• Recall: probabilistic Turing Machine
  – deterministic TM with extra tape for “coin flips”

• RL (Random Logspace)
  – \( L \in RL \) if there is a probabilistic logspace TM \( M \):
    \[
    x \in L \Rightarrow \Pr_y[M(x,y) \text{ accepts}] \geq \frac{1}{2}
    \]
    \[
    x \notin L \Rightarrow \Pr_y[M(x,y) \text{ rejects}] = 1
    \]
  – important detail #1: only allow one-way access to coin-flip tape
  – important detail #2: explicitly require to run in polynomial time

\[ L \subseteq RL \subseteq NL \subseteq SPACE(\log^2 n) \]

• Theorem (SZ): \( RL \subseteq SPACE(\log^{3/2} n) \)

• Belief: \( L = RL \) (major open problem)

---

Natural problem:
**Undirected STCONN**: given an undirected graph \( G = (V, E) \), nodes \( s, t \), is there a path from \( s \rightarrow t \)?

**Theorem**: \( USTCONN \in RL \).

(Recall: STCONN is NL-complete)

Proof sketch: (in Papadimitriou)
– add self-loop to each vertex (technical reasons)
– start at \( s \), random walk \( 2|V||E| \) steps, accept if see \( t \)
– Lemma: expected return time for any node \( i \) is \( 2|E|/d_i \)
– suppose \( s=v_1, v_2, \ldots, v_n=t \) is a path
  – expected time from \( v_i \) to \( v_{i+1} \) is \( (d_i/2)(2|E|/d_i) \)
  – expected time to reach \( v \), \( \leq |V||E| \)
  – \( \Pr \) fail reach \( t \) in \( 2|V||E| \) steps \( \leq \frac{1}{2} \)

Reingold 2005: \( USTCONN \in L \)
A motivating question

- Central problem in logic synthesis:
  - Given Boolean circuit $C$, integer $k$
  - Is there a circuit $C'$ of size at most $k$ that computes the same function as $C$?

- Complexity of this problem?
  - $\text{NP}$-hard? in $\text{NP}$? in $\text{coNP}$? in $\text{PSPACE}$?
  - Complete for any of these classes?

$\vee x_1 \land x_2 \land \neg x_3 \land \ldots \land x_n$.

Oracle Turing Machines

- Oracle Turing Machine (OTM):
  - Multitape $M$ with special "query" tape
  - Special states $q_1$, $q_{\text{yes}}$, $q_{\text{no}}$
  - On input $x$, with oracle language $A$
  - $M^A$ runs as usual, except…
  - When $M^A$ enters state $q_1$:
    - $y =$ contents of query tape
    - $y \in A \Rightarrow$ transition to $q_{\text{yes}}$
    - $y \notin A \Rightarrow$ transition to $q_{\text{no}}$

Oracle Turing Machines

- Nondeterministic OTM
  - Defined in the same way
  - (Transition relation, rather than function)
  - Oracle is like a subroutine, or function in your favorite programming language
    - But each call counts as single step
  - E.g.: given $\varphi_1, \varphi_2, \ldots, \varphi_n$ are even # satisfiable?
  - Poly-time OTM solves with SAT oracle

Oracle Turing Machines

Shorthand #1:
- Applying oracles to entire complexity classes:
  - Complexity class $C$
  - Language $A$
  - $C^A = \{ L \text{ decided by } OTM \text{ with oracle } A \text{ with } M \text{ "in" } C \}$
  - Example: $P^{SAT}$

Shorthand #2:
- Using complexity classes as oracles:
  - $OTM \text{ M}$
  - Complexity class $C$
  - $M^A$ decides language $L$ if for some language $A \in C$, $M^A$ decides $L$
  - Both together: $C^P = \text{languages decided by OTM "in" } C \text{ with oracle language from } D$
  - Exercise: show $P^{SAT} = P^{NP}$

