Michael Forbes
mforbes@illinois.edu

2019-02-07.4 2019-02-12.1
2019-02-12.2 cs579

cs579 Computational Complexity : Lecture 9 14
- Zander returning 02-21

admin: - ps 2 due thurs

last time: games, PSPACE completeness
L, NL completeness

today: NL completeness
PATH is NL complete
NL vs coNL

thm $NL \subseteq P$

Sketch: A decided by NTM N in space $O(\log n) \Rightarrow A \in P$

"on input x:

1) build configuration graph of N on x

$G = (V, E)$ $C_1 \to C_2$ under x

$O(\log n)$ space configs [$O(1)$]
- state
- read locations [two tapes]
- tape contents [only work tape]

} polynomially many

2) test if $C_{start} \to C_{acc}$ [unique accepting state]
[this is the path problem]
[is in P]

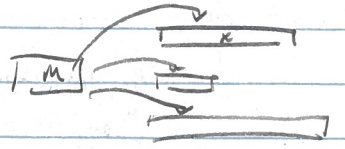
3) accept if path exists

Q.E.D.

Rmk: $\Rightarrow$ $A \leq_p$ PATH [reduction is configs graph] [is this NL-complete?]
but $\forall A \in P$ $A \leq_p$ PATH [as on ps 2 #3]
[so useless for L vs NL]

def: B NL complete if - $B \in NL$
- $\forall A \in NL$ $A \leq_L B$ [log space reduction]

def: a transducer TM M is:



M on x outputs y if output tape equals y at end of computation [rewrite]
runs in space $S(n)$ if - halts
- uses $\leq S(n)$ work tape cells

def: $A \leq_L B$ if st - f is computed by log space transducer
- $x \in A$ iff $f(x) \in B$

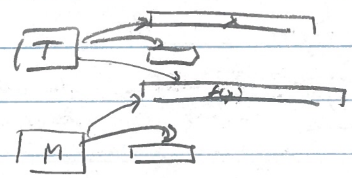
Michael Forbes
miforbes@illinois.edu
2019-02-12.2 ← 2019-02-12.1
→ 2019-02-12.3

Rmk: - log space transducer runs in poly time [poly # steps until halt]
$A \leq_L B \Rightarrow A \leq_P B$

prop: $A \leq_L B$, $B \in L \Rightarrow A \in L$ [so is helpful for L vs NL]

Pf: "on input x:
a) compute f(x) via TM T for $A \leq_L B$
b) decide if $f(x) \in B$ via TM M for $B \in L$
c) accept iff ↑ acc"

error: how to store f(x)?



[can't store f(x)]
[f(x) isn't there]

"on input x:
a) test if $f(x) \in B$
on each step | head of M is on some ith bit of f(x)
recompute from scratch using T on x and position i
[wastes time, but saves space]
b) accept iff ☒

Prop: $A \leq_L B$, $B \leq_L C \Rightarrow A \leq_L C$

Pf: same proof

thm: PATH is NL-complete

Pf: PATH ∈ NL: guess the path one vertex at a time

PATH NL-hard

$A \in NL$ decided by space $O(\lg n)$ NTM N

$x \in A$ iff $\langle G_{N,x}, c_{start}, c_{acc} \rangle \in PATH$

↳ via logspace transducer

"on input x:

$O(\lg n)$ space 1) for all $O(\lg n)$ space config c_1, c_2 [vertices V]

$O(\lg n)$ space { if $c_1 \to c_2$ in N on x in one step
output edge $(c_1, c_2) \in E$

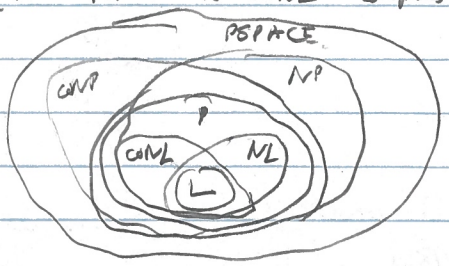
2) output V, c_{start}, c_{acc}

[no need to rewrite or look at output]
[encoding issues can be trickier w/ logspace, but still not an issue]

