CS151
Complexity Theory

Lecture 12.5

May 8, 2021

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

$\Sigma_0 = \Pi_0 = \mathbf{P}$

$\Delta_1 = \mathbf{P}^{\mathbf{P}}$ $\Sigma_1 = \mathbf{NP}$ $\Pi_1 = \mathbf{coNP}$

$\Delta_2 = \mathbf{P}^{\mathbf{NP}}$ $\Sigma_2 = \mathbf{NP}^{\mathbf{NP}}$ $\Pi_2 = \mathbf{coNP}^{\mathbf{NP}}$

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

Polynomial Hierarchy **PH =** $\cup_i \Sigma_i$

The Polynomial-Time Hierarchy

$\Sigma_0 = \Pi_0 = \mathbf{P}$

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

- 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 $\in \Sigma_2$

The Polynomial-Time Hierarchy

$\Sigma_0 = \Pi_0 = \mathbf{P}$

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

- 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 $\in \Delta_2$

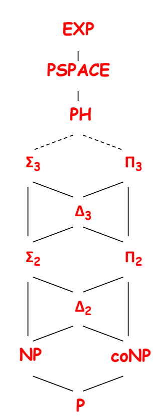
The PH

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 ∈ **NP** iff expressible as
  $L = \{ x \mid \exists y, |y| \le |x|^k, (x, y) \in R \}$
  where R ∈ **P**.
- Corollary: L ∈ **coNP** iff expressible as
  $L = \{ x \mid \forall y, |y| \le |x|^k, (x, y) \in R \}$
  where R ∈ **P**.

Useful characterization

Theorem: L ∈ Σ_i iff expressible as
$L = \{ x \mid \exists y, |y| \le |x|^k, (x, y) \in R \}$
where R ∈ Π_{i-1}.

- Corollary: L ∈ Π_i iff expressible as
  $L = \{ x \mid \forall y, |y| \le |x|^k, (x, y) \in R \}$
  where R ∈ Σ_{i-1}.

Useful characterization

Theorem: L ∈ Σ_i iff expressible as
$L = \{ x \mid \exists y, |y| \le |x|^k, (x, y) \in R \}$, where R ∈ Π_{i-1}.

- Proof of Theorem:
  – induction on i
  – base case (i =1) on previous slide
  (⇐)
  – we know $\Sigma_i = NP^{\Sigma_{i-1}} = NP^{\Pi_{i-1}}$
  – guess y, ask oracle if (x, y) ∈ R

Useful characterization

Theorem: L ∈ Σ_i iff expressible as
$L = \{ x \mid \exists y, |y| \le |x|^k, (x, y) \in R \}$, where R ∈ Π_{i-1}.

(⇒)
– given L ∈ $\Sigma_i = NP^{\Sigma_{i-1}}$ decided by ONTM M running in time n^k
– try: R = { (x, y) : y describes valid path of M's computation leading to q_{accept} }
– but how to recognize valid computation path when it depends on result of oracle queries?

Useful characterization

Theorem: L ∈ Σ_i iff expressible as
$L = \{ x \mid \exists y, |y| \le |x|^k, (x, y) \in R \}$, where R ∈ Π_{i-1}.

– try: R = { (x, y) : y describes valid path of M's computation leading to q_{accept} }
– valid path = step-by-step description including **correct** yes/no answer for each A-oracle query z_j (A ∈ Σ_{i-1})
– verify "no" queries in Π_{i-1}:
  e.g: $z_1 \notin A \wedge z_3 \notin A \wedge \ldots \wedge z_8 \notin A$
– for each "yes" query z_j: $\exists w_j, |w_j| \le |z_j|^k$ with $(z_j, w_j) \in R'$ for some R' ∈ Π_{i-2} by induction.
– for each "yes" query z_j put w_j in description of path y

Useful characterization

Theorem: L ∈ Σ_i iff expressible as
$L = \{ x \mid \exists y, |y| \le |x|^k, (x, y) \in R \}$, where R ∈ Π_{i-1}.