The Polynomial-Time Hierarchy

- Can define lots of complexity classes using oracles
  - The classes on the next slide stand out
    - They have natural complete problems
    - They have a natural interpretation in terms of alternating quantifiers
    - They help us state certain consequences and containments (more later)
The Polynomial-Time Hierarchy

\[ \Sigma_0 = \Pi_0 = P \]
\[ \Delta_1 = P^P \]
\[ \Sigma_1 = NP \]
\[ \Pi_1 = coNP \]
\[ \Delta_2 = P^{NP} \]
\[ \Sigma_2 = NP^{NP} \]
\[ \Pi_2 = coNP^{NP} \]
\[ \Delta_{i+1} = P^{\Sigma_i} \]
\[ \Sigma_{i+1} = NP^{\Sigma_i} \]
\[ \Pi_{i+1} = coNP^{\Sigma_i} \]

Polynomial Hierarchy \( PH = \bigcup_i \Sigma_i \)

\[ \Sigma_0 = \Pi_0 = P \]
\[ \Delta_1 = P^P \]
\[ \Sigma_1 = NP \]
\[ \Pi_1 = coNP \]
\[ \Delta_2 = P^{NP} \]
\[ \Sigma_2 = NP^{NP} \]
\[ \Pi_2 = coNP^{NP} \]
\[ \Delta_{i+1} = P^{\Sigma_i} \]
\[ \Sigma_{i+1} = NP^{\Sigma_i} \]
\[ \Pi_{i+1} = coNP^{\Sigma_i} \]

Example:

- **MIN CIRCUIT**: given Boolean circuit \( C \), integer \( k \); is there a circuit \( C' \) of size at most \( k \) that computes the same function \( C \) does?
  
  - **MIN CIRCUIT** \( \in \Sigma_2 \)

Example:

- **EXACT TSP**: given a weighted graph \( G \), and an integer \( k \); is the \( k \)-th bit of the length of the shortest TSP tour in \( G \) a 1?
  
  - **EXACT TSP** \( \in \Delta_2 \)

Useful characterization

- Recall: \( L \in NP \) iff expressible as
  \[ L = \{ x \mid \exists y, |y| \leq |x|^k, (x, y) \in R \} \]
  where \( R \in P \).

  - **Theorem**: \( L \in \Sigma_i \) iff expressible as
    \[ L = \{ x \mid \exists y, |y| \leq |x|^k, (x, y) \in R \} \]
    where \( R \in \Pi_{i+1} \).

- Corollary: \( L \in coNP \) iff expressible as
  \[ L = \{ x \mid \forall y, |y| \leq |x|^k, (x, y) \in R \} \]
  where \( R \in P \).

  - Corollary: \( L \in \Pi_i \) iff expressible as
    \[ L = \{ x \mid \forall y, |y| \leq |x|^k, (x, y) \in R \} \]
    where \( R \in \Sigma_{i+1} \).
Theorem: \( L \in \Sigma_i \) iff expressible as 
\[
L = \{ x | \exists y \in \Sigma^*, (x, y) \in R \}, \text{ where } R \in \Pi_{i+1}.
\]

- Proof of Theorem:
  - induction on \( i \)
  - base case \( (i = 1) \) on previous slide
  - we know \( \Sigma_1 = \text{NP}^{\oplus_1} = \text{NP}^{\Pi_1} \)
  - guess \( y \), ask oracle if \( (x, y) \in R \)

Useful characterization

Nicer, more usable version:

- \( \text{Le} \Sigma_1 \) iff expressible as 
  \[
  L = \{ x | \exists y_1 \forall y_2 \exists y_3 \ldots Q_1(x, y_1, y_2, 
  \ldots, y_{i-1}) \in R \} 
  \]
  where \( Q = \exists / \forall \) if \( i \) even/odd, and \( R \in \Sigma \)

- \( \text{Le} \Pi_1 \) iff expressible as 
  \[
  L = \{ x | \forall y_1 \exists y_2 \forall y_3 \ldots Q_1(x, y_1, y_2, \ldots, y_{i-1}) \in R \} 
  \]
  where \( Q = \exists / \forall \) if \( i \) even/odd, and \( R \in \Pi \)

Note: AND of polynomially many \( \Pi_1 \) predicates is in \( \Pi_1 \).