Michael Forbes
mforbes@illinois.edu

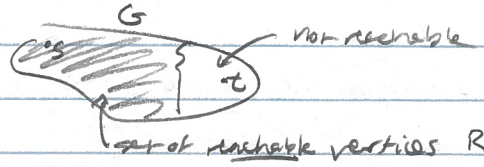
cor: $\overline{PATH}$ is coNL-complete : $A \leq_L B$ iff $\bar{A} \leq_L \bar{B}$

conj: $P \neq NP$
$NP \neq coNP$



thm [Immerman 88, Szelepcsényi 88] $NL = coNL$ [unexpected non-trivial behavior + tricky proofs]

Pf: Strategy: $\overline{PATH} \in NL$



idea: compute R, show $t \notin R$
too big to write down
didn't use nondeterminism ↔ verification/proofs

Key idea: if $|R|$ known, can prove $t \notin R$
- guess vertices in R and verify they are in R
- check you found all vertices in R
- observe t was not in list

"on input $\langle G, s, t \rangle$:

$O(\lg n)$ 1) $c = 0$ [count of things in R]
seen = true [seen t] [default is true] [some branch accepts if probably don't see t]

$O(\lg n)$ 2) for all $v \in V$

$O(1)$ a) guess if $v \in R$ [run nondeterministically]

$O(\lg n)$ b) if $v \in R$, guess path $s \rightsquigarrow v$ [$PATH \in NL$]

$O(\lg n)$ if found path: increment c
if no path: reject

$O(1)$ c) if $v \notin R$ $v = t$, seen = false

3) if $c = |R|$ [found all vertices in R]
accept if seen = false [$t \notin R$]

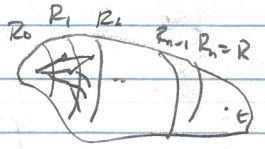
correctness: $t \notin R \Rightarrow$ some way to guess all $v \in R$ and paths $s \rightsquigarrow v$
so $c = |R|$, t not seen

$t \in R \Rightarrow$ any way of witnessing $c = |R|$ must go through $s \rightsquigarrow t$
so seen ≠ false

Michael Forbes
mforbes@illinois.edu
2019-02-12.4 ← 2019-02-12.3
→ 2019-02-14.1

complexity: $O(\lg n)$

[how $|R|$ not known]



idea: $R_i = \{ v$ reachable from s in $\le i$ steps $\}$

$R_0 = \{s\} \Rightarrow |R_0| = 1$

R_1: for any $v \in V$ can check if $v \in R_1$, given $|R_0|$

$\Rightarrow$ can compute $|R_1|$

R_2: $v \in R_2$, given $|R_1|$

...

R_n: $v \in R_n$, given $|R_{n-1}|$

$\Rightarrow$ can compute $|R_n| = |R| \to$ decide if $t \notin R$

computing $c_{i+1} = |R_{i+1}|$ from $c_i = |R_i|$

$O(\lg n)$ i) $c'_{i+1} = 0$ [counter for $|R_{i+1}|$]

$O(\lg n)$ ii) for $v \in V$ [potential $v \in R_{i+1}$]

$O(1)$ a) seen = false [have we proven $v \in R_{i+1}$]

$O(\lg n)$ b) $c'_i = 0$ [counter for R_i]

$O(\lg n)$ c) for $w \in V$ [guessing R_i]

i) guess if $w \in R_i$

$O(\lg n)$ ii) if $w \in R_i$, guess path $s \rightsquigarrow w$ in $\le i$ steps [in NL]

aa) if path

increment c'_i [found $w \in R_i$]

if $w \to v$, seen = true [proved $v \in R_{i+1}$]

bb) if no path

reject

d) if $c'_i = |R_i|$ [found R_i]

if seen = true: increment c_{i+1}

if $c'_i \neq |R_i|$: reject

3) output c_{i+1}

complexity:

correctness: as before

hence $|R_0| = 1 \to |R_1| \to \cdots \to |R_n| = |R| \to t \notin R$

↖ only store $|R_i|, |R_{i+1}|$ ⇝ PATH $\in$ NL.

today - PATH NL completeness

- $\overline{\text{PATH}} \in$ NL