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

Alternating quantifiers

Nicer, more usable version:

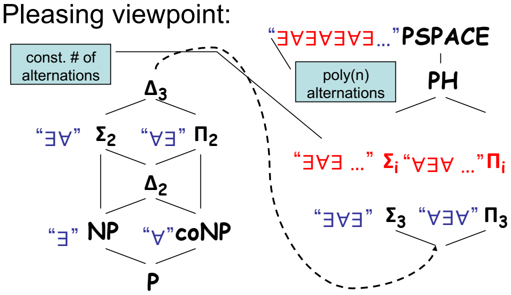
- $L \in \mathbf{\Sigma_i}$ iff expressible as
  $L = \{ x \mid \exists y_1 \forall y_2 \exists y_3 \ldots Q y_i (x, y_1, y_2, \ldots, y_i) \in R \}$
  where $Q = \forall/\exists$ if i even/odd, and $R \in \mathbf{P}$
- $L \in \mathbf{\Pi_i}$ iff expressible as
  $L = \{ x \mid \forall y_1 \exists y_2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

Alternating quantifiers

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

Alternating quantifiers

Pleasing viewpoint:



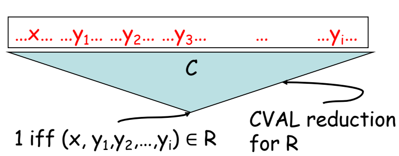
Complete problems

- three variants of SAT:
  - $QSAT_i$ (i odd) =
    {3-CNFs $\varphi(x_1, x_2, \ldots, x_i)$ for which
    $\exists x_1 \forall x_2 \exists x_3 \ldots \exists x_i\ \varphi(x_1, x_2, \ldots, x_i) = 1$}
  - $QSAT_i$ (i even) =
    {3-DNFs $\varphi(x_1, x_2, \ldots, x_i)$ for which
    $\exists x_1 \forall x_2 \exists x_3 \ldots \forall x_i\ \varphi(x_1, x_2, \ldots, x_i) = 1$}
  - QSAT = {3-CNFs φ for which
    $\exists x_1 \forall x_2 \exists x_3 \ldots Q x_n\ \varphi(x_1, x_2, \ldots, x_n) = 1$}

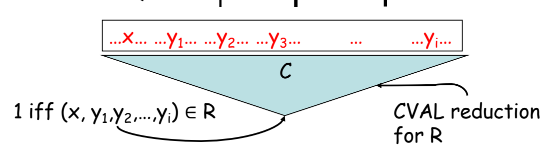
$QSAT_i$ is $\mathbf{\Sigma_i}$-complete

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

- Proof: (clearly in $\mathbf{\Sigma_i}$)
  - assume i odd; given $L \in \mathbf{\Sigma_i}$ in form
    $\{ x \mid \exists y_1 \forall y_2 \exists y_3 \ldots \exists y_i (x, y_1, y_2, \ldots, y_i) \in R \}$



$QSAT_i$ is $\mathbf{\Sigma_i}$-complete

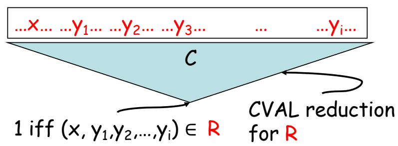


- Problem set: can construct 3-CNF φ from C:
  $\exists z\ \varphi(x, y_1, \ldots, y_i, z) = 1 \Leftrightarrow C(x, y_1, \ldots, y_i) = 1$
- we get:
  $\exists y_1 \forall y_2 \ldots \exists y_i \exists z\ \varphi(x, y_1, \ldots, y_i, z) = 1$
  $\Leftrightarrow \exists y_1 \forall y_2 \ldots \exists y_i\ C(x, y_1, \ldots, y_i) = 1 \Leftrightarrow x \in L$

