Michael Forbes
miforbes@illinois.edu

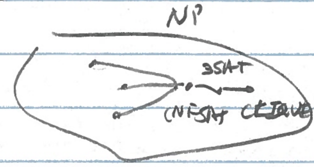
2019-01-29.4 → 2019-01-31.1
2019-01-31.2 ←
cs579

cs 579 Computational Complexity: Lecture 6

admin: ps 1 due now [late policy also]
include name/collab inside pset
ps 2 out tonight

last time: Cook Levin

NP



today: - space complexity
- PSPACE, NPSPACE

Q: what is a computational resource?
time - deterministic
- nondeterministic
space [deterministic, non deterministic]

ex

$\le t$ steps $\Rightarrow$ $\le t$ $\underbrace{\text{cells used}}_{\text{space}}$ [used in Cook Levin]

def: $s: \mathbb{N} \to \mathbb{N}$. A TM M runs in space $s(n)$ if it halts on all inputs of length n using $\le s(n)$ tape cells.
An NTM N runs in space $s(n)$ if all branches halt on all inputs "

def: $SPACE(s(n)) := \{ A : A = L(M)$, TM M runs in space $\le O(s(n)) \}$
NSPACE " N, NTM "

PSPACE = SPACE(poly(n))
NPSPACE = NSPACE(poly(n))

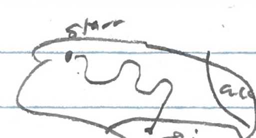
Prop: a) $TIME(t(n)) \subseteq SPACE(t(n))$
b) $SPACE(s(n)) \subseteq TIME(2^{O(s(n))}) = \bigcup_c TIME(2^{c \cdot s(n)})$

Pf: a) immediate [use same TM]
b) recall: configuration of TM

tape contents	$\le \lvert\Gamma\rvert^{s(n)}$	$\le 2^{O(s(n))}$
state	$\lvert Q\rvert$	$O(1)$
head position	$s(n)$	$\le 2^{O(s(n))}$

total: $\le 2^{O(s(n))}$

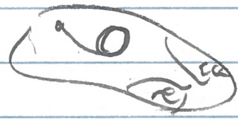
config space



Michael Forbes
mforbes@illinois.edu
2019-01-31.2 $\leftarrow$ 2019-01-31.1
$\rightarrow$ 2019-01-31.3

Clm: TM M configs repeats when computing on input x
$\implies$ M does not halt on x

Pf.



M halts on x $\implies$ no repeated configs
$\Uparrow$ $\qquad\qquad\qquad\qquad \implies$
M runs in space $S(n)$ $\implies$ $\leq 2^{O(S(n))}$ configs possible

$\leq 2^{O(S(n))}$ configurations seen

$\implies$ " time

Rmk: we did not change the TM

Q: why assume TM M halts in defn?

A: - nice assumption

- if does not halt and uses $\leq S(n)$ tape cells
  $\implies$ repeats configs, rejects input
  [could detect by listing configs — too much space] [want to change TM while still rejecting]

M' = "on input x:
1) run M for $2^{O(S(n))}$ steps
   - if acc accept
   - rej reject
   - has not halted, reject "

correctness: clear [changed TM]

complexity: [time, space, etc]
space $\leq S(n)$
always halts.

Cor: $P \subseteq PSPACE$

Prop: - $NTIME(t(n)) \subseteq NSPACE(t(n))$
- $NSPACE(s(n)) \subseteq NTIME(2^{O(s(n))})$ [same proof]

Q: NP vs PSPACE? [nondeterminism vs space]

Prop. $NP \subseteq PSPACE$

Pf: [direct proof is possible, lets do something different]

Michael Forbes
mforbes@illinois.edu
2019-01-31.2 $\rightleftarrows$ 2019-01-31.3
2019-01-31.4
CS579

lem: SAT $\in$ PSPACE

pf: "on input $\langle\varphi\rangle$:

$O(n)$ 1) for all $x \in \{0,1\}^n$ $n = \#$vars

$O(|\langle\varphi\rangle|)$ - if $\varphi(x) = 1$ <u>accept</u>

2) <u>reject</u> "

<u>correctness</u>: clear

<u>complexity</u>: $O(|\langle\varphi\rangle|)$

lem: $A \le_p B$, $B \in$ PSPACE $\Rightarrow$ $A \in$ PSPACE

pf: $x \mapsto f(x)$ M

"on input x: length $\le$ poly(n)

