CS151
Complexity Theory

Lecture 12

May 10, 2017

Useful characterization

Theorem: $L \in \mathbf{\Sigma_i}$ iff expressible as

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

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

- Corollary: $L \in \mathbf{\Pi_i}$ iff expressible as

$$L = \{ x \mid \forall\ y, |y| \leq |x|^k, (x, y) \in R \}$$

where $R \in \mathbf{\Sigma_{i-1}}$.

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$\in\mathbf{\Sigma_i}$:

$$L = \{x \mid \exists y_1\ (x, y_1) \in R' \},\ \text{for } R' \in \mathbf{\Pi_{i-1}}$$

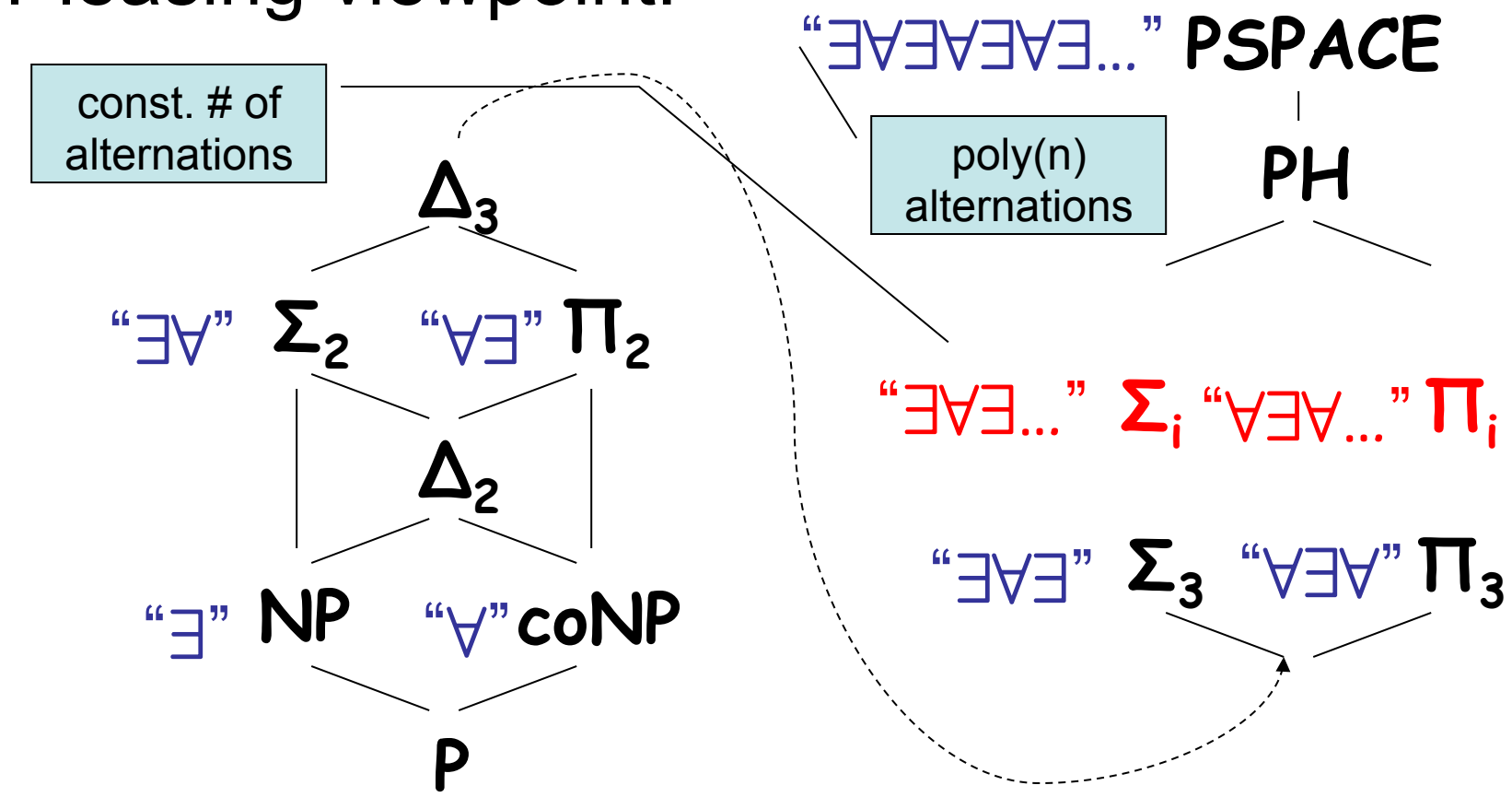
$$L = \{x \mid \exists y_1\ \ \forall y_2\, \exists y_3 \ldots Q y_i\ ((x, y_1), y_2, \ldots, y_i) \in R\}$$