QSAT$_i$ is $\mathbf{\Sigma_i}$-complete

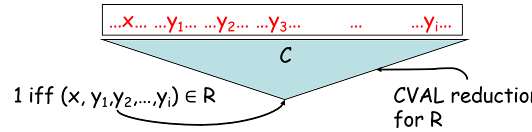
- Proof (continued)
  - assume i even; given $L \in \mathbf{\Sigma_i}$ in form

    $\{ x \mid \exists y_1\ \forall y_2\ \exists y_3 \ldots \forall y_i\ (x, y_1,y_2,\ldots,y_i) \in R \}$



May 8, 2021 CS151 Lecture 12.5 19

QSAT$_i$ is $\mathbf{\Sigma_i}$-complete



- Problem set: can construct 3-DNF φ from C:

  $\forall z\ \varphi(x,y_1,\ldots,y_i, z) = 1 \Leftrightarrow C(x,y_1,\ldots,y_i) = 1$
- we get:

  $\exists y_1\ \forall y_2 \ldots \forall y_i\ \forall z\ \varphi(x,y_1,y_2,\ldots,y_i, z) = 1$

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

May 8, 2021 CS151 Lecture 12.5 20

QSAT is **PSPACE**-complete

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

- Proof:
  - in **PSPACE**: $\forall x_1\ \exists x_2\ \forall x_3 \ldots Qx_n\ \varphi(x_1, x_2, \ldots, x_n)$? / $\exists x_1\ \forall x_2\ \exists x_3 \ldots Qx_n\ \varphi(x_1, x_2, \ldots, x_n)$?
  - "$\exists x_1$": 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"
  - "$\forall x_1$": for each x_1, recursively solve

    $\exists x_2\ \forall x_3 \ldots Qx_n\ \varphi(x_1, x_2, \ldots, x_n)$?
    - if encounter "no", return "no"
  - base case: evaluating a 3-CNF expression
  - poly(n) recursion depth
  - poly(n) bits of state at each level

May 8, 2021 CS151 Lecture 12.5 21

QSAT is **PSPACE**-complete

- given TM M deciding $L \in$ **PSPACE**; input x
- 2^{n^k} possible configurations
- single START configuration
- assume single ACCEPT configuration
- define:

  REACH(X, Y, i) $\Leftrightarrow$ configuration Y reachable from configuration X in at most 2^i steps.

May 8, 2021 CS151 Lecture 12.5 22

QSAT is **PSPACE**-complete

REACH(X, Y, i) $\Leftrightarrow$ configuration Y reachable from configuration X in at most 2^i steps.

- Goal: produce 3-CNF $\varphi(w_1,w_2,w_3,\ldots,w_m)$ such that

$\exists w_1\ \forall w_2 \ldots Qw_m\ \varphi(w_1,\ldots,w_m)$

REACH(START, ACCEPT, n^k)

May 8, 2021 CS151 Lecture 12.5 23

QSAT is **PSPACE**-complete

- for $i = 0, 1, \ldots n^k$ produce quantified Boolean **expressions** $\psi_i(A, B, W)$

  $\exists w_1\ \forall w_2 \ldots \psi_i(A, B, W) \Leftrightarrow$ REACH(A, B, i)
- convert ψ_{n^k} to 3-CNF φ
  - add variables V
  - $\exists w_1\ \forall w_2 \ldots \exists V\ \varphi(A, B, W, V) \Leftrightarrow$ REACH(A, B, n^k)
- hardwire A = START, B = ACCEPT

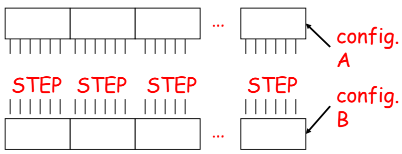
  $\exists w_1\ \forall w_2 \ldots \exists V\ \varphi(W, V) \Leftrightarrow x \in L$

May 8, 2021 CS151 Lecture 12.5 24

QSAT is **PSPACE**-complete

- $\psi_o(A, B)$ = true iff
  - A = B or
  - A yields B in one step of M

} Boolean expression of size $O(n^k)$



QSAT is **PSPACE**-complete

– Key observation #1:

$$\text{REACH}(A, B, i+1)$$
$$\Leftrightarrow$$
$$\exists Z\ [\text{REACH}(A, Z, i) \wedge \text{REACH}(Z, B, i)]$$

– cannot define $\psi_{i+1}(A, B)$ to be

$$\exists Z\ [\psi_i(A, Z) \wedge \psi_i(Z, B)]$$

(why?)

