Recall: probabilistic Turing Machine
- deterministic TM with extra tape for "coin flips"

**RL**
- **RL** (Random Logspace)
  - \( L \subseteq RL \) if there is a probabilistic logspace TM \( M \):
    - \( x \in L \Rightarrow \Pr[M(x,y) \text{ accepts}] \geq \frac{1}{2} \)
    - \( x \notin L \Rightarrow \Pr[M(x,y) \text{ rejects}] = 1 \)
  - 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:** \( L = RL \) (open problem)

---

**Undirected STCONN**

- **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 \( s=v_1, v_2, ..., v_n=t \) is a path
  - expected time from \( v_i \) to \( v_{i+1} \) is \( (d_i/2)(2|E|/d_i) = |E| \)
  - expected time to reach \( v_i \leq |V||E| \)
  - \( \Pr[\text{fail reach } t \text{ in } 2|V||E| \text{ steps}] \leq \frac{1}{2} \)

- **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 C does?

• Complexity of this problem?
  - \textbf{NP}-hard? in \textbf{NP}? in \textbf{coNP}? in \textbf{PSPACE}?
  - complete for any of these classes?

Oracle Turing Machines

• Oracle Turing Machine (OTM):
  - multitape TM M with special "query" tape
  - special states \text{q}_?, \text{q}_{yes}, \text{q}_{no}
  - on input x, with oracle language A
  - \text{M}^A runs as usual, except…
  - when \text{M}^A enters state \text{q}_?:
    - y = contents of query tape
    - y \in A \Rightarrow transition to \text{q}_{yes}
    - y \notin A \Rightarrow transition to \text{q}_{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
  - \text{e.g.}: given \varphi_1, \varphi_2, \ldots, \varphi_n are even # satisfiable?
  - poly-time OTM solves with SAT oracle

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

$\Sigma_0 = \Pi_0 = P$

$\Delta_i = P^\Sigma_i$, $\Sigma_i = NP$, $\Pi_i = coNP$

$\Delta_{i+1} = P^{\Sigma_i}$, $\Sigma_{i+1} = NP^{\Sigma_i}$, $\Pi_{i+1} = coNP^{\Sigma_i}$

Polynomial Hierarchy $PH = \bigcup_i \Sigma_i$

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 $\in \Sigma_2$

The PH

$PSPACE$: generalized geography, 2-person games...

3rd level: V-C dimension...

2nd level: MIN CIRCUIT, BPP...

1st level: SAT, UNSAT, factoring, etc...

Useful characterization

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

• Corollary: $L \in coNP$ iff expressible as
  $L = \{ x \mid \forall y, |y| \leq |x|^k, (x, y) \in R \}$
  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: $L \in \Pi_i$ iff expressible as
  $L = \{ x \mid \forall y, |y| \leq |x|^k, (x, y) \in R \}$
  where $R \in \Sigma_{i-1}$. 
Useful characterization

**Theorem:** $L \in \Sigma_i$ iff expressible as

$$L = \{ x \mid \exists y, |y| \leq |x|^i, (x, y) \in R \}, \text{ where } R \in \Pi_{i+1}.$$ 

- **Proof of Theorem:**
  - induction on $i$
  - base case ($i = 1$) on previous slide
  - try: $L = \Sigma_1 = \text{NP}^{\Pi_{i-1}}$ = $\text{NP}^{\Pi_{i-1}}$
  - guess $y$, ask oracle if $(x, y) \in R$
  - valid path
  - induction on $i$
  - $(\Leftarrow)$
  - given $L \in \Sigma_i = \text{NP}^{\Pi_{i-1}}$ decided by ONTM $M$
  - running in time $n^k$
  - try: $R = \{ (x, y) : y \text{ describes valid path of } M \text{'s computation leading to } q_{\text{accept}} \}$
  - but how to recognize valid computation path when it depends on result of oracle queries?

Useful characterization

**Theorem:** $L \in \Sigma_i$ iff expressible as

$$L = \{ x \mid \exists y, |y| \leq |x|^i, (x, y) \in R \}, \text{ where } R \in \Pi_{i+1}.$$ 

- try: $R = \{ (x, y) : y \text{ describes valid path of } M \text{'s computation leading to } q_{\text{accept}} \}$
- valid path = step-by-step description including correct yes/no answer for each $A$-oracle query $z_i$ ($A \in \Sigma_{i+1}$)
- verify "no" queries in $\Pi_{i+1}$:
  - e.g.: $z_i \notin A \land z_{i+1} \notin A \land \ldots \land z_k \notin A$
  - for each "yes" query $z_j$: $\exists y, |y| \leq |x|^j$ with $(z_j, w_j) \in R$ for some $R' \in \Pi_{i+2}$ by induction.
  - for each "yes" query $z_j$, put $w_j$ in description of path $y$

