Note Title 5/4/2013

Oracle Turing Machine: (OTM)
multi-tape TM with special "query" tape.
special states: $q_?$ q_{yes} q_{no}

on input x, w/ oracle language A
M_A runs as usual except...
if M_A enters $q_?$
y = contents of query tape.
$y \in A \Rightarrow$ transition to q_{yes}
$y \notin A \Rightarrow$ 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.

polytime OTM with a SAT oracle can solve
given $\phi_1\ \phi_2 \cdots \phi_n$ are an even # satisfiable?

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

$C^A = \{ L$ decidable by OTM M w/ oracle A
with M in $C \}$ example P^{SAT}

Another shorthand: using a complexity class
as an oracle:

OTM M
Complexity class $\mathcal{C}$

$M^{\mathcal{C}}$ decides language L if for some $A \in \mathcal{C}$, M^A decides L.

Both together: $\mathcal{C}^{\mathcal{D}}$ = languages decidable by OTM in $\mathcal{C}$ w/ oracle language in $\mathcal{D}$.

ex: $P^{SAT} = P^{NP}$

We can use these definitions to define lots of complexity classes.
- which ones have natural complete problems?
- have natural interpretation using alternating quantifiers.
- help us to state consequences, constraints.

$$\Sigma_0 = \Pi_0 = P.$$

$$\Delta_1 = P^P \qquad \Sigma_1 = NP \qquad \Pi_1 = \text{co-}NP$$
$$\Delta_2 = P^{NP} \qquad \Sigma_2 = NP^{NP} \qquad \Pi_2 = \text{co-}NP^{NP}$$
$$\vdots$$
$$\Delta_{i+1} = P^{\Sigma_i} \qquad \Sigma_{i+1} = NP^{\Sigma_i} \qquad \Pi_{i+1} = \text{co-}NP^{\Sigma_i}$$

Polynomial Hierarchy: $PH = \bigcup_i \Sigma_i$

Examples: MINCIRCUIT $\in \Sigma_2$.
↳ is there a circuit w/ fewer than s gates that computes the same function as C?

Input (C, s)
C: ckt
s: integer.

do C + C' compute the same fcn $\in$ co-NP.

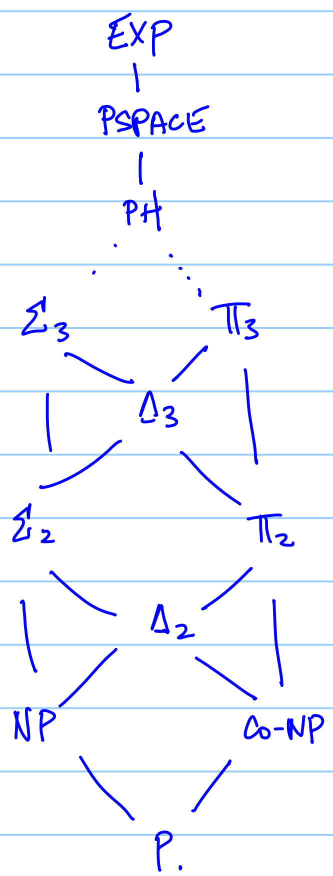
do C + C' differ on some input? $\in$ co-NP

Guess C' : consult oracle on equivalence.

↳ w/ at most s 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).



__Theorem__: $L \in \Sigma_i$ if it is expressible as:

$$L = \{ x \mid \exists y \ |y| \le |x|^k \ (x,y) \in R \}$$
$$R \in \Pi_i$$

$L \in \Pi_i$ if it is expressible as:

$$L = \{ x \mid \forall y \ |y| \le |x|^k \ (x,y) \in R \}$$
$$R \in \Sigma_{i-1}$$

Nicer more usable version:

$L \in \Sigma^i$ iff expressible as

$$L = \{ x \mid \exists y_1 \forall y_2 \exists y_3 \cdots Q y_i \ (x, y_1, \dots, y_i) \in R \}$$

if i odd : $Q = \exists$. if i even $Q = \forall$.

$L \in \Pi^i$ iff expessible as

$$L = \{ x \mid \forall y_1 \exists y_2 \cdots Q y_i \ (x, y, \dots y_i) \in R \}$$

$Q = \forall$ if i odd

$Q = \exists$ if i even.

$\Rightarrow \quad L \in \Sigma_i \qquad L \in NP^{\Sigma_{i-1}} = NP^{\Pi_{i-1}}$

↳ this because you can always reverse the output of the oracle.

