Lecture 22: PSPACE
VOTE VOTE VOTE

For your favorite course on automata and complexity

Please complete the online subject evaluation for 6.045
Final Exam Information

Who: You
On What: Everything through PSPACE
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 soon!
**Definition:** \( \text{SPACE}(s(n)) = \{ L \mid L \text{ is decided by a Turing machine with}\)
\[O(s(n))\text{ space complexity}\} \)

**Definition:** \( \text{NSPACE}(s(n)) = \{ L \mid L \text{ is decided by a } non-deterministic\)
Turing Machine with \(O(s(n))\text{ space complexity}\} \)
Theorem: Let $s : \mathbb{N} \rightarrow \mathbb{N}$ satisfy $s(n) \geq n$, for all $n$. Then every $s(n)$ space multi-tape TM has an equivalent $O(s(n))$ space one-tape TM.

The simulation of multitape TMs by one-tape TMs achieves this!

Corollary: The number of tapes doesn’t matter for space complexity! One tape TMs are as good as basically any other model!
Theorem:
NSPACE \( S(n) \) computations can also be simulated in at most \( 2^{O(S(n))} \) time steps.

\[
\text{NSPACE}(s(n)) \subseteq \bigcup_{c \in \mathbb{N}} \text{TIME}(2^c \cdot s(n))
\]

Key Idea: Think of the problem of simulating \( \text{NSPACE}(s(n)) \) as a problem on graphs.
**Def:** The configuration graph of $M$ on $x$ has nodes $C$ for every configuration $C$ of $M$ on $x$, and edges $(C, C')$ if and only if $C$ yields $C'$.

$G_{M,x}$

$M$ has space complexity $S(n)$  
$\Rightarrow G_{M,x}$ has $2^{c \cdot S(|x|)}$ nodes

$M$ is deterministic  
$\Rightarrow$ every node has outdegree $\leq 1$

$M$ is nondeterministic  
$\Rightarrow$ some nodes may have outdegree $> 1$

To simulate a non-deterministic $M$ in $2^{O(S(|x|))}$ time: do BFS in $G_{M,x}$ from the initial configuration!
PSPACE = $\bigcup_{k \in \mathbb{N}} \text{SPACE}(n^k)$

NPSPACE = $\bigcup_{k \in \mathbb{N}} \text{NSPACE}(n^k)$
Savitch’s Theorem

Theorem: For functions $s(n)$ where $s(n) \geq n$

$$\text{NSPACE}(s(n)) \subseteq \text{SPACE}(s(n)^2)$$

Proof Try:

Let $N$ be a non-deterministic TM with space complexity $s(n)$.

Construct a deterministic machine $M$ that tries every possible branch of $N$.

Since each branch of $N$ uses space at most $s(n)$, then $M$ uses space at most $s(n)^2$...

There are $2^{2^s(n)}$ branches to keep track of!
Given configurations $C_1$ and $C_2$ of a $s(n)$ space machine $N$, and a number $t$, want to know if $N$ can get from $C_1$ to $C_2$ within $t = 2^k$ steps

**Procedure SIM($C_1$, $C_2$, $t$):**

- If $t = 1$ then *accept* iff $C_1 = C_2$ or $C_1$ yields $C_2$ within one step.
  
- If $t > 1$, then for every configuration $C_m$ of size $s(n)$, if SIM($C_1$, $C_m$, $t/2$) and SIM($C_m$, $C_2$, $t/2$) accept then return *accept*
  return *reject* if no such $C_m$ is found

$\text{SIM}(C_1, C_2, t)$ has $O(\log t)$ levels of recursion
Each level of recursion uses $O(s(n))$ additional space.
Therefore $\text{SIM}(C_1, C_2, t)$ uses only $O(s(n) \log t)$ space!
**Theorem:** For functions $s(n)$ where $s(n) \geq n$

$$\text{NSPACE}(s(n)) \subseteq \text{SPACE}(s(n)^2)$$

**Proof:**

Let $N$ be a nondeterministic TM using $s(n)$ space

Let $d > 0$ be such that the number of configurations of $N(w)$ is at most $2^d s(|w|)$