Useful characterization

**Theorem:** $L \in \Sigma_i$ iff expressible as

$$L = \{ x \mid \exists y, |y| \leq |x|^i, (x, y) \in R \}, \text{ where } R \in \Pi_{i+1}.$$ 

- single language $R$ in $\Pi_{i+1}$:
  - $(x, y) \in R$
  - $\iff$
  - all "no" $z_i$ are not in $A$ and all "yes" $z_i$ have $(z_j, w_j) \in R'$ and $y$ is a path leading to $q_{\text{accept}}$.
- Note: AND of polynomially-many $\Pi_{i+1}$ predicates is in $\Pi_{i+1}$.

Useful characterization

**Theorem:** $L \in \Sigma_i$ iff expressible as

$$L = \{ x \mid \exists y, |y| \leq |x|^i, (x, y) \in R \}, \text{ where } R \in \Pi_{i+1}.$$ 

- $\Leftarrow$
  - $\forall R' \in \Pi_{i+2}$ by induction.
  - $\forall$ predicates is in $\Pi_{i+1}$

Useful characterization

**Theorem:** $L \in \Sigma_i$ iff expressible as

$$L = \{ x \mid \exists y, |y| \leq |x|^i, (x, y) \in R \}, \text{ where } R \in \Pi_{i+1}.$$ 

- try: $R = \{ (x, y) : y \text{ describes valid path of } M \text{'s computation leading to } q_{\text{accept}} \}$
- valid path = step-by-step description including correct yes/no answer for each $A$-oracle query $z_i$ ($A \in \Sigma_{i+1}$)
- verify "no" queries in $\Pi_{i+1}$:
  - e.g.: $z_i \notin A \land z_{i+1} \notin A \land \ldots \land z_k \notin A$
  - for each "yes" query $z_i$: $\exists y, |y| \leq |x|^j$ with $(z_i, w_i) \in R'$ for some $R' \in \Pi_{i+2}$ by induction.
  - for each "yes" query $z_i$, put $w_i$ in description of path $y$

Useful characterization

**Theorem:** $L \in \Sigma_i$ iff expressible as

$$L = \{ x \mid \exists y, |y| \leq |x|^i, (x, y) \in R \}, \text{ where } R \in \Pi_{i+1}.$$ 

- single language $R$ in $\Pi_{i+1}$:
  - $(x, y) \in R$
  - $\iff$
  - all "no" $z_i$ are not in $A$ and all "yes" $z_i$ have $(z_j, w_j) \in R'$ and $y$ is a path leading to $q_{\text{accept}}$.
  - Note: AND of polynomially-many $\Pi_{i+1}$ predicates is in $\Pi_{i+1}$.

Useful characterization

**Theorem:** $L \in \Sigma_i$ iff expressible as

$$L = \{ x \mid \exists y, |y| \leq |x|^i, (x, y) \in R \}, \text{ where } R \in \Pi_{i+1}.$$ 

- try: $R = \{ (x, y) : y \text{ describes valid path of } M \text{'s computation leading to } q_{\text{accept}} \}$
- valid path = step-by-step description including correct yes/no answer for each $A$-oracle query $z_i$ ($A \in \Sigma_{i+1}$)
- verify "no" queries in $\Pi_{i+1}$:
  - e.g.: $z_i \notin A \land z_{i+1} \notin A \land \ldots \land z_k \notin A$
  - for each "yes" query $z_i$: $\exists y, |y| \leq |x|^j$ with $(z_i, w_i) \in R'$ for some $R' \in \Pi_{i+2}$ by induction.
  - for each "yes" query $z_i$, put $w_i$ in description of path $y$

Alternating quantifiers

Nicer, more usable version:

- $L \in \Sigma_i$ iff expressible as
  $$L = \{ x \mid \exists y_1 \exists y_2 \exists y_3 \ldots \exists y_i (x, y_1, y_2, \ldots, y_i) \in R \}$$
  where $Q = \forall \exists$ if $i$ even/odd, and $R \in \text{P}$

- $L \in \Pi_i$ iff expressible as
  $$L = \{ x \mid \forall y_1 \exists y_2 \exists y_3 \ldots \exists y_i (x, y_1, y_2, \ldots, y_i) \in R \}$$
  where $Q = \exists \forall$ if $i$ even/odd, and $R \in \text{P}$

