Oracle Turing Machine: (oTM)

- multi-tape TM with special "query" tape.
- Special states: \( q_? \) , \( q_{yes} \) , \( q_{no} \)

on input \( x \), \( A \)/ oracle language \( A \)
- \( M_A \) runs as usual except...
  - if \( M_A \) enters \( q_? \):
    - \( y = \) contents of query tape.
    - if \( y \in A \) \( \Rightarrow \) transition to \( q_{yes} \)
    - if \( y \notin A \) \( \Rightarrow \) transition to \( q_{no} \).

Non-deterministic oTM: defined the same way
- transition is a relation instead of a function.

Oracle is like a subroutine,
- each call counts as only one step.

polynomial oTM with a SAT oracle can solve
- given \( \phi_1, \phi_2, \ldots, \phi_n \) are \( \Phi \), even if \( \Phi \) is in PSPACE?

Shorthand: applying oracle to entire complexity class.
- complexity class \( C \)
- language \( A \).

\( C^A = \{ L \text{ decidable by } OTM \ M \text{ w/ oracle } A \} \)
- with \( M \) in \( C \), \( \text{example } P^{NP} \)

Another shorthand: using a complexity class
as an oracle:
OM M
Complexity class C

\[ M^C \text{ decides language } L \text{ if for some } A \in C, \]
\[ M^A \text{ decides } L. \]

Both together: \( \mathcal{C}^D = \) languages decidable by OM in C
w/ oracle language in D.
\[ \text{ex: } \text{PSPACE} = \text{P}^{\text{NP}} \]

We can use these definitions to define lots of complexity classes.
- Which ones have natural complete problems?
- Have natural interpretation using alternation.
- Help vs to state consequences, constraints.

\[ \Sigma_0^p = \Pi_0^p = \text{P}. \]
\[ \Delta_1 = \text{P}^\text{P} \quad \Sigma_1 = \text{NP} \quad \Pi_1 = \text{co-NP} \]
\[ \Delta_2 = \text{P}^{\text{NP}} \quad \Sigma_2 = \text{NP}^{\text{NP}} \quad \Pi_2 = \text{co-NP}^{\text{NP}} \]
\[ \Delta_i \Pi_i = \text{P}^{\Sigma_i} \quad \Sigma_i \Pi_i = \text{NP}^{\Sigma_i} \quad \Pi_i \Pi_i = \text{co-NP}^{\Sigma_i} \]

**Polynomial Hierarchy:** \( \text{PH} = \bigcup_i \Sigma_i \)

**Examples:** \text{MINCIRCUIT} \( \in \Sigma_2 \).
\( \rightarrow \) is there a circuit of fewer than \( s \) gates

**Input:** \((C, s)\) that computes the same function as \( \text{C} \)?
1. \( C \) is a circuit
2. \( s \) is integer.
do C + C' empirically the same for \( \leq \text{co-NP} \).
do C + C' differ on some input? \( \leq \text{co-NP} \)

guess C': consult oracle on equivalence.

\( \leq \text{co-NP} \) at most 8 gates.

**Exact TSP:** Given a weighted graph \( G \), integer \( k \), is the \( k \)-th bit of the description of the shortest TSP tour in \( G \) a 1?

**Exact TSP:** (Binary search on TSP length).

\( \text{Exp} \) \( \text{PSPACE} \)

\( \text{P} \)

\( \text{NP} \)

\( \text{co-NP} \)

\[ L \leq \Sigma_i \text{ if it is expressible as:} \]
\[ L = \exists x \exists y \mid y \leq |x|^k \mid (x,y) \in R^3 \]
\[ R \leq \Pi_i \]

\[ L \leq \Pi_i \text{ if it is expressible as:} \]
\[ L = \exists x \mid \forall y \mid y \leq |x|^k \mid (x,y) \in R^3 \]
\[ R \leq \Sigma_{i-1} \]

**Niuw more usable version:**

\[ L \leq \Sigma_i \text{ iff expressible as} \]
\[ L = \exists x \mid \forall y_1, \forall y_2, \forall y_3 \ldots \forall y_i \mid (x,y_1,\ldots,y_i) \in R^3 \]
\[ \text{if } \text{i odd: } Q = \bot \text{ if i even: } Q = \top. \]

