

Today

- Space complexity wrap up
- interactive pfs
- $NL = NSPACE(\log(n)) \in P$
- Savitch's thm: $NSPACE(s(n)) \subseteq SPACE(s(n)^2)$
- Immerman - ~~Savitch's thm~~ ^{Szelepcsenyi}: $NL = co NL$

$DIRPATH = \{ \langle G, s, t \rangle \mid \exists s-t \text{ directed path in } G \}$

Thm $DIRPATH$ is NL -complete.

Pf. Clearly $DIRPATH \in NL$.

For the other direction, define LOGSPACE REDUCTION:

The ~~reduction~~ transformation

$x \mapsto (G_x, C_{start}, C_{end})$

that we saw last time in

fact can be implemented in logspace

$L' \leq L$ means $\exists$
 $f: \Sigma^* \rightarrow \Sigma^*$ s.t.

f can be computed in logspace
and $\rightarrow$ aka, work tapes have logspace

$f(x) \in L' \iff x \in L$

NOTICE that transitivity takes a bit of thought, since you can't store the output, but you can do it.

See Sipser's book for more details.

Thm 2-SAT is NL -complete.

We can also use logspace reductions to define completeness for P :

Def A is P -complete if

(1) $A \in P$

(2) $\forall B \in P, B \leq_P A$.

Thm $CIRCUIT-VALUE = \{ \langle C, x \rangle : C(x)=1, C \text{ is a circuit} \}$ is P -complete.

①

SAVITCH'S THM: $NL \subseteq SPACE(\log^2(n))$

more generally, $NSPACE(s(n)) \subseteq SPACE(s^2(n))$

(OPEN: Is there a polytime alg for, say, $DIRPATH$, that has space $\text{polylog}(n)$.)

Proof Let $L \in NSPACE(s(n))$.

We want to know if there's some path from the start configuration to the accept configuration.

As before, $|configurations| \leq 2^{O(s(n))}$

Idea: "guess the midpoint and recurse"

More formally:

- define $REACH(C_1, C_2, T) = \text{TRUE} \iff \exists \text{ path from } C_1 \text{ to } C_2$
- goal is $REACH(C_{start}, C_{accept}, 2^{O(s(n))})$ in T steps.

- $REACH(C_1, C_2, T) \iff \exists C_{mid} [REACH(C_1, C_{mid}, T/2) \wedge REACH(C_{mid}, C_2, T/2)]$

Define
 $UST-CONN = \{ \langle G, s, t \rangle \mid G \text{ undirected w/ path } s \rightarrow t \}$

This is easier than
 $ST-CONN$: hms out,
 $UST-CONN \in L$.

(Reingold's Thm: more on that later.)

Turn this into a recursive algorithm in the natural way:

for each C_{mid} ;

check both of the $REACH$ statements recursively.

(and if $T=1$, base case is easy)

The space complexity is $O(S^2(n))$:

- The recursion goes $O(s(n))$ deep
- at each level you need to store a few configurations, which is $O(s(n))$

OPEN: $RL = L$?

$NL = L$?

$NL \subseteq SPACE(\log^{1.9}(n))$?

$RL \subseteq SPACE(\log^{3/2-\epsilon}(n))$?

②

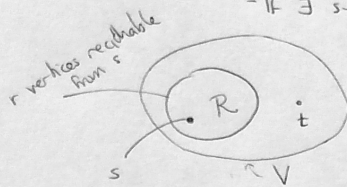
IMMERMAN-SZEPCEŃYI

THM: NL = coNL (1988)

Pr. Suffices to show ST-CONN $\in$ coNL.

Idea 1. Given $\langle G, s, t \rangle$, want an NL machine st.

- If NO s - t path, some computation path should accept
- If $\exists$ s - t path, then NO ~~computation~~ computation path should accept.



Suppose you know how many vertices r are reachable from s .

This algorithm will accept only if $t \in R$, aka, NO s - t path.

But, how do we know r ?

ATTEMPT:

$c \leftarrow 0$

for all $v \in V$:

guess if $v \in R$

if guess so:

guess a path from s
if it ends at v :

$c \leftarrow 1$

else reject

return c

counter $\leftarrow 0$
for $v \in V \setminus \{t\}$:
guess if $v \in R$, if not, continue
~~guess if $v \in R$~~
~~if it ends at v~~ guess a path of len $(n-1)$, $s \rightarrow v$
if it ends in v :
increment counter
else:
reject.
if $c = r$:
accept
else:
reject.

This algorithm might underestimate r .
But there's SOME branch, with all the correct guesses, that returns the correct count.

Instead, IDEA 2.

$R_i = \{v \mid v \text{ is reachable from } s \text{ in } \leq i \text{ steps}\}$

$R_0 = \{s\}$, $R_{n-1} = R$

define $r_i = |R_i|$, so we want $r = r_{n-1}$.

Now compute this recursively:

Given G, i, s, r_{i-1} ; return r_i

That is, we want a computation tree



Then we can hook all these trees together to get our final alg

NL = coNL ctd.

(Idea 2 ctd.)