QSAT is **PSPACE**-complete

– Key idea #2: use quantifiers

– couldn't do $\psi_{i+1}(A, B) = \exists Z\ [\psi_i(A, Z) \wedge \psi_i(Z, B)]$

– define $\psi_{i+1}(A, B)$ to be

$$\exists Z \forall X \forall Y\ [((X=A \wedge Y=Z) \vee (X=Z \wedge Y=B)) \Rightarrow \psi_i(X, Y)]$$

– $\psi_i(X, Y)$ is preceded by quantifiers

– move to front (they don't involve X,Y,Z,A,B)

QSAT is **PSPACE**-complete

$\psi_o(A, B)$ = true iff A = B or A yields B in 1 step

$\psi_{i+1}(A, B)$ =

$\exists Z \forall X \forall Y[((X=A \wedge Y=Z) \vee (X=Z \wedge Y=B)) \Rightarrow \psi_i(X, Y)]$

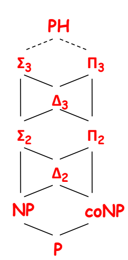
– $|\psi_0| = O(n^k)$

– $|\psi_{i+1}| = O(n^k) + |\psi_i|$

– total size of ψ_{nk} is $O(n^k)^2$ = poly(n)

– logspace reduction

PH collapse



Theorem: if $\mathbf{\Sigma_i = \Pi_i}$ then for all j > i

$$\Sigma_j = \Pi_j = \Delta_j = \Sigma_i$$

"the polynomial hierarchy **collapses** to the i-th level"

- Proof:
  – sufficient to show $\mathbf{\Sigma_i = \Sigma_{i+1}}$
  – then $\mathbf{\Sigma_{i+1} = \Sigma_i = \Pi_i = \Pi_{i+1}}$; apply theorem again

PH collapse

– recall: $L \in \mathbf{\Sigma_{i+1}}$ iff expressible as

$$L = \{ x \mid \exists y\ (x, y) \in R \}$$

where $R \in \mathbf{\Pi_i}$

– since $\mathbf{\Pi_i = \Sigma_i}$, R expressible as

$$R = \{ (x,y) \mid \exists z\ ((x, y), z) \in R' \}$$

where $R' \in \mathbf{\Pi_{i-1}}$

– together: $L = \{ x \mid \exists (y, z)\ \ (x, (y, z)) \in R' \}$

– conclude $L \in \mathbf{\Sigma_i}$

Oracles vs. Algorithms

A point to ponder:

- given poly-time **algorithm** for SAT
  - can you solve MIN CIRCUIT efficiently?
  - what other problems? Entire complexity classes?
- given SAT **oracle**
  - same input/output behavior
  - can you solve MIN CIRCUIT efficiently?

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

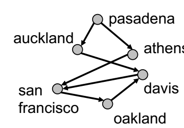
Natural complete problems in PH

- MIN CIRCUIT
  - good candidate to be Σ_2-complete, still open
- MIN DNF: given DNF φ, integer k; is there a DNF φ' of size at most k computing same function φ does?