\( Q = \top \text{ if } \text{i odd} \)
\( Q = \bot \text{ if } \text{i even} \).

\[ L \leq \Pi_i \text{ iff expressible as} \]
\[ L = \exists x \mid \forall y_1, \forall y_2 \ldots \forall y_i \mid (x,y_1,\ldots,y_i) \in R^3 \]
By induction on $i$:

For $i = 1$:

$\Sigma_1 = \text{NP}

L \in \text{NP} \iff \exists k, R \text{ poly time}

L = \{x \mid \exists y_1, y_2, \ldots, y_{k-1}, (x, y_1, y_2, \ldots, y_{k-1}) \in R \}

(i)$

Note: if $x \in L$ then $y_1, y_2, \ldots, y_{k-1} \notin R$.

Now consider $L \in \Sigma_1 = \text{NP}$.

This $\Sigma_1$ gets many queries to $\Sigma_{i-1}$
but we need to compress these into a single query.

There is a poly-time NTM $M$ on alphabet $A \in \Sigma_{i-1}$

Call this machine $M_A$

$x \in L \iff M_A \text{ accepts } x.$

Now suppose we guess $M_A$'s non-deterministic choices: $y_1, \ldots, y_k$.

We also guess the inputs to the queries it makes to $A$:

$U_1, \ldots, U_k$.

We also guess the outcomes of those queries.

If the guesses are all correct:

$U_j = 1 \Rightarrow \exists y_1, y_2, \ldots, y_{k-1}, (u_j, y_1, y_2, \ldots, y_{k-1}) \in R.$

$U_j = 0 \Rightarrow \forall y_1, y_2, \ldots, y_{k-1}, (u_j, y_1, y_2, \ldots, y_{k-1}) \notin R.$

$x \in L$ iff there is a good set of choices.

$M_A$ on non-deterministic choices $y$ asks $y_1, \ldots, y_k$.

It answers $U_1, \ldots, U_k$.

Accepts if $x \in L$. 

[Handwritten notes and arrows]
Note: this is assuming answers are correct. The query $z_i$ can depend on answers $y_1, \ldots, y_i$, as well as $y$.  

This part verifies that the oracle queries are correct.

We can rearrange the above expression to look like:

$$\forall x \exists y_1 \forall y_2 \ldots \forall y_i \exists y_{i+1} (x, y_1, \ldots, y_i) \in R$$

This is poly time decidable.

$$L = \exists x \forall y_1 \forall y_2 \ldots \forall y_i \exists y_{i+1} (x, y_1, \ldots, y_i, y_{i+1}) \in R$$

**Proof:** $L \in NP^{z_i}$

Consider the language $L' = \exists(x, y_1) \forall y_2 \ldots \forall y_i \exists y_{i+1} (x, y_1, \ldots, y_i) \in R$

$L' \in \Pi_{i-1} \Rightarrow \overline{L} \in \Sigma_{i-1}$

An NP machine $M$ with oracle $L'$:

Guess $y_1$

Then guess whether $(x, y') \in \overline{L}$

If oracle answers 'yes' → reject

If oracle answers 'no' → accept

$x \notin L' \Rightarrow x \in \overline{L}$.

**PSPACE:** $\forall \forall \ldots \exists \varphi(x_1, \ldots, x_n) \leftrightarrow \text{poly} \# \# \text{alternations}$.

# Alternations can depend on the size of the input.

**PH:** any constant # of alternations.
Polytope Hierarchy: Complete problems.

Three variations of SAT

$QSAT_i \quad (i$ odd$) \quad \exists 3 - \text{CNF}'s \quad \phi(x_1, \ldots, x_i) \text{ for which}$

$\exists x, \forall x_2 \ldots \exists x_i \phi(x, x_2, \ldots, x_i) = 1 \land$

$QSAT_i \quad (i$ even$) \quad \exists 3 - \text{DNF}'s \quad \phi(x_1, \ldots, x_i) \text{ for which}$