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

  - same argument for L $\in\mathbf{\Pi_i}$
  - ($\Leftarrow$) exercise.

Alternating quantifiers

Pleasing viewpoint:



Complete problems

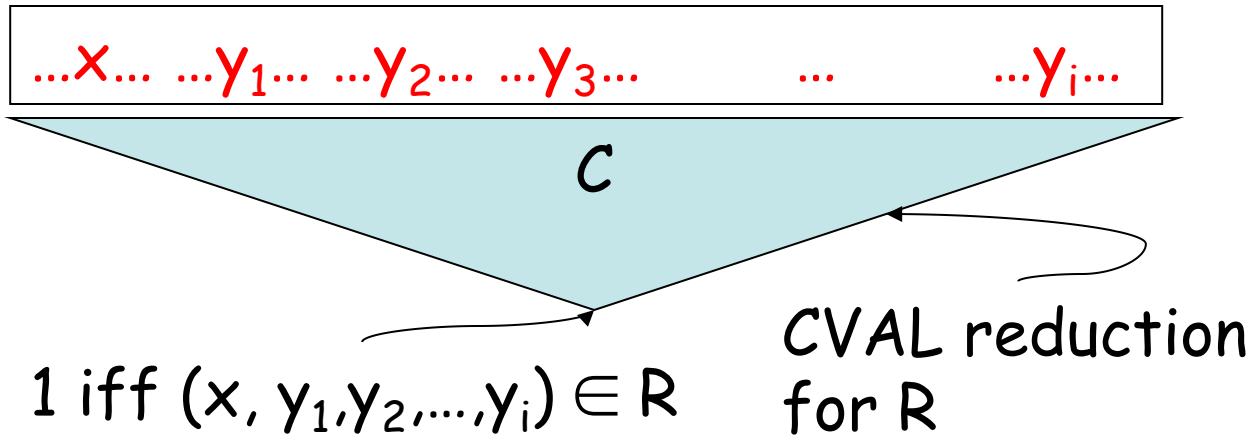
- three variants of SAT:
  - $QSAT_i$ (i odd) =
    {3-CNFs $\varphi(x_1, x_2, \ldots, x_i)$ for which
    $\exists x_1 \forall x_2 \exists x_3 \ldots \exists x_i \; \varphi(x_1, x_2, \ldots, x_i) = 1\}$
  - $QSAT_i$ (i even) =
    {3-DNFs $\varphi(x_1, x_2, \ldots, x_i)$ for which
    $\exists x_1 \forall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

