BU CS 332 – Theory of Computation

Lecture 23:

- Savitch's Theorem
- PSPACE-Completeness
- Unconditional Hardness
- Course Evaluations

Reading:

Sipser Ch 8.1-8.3, 9.1

Mark Bun

April 27, 2020

Space analysis

Space complexity of a TM (algorithm) = maximum number of tape cell it uses on a worst-case input

Formally: Let $f : \mathbb{N} \rightarrow \mathbb{N}$. A TM M runs in space $f(n)$ if on every input $w \in \Sigma^*$, M halts on w using at most $f(n)$ cells

For nondeterministic machines: Let $f : \mathbb{N} \rightarrow \mathbb{N}$. An NTM N runs in space $f(n)$ if on every input $w \in \Sigma^*$, N halts on w using at most $f(n)$ cells on every computational branch

Space complexity classes

Let $f : \mathbb{N} \rightarrow \mathbb{N}$

A language $A \in \mathrm{SPACE}(f(n))$ if there exists a basic single-tape (deterministic) TM M that

1) Decides A, and
2) Runs in space $O(f(n))$

A language $A \in \mathrm{NSPACE}(f(n))$ if there exists a single-tape nondeterministic TM N that

1) Decides A, and
2) Runs in space $O(f(n))$

Savitch's Theorem

Savitch's Theorem: Deterministic vs. Nondeterministic Space

Theorem: Let f be a function with $f(n) \geq n$. Then $NSPACE(f(n)) \subseteq SPACE\left(\left(f(n)\right)^2\right)$.

Proof idea:

- Let N be an NTM deciding f in space $f(n)$
- We construct a TM M deciding f in space $O\left(\left(f(n)\right)^2\right)$
- Actually solve a more general problem:
  - Given configurations c_1, c_2 of N and natural number t, decide whether N can go from c_1 to c_2 in $\leq t$ steps on some nondeterministic path.
  - Design procedure CANYIELD(c_1, c_2, t)

Savitch's Theorem

Theorem: Let f be a function with $f(n) \geq n$. Then $NSPACE\big(f(n)\big) \subseteq SPACE\left(\big(f(n)\big)^2\right)$.

Proof idea:

- Let N be an NTM deciding f in space $f(n)$

M = "On input w:

1. Output the result of CANYIELD$(c_1, c_2, 2^{df(n)})$"

where CANYIELD(c_1, c_2, t) decides whether N can go from configuration c_1 to c_2 in $\leq t$ steps on some nondeterministic path

Savitch's Theorem

CANYIELD(c_1, c_2, t) decides whether N can go from configuration c_1 to c_2 in $\leq t$ steps on some nondeterministic path:

CANYIELD(c_1, c_2, t) =

1. If $t = 1$, accept if $c_1 = c_2$ or c_1 yields c_2 in one transition.
   Else, reject.
2. If $t > 1$, then for each config c_{mid} of N with $\leq f(n)$ cells:
3. Run CANYIELD($\langle c_1, c_{mid}, t/2 \rangle$).
4. Run CANYIELD($\langle c_{mid}, c_2, t/2 \rangle$).
5. If both runs accept, accept.
6. Reject.

Complexity class PSPACE

Definition: PSPACE is the class of languages decidable in polynomial space on a basic single-tape (deterministic) TM

$$\text{PSPACE} = \bigcup_{k=1}^{\infty} \text{SPACE}(n^k)$$

Definition: NPSPACE is the class of languages decidable in polynomial space on a single-tape (nondeterministic) TM

