INSIDE P

this is transitive

We need to use log-space reductions $\leq_m^{\log}$ (when using $\leq_m^P$, all problems in P except $\emptyset$ and $\{0,1\}^*$ are equivalent)

Important classes: $L := \mathrm{DSPACE}(\log n)$, $NL := \mathrm{NSPACE}(\log n)$

We know: $L \subseteq NL = \text{co-}NL \subseteq P$ — inclusions not known to be strict

trivial, Imm.-Sz., reachability & $2^{c \log n} = n^c$

Theorem: CIRCUIT-EVAL is P-complete wrt. $\leq_m^{\log}$.
given Boolean circuit & input, is the output true?

Proof: Verify that the circuit construction we used when proving Cook's thm can be carried out in log. space.

open: CIRCUIT-EVAL $\in$ NL

Theorem: REACH is NL-complete wrt. $\leq_m^{\log}$.

Proof: Configuration graph of a NTM can be constructed in log space.

open: REACH $\in$ L

2-SAT (CNF formulas, all clauses have ≤ 2 literals)
exactly 2 if we replace (x) by $(x \vee x)$

$(\alpha \vee \beta)$ is an implication $\neg\alpha \Rightarrow \beta$, which is also $\neg\beta \Rightarrow \alpha$

For a 2-CNF formula φ, construct its <u>implication graph</u>: vertices = variables & their negations
edges = implications (clause produces two)

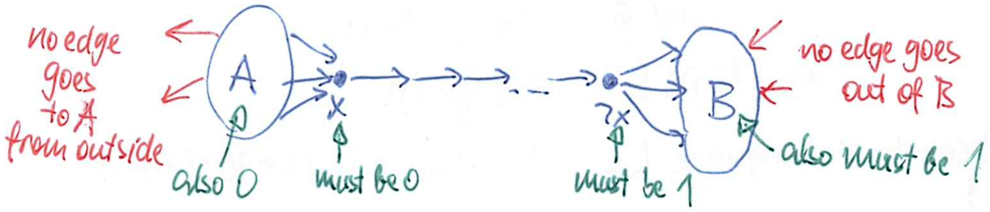
Lemma: φ is unsatisfiable $\Leftrightarrow$ $\exists$ var. v s.t. G_φ contains both a path $v \to \neg v$ and a path $\neg v \to v$ — "contradictory cycle"

Proof: First observe: if literal α is set to 1, all literals reachable from α must be also 1.
... if v set to 0, all literals from which v is reachable must be also 0.

Hence $\Leftarrow$ is true.

$\Rightarrow$: prove contra-positive: If there $\nexists$ a contradictory cycle, we construct a satisfying assignment.

① If there exists a path $x \to \neg x$:



Also, A is the mirror image of B:
- literals negated
- edge directions flipped

If $A \cap B \neq \emptyset$: $\exists$ contradictory cycle.

Otherwise: remove $A, B, x, \neg x$ & continue (because of red note, the removed part cannot affect SAT'ability of the rest)

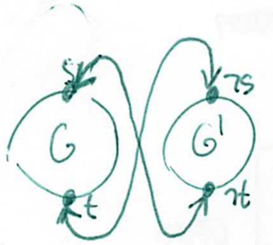
② $\exists$ path $\neg x \to x$: symetrically.

③ no such paths exist: add edge $x \to \neg x$ for some remaining variable x (effectively setting $x=0$) and continue (this couldn't have created a new contradictory cycle)

Corollary: 2-SAT $\in$ P (in fact, there is an $O(n)$-time alg. on the RAM)

Thus 2-SAT is NL-complete.

Proof: REACH $\in$ NL, which is also co-NL, so $\overline{\text{REACH}} \in$ NL.
We can decide 2-SAT using a subroutine for $\overline{\text{REACH}}$, so 2-SAT $\in$ NL.
NL-hardness: REACH is NL-complete, NL = co-NL, so $\overline{\text{REACH}}$ is also NL-complete.
Let's reduce $\overline{\text{REACH}}$ to 2-SAT in log space:



- given graph G and vertices s, t (if s = t, REJECT) ← by producing a constant non-SATable formula
- build a formula with implication graph G, containing only positive literals (a disjoint copy of G gets created with the mirror image ...)
- add implications $t \Rightarrow \neg s$, $\neg t \Rightarrow s$ (this creates $s \Rightarrow \neg t$, $\neg s \Rightarrow t$)
- resulting formula is SATable $\Leftrightarrow \nexists$ path $s \to t$ (no other contradictory cycle is possible)

HIERARCHY THEOREMS

Goal: Show that some classes are different :)

Tools:
- time/space-constructibility of functions
- enumeration of machines M_α (we can also use integer codes instead of strings)
- Universal Turing Machine (UTM): given $\langle \alpha, \beta \rangle$, simulates M_α on input β.
  - complexity: if M_α runs in time T and space S, on input β, UTM(α, β) runs in:
    - space $\in O(S)$
    - time $\in O(T^2)$ or $O(T \log T)$

    constants dependent on α (e.g., size of work alphabet)

    $O(T^2)$: because of reduction k tapes $\Rightarrow$ 1 tape

    $O(T \log T)$: can use a better reduction $k \to 2$ tapes (we haven't proven that)
  - can extend the UTM to count space/time used
    - by the simulated machine
    - by the UTM itself

    & stop simulation if limit exceeded

Theorem (space hierarchy): If f, g are non-decreasing space-constructible functions, $f \in o(g)$ and $g(n) \geq \log n$, then
$$\text{DSPACE}(f(n)) \subsetneq \text{DSPACE}(g(n)).$$

Proof: $\subseteq$ trivial, will construct a language $L \in \text{DSPACE}(g(n)) \setminus \text{DSPACE}(f(n))$.