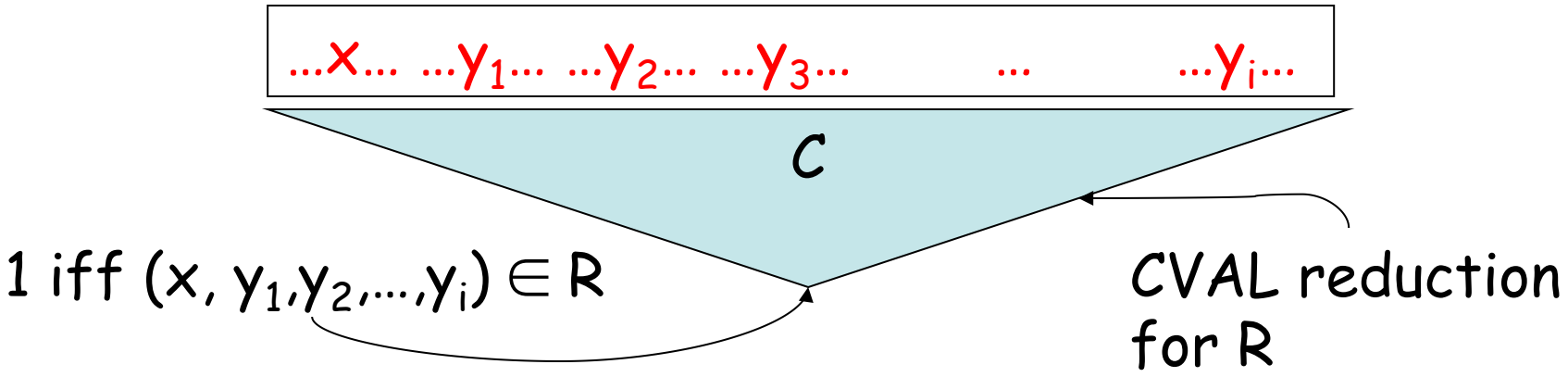
Theorem: $QSAT_i$ is Σ_i-complete.

- Proof: (clearly in Σ_i)
  - assume i odd; given $L \in \Sigma_i$ in form

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



$QSAT_i$ is Σ_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:

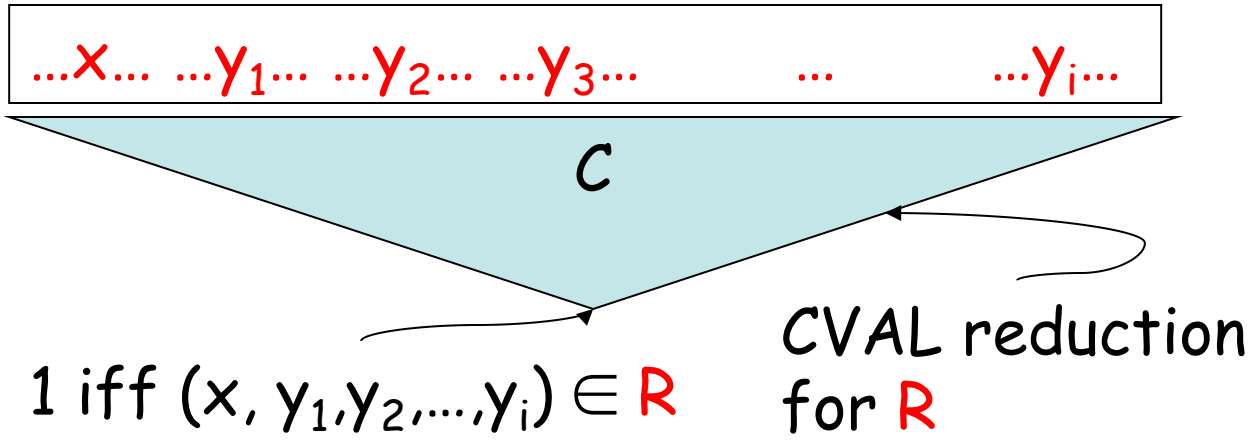
$$\exists y_1 \forall y_2 \ldots \exists y_i\ \exists z\ \varphi(x,y_1,\ldots,y_i,\ z) = 1$$

