AB Chap. 4, Sipser Chap. 8, Cai Chap. 3

Last time: start space complexity unit

- nondet space, Savitch's Theorem

  $f$ a p.c.f.: $NSPACE(f) \subseteq SPACE(f^2)$

AB Chap. 4.2, Papad. 19.1, Sipser 8.3, Cai 3.4

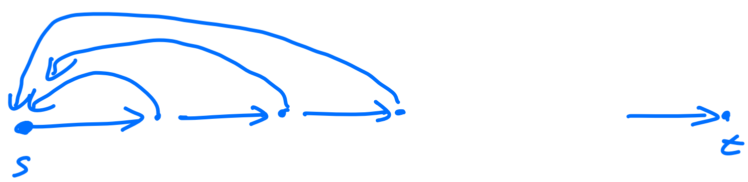
Today:
- NL, NL-completeness (under logspace reduc.)
- PSPACE, PSPACE-completeness (under poly-time reduc.)

Questions?

---

Q: Is $L = NL$? Who knows...? Probably not...?

REACH $\in$ NL, seems (?) not in L...




---

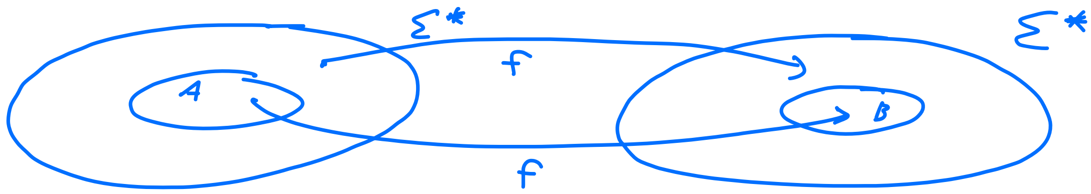
Analogue of NPC theory: NL-completeness.

REACH is NL-complete: if it's in L, then NL = L.

New notion of reduc (poly-time: too strong, since $NL \subseteq P$): logspace reducibility.

Def: Lang $A$ is logspace reducible to lang $B$ ($A \leq_L B$) means: there is a mapping $f: \Sigma^* \to \Sigma^*$, computable in logspace, s.t. $\forall x$, $x \in A \iff f(x) \in B$. does not mean $|f(x)| \leq \log(|x|)$.

(Recall: "f comp. in logspace" means worktape usage is $\leq \log n$;




---

<u>Def:</u> B is <u>NL-complete</u> if
① for every $A \in NL$, have $A \le_L B$ (NL-hard)
② $B \in NL$.

---

<u>Useful fact:</u> If $A \le_L B$ & $\underline{B \in L}$, then $A \in L$.

logspace $M_B$

<u>Pf:</u> <u>Wrong arg:</u> on input $x \stackrel{?}{=} A$,
1) run logspace f to compute f(x)
2) use $M_B$ (logspace TM deciding B) on f(x).

Not ok;


can't write down f(x).

Right arg: our machine $M_A$ computes indiv. characters of f(x) as required by $M_B$ "on the fly" as needed.

$M_A$ simulates $M_B$ on f(x), keeping track of where $M_B$'s input head would be on f(x), <u>without explicitly writing f(x).</u>

Every time $M_B$ would move input head on f(x) (to, say, $i^{th}$ char. of f(x)), $M_A$ restarts comput. of f on x from start, not actually producing output but instead incrementing a counter for each char. of f(x) that would be output.
When counter = i, have needed char. of f(x) for $M_B$.

Since f is log-space computable, on input |x|=n, have $|f(x)| \le poly(n)$, so $i \le poly(n)$ + O(log n) bits of memory is enough for counter.

---

**Thm:** REACH is NL-complete.

**Pf:** Know REACH $\in$ NL (last time); so need only show REACH is NL-hard.

Fix any $A \in NL$, let $M_A$ be logspace NTM for A.

Need logspace reduc: $x \xrightarrow{\text{logspace}} (G, s, t)$ of REACH

$G$ = config graph of $M_A$ on $x$.

Nodes of $G$: config of $M_A$ on $x$.

$(c_1, c_2)$ edge present in $G \Leftrightarrow c_2$ is a poss. next config. of $M_A$ from $c_1$.

$s = \underset{\text{unique}}{!}$ start config. of $M_A$ on $x$

$t = !$ acc " " " " " (wlog $M_A$ is "standardized")

Is this mapping logspace-computable? yes.

Machine outputs two lists (nodes of $G$, edges of $G$)

List of nodes easy: generate all of them (each node is $O(\log n)$ descrip. length) sequentially.