Define machine M: Given input β:
1. Check that β has the form $\alpha 1 0^l$ for some α, l.
2. Write $g(|\beta|)$ 1s on a work tape X.
3. Simulate M_α on input β using an UTM.
   Stop if more than $g(|\beta|)$ cells are used by the UTM (& assume M_α rejected then)
4. If M_α accepted, reject.
   If rejected, accept.

then $L := L(M)$

We check that M runs in space $O(g(n))$, so $L \in DSPACE(g(n))$.

Let's show that $L \notin DSPACE(f(n))$... if it were true, there $\exists M_\alpha$ deciding L in space $f'(n) \in O(f(n))$.

So the UTM can simulate M_α in space $\underbrace{c \cdot f'(n)}$ for some c (depending on α).

this is in $o(g(n))$, so $cf'(n) < g(n)$ for n large enough

Construct input $\beta := \alpha 10^l$ for l large enough.

Then the UTM fits in the time bound $g(|\beta|) \Rightarrow$ on this input, M_α doesn't agree with M ↯ ■

<u>Notes</u>: The trick with padding α by 10^l is actually not necessary, because for every machine, there are infinitely many equivalent codes $\Rightarrow$ just pick code α large enough.

<u>Corollaries</u>: $DSPACE(n) \subsetneq DSPACE(n^2) \subsetneq DSPACE(n^3) \subsetneq \dots$, so $PSPACE \neq DSPACE(n^k)$ for every k.

$DSPACE(n) \subsetneq DSPACE(n \log\log n) \subsetneq DSPACE(n \log n) \subsetneq DSPACE(n^2)$

$NL \subseteq DSPACE(\log^2 n) \subsetneq DSPACE(n) \subsetneq PSPACE$] so $NL \neq PSPACE$ and $QBF \notin NL$

Savitch's thm.

$PSPACE \subseteq DSPACE(2^n) \subsetneq DSPACE(2^{n^2}) \subseteq EXPSPACE$] so $PSPACE \neq EXPSPACE$

<u>Theorem (time hierarchy)</u>: If f, g are time-constructible non-decreasing functions such that $f \cdot \log f \in o(g)$, then $DTIME(f(n)) \subsetneq DTIME(g(n))$.

<u>Proof</u>: Almost identical, modify step 3 to stop the UTM after $g(n)$ steps.

If M_α decides M in time $f' \in O(f)$, then UTM simulates M_α in time at most $f'(n) \log f'(n) \in o(g(n))$.

So for large enough equivalent code α, UTM completes simulation of $M_\alpha(\alpha)$ in time $g(|\alpha|)$.

Therefore M_α disagrees with M on input α ↯

<u>Corollaries</u>: $DTIME(n) \subsetneq DTIME(n^2) \subsetneq \dots$ (But we cannot separate $DTIME(n \log n)$ from $DTIME(n)$ this way)

$DTIME(n^k) \subsetneq DTIME(n^{\log n}) \subsetneq EXP$ ... so $P \subsetneq EXP$.

$P \neq DTIME(n^k)$ for every k.

So we have: $L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXP \subseteq NEXP \subseteq EXPSPACE$.

$\neq$ $\neq$ $\neq$

<u>Note</u>: What about non-deterministic classes?

We have $NTIME(f(n)) \subsetneq NTIME(g(n))$ and $NSPACE(f(n)) \subsetneq NSPACE(g(n))$ } whenever $f \in o(g)$

- non-deterministic reduction of tapes can be done with <u>constant</u> overhead in both time & space
- we have non-deterministic UTM
- but how do we negate its output ???
  - for space-bounded classes, use Immerman–Szelepcsényi thm.
  - for time-bounded, a more involved proof is needed (not covered here)

RELATIVE CLASSES

We can define complexity classes for machines with an oracle.
For example $P[A]$ a.k.a. P^A and $NP[A]$ a.k.a. NP^A. ← called "relative" classes wrt. A
Many proofs apply to relativized statements of theorems, too.
But P vs. NP cannot be relativized: ← this limits proof techniques which could separate P from NP (e.g., diagonalization as in proofs in hierarchy thm's doesn't work)

Theorem: There exist languages A, B s.t. $P[A] = NP[A]$, but $P[B] \neq NP[B]$.

Proof: (A) Let $A = QBF$. Then $P[A] = PSPACE[A] = PSPACE$
$NP[A] \subseteq PSPACE[A] = PSPACE$.

(B) For every language B, define $U_B := \{1^n \mid \exists \beta \in B \text{ with } |\beta| = n\}$. ← "shadow cast by the language B"
We have $U_B \in NP[B]$: just guess β and check it's in B.
Construct B s.t. $U_B \notin P[B]$: in step i, we make sure that $M_i[B]$ doesn't decide U_B within $2^n/10$ steps for inputs of size n. To achieve that, we put finitely many strings inside or forever outside B ← we "decide their fate"

Step i: Choose $n >$ lengths of all strings whose fate we already decided.
Run $M_i[B]$ on 1^n for $2^n/10$ steps.

- when it queries B for a string β:
  - if fate of β was already decided, answer consistently
  - if not, put β outside B and answer NO
- if it accepted 1^n, arrange $1^n \notin U_B$: so far, no string of length n is in B, put the remaining ones outside B
- if it rejected 1^n, add one string of length n to B (so $1^n \in U_B$)
  ← so far, we met at most $2^n/10$ such strings, so some undecided strings must remain.

Now if some machine $M[B]$ decides U_B in time $f(n) \in poly(n)$,
we have $f(n) < 2^n/10$ for n large enough.
For large enough i s.t. M_i is equivalent to M, n is also large enough
$\Rightarrow L(M)$ disagrees with U_B on input 1^n.

So $U_B \notin P[B]$.