$$\text{NPSPACE} = \bigcup_{k=1}^{\infty} \text{NSPACE}(n^k)$$

Relationships between complexity classes

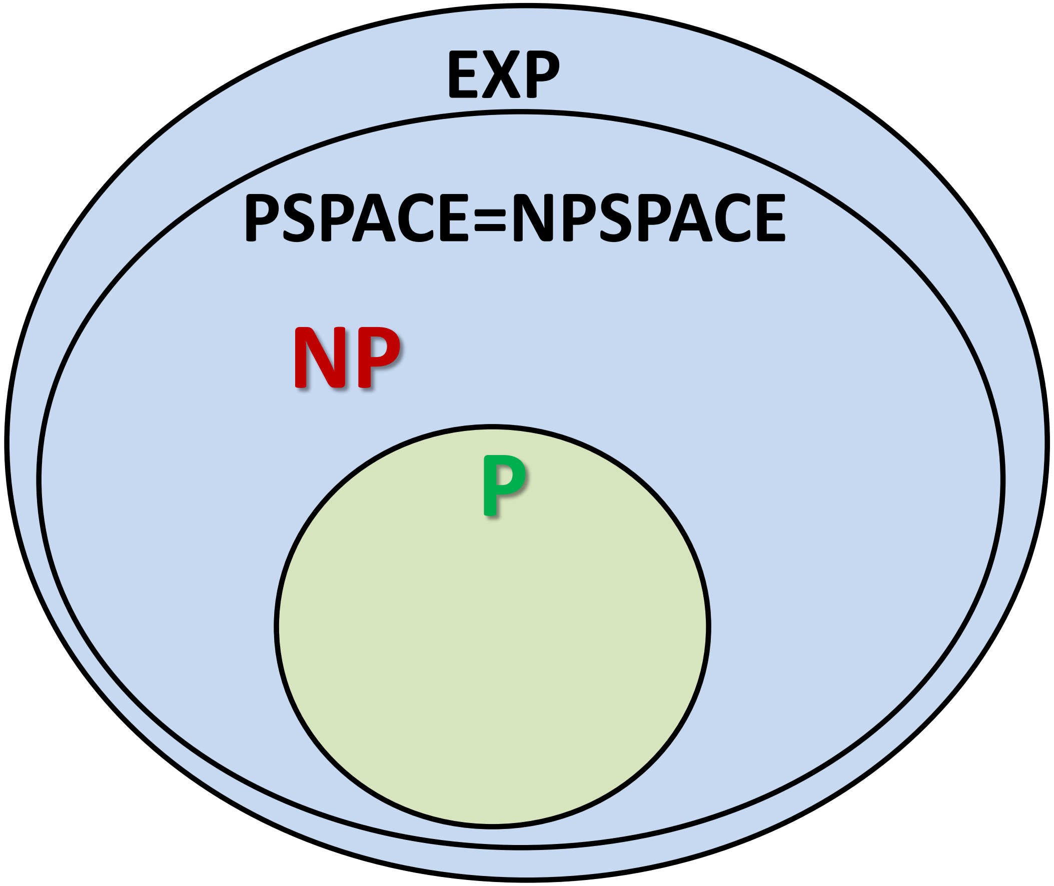
1. $\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXP}$

since $SPACE\big(f(n)\big) \subseteq TIME(2^{O(f(n))})$

2. $\mathrm{P} \neq \mathrm{EXP}$ (Monday)

Which containments in (1) are proper?

Unknown!



PSPACE-Completeness

What happens in a world where $\mathrm{P} \neq \mathrm{PSPACE}$?

Even more believable than $\mathrm{P} \neq \mathrm{NP}$, but still(!) very far from proving it

Question: What would $\mathrm{P} \neq \mathrm{PSPACE}$ allow us to conclude about problems we care about?

PSPACE-completeness: Find the "hardest" problems in PSPACE

Find $L \in \mathrm{PSPACE}$ such that $L \in \mathrm{P}$ iff $\mathrm{P} = \mathrm{PSPACE}$

Reminder: NP-completeness

Definition: A language B is NP-complete if

1) $B \in \mathrm{NP}$, and
2) Every language $A \in \mathrm{NP}$ is poly-time reducible to B, i.e., $A \leq_{\mathrm{p}} B$ ("B is NP-hard")

PSPACE-completeness

Definition: A language B is PSPACE-complete if

1) $B \in \mathrm{PSPACE}$, and
2) Every language $A \in \mathrm{PSPACE}$ is poly-time reducible to B, i.e., $A \leq_{\mathrm{p}} B$ ("B is PSPACE-hard")

A PSPACE-complete problem: TQBF

"Is a fully quantified logical formula true?"