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

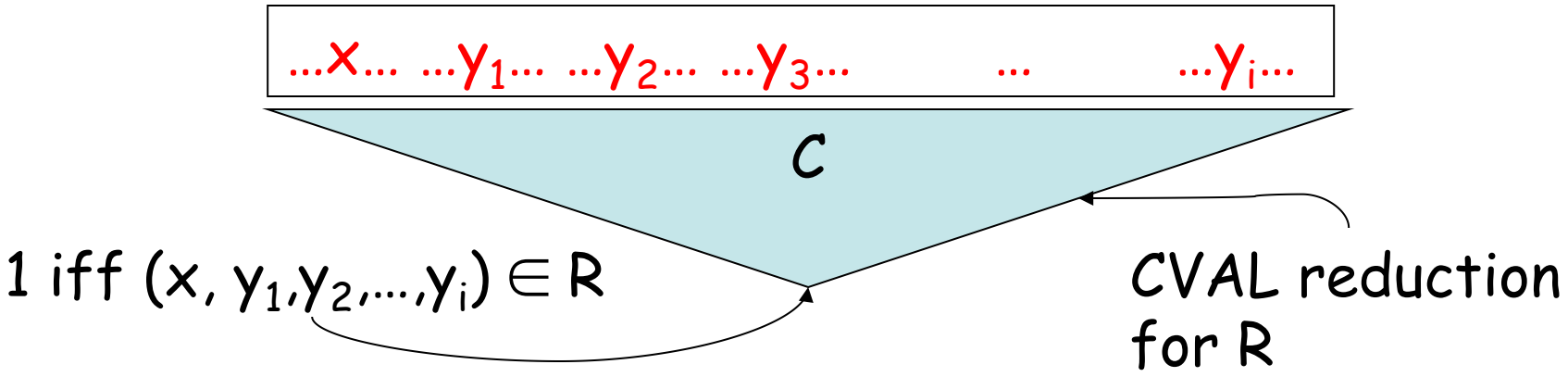
$QSAT_i$ is $\mathbf{\Sigma_i}$-complete

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

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



$QSAT_i$ is Σ_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:

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

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

QSAT is **PSPACE**-complete

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

- Proof:
  - in **PSPACE:** $\forall x_1 \exists x_2 \forall x_3 \ldots Q x_n \, \varphi(x_1, x_2, \ldots, x_n)?$ / $\exists x_1 \forall x_2 \exists x_3 \ldots Q x_n \, \varphi(x_1, x_2, \ldots, x_n)?$
  - "$\exists x_1$": for each x_1, recursively solve
    $\forall x_2 \exists x_3 \ldots Q x_n \, \varphi(x_1, x_2, \ldots, x_n)?$
    - if encounter "yes", return "yes"
  - "$\forall x_1$": for each x_1, recursively solve
    $\exists x_2 \forall x_3 \ldots Q x_n \, \varphi(x_1, x_2, \ldots, x_n)?$
    - if encounter "no", return "no"
  - base case: evaluating a 3-CNF expression
  - poly(n) recursion depth
  - poly(n) bits of state at each level

QSAT is **PSPACE**-complete

– given TM M deciding L ∈ **PSPACE**; input x

– 2^{n^k} possible configurations

– single START configuration

– assume single ACCEPT configuration

– define:

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

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

$$\exists w_1 \forall w_2 \ldots Q w_m \, \varphi(w_1,\ldots,w_m)$$
$$\Leftrightarrow \text{REACH(START, ACCEPT, } n^k)$$

QSAT is **PSPACE**-complete

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

$$\exists w_1 \forall w_2 \ldots \psi_i(A, B, W) \Leftrightarrow \text{REACH}(A, B, i)$$

– convert ψ_{nk} to 3-CNF φ

- add variables V
- $\exists w_1 \forall w_2 \ldots \exists V \varphi(A, B, W, V) \Leftrightarrow \text{REACH}(A, B, n^k)$