$\exists x, \forall x_2 \ldots \forall x_i \phi(x_1, \ldots, x_i) = 1 \land$

$QSAT \quad \exists 3 - \text{CNF}s \quad \phi \text{ for which}$

$\forall x_1 \forall x_2 \ldots \forall x_n \phi(x_1, \ldots, x_n) = 1 \land$

Theorem: $QSAT_i \text{ is } \Sigma_i\text{-complete.}$

Clearly $QSAT_i \in \Sigma_i.$

Assume $i$ odd, let $\Sigma_i$ in form

$\exists x_1 \exists y_1, \forall x_2 \ldots \exists y_i \phi(x_1, y_1, \ldots, y_i) \in R?.$

If $(x, y_1, \ldots, y_i)$ are fixed we have polynomial time TH that decides $R.$

Take computation tableau for $R$'s computation on input $(x, y_1, \ldots, y_i)$

and make it into a poly-sized circuit as we did

in the proof that $CVAL$ is $P$-complete.

Use the circuit $c$ to form a $\text{CNF}$ formula:

for example:

Use the circuit $c$ to form a $\text{CNF}$ formula:

for example:

$\overline{z} \iff x_1 \land \overline{x}_2$

$\iff (\overline{z} \lor x_1) \land (\overline{z} \lor \overline{x}_2)$

$\overline{z} \lor (x_1 \land \overline{x}_2)$

$\iff (\overline{z} \lor \overline{x}_1) \land (\overline{z} \lor \overline{x}_2)$

$\overline{z} \lor (x_1 \land \overline{x}_2)$

These are auxiliary variables.
Then add the clause \((2n)\).

\[ \exists \exists \phi(x, y_1, \ldots, y_i, z) = 1 \iff c(x, y_1, \ldots, y_i) = 1. \]

So:

\[ x \in L \iff \exists y_1 \forall y_2 \ldots \exists y_i : c(x, y_1, \ldots, y_i) = 1 \iff \exists y_1 \forall y_2 \ldots \forall y_i : \phi(x, y_1, \ldots, y_i, z) = 1. \]

For even \(i\): \( L \in \Sigma_i \)

\[ \exists x | \exists y_1 \forall y_2 \ldots \forall y_i (x, y_1, \ldots, y_i) \in R \subseteq. \]

\[ \begin{tikzpicture}
  \node (L) at (0,0) {Lift \((x, y_1, \ldots, y_i) \in R\)};
  \node (C) at (0,2) {C \rightarrow \text{Sieve circuit from CVAL reduction for } R.};
  \node (Z) at (0,4) {\text{Convert as before to } 3\text{CNF formula } \phi.}
  \node (N) at (0,6) {\text{except add a NOT gate at the end.}};
  \node (B) at (0,8) {\text{3CNF}};
  \node (D) at (0,10) {\text{By DeMorgan, this becomes } 3\text{DNF}};
\end{tikzpicture} \]

For a fixed \(x, y_1, \ldots, y_i\): \(c(x, y_1, \ldots, y_i) = 0 \iff \forall z \phi(x, y_1, \ldots, y_i) = 0. \iff \forall z \neg \phi(x, y_1, \ldots, y_i) = 1 \]

\[ \exists y_1 \forall y_2 \ldots \forall y_i : \forall z : \phi(x, y_1, \ldots, y_i) = 1 \]

\[ \iff \exists y_1 \forall y_2 \ldots \forall y_i : c(x, y_1, \ldots, y_i) = 0 \iff x \in L. \]

Because of the NOT gate at the end.

\(Q\text{SAT} \text{ is } PSPACE\text{-complete.}\)

\[ \exists x_1 \forall x_2 \ldots Q x_n \phi(x_1, \ldots, x_n) \]

\[ \exists x_1 \forall x_2 \ldots Q x_n \phi(x_1, \ldots, x_n). \]
QSAT ∈ PSPACE: \( \exists x_1 \ldots \exists x_n \)

For each value \( x_1 \), recursively solve

\[ \forall x_2 \ldots \exists x_n \phi(x_1, \ldots, x_n) \]

if yes, then return yes.

