**Last time:**  NL-completeness (under logspace redu.)  
PSPACE-completeness (under p-time redu.)

**Today:** - Finish argument that QBF is PSPACE-complete  
- PSPACE + 2-player games  
- Generalized Geography is PSPACE-complete  
- Start Immerman–Szelepcsényi theorem:
  - nondist space closed under complement  
  \( L \subseteq \text{NC} = \text{co-NC} \)

**Admin:** - PS 2 due today  
- OH next week strongly recommended  
  (no OH during midterm week)  
  Midterm Mar 6-12 for material thru next Thurs

**Questions?**  
Late days

**Recall:** we have \( L \subseteq \text{PSPACE} \), decided by DTM \( M \) running in \( n^k \) space.  
- If \( M \) accepts, always does so (always halts) in time \( 2^{O(n)} \) s.m.e. (or \( \infty \) loop run forever).  
- We need to give polytime red. which, on input \( x \in L \), output a QBF which is true iff \( x \in L \).

3 ideas/ingredients:

1) Cook-Levin thm: computation tableau, cell \( T_{ij} \) contains contents of TM \( M \)'s tape cell \( i \) at time \( t \).
As in CL pf, can use Bool form. to enforce consistency of $T$ with MS comp.

**NOTE**: in $n^k$ space, $M$ can run for as long as $2^{d^k}$ time, so tableau has $n^k$ columns, $2^{d^k}$ rows

2) Savitch's theorem: recursively find midpoint of tableau.
   This could require $2^{n^{d^k}}$ many $\exists \cdots \exists$ quantifiers...

3) **Use univ. quantifiers to save (expon.) on length of formula.**
   (enables poly size QBF to capture the computation)

More detail:

1) As in CL pf, given $|x| = n$, any config of $M$ on $x$
   (row of tableau) has bin. encoding of length $n^k$.
   (asst to $n^k$ Bool. vars).

   - We assume WLOG $M$ has 1 accept config $C_{acc}$.

   Let $C_1, C_2$ be two configs (Boolean encodings of configs; bitstrings of length $n^k$)
   Let $\text{LEGIT}(c_1, c_2, i)$ be:

   $$\text{LEGIT}(c_1, c_2, i) = \begin{cases} T & \text{if } M \text{ goes from } c_1 \text{ to } c_2 \text{ in } \leq 2^i \\ F \text{ or } \# \text{ or } 7 \text{ or } 9 \text{ or } \ldots \end{cases}$$
\[
\text{Mac} x \iff \text{LEGIT}(c_{\text{start}}, c_{\text{acc}}, d^n) \text{ is } T
\]
(Here \(c_{\text{start}} \in \{1, \ldots, n\}^n\) is bin. encoding of M's init config. on input x). \(c_{\text{acc}} = \)

We want an almost-finitely-quantified Bool formula for \(\text{LEGIT}(c, c_i; i)\); the vars for \(c, c_i\) are free, all others quantified — so when we plug in \(c_{\text{start}}, c_{\text{acc}}\), it is tot. quant. \& T/F as should be.

Remains to describe/construct \(\text{LEGIT}(c, c_i; i)\)

\(i \odot 0: \quad \text{LEGIT}(c, c_2, 0) = \varnothing_0(c, c_2) \lor \varnothing_1(c, c_2)\)

where

\[\varnothing_0(c, c_2) = T \iff M \text{ yields } c_2 \text{ from } c_i \text{ in 0 steps,}
\]
i.e. \(c_i = c_2\) (bit by bit).

\[\varnothing_1(c, c_2) = T \iff M \text{ yields } c_2 \text{ from } c_i \text{ in 1 step;}
\]
as in CL thm pf, a Bool formula for this \(O(\cdot^n)\) size.

"STEP" =

\(i \odot 0: \quad \text{Most naive approach: do time step}
\]
\[
\text{LEGIT}(c, b; i) = \exists c, c_2, c_3, \ldots c_{i-1}\, \text{LEGIT}(c, c_0) \land \text{LEGIT}(c, c_2, 0) \land \ldots \land \text{LEGIT}(c_{i-2}, c_{i-1}; 0) \land \text{LEGIT}(c_{i-1}, b; 0)
\]

(1) Correct semantically, but way too long to have \(\text{LEGIT}(c_{\text{start}}, c_{\text{acc}}, d^n)\) be poly-time computable.
2) Next attempt: use Switch/recursive midpoint idea:

\[ \text{LEGIT}(c_1, c_2, i) = \exists c' \text{ LEGIT}(c, c', i - 1) \land \text{LEGIT}(c', c, i - 1) \]

Looks good, but actually same size as 1.