$L \in \Sigma_{i-1} \iff co\text{-}L \in \Pi_{i-1}$

By induction on i: $\quad i = 1: \quad \Sigma_1 = NP$

$L \in NP \iff \exists\, k, R$ poly time

$L = \{x \mid \exists y \ |y| \le |x|^k \ \ R(x,y) = \text{yes}\}$

Assume $L \in \Sigma_{i-1} \iff \exists\, k$, poly time R

$L = \{x \mid \exists y_1, \forall y_2, \ldots, Q\, y_{i-1}, \ |y_j| \le |x|^k, \ (x, y_1, \ldots y_{i-1}) \in R\}$

note: if $x \notin L$ then $\forall y_1 \exists y_2 \cdots \bar{Q} y_{i-1} \ (x, y_1, \ldots, y_{i-1}) \notin R$.

Now consider $L \in \Sigma_i = NP^{\Sigma_{i-1}}$ → this NP gets many queries to Σ_{i-1} but we need to compress these into a single query.

There is a poly-time NTM w/ oracle $A \in \Sigma_{i-1}$

Call this machine M_A

$x \in L \iff M_A$ accepts x.

($A = \{x \mid \exists y_1 \forall y_2 \cdots Q\, y_{i-1} \ (x, y_1, y_2 \cdots y_{i-1}) \in R\}$ — poly time

y. Now suppose we guess M_A's non-deterministic choices: y

$z_1 \cdots z_k$ We also guess the inputs to the queries it makes to A

$u_1 \cdots u_k$ We also guess the outcomes of those queries.

If the guesses are all correct

$u_j = 1 \quad \Rightarrow \quad \exists y_1 \forall y_2 \cdots Q\, y_{i-1} \ (u_j, y_1, \ldots, y_{i-1}) \in R$

$u_j = 0 \quad \Rightarrow \quad \forall y_1 \exists y_2 \cdots \bar{Q}\, y_{i-1} \ (u_j, y_1, \ldots y_{i-1}) \notin R$

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

$\exists y\, z_1 \cdots z_k\, u_1 \cdots u_k$ [M_A w/ non-det choices y asks queries $z_1 \cdots z_k$ gets answers $u_1 \cdots u_k$ accepts.] poly-time checkable.

this part verifies that the oracle queries are correct.

Note this is assuming answers are correct. The query z_j can depend on answers $u_1 \cdots u_{j-1}$ as well as y.

AND for $u_j = 1$ $\exists y_1 \forall y_2 \cdots Q y_{i-1} \ (u_j, y_1, \ldots, y_{i-1}) \in R$

AND for $u_j = 0$ $\underline{\forall y_1} \exists y_2 \cdots \bar{Q} y_{i-1} \ (u_j, y_1, \ldots, y_{i-1}) \notin R$

the additional layer of alternation comes from this term.

We can rearrange the above expression to look like:

$$\{ x \mid \exists y_1 \forall y_2 \cdots Q y_i \ (x, y_1, \ldots, y_i) \in R \}$$

poly time decidable.

$\Leftarrow$ $L = \{ x \mid \exists y_1 \forall y_2 \cdots Q y_i \ (x, y_1, \ldots, y_{i-1}) \in R \}$.

prove $L \in NP^{\Sigma_{i-1}}$

Consider the language $L' = \{ (x, y_1) \mid \forall y_2 \cdots Q y_i \ (x, y_1, \ldots, y_i) \in R \}$

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

An NP machine w/ oracle L':

Guess y_1

then query whether $(x, y') \in co\text{-}L'$

if oracle answers 'yes' $\rightarrow$ reject

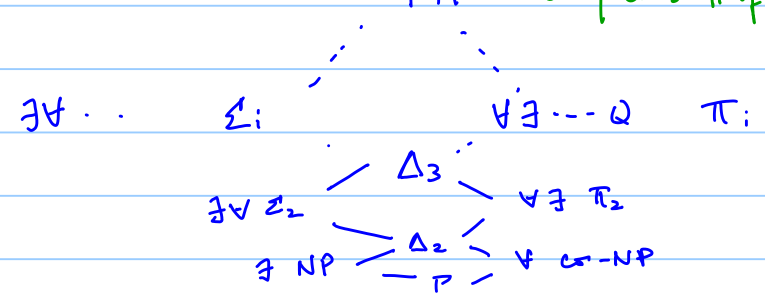
if oracle answers 'no' $\rightarrow$ accept.