Return 'no'.

\[ \forall x_1 \ldots \exists x_n \phi(x_1, \ldots, x_n) \]

For each value for \( x_1 \), recursively solve

\[ \exists x_2 \ldots \forall x_n \phi(x_1, \ldots, x_n) \]

if no, return no.

Return 'yes'.

Base case 3 CNF expression w/ all variables determined (CVAL).

poly (n) recursive depth.

poly (n) bits of state @ each level.

Now for each L ∈ PSPACE \( \Leftarrow \) QSAT.

\( 2^{nk} \) possible configurations expressible as a vector

of variables \( \hat{a}, \hat{b} \).

Single start, single accept.

Define \( \text{REACH} \) \( (X, Y; i) \leftrightarrow \) configuration \( Y \) reachable from

\( X \) in \( \leq 2^i \) steps.

Produce 3CNF \( \phi(w_1, w_2, \ldots, w_m) \) s.t.

\[ \exists w_1 \forall w_2 \ldots \exists w_m \phi(w_1, \ldots, w_m) \leftrightarrow \text{REACH}(\text{start}, \text{accept}, n^k) \]

Def: \( \alpha_i (A, B) = \exists w_1 \forall w_2 \ldots \forall w_i \phi_i (A, B, w_1 \ldots w_i) \)

\[ \leftrightarrow \text{REACH}(A, B, i) \]
\[ \Phi_0 = \Psi_0(A,B) = \text{true} \iff A = B \text{ or } A \text{ yields } B \text{ in one step of } M. \]

This can be expressed as a Boolean expression of size \(n^k\).

The length of \(A + B\) depends on \(x\) but otherwise, \(i\)'s depends only on \(M\).

Key idea: \( \text{REACH}(A, B, i+1) \Leftrightarrow \exists z \left[ \text{REACH}(A, z, i) \land \text{REACH}(z, B, i) \right] \)

This would get exponentially large!

So we can't do: \( \Psi_{i+1}(A, B) = \exists z \left[ \Psi_i(A, z) \land \Psi_i(z, B) \right] \)

Instead: \( \Psi_{i+1}(A, B) = \exists z \forall x \forall y \left[ (x = A \land y = z) \lor (x = z \land y = B) \Rightarrow \Psi_i(x, y) \right] \)

Note that \(\Psi_i\) has quantifiers, but they don't bind \(A, B, x, y\) on \(z\), so they can be moved to the front.

\[ |\Phi_0| = O(n^k) \]
\[ |\Psi_i| = O(n^k) + |\Psi_i|\]  
Total size: \(\text{poly}(n)\).

This is a log-space reduction.

The only part specific to \(\chi\) is \(\Psi_0\) (and then only the size of the tape). Also hard coding \(\text{START} \Rightarrow \text{ACCEPT}\).

Final: \( \exists z \forall x \forall y \left[ (x = \text{START} \land y = z) \lor (x = z \land y = \text{acc}) \Rightarrow \Psi_i(x, y) \right] \)
**PH Collapse**

Theorem: If \( \Sigma_i = \Pi_i \) then for all \( j > i \),
\[
\Sigma_j = \Pi_j = \Delta_j = \Sigma_j
\]

"The polynomial hierarchy collapses to the \( i \)th level."

**Proof**

It's enough to show \( \Sigma_i = \Sigma_{i+1} \)

\[
L \in \Pi_{i+1} \Rightarrow L \in \Sigma_{i+1} \Rightarrow \exists L \in \Sigma_i \Rightarrow L \in \Pi_i \Rightarrow \Sigma_i \text{ and } \Pi_{i+1}
\]

then \( \Sigma_{i+2} = \text{NP} \Sigma_{i+1} = \text{NP} \Sigma_{i} = \Sigma_{i+1} \)

Now to show \( \Sigma_i = \Sigma_{i+1} \)

\[
L \in \Sigma_{i+1} \iff \text{expressible as:}
L = \exists x \mid \exists y, (x,y) \in R \exists z \mid R \in \Pi_i.
\]

Hypothesis is \( \Pi_i = \Sigma_i \)

\[
R = \exists (x,y) \mid \exists z, \ (x,y,z) \in R' \exists z \mid R' \in \Pi_{i-1}
\]

\[
L = \exists x \mid \exists y, z, \ (x,y,z) \in R' \exists z \mid R' \in \Pi_{i-1} \Rightarrow L \in \Sigma_i.
\]

**Natural Complete Problems in PH**

We have already seen versions of SAT that are complete for each level of the PH = PSPACE.