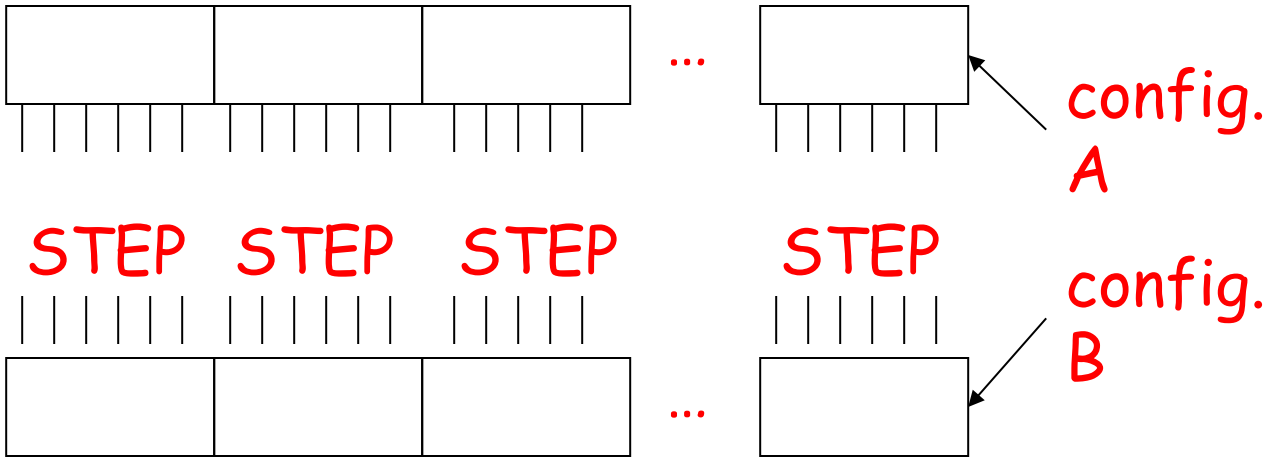
– hardwire A = START, B = ACCEPT

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

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:

$$\text{REACH}(A, B, i+1)$$

$$\Leftrightarrow$$

$$\exists Z \left[\text{REACH}(A, Z, i) \wedge \text{REACH}(Z, B, i)\right]$$

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

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

(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

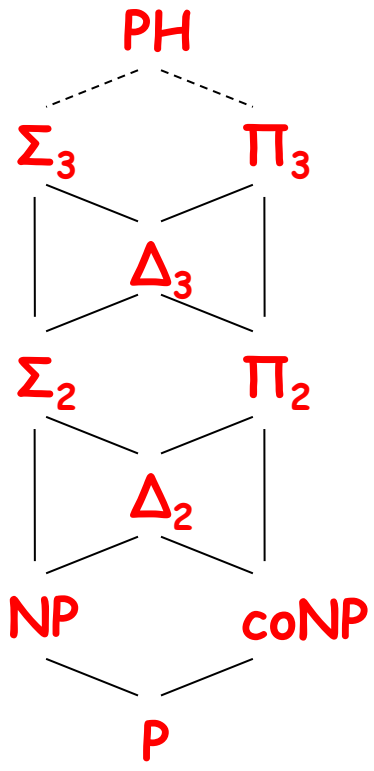
$\psi_{i+1}(A, B) =$

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

– $|\psi_0| = O(n^k)$
– $|\psi_{i+1}| = O(n^k) + |\psi_i|$

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

PH collapse



Theorem: if $\Sigma_i = \Pi_i$ 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 $\Sigma_i = \Sigma_{i+1}$
  - then $\Sigma_{i+1} = \Sigma_i = \Pi_i = \Pi_{i+1}$; 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

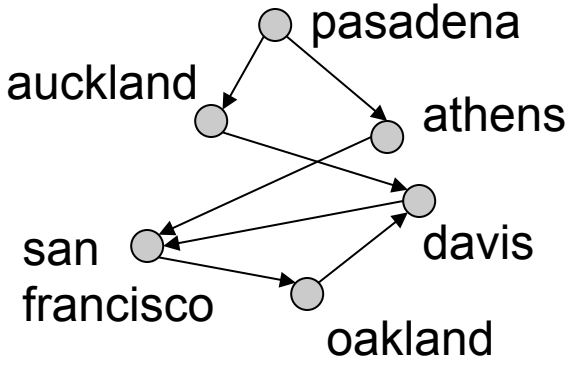
– MIN CIRCUIT
  - good candidate to be Σ_2-complete, still open