$x \notin co\text{-}L' \Rightarrow x \in L'$.

PSPACE $\exists \forall \exists \forall \cdots \phi(x_1, \ldots, x_n)$ $\leftarrow$ poly # of alternations.

\# of alternations can depend on the size of the input.

PH ~ any const # of alternations.



Polynomial Hierarchy: Complete problems.

Three variations of SAT

QSAT$_i$ (i odd) $\{$ 3-CNF's $\phi(\vec{x}_1, .., \vec{x}_i)$ for which $\exists \vec{x}_1 \forall \vec{x}_2 \cdots \exists \vec{x}_i\ \phi(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_i) = 1 \}$

QSAT$_i$ (i even) $\{$ 3-DNF's $\phi(\vec{x}_1, .., \vec{x}_i)$ for which $\exists x_1 \forall x_2 \ldots\ldots \forall x_i\ \phi(\vec{x}_1, .., \vec{x}_i) = 1 \}$

QSAT $\{$ 3-CNFs ϕ for which $\exists x_1 \forall x_2 \cdots Q_n\ \phi(x_1, \ldots, x_n) = 1 \}$

Theorem: QSAT$_i$ is Σ_i - complete.

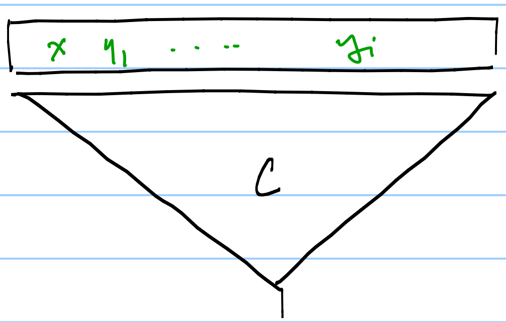
Clearly QSAT$_i \in \Sigma_i$

Assume i odd. $L \in \Sigma_i$ in form

$\{ x \mid \exists y_1 \forall y_2 \ldots \exists y_i\ (x, y_1, \ldots, y_i) \in R \}$

If $(x, y_1, \ldots, y_i)$ are fixed we have poly time TM 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.



1 iff $(x, y_1 \cdots y_i) \in R$.

Use the circuit c to construct a CNF formula:

for example:

$(z \Leftrightarrow x_1 \wedge x_2)$

new clauses.

$\neg z \vee (x_1 \wedge x_2) = (\neg z \vee x_1) \wedge (\neg z \vee x_2)$

$\neg(x_1 \wedge x_2) \vee z \equiv (\neg x_1 \vee \neg x_2 \vee z)$

the z's are auxiliary varibles.

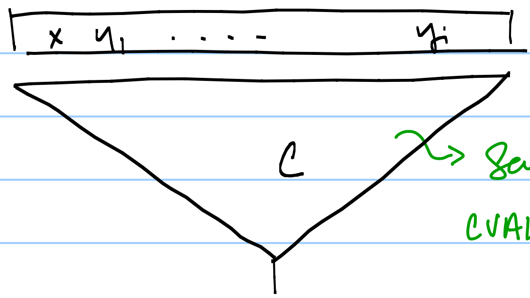
then add the clause (z_n).

So: $\exists \vec{z}\ \phi(x, y_1, \ldots, y_i, z) = 1 \iff C(x, y_1, \ldots, y_i) = 1.$

$x \in L \iff \exists y_1 \forall y_2 \ldots \exists y_i\ C(x, y_1, \ldots, y_i) = 1$

$\iff \exists y_1 \forall y_2 \ldots \exists y_i \exists z\ \phi(x, y_1, \ldots, y_i, z) = 1.$

for even i: $L \in \Sigma_i$ $\{x \mid \exists y_1 \forall y_2 \ldots \forall y_i\ (x, y_1 \cdots y_i) \in R\}$.



Same circuit from CVAL reduction for R.

1 iff $(x, y_1, \ldots, y_i) \in R$.

Convert as before to 3CNF formula Φ. except add a NOT gate at the end.