In the PH, almost all natural complete problems lie in 2nd or 3rd tier of the hierarchy.
Natural complete problem for PSPACE: games.

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

In PSPACE: $\phi_i(V_1, \ldots, V_i): s V_1 \cdots V_i$ is a losing pass.

Expressible as a poly-size boolean formula.

$\exists s, V_1, V_2, \ldots, V_i$ such that

$(s, V_1, V_2) \ldots (V_i-1, V_i)$ all edges.

$s, V_1, \ldots, V_i$ all distinct

$(V_i = s) \lor (V_i = V_1) \lor \ldots \lor (V_i = V_i)$

$\forall V_1 \forall V_2 \ldots \forall V_i \lor \phi_i$.

Now: QSAT $\leq$ GEOG:

Player 1 trying to make every clause true.

Player 2 trying to make a clause false.

Q Boolean Formula $\rightarrow (G, s)$.

Player tries to pick an unsatisfied clause.

Given clause, Player 1 tries to find a literal inside the chosen clause that is true.
Karp - Lipton Theorem

We know that if $P = NP$ then SAT has poly-sized circuits. What about the converse of this statement? The converse holds if we restrict our attention to uniform circuit families.

We will show that if SAT has poly-size (non-uniform) circuits, then the PH collapses to the 2nd level.

**Theorem:** $NP \leq P/poly \implies PH = \Sigma_2$

It suffices to show that $\Sigma_2 \leq \Sigma_2$.

Will show a $\Pi_2$-complete problem can be done in $\Sigma_2$.

\[
\forall u \in \{0,1\}^n \exists v \in \{0,1\}^n \phi(u,v) = 1. \]

Fixing $v$ results in an instance of SAT w.r.t. $v$.

$NP \leq P/poly \implies p(n)$-sized circuit family

that solves SAT.

For every boolean formula $\phi \in \{0,1\}^n$

$C_m(\phi, u) = 1$ if $\exists v \phi(u,v) = 1$.

$m = |\phi(u)|$ (why is this true?)

$C_m$ solves the decision problem for SAT. This can be converted to a circuit that finds the solution $v$ if it exists.
For $i = 1$ to $n$

- IS $\phi_n(t_1, ..., t_i-1, 0, v_i, ..., v_n)$ Satisfies $\phi$ here?
  - YES $\Rightarrow t_i = 0$
  - NO $\Rightarrow t_i = 1$

This algorithm can be encoded in a circuit $C'_n$.

This gives a $g(n)$-size circuit family $\{C'_n\}_{n \in \mathbb{N}}$.

For every $\phi + n$ if $\exists v$ s.t. $\phi(v, v) = 1$, then $v$ is a

$NP \subseteq P/poly$ implies the existence of such a $C'$.

$C'$ can be guessed using the $\exists$ quantifier.

$C'_n$ can be guessed using $g(n)$ bits.

$(+) \exists \bar{w} \in \{0,1\}^{g(n)}$ $\forall n \in \mathbb{N}$

using $\bar{w}$ to describe $C'_n \phi(u, C'_n(\bar{w}, u)) = 1$.

This can be done in poly time.

holds if $\forall u \exists v \phi(u, v) = 1$ 

(*)

If $(*)$ does not hold then

$\exists u \forall v \neg \phi(u, v)$

which means that $(+)$ will fail too.

because no circuit $C'$ will be able to find a $v$

that s.t. $\phi(u, v) = 1$.

Theorem: $BPP \subseteq \Sigma_2 \cap \Pi_2$

(We don't even know if $BPP \neq EXP$

but we expect that $\Sigma_2 \cap \Pi_2$

is much weaker than EXP)
It's enough to show that $\text{BPP} \subseteq \Sigma_2$ since $\text{BPP}$ is closed under complement.

