

Given a fully quantified Boolean formula

$$\exists y \forall x \ [ \ (x \lor y) \land (\neg x \lor \neg y) \ ]$$

E chooses values for variables quantified by $$\exists$$

A chooses values for variables quantified by $$\forall$$

The game starts at the leftmost quantifier

E wins if the resulting formula evaluates to true

A wins otherwise

Who wins the above QBF?
Examples: \( \forall x \exists y \left[ (x \lor y) \land (\neg x \lor \neg y) \right] \)

E has a winning strategy

\( \exists x \forall y \left[ x \lor \neg y \right] \)

E has a winning strategy

\[ \text{FG} = \{ \phi \mid \text{Player E has a winning strategy in } \phi \} \]

**Theorem:** FG is PSPACE-Complete

**Proof:**

\[ \text{FG} = \text{TQBF} \]
The Geography Game

Two players take turns naming cities from anywhere in the world

Each city chosen must begin with the same letter that the previous city ended with

Cities cannot be repeated

Austin → Newark → Kalamazoo → Opelika

Whenever someone can no longer name any more cities, they lose and other player wins
GG = \{ (G, a) \mid \text{Player 1 has a winning strategy for geography on graph } G \text{ starting at node } a \} 

Theorem: GG is PSPACE-Complete
GG ∈ PSPACE

Want: PSPACE machine GGM that accepts \((G, a)\)

\[ \iff \text{Player 1 has a winning strategy on } (G, a) \]

**GGM**\((G, a)\): If node \(a\) has no outgoing edges, *reject*

Remove node \(a\) and all adjacent edges, getting a smaller graph \(G_1\)

For all nodes \(a_1, a_2, ..., a_k\) that node \(a\) pointed to, Recursively call **GGM**\((G_1, a_i)\).

If all the calls accept, then *reject* else *accept*

Claim: All the recursive calls accept

\[ \iff \text{Player 2 has a winning strategy!} \]
GG is PSPACE-hard

We show that $FG \leq_p GG$

We transform a formula $\phi$ into $(G, a)$ such that:

**Player E** has winning strategy in $\phi$ if and only if

**Player 1** has winning strategy in $(G, a)$

For simplicity we assume $\phi$ is of the form:

$$\phi = \exists x_1 \forall x_2 \exists x_3 \ldots \exists x_k [F]$$

where $F$ is in CNF: an AND of ORs of literals.

(Quantifiers alternate, and the last move is E’s)
\[\exists x_1 \forall x_2 \ldots \exists x_k (x_1 \lor x_k \lor x_2) \land \neg x_1 \lor \neg x_2 \lor \neg x_2 \land \ldots\]
$\exists x_1 \forall x_2 \ldots \exists x_k (x_1 \lor x_k \lor x_2) \land (\neg x_1 \lor \neg x_2 \lor \neg x_2) \land \ldots$
\[ \exists x_1 \, [ \, (x_1 \lor x_1 \lor x_1) \, ] \, \]
GG = \{ (G, a) \mid \text{Player 1 has a winning strategy for geography on graph G starting at node a} \}

Theorem: GG is PSPACE-Complete
Question:
Is Chess a PSPACE-complete problem?

No, because determining whether a player has a winning strategy takes CONSTANT time and space (OK, the constant is large...)

But generalized versions of Chess, GO, and Checkers (on $n \times n$ boards) can be shown to be PSPACE-hard
Randomized Complexity
A probabilistic TM $M$ is a nondeterministic TM where:

- Each nondeterministic step is called a **coin flip**
- Each nondeterministic step has only two legal next moves (heads or tails)

The probability that $M$ runs on a branch $b$ is:

$$\Pr[b] = 2^{-k}$$

where $k$ is the number of coin flips that occur on branch $b$
Probabilistic/Randomized Algorithms

Why study randomized algorithms?

1. They can be *simpler* than deterministic algorithms

2. They can be *more efficient* than deterministic algorithms

3. Can randomness be used to solve problems *provably* much faster than deterministic algorithms?

*This is a completely open question!*
Pr [ M accepts w ] = \sum \text{Pr [ } b \text{ ]}

b is a branch on which M on w accepts

**Theorem:** A language A is in NP if there is a nondeterministic polynomial time TM M such that for all strings w:

\[
\begin{align*}
\text{w } \in \text{ A } & \Rightarrow \text{ Pr[ M accepts w ] } > 0 \\
\text{w } \notin \text{ A } & \Rightarrow \text{ Pr[ M accepts w ] } = 0
\end{align*}
\]
**Definition.** A probabilistic TM $M$ decides a language $A$ with error $\varepsilon$ if for all strings $w$,

$$w \in A \implies \Pr[ M \text{ accepts } w ] \geq 1 - \varepsilon$$

$$w \notin A \implies \Pr[ M \text{ doesn’t accept } w ] \geq 1 - \varepsilon$$

**Theorem:** A language $A$ is in $NP$ if there is a nondeterministic polynomial time TM $M$ such that for all strings $w$:

$$w \in A \implies \Pr[ M \text{ accepts } w ] > 0$$

$$w \notin A \implies \Pr[ M \text{ accepts } w ] = 0$$
Error Reduction Lemma

Lemma: Let $\varepsilon$ be a constant, $0 < \varepsilon < 1/2$, let $k \in \mathbb{N}$. If $M_1$ has error $\varepsilon$ and runs in $t(n)$ time then there is an equivalent machine $M_2$ such that $M_2$ has error $2^{-n^k}$ and runs in $O(n^k \cdot t(n))$ time

Proof Idea:
On input $w$, $M_2$ runs $M_1$ on $w$, for $10 n^k$ independent random trials, records the answer of $M_1$ each time, and then returns the most popular answer (accept or reject)
BPP = Bounded Probabilistic P

BPP = \{ L \mid L \text{ is recognized by a probabilistic polynomial-time TM with error at most } 1/3 \} 

Why 1/3?

It doesn’t matter what error value we pick, as long as the error is smaller than 1/2.

When the error is smaller than 1/2, we can apply the error reduction lemma.