For a fixed $x\, y_1 \cdots y_i$ $C(x, y_1, \ldots, y_i) = 0 \iff \forall z\ \overbrace{\phi(x_1, y_1, \ldots, y_i) = 0}^{3CNF}.$

$\iff \forall z\ \underbrace{\neg \phi(x_1, \ldots, y_i)}_{\text{By DeMorgan, this becomes 3DNF}} = 1$

$\exists y_1 \forall y_2 \ldots \forall y_i \forall z\ \underbrace{\phi'(x, y_1, \ldots, y_i, z)}_{\phi' = \neg\phi} = 1$

$\iff \exists y_1 \forall y_2 \ldots \forall y_i\ C(x, y_1, \ldots, y_i) = 0 \iff x \in L.$ //

because of the NOT gate at the end.

QSAT is PSPACE-complete.

$\{\forall x_1 \exists x_2 \ldots Q x_n\ \phi(x_1, \ldots, x_n)$

$\exists x_1 \forall x_2 \ldots Q x_n\ \phi(x_1, \ldots, x_n).$

QSAT $\in$ PSPACE: $\exists x_1 \cdots Qx_n$

For each value x_1 recursively solve

$\forall x_2 \cdots Qx_n\ \phi(x_1,\ldots,x_n)$

if yes, then return yes.

Return 'no'.

$\forall x_1 \cdots Qx_n\ \phi(x_1,\ldots,x_n)$

For each value for x_1, recursively solve

$\exists x_2 \cdots Qx_n\ \phi(x_1,\ldots,x_n)$

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 $L \in$ PSPACE $L \propto$ QSAT.

2^{n^k} possible configurations expressible as a vector of variables $\vec{a}$, $\vec{b}$.

Single start, single accept.

define REACH $(X, Y, i) \longleftrightarrow$ configuration Y reachable from X in $\leq 2^i$ steps.

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

$\exists w_1 \forall w_2 \ldots Qw_n\ \phi(w_1,\ldots,w_m) \longleftrightarrow$ REACH(start, accept, n^k).

Def: $\psi_i(A,B) = \exists w_1 \forall w_2 \cdots Qw_i\ \phi_i(A, B, w_1 \ldots w_i)$

$\Longleftrightarrow$ REACH(A, B, i)

$\phi_0 = \psi_0(A,B) =$ true iff $A = B$ or A yields B in one step of M.

this can be expressed as a Boolean expression of size n^k.
the length of A & B depend on x but otherwise, this depends only on M.

Key idea: $REACH(A,B,i+1) \iff \exists z\,[REACH(A,z,i) \wedge REACH(z,B,i)]$

this would get exponentially large!

So we can't do: $\psi_{i+1}(A,B) = \exists z\,[\psi_i(A,z) \wedge \psi_i(z,B)]$

Instead: $\psi_{i+1}(A,B) = \exists z\,\forall x\,\forall y\,[(X=A \wedge Y=z) \vee (X=z \wedge Y=B) \Rightarrow \psi_i(X,Y)]$

Note that ψ_i has quantifiers, but they don't bind A, B, X, Y or z, so they can be moved to the front

$|\psi_0| = O(n^k)$

$|\psi_{i+1}| = O(n^k) + |\psi_i|$ total size: poly(n).

This is a log-space reduction.

The only part specific to x is ψ_0 (and then only the size of the tape). Also hard coding START + ACCEPT.

Final: $\exists z\,\forall x\,\forall z\,[(X = START \wedge Y = z) \vee (X = z \wedge Y = ACC) \Rightarrow \psi_{nk}(X,Y)]$

PH Collapse

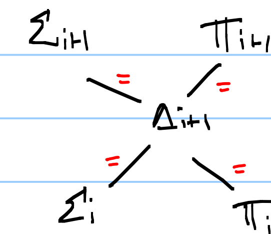
Theorem: if $\Sigma_i = \Pi_i$ then for all $j > i$,
$$\Sigma_j = \Pi_j = \Delta_j = \Sigma_i$$
"the polynomial hierarchy collapses to the i^{th} level."

Proof

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

$$L \in \Pi_{i+1} \Rightarrow \bar{L} \in \Sigma_{i+1} \Rightarrow \bar{L} \in \Sigma_i \Rightarrow L \in \Pi_i$$

then $\Sigma_{i+2} = NP^{\Sigma_{i+1}} = NP^{\Sigma_i} = \Sigma_{i+1}$, etc.



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

$L \in \Sigma_{i+1}$ iff expressible as:
$$L = \{ x \mid \exists y \ (x,y) \in R \} \qquad R \in \Pi_i$$
Hypothesis is $\Pi_i = \Sigma_i$

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

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

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 2^{nd} or 3^{rd} tier of the hierarchy.