– 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 pick-ing 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

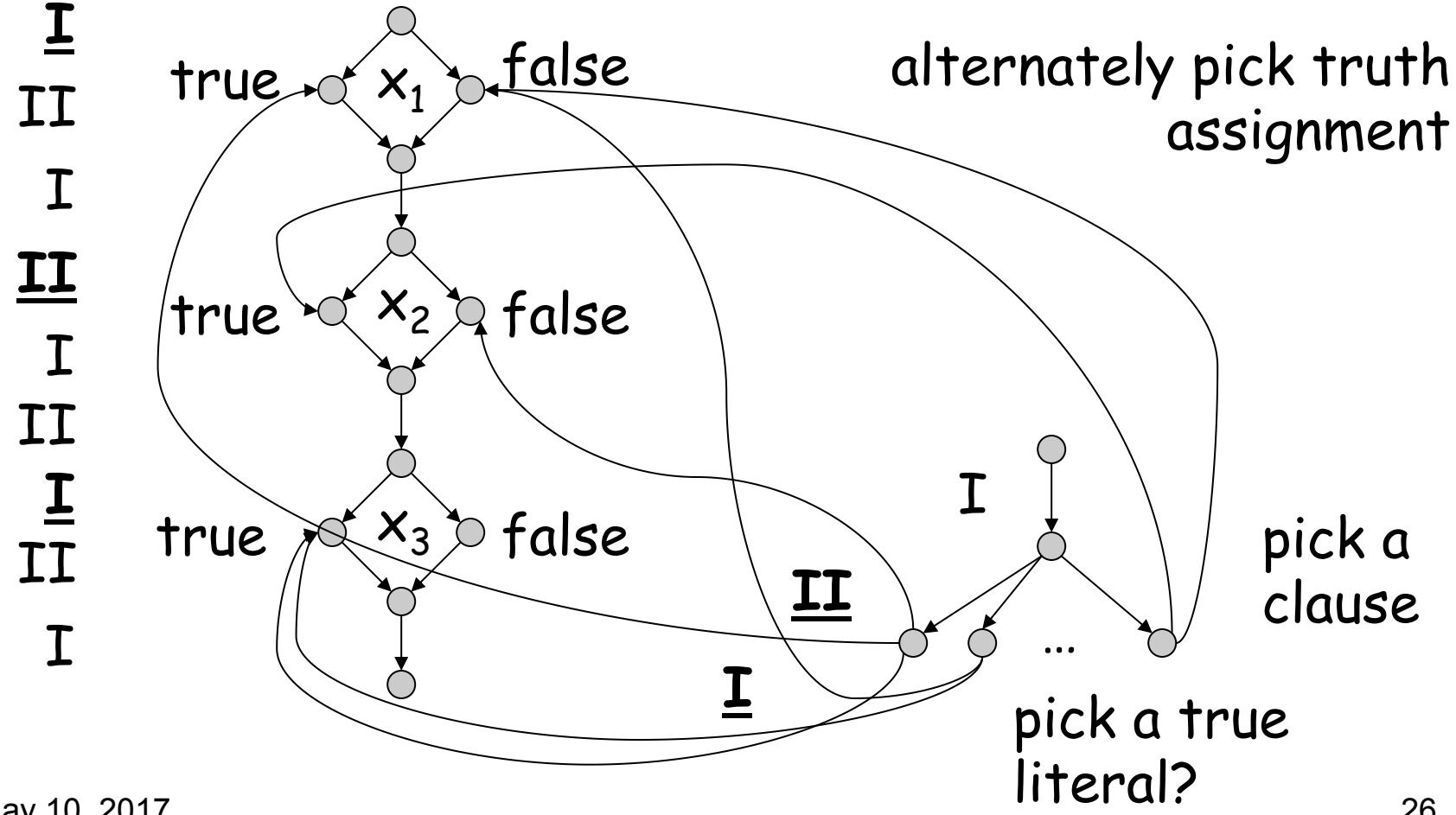
Theorem: GEOGRAPHY is PSPACE-complete.