- Boolean variable: Variable that can take on the value true/false (encoded as 0/1)
- Boolean operations: $\wedge$ (AND), $\vee$ (OR), $\neg$ (NOT)
- Boolean formula: Expression made of Boolean variables and operations. Ex: $(x_1 \vee \overline{x_2}) \wedge x_3$
- Fully quantified Boolean formula: Boolean formula with all variables quantified ($\forall, \exists$) Ex: $\forall x_1 \exists x_3 \forall x_2 \quad (x_1 \vee \overline{x_2}) \wedge x_3$
- Every fully quantified Boolean formula is either true or false

- $TQBF = \{\langle \varphi \rangle | \varphi \text{ is a true fully quantified formula}\}$

Theorem: TQBF is PSPACE-complete

Need to prove two things…

1) $TQBF \in \mathrm{PSPACE}$

2) Every problem in PSPACE is poly-time reducible to $TQBF$ ($TQBF$ is PSPACE-hard)

1) TQBF is in PSPACE

T = "On input $\langle \varphi \rangle$,
where φ is a fully quantified Boolean formula:

1. If φ has no quantifiers, it has only constants (and no variables). Evaluate φ.
   If true, accept; else, reject.
2. If φ is of the form $\exists x\ \psi$, recursively call T on ψ with $x = 0$ and then on ψ with $x = 1$.
   If either call accepts, accept; else, reject.
3. If φ is of the form $\forall x\ \psi$, recursively call T on ψ with $x = 0$ and then on ψ with $x = 1$.
   If both calls accept, accept; else, reject."

- If n is the input length, T uses space $O(n)$.

2) TQBF is PSPACE-hard

Theorem: Every language $A \in \mathrm{PSPACE}$ is poly-time reducible to $TQBF$

Proof idea:

Let $A \in \mathrm{PSPACE}$ be decided by a poly-space deterministic TM M. Using proof of Cook-Levin Theorem,

M accepts input $w \Leftrightarrow$ formula $\varphi_{M,w}$ is true

Using idea of Savitch's Theorem, replace $\varphi_{M,w}$ with a quantified formula of poly-size that can be computed in poly-time

Unconditional Hardness

Hardness results so far

- If $\mathrm{P} \neq \mathrm{NP}$, then $3SAT \notin P$
- If $\mathrm{P} \neq \mathrm{PSPACE}$, then $TQBF \notin P$

Question: Are there decidable languages that we can show are not in P?

Diagonalization redux

TM M						
M_1						
M_2						
M_3						
M_4						
$\vdots$						

Diagonalization redux

TM M	$M(\langle M_1 \rangle)$?	$M(\langle M_2 \rangle)$?	$M(\langle M_3 \rangle)$?	$M(\langle M_4 \rangle)$?		$D(\langle D \rangle)$?
M_1	Y	N	Y	Y	…	
M_2	N	N	Y	Y		
M_3	Y	Y	Y	N		
M_4	N	N	Y	N		
⋮					⋱	
D						

$\overline{SA_{\text{TM}}} = \{\langle M \rangle \mid M$ is a TM that does not accept input $\langle M \rangle\}$

$\overline{SA_{\text{TM},EXP}} = \{\langle M \rangle \mid M$ is a TM that does not accept input $\langle M \rangle$ within $2^{|\langle M \rangle|}$ steps$\}$

An explicit undecidable language

- Theorem: $L = \overline{SA_{\mathrm{TM},EXP}} = \{\langle M \rangle \mid M \text{ is a TM that does not accept input } \langle M \rangle \text{ within } 2^{|\langle M \rangle|} \text{ steps}\}$

is in EXP, but not in P

Proof:

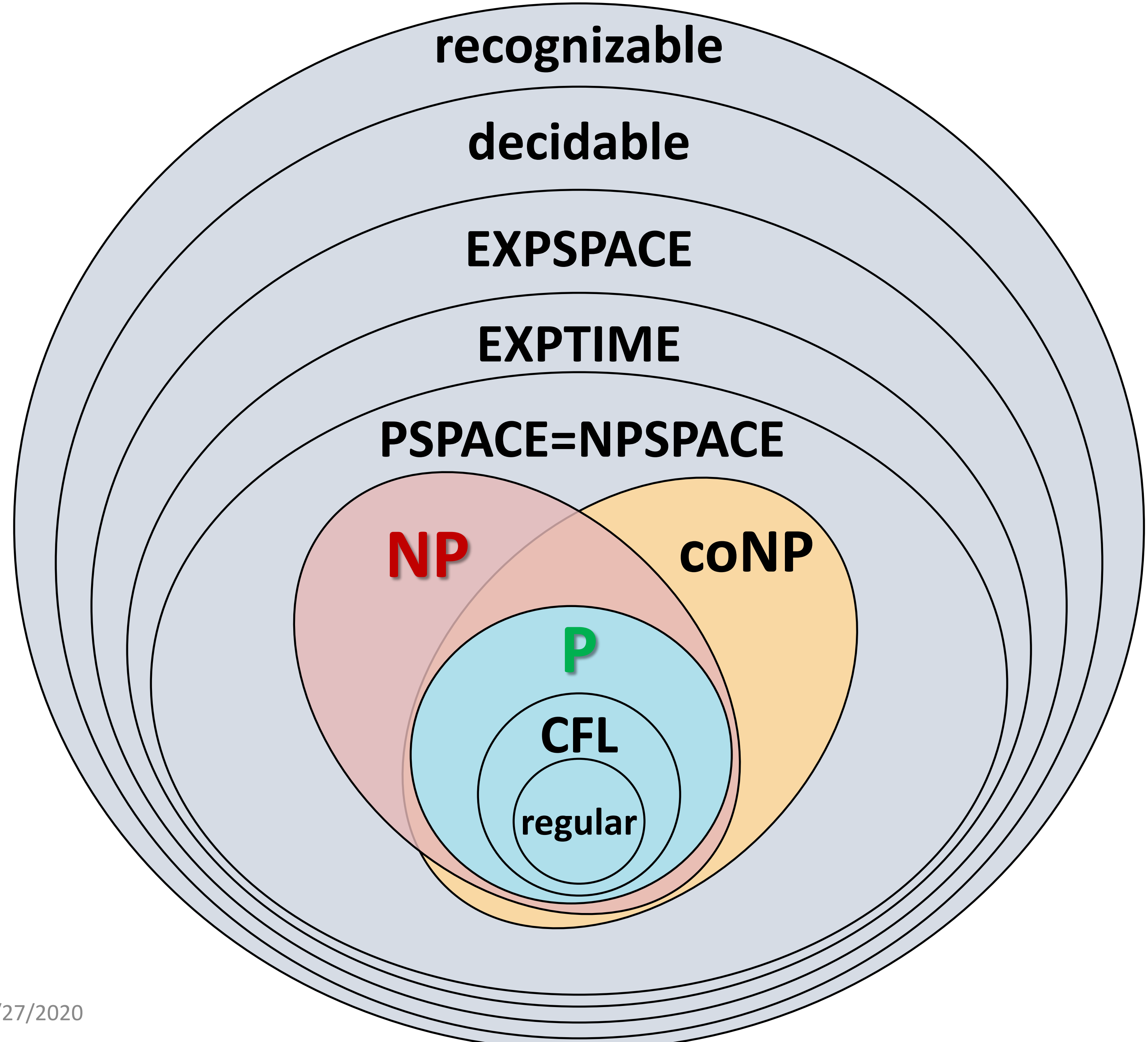
- In EXP: Simulate M on input $\langle M \rangle$ for $2^{|\langle M \rangle|}$ steps and flip its decision
- Not in P: Suppose for contradiction that D decides L in time n^k

Time and space hierarchy theorems

- For any* function $f(n) \geq n \log n$, a language exists that is decidable in $f(n)$ time, but not in $o\left(\frac{f(n)}{\log f(n)}\right)$ time.

- For any* function $f(n) \geq n \log n$, a language exists that is decidable in $f(n)$ space, but not in $o(f(n))$ space.

*time constructible and space constructible, respectively


recognizable
decidable
EXPSPACE
EXPTIME
PSPACE=NPSPACE
NP
coNP
P
CFL
regular

Course evaluations

https://bu.campuslabs.com/courseeval