Alternating quantifiers

**Proof:**

- $(\Rightarrow)$ induction on $i$
  - base case: true for $\Sigma_1 = \text{NP}$ and $\Pi_1 = \text{coNP}$
  - consider $L \in \Sigma_i$:
    - $L = \{ x \mid \exists y_1 (x, y_1) \in R' \}$, for $R' \in \Pi_{i+1}$
    - $L = \{ x \mid \exists y_1 \forall y_2 \exists y_3 \ldots \exists y_i (x, y_1, y_2, \ldots, y_i) \in R \}$
    - $L = \{ x \mid \exists y_1 \forall y_2 \exists y_3 \ldots \exists y_i (x, y_1, y_2, \ldots, y_i) \in R \}$
    - same argument for $L \in \Pi_i$
  - $(\Leftarrow)$ exercise.
Three variants of SAT:

- **QSAT**
  - Problem set: can construct
  - Proof: (clearly we get:

**QSAT**

- **Problem (continued)**
  - Assume i even; given L ∈ Σ, in form
    \[ \{ x | \exists y_1 \forall y_2 \exists y_3 \ldots \exists y_i (x, y_1, y_2, \ldots, y_i) \in R \} \]
    - 1 iff \( (x, y_1, y_2, \ldots, y_i) \in R \)

**QSAT**

<table>
<thead>
<tr>
<th>May 4, 2015</th>
<th>25</th>
</tr>
</thead>
</table>

Complete problems

- Three variants of SAT:
  - **QSAT** (i odd) =
    - (3-CNFs \( \phi(x_1, x_2, \ldots, x_i) \) for which
      \[ \exists x_1 \forall x_2 \exists x_3 \ldots \exists x_i \phi(x_1, x_2, \ldots, x_i) = 1 \}
  - **QSAT** (i even) =
    - (3-DNFs \( \phi(x_1, x_2, \ldots, x_i) \) for which
      \[ \exists x_1 \forall x_2 \exists x_3 \ldots \forall x_i \phi(x_1, x_2, \ldots, x_i) = 1 \}
  - **QSAT** = (3-CNFs \( \phi \) for which
    \[ \exists x_1 \forall x_2 \exists x_3 \ldots Qx_n \phi(x_1, x_2, \ldots, x_n) = 1 \}

QSAT is \( \Sigma_i \)-complete

**Theorem:** QSAT is \( \Sigma_i \)-complete.

- Proof: (clearly in \( \Sigma_i \))
  - Assume i odd; given \( L \in \Sigma_i \) in form
    \[ \{ x | \exists y_1 \forall y_2 \exists y_3 \ldots \exists y_i (x, y_1, y_2, \ldots, y_i) \in R \} \]
      - 1 iff \( (x, y_1, y_2, \ldots, y_i) \in R \)

QSAT is \( \Sigma_i \)-complete

**Proof (continued)**

- Assume i even; given \( L \in \Sigma_i \) in form
  \[ \{ x | \exists y_1 \forall y_2 \exists y_3 \ldots \exists y_i (x, y_1, y_2, \ldots, y_i) \in R \} \]
    - 1 iff \( (x, y_1, y_2, \ldots, y_i) \in R \)

QSAT is \( \Sigma_i \)-complete

<table>
<thead>
<tr>
<th>May 4, 2015</th>
<th>27</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>May 4, 2015</th>
<th>28</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>May 4, 2015</th>
<th>29</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>May 4, 2015</th>
<th>30</th>
</tr>
</thead>
</table>
QSAT is **PSPACE**-complete

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

- **Proof:**
  - **in PSPACE:** \( \exists x_1 \forall x_2 \exists x_3 \cdots \exists x_n \varphi(x_1, x_2, \ldots, x_n) \)?
  - “\( \exists x_i \): for each \( x_i \), recursively solve \( \forall x_1 \exists x_2 \exists x_3 \cdots \exists x_n \varphi(x_1, x_2, \ldots, x_n) \)?
    - if encounter “yes”, return “yes”
  - “\( \forall x_i \): for each \( x_i \), recursively solve \( \exists x_1 \forall x_2 \exists x_3 \cdots \exists x_n \varphi(x_1, x_2, \ldots, x_n) \)?
    - 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

---

QSAT is **PSPACE**-complete

- for \( i = 0, 1, \ldots, n^k \) produce quantified Boolean expressions \( \psi_i(A, B, W) \)
- convert \( \psi_i \) to 3-CNF \( \varphi \)
  - add variables \( V \)
  - hardwire \( A = \text{START}, B = \text{ACCEPT} \)
  - \( \exists w_1 \forall w_2 \cdots \forall w_c \varphi(W, V) \Leftrightarrow x \in L \)