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 $\mathbf{\Pi_2} \subset \mathbf{\Sigma_2}$
  - $L \in \mathbf{\Pi_2}$ implies L expressible as:

$$L = \{ x : \forall y \, \exists z \, (x, y, z) \in R \}$$

with $R \in P$.

Karp-Lipton

$L = \{ x : \forall y \, \exists z \, (x, y, z) \in R \}$

– given (x, y), "$\exists z \, (x, y, z) \in R$?" is in **NP**

– *pretend* C solves SAT, use self-reducibility

– Claim: if SAT $\in$ P/poly, then L =

$\{ x : \exists C \, \forall y$

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

poly time

Karp-Lipton

$$L = \{ x : \forall y \, \exists z \, (x, y, z) \in R \}$$

{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

$\mathbf{BPP} \subset \mathbf{PH}$

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

Theorem (S,L,GZ): $\mathbf{BPP} \subset (\mathbf{\Pi_2} \cap \mathbf{\Sigma_2})$

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

BPP ⊂ PH

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

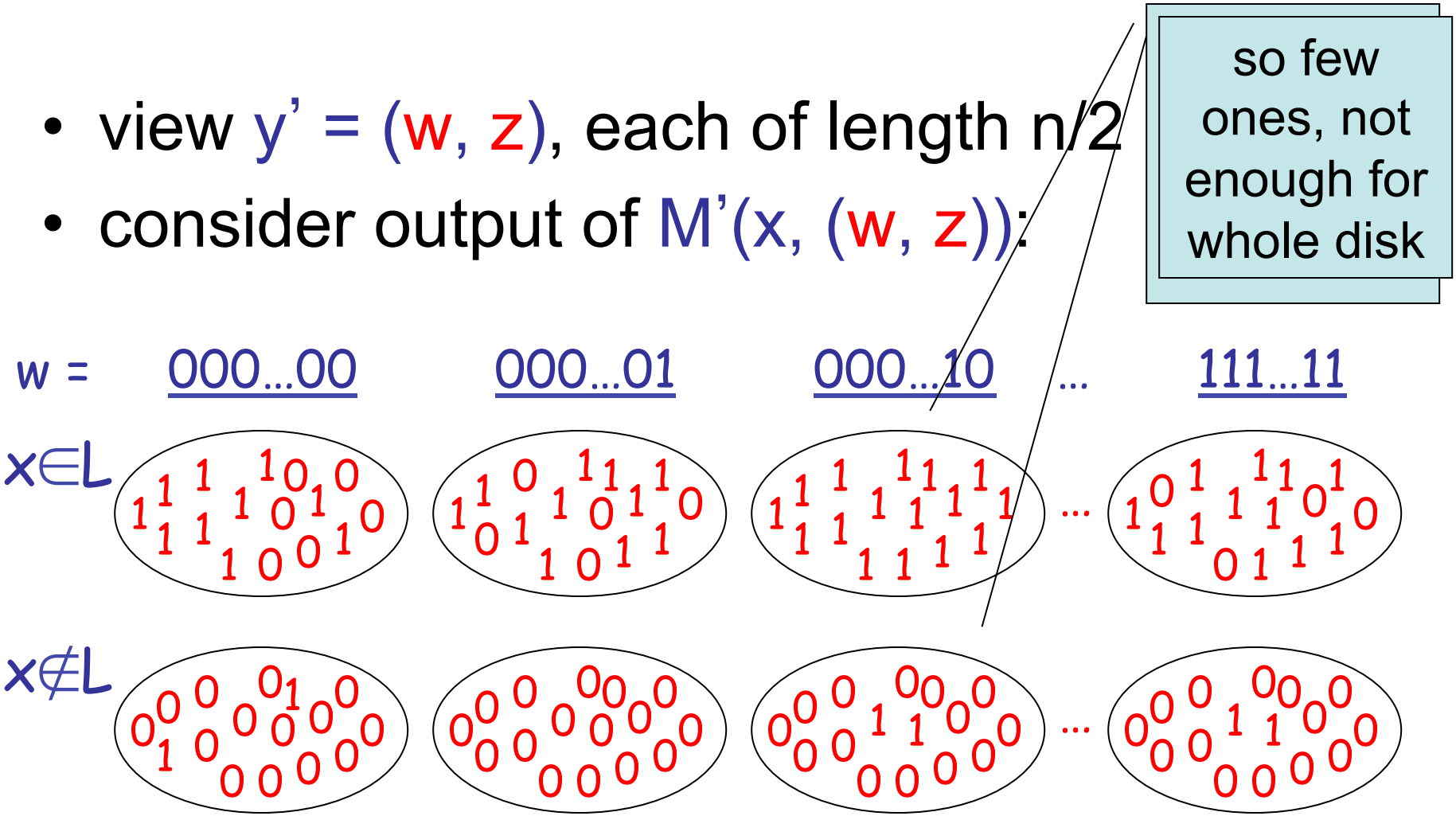
$$x \in L \Rightarrow Pr_y[M(x,y) \text{ accepts}] \geq 2/3$$