Alternating quantifiers

- \( \text{Le} \Pi_1 = \text{NP} \)
- \( \text{Le} \Pi_1 = \text{coNP} \)
- consider \( \text{Le} \Sigma_1 \):
  \[
  L = \{ x | \exists y_1 (x, y_1) \in R \}, \text{ for } R \in \Pi_{i+1}
  \]
  \[
  L = \{ x | \exists y_2 \forall y_3 \exists y_4 \ldots Q_1(x, y_1, y_2, 
  \ldots, y_{i-1}) \in R \}
  \]
  \[
  L = \{ x | \exists y_1 \forall y_2 \exists y_3 \ldots Q_1(x, y_1, y_2, \ldots, y_{i-1}) \in R \}
  \]

- same argument for \( L \in \Pi_1 \)

- \( \text{Le} \Pi_1 = \Sigma_1 \) 
- \( \text{Le} \Pi_1 = \Sigma_1 \) 
- exercise.
Alternating quantifiers

Pleasing viewpoint:

Complete problems

- three variants of SAT:
  - QSAT$_i$: (i odd) = 
    \begin{align*}
    &\{\text{3-CNFs } \phi(x_1, x_2, \ldots, x) \text{ for which} \\
    &\exists x_1 \forall x_2 \exists x_3 \ldots \exists x_i \phi(x_1, x_2, \ldots, x) = 1\} \\
    &\text{QSAT$_i$: (i even) =} \\
    &\{\text{3-DNFs } \phi(x_1, x_2, \ldots, x) \text{ for which} \\
    &\exists x_1 \forall x_2 \exists x_3 \ldots \forall x_i \phi(x_1, x_2, \ldots, x) = 1\} \\
    &\text{QSAT = (3-CNFs } \phi \text{ for which} \\
    &\exists x_1 \forall x_2 \exists x_3 \ldots \exists x_i \phi(x_1, x_2, \ldots, x_i) = 1\} 
    \end{align*}

QSAT$_i$ is $\Sigma_i$-complete

Theorem: QSAT$_i$ is $\Sigma_i$-complete.

- Proof: (clearly in $\Sigma_i$)
  - assume i odd; given $L \in \Sigma_i$ in form 
    \{ $x \mid \exists y_1 \forall y_2 \exists y_3 \ldots \exists y_i (x, y_1, y_2, \ldots, y) \in R$ \}
  - we get:
    \begin{align*}
    \exists y_1 \forall y_2 \exists y_3 \ldots \exists y_i \phi(x, y_1, y_2, \ldots, y_i) = 1 \iff x \in L \\
    \end{align*}

QSAT$_i$ is $\Sigma_i$-complete

Proof (continued)

- assume i even; given $L \in \Sigma_i$ in form 
  \{ $x \mid \exists y_1 \forall y_2 \exists y_3 \ldots \forall y_i (x, y_1, y_2, \ldots, y) \in R$ \}
  \begin{align*}
  \exists y_1 \forall y_2 \exists y_3 \ldots \forall y_i \phi(x, y_1, y_2, \ldots, y_i) = 1 \iff x \in L \\
  \end{align*}

QSAT$_i$ is $\Sigma_i$-complete

Proof (continued)

- Problem set: can construct 3-CNF $\phi$ from $C$:
  \begin{align*}
  &\exists z \phi(x, y_1, \ldots, y_i, z) = 1 \iff C(x, y_1, \ldots, y_i) = 1 \\
  \end{align*}

- we get:
  \begin{align*}
  &\exists y_1 \forall y_2 \exists y_3 \ldots y_i \phi(x, y_1, \ldots, y_i, z) = 1 \\
  &\iff \exists y_1 \forall y_2 \exists y_3 \ldots \exists y_i C(x, y_1, \ldots, y_i) = 1 \iff x \in L \\
  \end{align*}
QSAT is \textbf{PSPACE}-complete

\textbf{Theorem:} QSAT is \textbf{PSPACE}-complete.