Let $\text{BPP} \supseteq \bigcap \text{BPP} \rightarrow \bigcap \Sigma_2 \rightarrow \bigcap \Pi_2$.

First use error reduction on input of length $n$, use $m = \text{poly}(n)$ random bits to get:

$x \in L \Rightarrow \Pr_r [M(x,r) \text{ accepts}] \geq 1 - 2^{-n}$

$x \notin L \Rightarrow \Pr_r [M(x,r) \text{ accepts}] \leq 2^{-n}$.

For $x \in \{0,1\}^n$, let $S_x = \text{Set of Strings } r \text{ for which } M(x,r) \text{ accepts}$.

For $x \in L$ we will show how to check $|S_x| \geq (1 - 2^{-n})2^m$.

For $x \notin L$ we will show how to check $|S_x| \leq 2^{-n}2^m$.

For $S \subseteq \{0,1\}^m$, $u \in \{0,1\}^m$, define $S + u = \{x + u | x \in S\}$ (bit-wise x-or).

Let $k = \left\lceil \frac{m}{n} \right\rceil + 1$.

Lemma 1: for every $S \subseteq \{0,1\}^m$, $|S| \leq 2^{m-n}$

for every choice of $u_1, \ldots, u_k$ $U_{i=1}^k (S + u_i) \neq \{0,1\}^m$.

Proof (Simple counting argument)

$|S + u_i| = |S| = 2^{m-n}$

$U_{i=1}^k (S + u_i) = k \cdot |S| = k \cdot 2^{m-n}$

$= \left\lceil \frac{m}{n} \right\rceil 2^{m-n} < 2^m$.

Lemma 2: For every $S \subseteq \{0,1\}^m$ s.t. $|S| > (1 - 2^{-n})2^m$ there exist $u_1, \ldots, u_k$ s.t. $U_{i=1}^k (S + u_i) = \{0,1\}^m$.

(Proven below).
**Proof of Theorem from Lemma:**

\[ x \in L \iff \exists U_1 \ldots U_k \in \{0,1\}^n \forall r \in \{0,1\}^n \quad r \in U_{i=1}^k (S_x + u_i) \]

\[ \iff \exists U_1 \ldots U_k \in \{0,1\}^n \quad \forall r \in \{0,1\}^n \quad \forall i = 1, \ldots, k \quad \mathcal{M}(x, r + u_i) \text{ accepts} \]

**Note:**

\( \mathcal{M} \text{ accepts } (x, r + u_i) \iff r + u_i \in S_x \iff r \in S_x + u_i \)

\[ r \in U_{i=1}^k S_x + u_i \iff \forall i = 1, \ldots, k \quad r + u_i \in S_x \iff \forall i = 1, \ldots, k \quad \mathcal{M}(x, r + u_i) \text{ accepts} \]

Poly-time procedure expressible as a boolean formula.

**Proof of Lemma 2: Probabilistic Method.**

Pick \( U_1 \ldots U_k \) at random.

Will show \( \Pr \left[ \bigcup_{i=1}^k S_x + u_i = \{0,1\}^n \right] > 0 \)

So there exist \( U_1 \ldots U_k \) for which \( \mathcal{E} \) holds.

\[ \Pr \exists \text{ bad } r \text{ which is not in } \bigcup_{i=1}^k S_x + u_i = 1 - \Pr \left[ \bigcup_{i=1}^k S_x + u_i = \{0,1\}^n \right] \]

\[ \text{will show } < 1 \]

For fixed \( r \): \( r \in S_x + u_i \iff u_i \in S_x + r \)

\[ \Pr [u_i \notin S_x + r] < 1 - \frac{(1-2^{-n})2^k}{2^n} = 2^{-n} \]

\[ \Pr [r \text{ not in any } S_x + u_i] \leq \Pr [\text{all } u_i \notin S_x + r] \leq (2^{-n})^k < 2^{-n} \]

(All \( u_i \)'s chosen independently)

\[ \Pr [\text{fixed } r \notin \bigcup_{i=1}^k S_x + u_i] < 2^{-n} \]

\[ \Pr [\exists r \notin \bigcup_{i=1}^k S_x + u_i] < 2^{-n}. 2^{-n} < 1. \]