$\Rightarrow$ poly(n) time / space 1) compute $f(x)$

2) Run M on $f(x)$ / <u>accept</u> iff M acc"

$\le$ poly(poly(n)) $\in$ poly(n) space

def: $coNP = \{A : \bar{A} \in NP\}$

$\bar{A} = \Sigma^* \setminus A$

eg: TAUTOLOGY $= \{\langle\varphi\rangle : \varphi$ boolean formula $\forall x\ \varphi(x) = 1\}$

lem: L NP complete iff $\bar{L}$ coNP-complete

pf: - $L \in NP$ iff $\bar{L} \in coNP$

- $A \in NP$, $A \le_p L$ $\bar{A} \in coNP$ $\bar{A} \le_p \bar{L}$

$x \in A$ iff $f(x) \in L$ $x \in \bar{A}$ iff $f(x) \in \bar{L}$

[f is same reduction]

Cor: TAUTOLOGY is coNP-complete

pf: $\langle\varphi\rangle \in$ TAUTOLOGY iff $\langle\neg\varphi\rangle \notin$ SAT [encoding issues]

$\forall x\ \varphi(x) = 1$

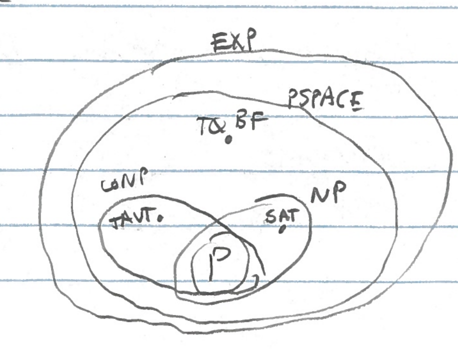
$\neg \forall x\ \varphi(x) = 1 \equiv \exists x\ (\neg\varphi)(x) = 1$

Prop: $A \in$ PSPACE iff $\bar{A} \in$ PSPACE

pf: $\hookrightarrow$ TM M $\hookrightarrow$ TM M' w/ acc/rej swapped

Cor: coNP $\subseteq$ PSPACE.

<u>the world</u>:



<u>conj</u>: $P \ne NP$
$P \ne coNP$
$P \ne PSPACE \ne NP$

Michael Forbes
mforbes@illinois.edu
2019-01-31.4 ← 2019-01-31.3
→ 2019-02-05
cs579

def: a true quantified boolean formula is an expression

$$\varphi = \underset{\exists,\forall}{Q_1} x_1 \underset{\exists\forall}{Q_2} x_2 \cdots Q_k x_k \; \psi$$

← boolean formula on $x_1, \dots, x_k$
[no free variables]

that is true.

ex: $\varphi = \forall x \; \exists y \; \forall z \; ((x \vee y) \wedge (\bar{y} \wedge z))$

is false: if $x = 0 \Rightarrow y = 1$ must hold
$\Rightarrow \forall z \; z$, which is false

TQBF = { $\langle\varphi\rangle$: φ is true quantified boolean formula }

ex: SAT, TAUT ⊆ TQBF

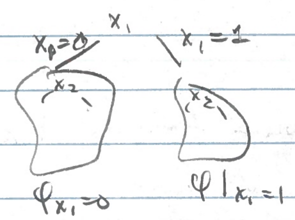
Thm: TQBF is PSPACE-complete

Prop: $\in$ PSPACE

pf: $\varphi = Q_1 x_1 Q_2 x_2 \cdots Q_k x_k \; \psi$

idea: use recursion, reusing space

eg:



space → linear in # variables
[and rewriting φ]

algo: " on input $\langle\varphi\rangle$:

1) if no quantifiers, eval φ directly [no variables]
2) if $\varphi = \exists x_1 \psi$ — solve recursively $\psi|_{x_1=0}$
   " $|_{x_1=1}$
   accept if either accept
3) $\forall x_1$ "
   accept if both accept

correctness: clear
complexity: always halts
space: $\underbrace{O(n)}_{\text{recursion stack}} + \underbrace{|\langle\varphi\rangle|}_{\text{manipulating } \varphi}$)

today: space complexity
next time: PSPACE vs NPSPACE