Natural complete problem for PSPACE: games.

To review: players take turns selecting an edge from the current node to any unvisited node. Player loses if they have no valid edge to pick.

GEOGRAPHY = { (G, s) : G is an undirected graph and player 1 can win from starting point s }.

Theorem: GEOGRAPHY is PSPACE-complete.

In PSPACE: $\phi_i(v_1, \ldots, v_i)$: $s v_1 \cdots v_i$ is a losing path.

i even

expressible as poly-size boolean formula.

$(s, v_1)\ (v_1, v_2)\ \cdots\ (v_{i-1}, v_i)$ all edges.

$\{s, \ldots, v_{i-1}\}$ all distinct

$(v_i = s) \vee (v_i = v_1) \cdots (v_i = v_{i-1})$

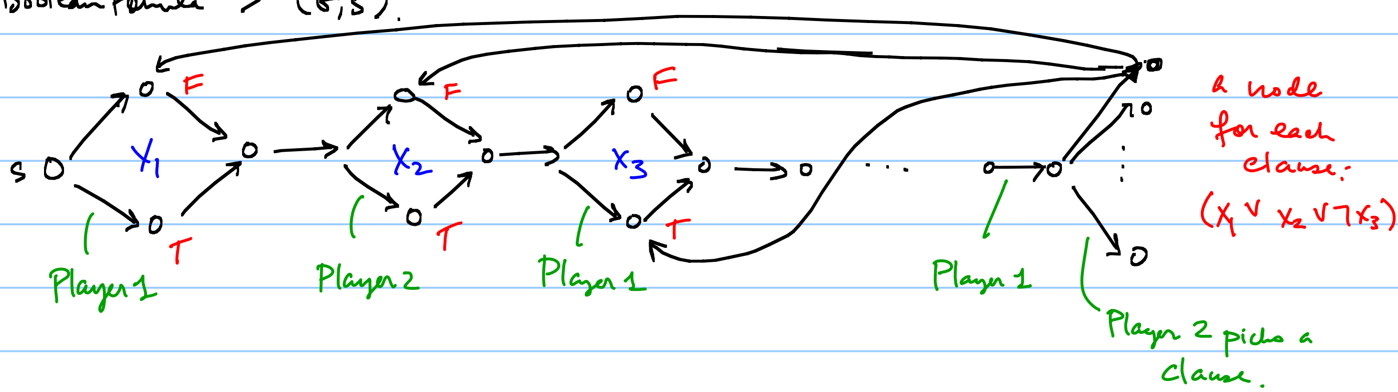
$$\exists v_1 \forall v_2 \cdots \forall v_n \bigvee_{i \text{ even}} \phi_i.$$

Now: QSAT $\propto$ GEOG:

Player 1 trying to make every clause true

Player 2 trying to make a clause false.

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



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: $NP \subseteq P/poly \Rightarrow PH = \Sigma_2$

It suffices to show that $\Pi_2 \subseteq \Sigma_2$.

Will show a Π_2-complete problem can be done in Σ_2.

$$\forall u \in \{0,1\}^n \ \exists v \in \{0,1\}^n \ \underbrace{\phi(u,v)}_{\text{Boolean formula.}} = 1.$$

Fixing u results in an instance of SAT w/ n vars.

$NP \subseteq P/poly \rightarrow p(n)$- sized circuit family that solves SAT.

For every boolean formula ϕ & $u \in \{0,1\}^n$

$$C_m(\phi, u) = 1 \quad \text{iff} \quad \exists v \ \phi(u,v) = 1.$$

$m = |(\phi,u)|$ (why not size of ϕ?).

C_m solves the decision problem for SAT. This can be converted to a circuit that finds the solution. v if it exists.

Hard-code some of the input vars of ϕ: Variable inputs $v_1 \dots v_n$
hard coded inputs $t_1 \dots t_n$.

For $i = 1$ to n
  Is $\phi_n(t_1, \dots, t_{i-1}, 0, v_{i+1}, \dots, v_n)$ satisfiable?
    YES $\Rightarrow$ $t_i \leftarrow 0$
    NO $\Rightarrow$ $t_i \leftarrow 1$

it may be a different m for C_m in each iteration because input size may change. That's ok.

this algorithm can be encoded in a circuit C'_m

This gives a $q(n)$-sized circuit family $\{C'_n\}_{n \in \mathbb{N}}$.
For every ϕ + u if $\exists v$ s.t. $\phi(u,v) = 1$, it outputs v.