List of edges: go over all pairs of nodes; in logspace, easy to check, given $c_1$ + $c_2$, whether $c_2$ follows from $c_1$ under $M_A$ in one step. ∎

---

# PSPACE + PSPACE-completeness

$\varphi(x_1, \dots, x_n)$

Usual SAT problem: determine T/F of

"$\exists x_1 \exists x_2 \dots \exists x_n \, \varphi(x_1, \dots, x_n)$"

$0/1 \quad 0/1 \quad 0/1$

Up our game: generalize to expr. like

(n even)

"$\forall x_1 \exists x_2 \forall x_3 \exists x_4 \dots \exists x_n \; \varphi(x_1, \dots, x_n)$"

(altern. wLOG;

$\exists x_1 \exists x_2$

$\forall x_0 \exists x_1 \forall x_1' \exists x_2$

won't appear)

a totally quantified Bool formula

Ex $\forall x \exists y (x \vee y) \wedge (\bar{x} \vee \bar{y}) \rightarrow$ is T ✓

$x = 0$ : take $y = 1$

$x = 1$ : take $y = 0$

$\exists x \forall y (x \wedge y) \rightarrow$ is F ✗ $\quad x = 0$: no $y$

$\exists x (x \vee y)$ : not legit

---

Def: QSAT (TQBF) is

$\{ \Phi : \Phi \text{ is a true quantif. Bool. formula} \}$

---

Seem like QSAT $\notin$ NP: what's witness?

Evidence that QSAT $\notin$ NP:

**Thm:** QSAT is PSPACE-complete (under $\leq_p$).

**Pf:** Must show (I) QSAT $\in$ PSPACE
(II) every $L \in$ PSPACE is $L \leq_p$ QSAT.

(I): Here's a PSPACE alg $A$ for QSAT:

input $\Phi = \exists x_1 \forall x_2 \ldots \varphi(x_1, \ldots, x_n)$

Alg $A$:

- check all vars are quantified
- If $\Phi$ is "$\exists x\, \alpha$", recursively call $A$ twice, one on $\alpha|_{x \leftarrow 0}$, once on $\alpha|_{x \leftarrow 1}$. Reuse space used in 1st call for second call. If either call returns T, return T, else return F.
- If $\Phi$ is "$\forall x\, \alpha$", recursively call $A$ twice, one on $\alpha|_{x \leftarrow 0}$, once on $\alpha|_{x \leftarrow 1}$. Reuse space used in 1st call for second call. If both calls return T, return T, else return F.
- If $\Phi$ has no quantif. in front: it'll also have no vars (all are set to 0 or 1) - evaluate it + return truth value.

Correct. Recursion depth $= n$.
Space usage: to keep track of loc. in tree of recursive calls; + poly(n) per level of depth; poly(n) space.

II) Now: show every $L \in PSPACE$ is $L \leq_p QSAT$.

Fix $L \in PSPACE$. $M =$ $n^k$-space TM deciding $L$.
↳ (1 tape).

(• $M$'s comput. always runs in $2^{dn^k}$ time (decider!)).

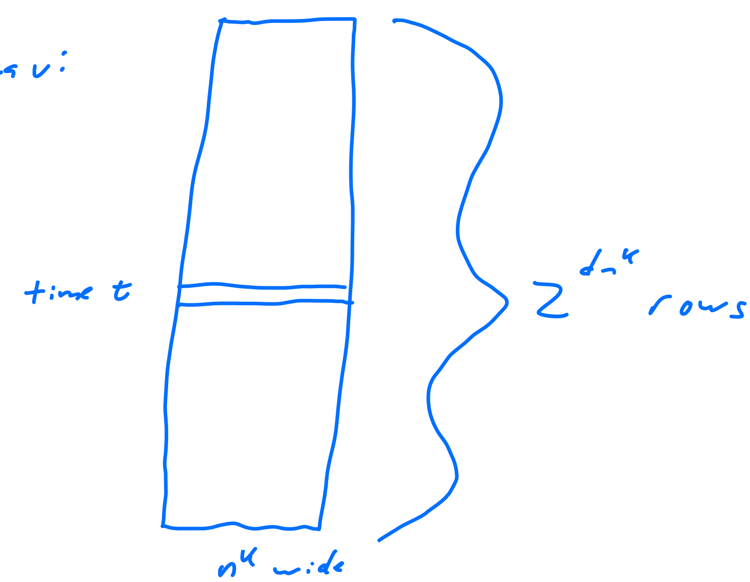
Need poly-time mapping: given $x$, outputs a quant. Bool. formula that's true iff $M$ acc. $x$.

3 ideas:

① Cook-Levin thm: computation tableau of $M$ on $x$.
Grid $T$; cell $T_{t,j}$ of $T$ is $j^{th}$ cell of config of $M$ on $x$ at time $t$ (tape cell contents, etc.)

Like in C-L thm, Bool. expressions enforce/check local consistency of tableau $T$ (ensures $T$ faithfully encodes comput. of $M$ on $x$).

Tableau:




(big!)

② Savitch's thm: recursively find midpoint of tableau.

Naively: one "$\exists$" for each midpoint/row of T: gives formula

$$\underbrace{\exists x_1 \cdots\cdots\cdots \exists}_{2^{dn^k} \text{ rows}} \varphi$$

(3) Use univ. $\forall$ quant. to (expon!) save on size of formula.

---

Finish next time.