Here’s a deterministic $O(s(n)^2)$ space algorithm for $N$:

**M(w):** For all configurations $C_{\text{acc}}$ of $N(w)$ in the accept state, if $\text{SIM}(q_0, w, C_{\text{acc}}, 2^d s(|w|))$ accepts, then accept else reject

**Claim:** $L(M) = L(N)$
Theorem: For functions \( s(n) \) where \( s(n) \geq n \)
\[
\text{NSPACE}(s(n)) \subseteq \text{SPACE}(s(n)^2)
\]

Proof:
Let \( N \) be a nondeterministic TM using \( s(n) \) space
Let \( d > 0 \) be such that the number of configurations of \( N(w) \) is at most \( 2^d s(|w|) \)

Here’s a deterministic \( O(s(n)^2) \) space algorithm for \( N \):

\( M(w) \): For all configurations \( C_{\text{acc}} \) of \( N(w) \) in the accept state,
If \( \text{SIM}(q_0, w, C_{\text{acc}}, 2^d s(|w|)) \) accepts, then accept
else reject

Why does it take only \( s(n)^2 \) space?
Theorem: For functions \( s(n) \) where \( s(n) \geq n \)

\[
\text{NSPACE}(s(n)) \subseteq \text{SPACE}(s(n)^2)
\]

Proof:

Let \( N \) be a nondeterministic TM using \( s(n) \) space.

Let \( d > 0 \) be such that the number of configurations of \( N(w) \) is at most \( 2^d s(|w|) \).

Here's a deterministic \( O(s(n)^2) \) space algorithm for \( N \):

\( M(w) \): For all configurations \( C_{\text{acc}} \) of \( N(w) \) in the accept state,

- If \( \text{SIM}(q_0, w, C_{\text{acc}}, 2^d s(|w|)) \) accepts, then \( \text{accept} \)
- Else \( \text{reject} \)

\( \text{SIM} \) uses \( O(s(n) \log t) \) space to simulate \( t \) steps of \( N \).

Set \( t = 2^d s(|w|) \). Uses \( O(s(n)^2) \) space overall!
$$\text{PSPACE} = \bigcup_{k \in \mathbb{N}} \text{SPACE}(n^k)$$

$$\text{NPSPACE} = \bigcup_{k \in \mathbb{N}} \text{NSPACE}(n^k)$$

$$\text{PSPACE} = \text{NPSPACE}$$
PSPACE-complete problems
**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)

**Why poly-time?**

**Theorem:** If B is PSPACE-complete and B is in $\text{P}$ then $P = \text{PSPACE}$

**Theorem:** If B is PSPACE-complete and B is in $\text{NP}$ then $NP = \text{PSPACE}$
Definition:
A fully quantified Boolean formula is a Boolean formula where every variable in the formula is quantified (∃ or ∀) at the beginning the formula. These formulas are either true or false

∃x∃y [ x ∨ ¬y ]

∀x [ x ∨ ¬x ]

∀x [ x ]

∀x∃y [ (x ∨ y) ∧ (¬x ∨ ¬y) ]
**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, SAT $\leq_P$ TQBF and TAUTOLOGY $\leq_P$ TQBF

**Theorem (Meyer-Stockmeyer):**

TQBF is PSPACE-complete
TQBF is in PSPACE

QBF-SOLVER(\phi):

1. If \phi has no quantifiers, then it is an expression with only constants. Evaluate \phi. Accept iff \phi evaluates to 1.

2. If \phi = \exists x \psi, call QBF-SOLVER on \psi twice: first with x set to 0, then with x set to 1. Accept iff at least one call accepts.

3. If \phi = \forall x \psi, call QBF-SOLVER on \psi twice: first with x set to 0, then with x set to 1. Accept iff both calls accept.
TQBF is PSPACE-hard: Every language $A$ in PSPACE is polynomial time reducible to TQBF

We’ll outline a proof of this. The missing details aren’t necessary, but please ask questions!