Theorem (U): MIN DNF is $\boldsymbol{\Sigma_2}$-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}

Natural complete problems in PSPACE

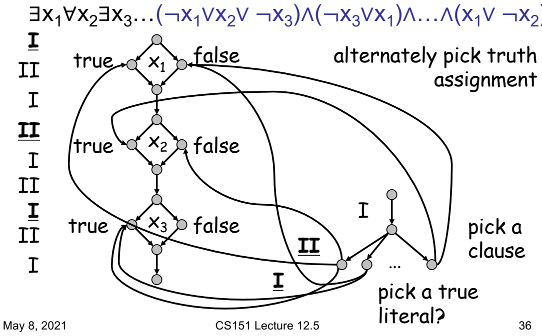
Theorem: GEOGRAPHY is PSPACE-complete.

Proof:

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

Natural complete problems in PSPACE

$\exists x_1 \forall x_2 \exists x_3 \ldots (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_3 \vee x_1) \wedge \ldots \wedge (x_1 \vee \neg x_2)$



Karp-Lipton

- we know that **P** = **NP** implies SAT has polynomial-size circuits.
  - (showing SAT does *not* have poly-size circuits is one route to proving **P** ≠ **NP**)
- suppose SAT has poly-size circuits
  - any consequences?
  - might hope: SAT ∈ **P/poly** ⇒ **PH** collapses to **P**, same as if SAT ∈ **P**

Karp-Lipton

Theorem (KL): if SAT has poly-size circuits then **PH** collapses to the second level.

- Proof:
  - suffices to show $\Pi_2 \subseteq \Sigma_2$
  - L ∈ Π_2 implies L expressible as:

    L = {x : ∀y ∃z (x, y, z) ∈ R}

    with R ∈ P.

Karp-Lipton

L = {x : ∀y ∃z (x, y, z) ∈ R}

- given (x, y), "∃z (x, y, z) ∈ R?" is in **NP**
- *pretend* C solves SAT, use self-reducibility
- Claim: if SAT ∈ P/poly, then L =

  { x : ∃C ∀y

  [use C repeatedly to **find** some z for which (x, y, z) ∈ R; accept iff (x, y, z) ∈ R] }

  (poly time)

Karp-Lipton

L = {x : ∀y ∃z (x, y, z) ∈ R}

{x : ∃C ∀y [use C repeatedly to **find** some z for which (x,y,z) ∈ R; accept iff (x,y,z) ∈ R] }

- x ∈ L:
  - some C decides SAT ⇒ ∃C ∀y […] accepts
- x ∉ L:
  - ∃y ∀z (x, y, z) ∉ R ⇒ ∀C ∃y […] rejects

BPP ⊆ PH

- Recall: don't know **BPP** different from **EXP**

Theorem (S,L,GZ): **BPP** ⊆ ($\Pi_2 \cap \Sigma_2$)

- don't know $\Pi_2 \cap \Sigma_2$ different from **EXP** but believe much weaker

BPP ⊆ PH

- Proof:
  - **BPP** language L: p.p.t. TM M:

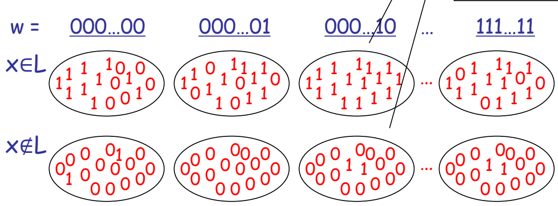
    x ∈ L ⇒ Pr_y[M(x,y) accepts] ≥ 2/3

    x ∉ L ⇒ Pr_y[M(x,y) rejects] ≥ 2/3
  - strong error reduction: p.p.t. TM M'
    - use n random bits (|y'| = n)
    - # strings y' for which M'(x, y') incorrect is at most $2^{n/3}$
    - (can't achieve with naïve amplification)

BPP ⊆ PH

- view $y' = (w, z)$, each of length $n/2$
- consider output of $M'(x, (w, z))$:

so few ones, not enough for whole disk



BPP ⊆ PH

- proof (continued):
  - strong error reduction: # bad $y' < 2^{n/3}$
  - $y' = (w, z)$ with $|w| = |z| = n/2$
  - Claim: $L = \{x : \exists w\ \forall z\ M'(x, (w, z)) = 1\}$
  - $x \in L$: suppose $\forall w \exists z\ M'(x, (w, z)) = 0$
    - implies $\geq 2^{n/2}$ 0's; contradiction
  - $x \notin L$: suppose $\exists w \forall z\ M'(x, (w, z)) = 1$
    - implies $\geq 2^{n/2}$ 1's; contradiction