3) Use A to avoid formula size doubling in

\[ \text{LEGIT}(c_1, c_2, i) = \exists c \forall c_3 \forall c_4 \]

\[ (((c_3 = c) \land (c_4 = c')) \lor ((c_3 = c') \land (c_4 = c))) \Rightarrow \text{LEGIT}(c_3, c_4, i - 1) \]

\[ \text{size of } \text{LEGIT}(c_1, c_2, i) = O(n^i) + \quad \text{size of} \]

\[ i \cdot O(n^i) \]

\[ \leq O(n^{2i}) \text{ even for } i = \frac{n^i}{i} \]

So \( \text{LEGIT}(c_1, c_2, d^{n^i}) \) is poly size + poly time computable, &

\[ \text{Macc} \iff \text{LEGIT}(c_{\text{stat}} + c_{\text{acc}} d^{n^i}) \text{ is } \top. \]

**PSPACE:** class that captures optimal play in 2-player games.

Can view QSAT inst. \( \exists x_1 \forall x_2 \exists x_3 \forall x_4 \ldots (\forall x_i \exists x_{i+1}) \) as a game:
Â, E players take turns
E goes first b/c E chooses value for x,
A goes : chooses an x₂ value. Etc.
E wins iff final \( \varphi(x₁, x₂, \ldots xₙ) = T \).

Clear that
E has a winning strat. \( \iff \) is T.

Here's a game: GEOGRAPHY
R: PARIS
M: SYDNEY
R: YPSILANTI
M: INDIANAPOLIS
R: STOCKHOLM
M: MEXICO CITY

Can consider a fixed set of cities...
Game corr. to digraph: nodes for cities, edges based on 1st/last letters. Players alternate steps, first player who can't move (all out-neighbors already used) is loser.
**Def** \( GG = \text{GENERALIZED GEOGRAPHY} \)

\[ GG = \exists (G, v) : G \text{ disrep, player 1 has a winning strat. For the game played on } G \text{ starting at node } v. \]

**Thm:** \( GG \) is PSPACE-complete.

**PF**
1) Show \( GG \in \text{PSPACE} \): here’s an \( A \):

**A:** on input \((G, v)\):

1) if no out-edges from \( v \), return FALSE/LOSE;
2) if \( v \) has out-edges in \( G \), remove them + \( v \) to get \( G' \);
3) for each node \( v' \) in \( G' \) s.t. \( v \to v' \) was an edge in \( G \),

- call \( A \) on \((G', v')\) (using space for each call)

If any call returns F/L, player 2 can be made to lose, return TRUE/WIN.
If every call returns T/W, return F/L.

**Correct.** Depth of recursion \( \leq n = \# \text{nodes} \), so space used \( \leq \text{poly}(n) \).
Must show any \( \mathsf{L} \subseteq \mathsf{PSPACE} \) has \( \mathsf{L} \subseteq \mathsf{P} \mathsf{GG} \).

Suff to show \( \mathsf{ABF} \subseteq \mathsf{P} \mathsf{GG} \).

Do this by example: consider

\[
\Phi = \exists x \forall y \exists z (\overline{x} \land y \land z) \land (w \land x \land \overline{y}) \land (w \land y \land \overline{z})
\]

Observe: is \( \mathsf{T} \): take \( w = \mathsf{T} \), take any \( x \), can take \( y = \mathsf{T} \)

Here's the \( \mathsf{GG} \) instance:

- \( 4 \) nodes for each var;
  - diamond, hooked up as shown
  - left = \( \text{neg} \), right = \( \text{neg} \)
- node for each clause;
  - that node has edge to each lit. it contains

- Any path from start node corr. to \( \mathsf{F} \) (1st player)
  - choosing value for 1st var, \( A \) choosing value for 2nd var;
  - etc. Player decides to set var \( x \) to \( \mathsf{T} \) \( \implies \) choose \( \overline{x} \) node, \( \lor \) vice versa.

- A player's turn at \( * \): choose a clause. (Clause \( \iff \) the assn falsifies the CNF)
Then "∃" choices are exactly the lits in the clause.

If no true lit. in clause, no move for "∃ player" (all poss lits for him were on path), "∃ player" loses.
If some lit. is true in clause, then "∃ player" chooses it, "∀ player" will have no move.

Next time: Immenerman–Szelapcsényi theorem