For every language $A$ is in PSPACE, there is some $k$ and some deterministic TM $M$ that decides $A$ using space $\leq kn^k$

Our polynomial-time reduction will map every string $w$ to a fully quantified Boolean formula $\phi$ of $O(n^{2k})$ size that simulates $M$ on $w$
A **tableau for M on w** is an table whose rows are the configurations of M on input w

\[
\begin{array}{cccccc}
\# & q_0 & w_1 & w_2 & \ldots & w_n & \square & \ldots & \square & \# \\
\# & & & & & & & & & \\
\# & & & & & & & & & \\
\# & & & & & & & & & \\
\end{array}
\]

\[2^{O(n^k)}\]
We’ll construct a QBF $\phi$ that is true if and only if $M$ accepts $w$ of length $n$.

Let $s(n) = n^k$. Suppose $M(w)$ has $\leq 2^b s(n)$ configs.

Using two blocks of $b \cdot s(n)$ Boolean variables denoted $C$ and $D$ representing two configurations of $M$ on $w$, and integer $t = 2^k$, we’ll construct a QBF $\phi_{C,D,t}$

$\phi_{C,D,t}$ is true if and only if $M$ starting in config $C$ reaches config $D$ in $\leq t$ steps.

Then we’ll set $\phi = \phi_{C_{\text{start}}, C_{\text{acc}}, h}$, where

$h = 2^b s(n)$, upper bounds the number of configurations of $M$ on $w$ of length $n$.

$C_{\text{start}}$ = initial configuration of $M$ on $w$,
$C_{\text{acc}}$ = (unique) accepting configuration of $M$.
IDEA:

Work like Savitch’s theorem.

Guess the configuration in the “middle” of the computation, and use recursion!

\[ \phi_{C,D,t} \] will informally say:

“there exists a configuration E such that \[ \phi_{C,E,t/2} \] is true and \[ \phi_{E,D,t/2} \] is true”

Goal: If M uses \( n^k \) space on inputs of length \( n \), then our QBF \( \phi \) will have size \( O(n^{2k}) \)
If $t = 1$, then $\phi_{C,D,t}$ should look like:

$$\phi_{C,D,1} = \text{“C equals D” OR “D follows from C in a single step of M”}$$

How do we logically express “C equals D”?
Write a Boolean formula saying that the block of $b \ s(n)$ variables representing C equals the block of $b \ s(n)$ variables representing D

$$\wedge_{i=1}^{b \ s(n)} (C_i = D_i) = \wedge_i ( (C_i \lor \neg D_i) \land (\neg C_i \lor D_i) )$$

“D follows from C in a single step of M”?
Use 2 x 3 windows as in the Cook-Levin theorem, and write a CNF formula
For \( t > 1 \), let’s try to construct \( \phi_{C,D,t} \) recursively:

\[
\phi_{C,D,t} = \exists E \left[ \phi_{C,E,t/2} \land \phi_{E,D,t/2} \right]
\]

\( \exists e_1 \exists e_2 \ldots \exists e_S \quad \text{where } S = b n^k \)

**But how long is this formula??**

Every level of the recursion cuts \( t \) *in half* but roughly *doubles* the size of the formula...!

We can get around this. Modify the formula to be:

\[
\phi_{C,D,t} = \exists E \forall X,Y \left[ \left( (X,Y) = (C,E) \lor (X,Y) = (E,D) \right) \Rightarrow \phi_{x,y,t/2} \right]
\]

This folds the two recursive sub-formulas into one!
\[ \phi_{c,D,t} = \exists E \ \forall X,Y \ [ ( (X,Y) = (C,E) \lor (X,Y) = (E,D) ) \ \Rightarrow \ \phi_{x,y,t/2} ] \]

Set \( \phi = \phi_{c_{\text{start}}, c_{\text{acc}}, h} \) where \( h = 2^{b \cdot s(n)} \)

Each recursive step adds a part that is linear in the size of the configurations, so has size \( O(s(n)) \)

Number of levels of recursion is \( \log h \leq O(s(n)) \)

Therefore the size of \( \phi \) is \( O(s(n)^2) \)
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