To compute r_i :

Input: r_{i-1}, i, s, G

$c \leftarrow 0$

for all $v \in V$:

$b \leftarrow 0$; flag $\leftarrow 0$

for all $w \in V$:

guess if $w \in R_{i-1}$

if guess so:

- verify it (aka: guess a path of len $i-1$)

- if that checks out:

$b \leftarrow 1$

flag $\leftarrow$ TRUE if w

if $(w, v) \in E$ or $w = v$:

flag $\leftarrow$ TRUE

else reject.

If $b < r_{i-1}$:

reject

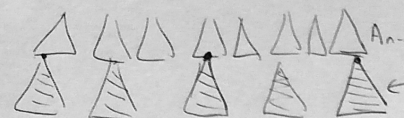
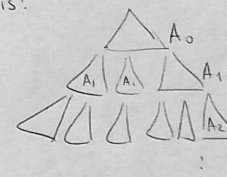
If FLAG:

$c \leftarrow 1$

return c

call this A_i

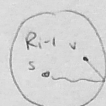
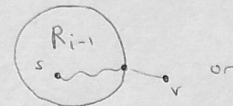
Now the final alg is:



at this level either we have rejected or we successfully computed $r = r_{n-1}$.

the computation tree from idea 1

$v \in R_i$ means:



CLAIM. In any non-rejecting branch, $c = r_i$.

Thm TQBF is PSPACE-complete.

Pf sketch. TQBF $\in$ PSPACE (last time)

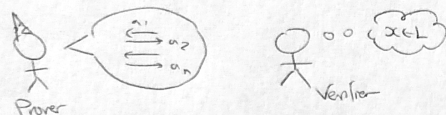
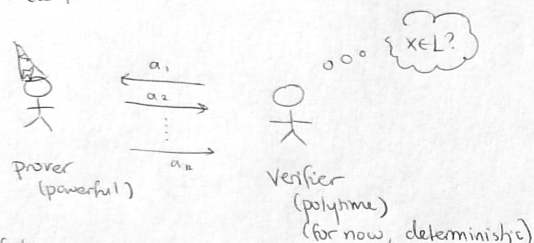
The fact that this is PSPACE-hard follows from a proof very similar to that of Savitch's theorem.

(See supplementary notes for more details).

INTERACTIVE PROOFS.

Q: Does interaction help to prove "more stuff", aka, languages beyond NP.

Model:



A: Interaction ALONE does not help.

Q2: What about a randomized verifier? (and sometimes he accepts $x \notin L$)

~~Example Graph isomorphism~~

DEF $L \in IP$ if there is a randomized protocol between Prover + Verifier st.

$$x \in L \Rightarrow \Pr\{(P \leftrightarrow V)(x) \text{ accepts}\} \geq 2/3$$

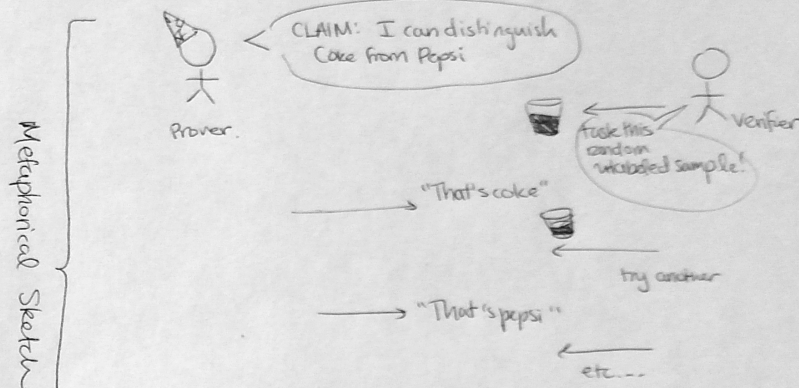
$$x \notin L \Rightarrow \Pr\{(P \leftrightarrow V)(x) \text{ accepts}\} \leq 1/3$$

← actually we can replace $2/3$ w/ 1 and not change the def

EXAMPLE: GRAPH NON ISOMORPHISM (GNI).

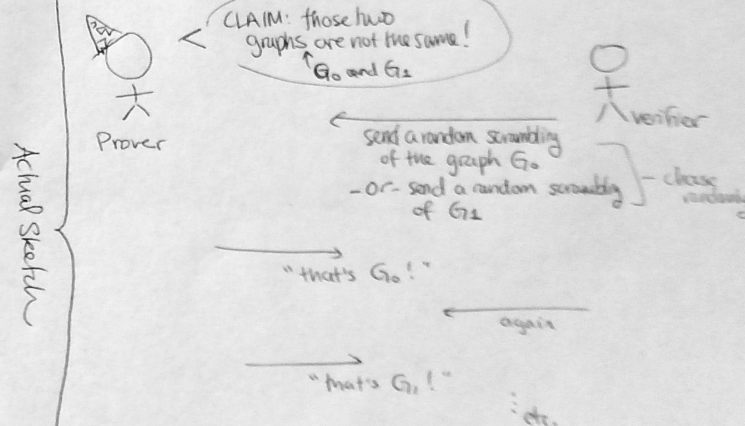
OPEN: $GNI \in NP?$

BUT. Here's an easy interactive protocol. (~~Metaphorical sketch~~)



If the prover is not lying, he'll always be correct. Else, sooner or later he'll surely be caught!

for graph isomorphism then,



If the graphs are really distinct, the prover will always be right; else, sooner or later he will be caught.