NP $\subseteq$ P/poly implies the existence of such a C'
C' can be guessed using the $\exists$ quantifier.
C'_m can be guessed using $c\,q(n)$ bits.

(+) $\exists w \in \{0,1\}^{c\,q(n)}$ $\forall u \in \{0,1\}^n$
$\underbrace{\text{using } w \text{ to describe } C'_m \quad \phi(u, C'_m(\phi, u)) = 1}_{\text{this can be done in poly time.}}$.

holds if $\underbrace{\forall u \, \exists v \; \phi(u,v) = 1}_{(*)}$

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 forces $\phi(u,v) = 1$.

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 $BPP \subseteq \Sigma_2$ since BPP is closed under complement.

$$L \in BPP \Rightarrow \bar{L} \in BPP \Rightarrow \bar{L} \in \Sigma_2 \Rightarrow L \in \Pi_2.$$

First use error reduction on input of length n, use $m = poly(n)$ 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 = set of strings r for which $M(x,r)$ accepts.

$$\left.\begin{array}{ll} \text{for } x \in L & |S_x| \geq (1-2^{-n})2^m \\ \text{for } x \notin L & |S_x| \leq 2^{-n}2^m \end{array}\right\} \text{ we will show how to check which using only two quantifiers.}$$

For $S \subseteq \{0,1\}^m$, $u \in \{0,1\}^m$, define $S+u = \{x+u \mid x \in S\}$

↳ bit-wise x-or.

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

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

for every choice of $u_1, \ldots, u_k$

$$\cup_{i=1}^{k} (S+u_i) \neq \{0,1\}^m$$

<u>Proof</u> (Simple counting argument)

$$|S+u_i| = |S| = 2^{m-n} \qquad \left|\cup_{i=1}^{k}(S+u_i)\right| \leq k|S| = k\,2^{m-n}$$
$$= \lceil \tfrac{m}{n} \rceil 2^{m-n} < 2^m$$

//.

<u>Lemma 2</u>: For every $S \subseteq \{0,1\}^m$ s.t. $|S| > (1-2^{-n})2^m$

$\exists u_1 \cdots u_k$ s.t. $\cup_{i=1}^{k} (S+u_i) = \{0,1\}^m$

(Proven below).

Proof of theorem from Lemma:

$$x \in L \iff \exists\, u_1 \cdots u_k \in \{0,1\}^m\ \forall r \in \{0,1\}^m \quad r \in \bigcup_{i=1}^{k} (S_x + u_i)$$

$$\iff \exists\, u_1 \cdots u_k \in \{0,1\}^m\ \forall r \in \{0,1\}^m \quad \underbrace{\bigvee_{i=1}^{k} M(x, r+u_i) \text{ accepts.}}_{\text{poly-time procedure expressable as a boolean formula.}}$$

Note M accepts $(x, r+u_i) \iff r+u_i \in S_x \iff r \in S_x + u_i$

$r \in \bigcup_{i=1}^{k} S_x + u_i \iff \bigvee_{i=1}^{k} r+u_i \in S_x \iff \bigvee_{i=1}^{k} M(x, r+u_i) = \text{accept.}$

Proof of Lemma 2: Probabilistic Method.

pick $u_1 \cdots u_k$ at random.

will show $\Pr[\underline{\bigcup_{i=1}^{k} S_x + u_i = \{0,1\}^n}] > 0$

so there exist $u_1 \cdots u_k$ for which $\uparrow$ holds.

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

$\llcorner$ will show < 1

For fixed r: $r \in S_x + u_i \iff u_i \in S_x + r$

$$\text{Prob}[u_i \notin S_x + r] < 1 - \frac{(1-2^{-n})2^m}{2^m} = 2^{-n}$$

$$\text{Prob}[r \text{ not in any } S_x + u_i] \iff \text{Prob}[\text{all } u_i \notin S_x + r]$$

$$\le \underbrace{(2^{-n})^k}_{\text{all } u_i\text{'s chosen independently.}} < 2^{-m}$$

$$\text{Prob}[\text{fixed } r \notin \bigcup_{i=1}^{k} S_x + u_i] < 2^{-m}.$$

$$\text{Prob}[\exists r \in \bigcup_{i=1}^{k} S_x + u_i] < 2^{-m} \cdot 2^m < 1.$$