BPP ⊆ PH

- given **BPP** language L: p.p.t. TM M:
  - $x \in L \Rightarrow Pr_y[M(x,y) \text{ accepts}] \geq 2/3$
  - $x \notin L \Rightarrow Pr_y[M(x,y) \text{ rejects}] \geq 2/3$
- showed $L = \{x : \exists w\ \forall z\ M'(x, (w, z)) = 1\}$

- thus **BPP** $\subseteq \Sigma_2$
- **BPP** closed under complement $\Rightarrow$ **BPP** $\subseteq \Pi_2$
- conclude: **BPP** $\subseteq (\Pi_2 \cap \Sigma_2)$

New Topic

The complexity of

counting

Counting problems

- So far, we have ignored function problems
  - given x, compute f(x)
- important class of function problems:

  counting problems

  - e.g. given 3-CNF φ how many satisfying assignments are there?

Counting problems

- **#P** is the class of function problems expressible as:

  input x $f(x) = |\{y : (x, y) \in R\}|$

  where R ∈ **P**.

- compare to **NP** (decision problem)

  input x $f(x) = \exists y : (x, y) \in R$?

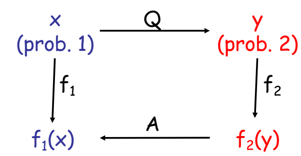
  where R ∈ **P**.

Counting problems

- examples
  - #SAT: given 3-CNF φ how many satisfying assignments are there?
  - #CLIQUE: given (G, k) how many cliques of size at least k are there?

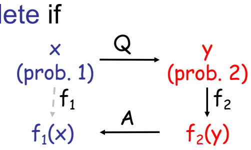
Reductions

- Reduction from function problem f_1 to function problem f_2
  - two efficiently computable functions Q, A



Reductions

- problem f is **#P**-complete if
  - f is in **#P**
  - every problem in **#P** reduces to f



- "parsimonious reduction": A is identity
  - many standard **NP**-completeness reductions are parsimonious
  - therefore: if #SAT is **#P**-complete we get lots of **#P**-complete problems

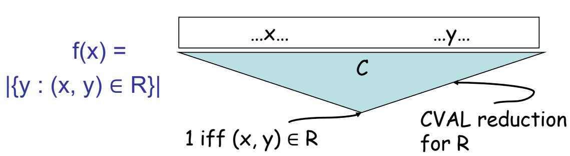
#SAT

#SAT: given 3-CNF φ how many satisfying assignments are there?

Theorem: #SAT is **#P**-complete.

- Proof:
  - clearly in **#P:** (φ, A) ∈ R ⇔ A satisfies φ
  - take any f ∈ **#P** defined by R ∈ **P**

#SAT



- add new variables z, produce φ such that
  ∃z φ(x, y, z) = 1 ⇔ C(x, y) = 1
- for (x, y) such that C(x, y) = 1 this z is *unique*
- hardwire x
- # satisfying assignments = |{y : (x, y) ∈ R}|

Relationship to other classes

- To compare to classes of decision problems, usually consider

  $\mathbf{P}^{\#\mathbf{P}}$

  which is a decision class…
- easy: **NP**, **coNP** ⊆ $\mathbf{P}^{\#\mathbf{P}}$
- easy: $\mathbf{P}^{\#\mathbf{P}}$ ⊆ **PSPACE**

Toda's Theorem (homework): **PH** ⊆ $\mathbf{P}^{\#\mathbf{P}}$.

Relationship to other classes

Question: is **#P** hard because it entails *finding* **NP** witnesses?

…or is *counting* difficult by itself?