\textbf{Proof:}

- \( \forall x_1 \exists x_2 \forall x_3 \ldots Qx_n \varphi(x_1, x_2, \ldots, x_n) \)
  - \( \exists x_1 \): for each \( x_1 \), recursively solve \( \forall x_2 \exists x_3 \ldots Qx_n \varphi(x_1, x_2, \ldots, x_n) \)
  - if encounter “yes”, return “yes”
  - if encounter “no”, return “no”
  - base case: evaluating a 3-CNF expression
  - \( \text{poly}(n) \) recursion depth
  - \( \text{poly}(n) \) bits of state at each level

\begin{itemize}
  \item for \( i = 0, 1, \ldots, n \) produce quantified Boolean expressions \( \psi_i(A, B, W) \)
  \item convert \( \psi_i \) to 3-CNF \( \varphi \)
  \item add variables \( V \)
  \item hardware \( A = \text{START}, B = \text{ACCEPT} \)
  \item boolean expression of size \( O(n^k) \)
\end{itemize}

\begin{itemize}
  \item prove \( \exists y \in L \) \quad \text{by \textbf{PSPACE}-complete}\textbf{REACH}(A, B, i) \rightarrow \text{configuration } Y \text{ reachable from configuration } X \text{ in at most } 2^i \text{ steps.}
  \item key observation \#1:
  \begin{align*}
    \text{REACH}(A, B, i) \implies \text{configuration } Y \text{ reachable from configuration } X \text{ in at most } 2^i \text{ steps.}
  \end{align*}
\end{itemize}

- cannot define \( \psi_i(A, B) \) to be
  \begin{align*}
    \exists Z \left[ \text{REACH}(A, Z, i) \land \text{REACH}(Z, B, i) \right]
  \end{align*}

(why?)
QSAT is PSPACE-complete

- Key idea #2: use quantifiers
- couldn’t do \( \psi_{i+1}(A, B) = \exists Z [\psi_i(A, Z) \land \psi_i(Z, B)] \)
- define \( \psi_{i+1}(A, B) \) to be
  \[ \exists Z \forall X \forall Y [((X=A \land Y=Z) \lor (X=Z \land Y=B)) \Rightarrow \psi(X, Y)] \]
- total size of \( \psi_i \) is \( O(n^2) = \text{poly}(n) \)
- logspace reduction

PH collapse

- recall: \( L \in \Sigma_{i+1} \) iff expressible as
  \[ L = \{ x \mid \exists y (x, y) \in R \} \]
  where \( R \in \Pi_i \)
- since \( \Pi_i = \Sigma_i \), \( R \) expressible as
  \[ R = \{ (x, y) \mid \exists z (x, y, z) \in R' \} \]
  where \( R' \in \Pi_{i+1} \)
- together: \( L = \{ x \mid \exists (y, z) (x, (y, z)) \in R' \} \)
- conclude \( L \in \Sigma_i \)

Natural complete problems

- We now have versions of SAT complete for levels in PH, PSPACE
- Natural complete problems?
  - PSPACE: games
  - PH: almost all natural problems lie in the second and third level

Natural complete problems in PH

- MIN CIRCUIT
  - good candidate to be \( \Sigma_i \)-complete, still open
- MIN DNF: given DNF \( \varphi \), integer \( k \); is there a DNF \( \varphi' \) of size at most \( k \) computing same function \( \varphi \) does?

Theorem (U): MIN DNF is \( \Sigma_2 \)-complete.
Natural complete problems in PSPACE

- General phenomenon: many 2-player games are PSPACE-complete.
  - 2 players I, II
  - alternate picking edges
  - lose when no unvisited choice
- GEOGRAPHY = \{(G, s) : G is a directed graph and player I can win from node s\}

**Theorem:** GEOGRAPHY is PSPACE-complete.

**Proof:**
- in PSPACE
  - easily expressed with alternating quantifiers
- PSPACE-hard
  - reduction from QSAT

 alternately pick truth assignment

pick a clause

pick a true literal?