$$x \notin L \Rightarrow Pr_y[M(x,y) \text{ rejects}] \geq 2/3$$

  - 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)):



BPP ⊂ PH

- proof (continued):
  - strong error reduction: # bad y' < $2^{n/3}$
  - y' = (w, z) with |w| = |z| = n/2
  - Claim: L = {x : ∃w ∀z M'(x, (w, z)) = 1 }
  - x∈L: suppose ∀w∃z M'(x, (w, z)) = 0
    - implies ≥ $2^{n/2}$ 0's; contradiction
  - x∉L: suppose ∃w∀z M'(x, (w, z)) = 1
    - implies ≥ $2^{n/2}$ 1's; contradiction

$\mathbf{BPP} \subset \mathbf{PH}$

– 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 $\mathbf{BPP} \subset \mathbf{\Sigma_2}$

– **BPP** closed under complement $\Rightarrow \mathbf{BPP} \subset \mathbf{\Pi_2}$

– conclude: $\mathbf{BPP} \subset (\mathbf{\Pi_2} \cap \mathbf{\Sigma_2})$

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 $R \in \mathbf{P}$.

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

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

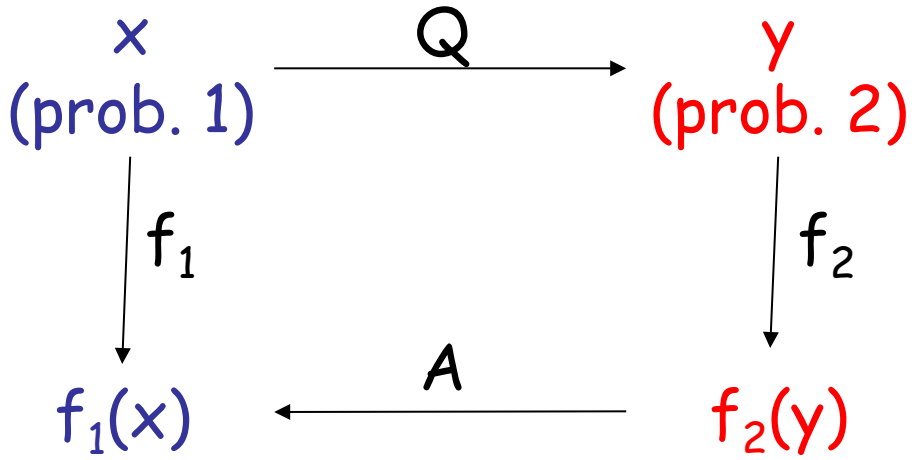
  where $R \in \mathbf{P}$.

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?

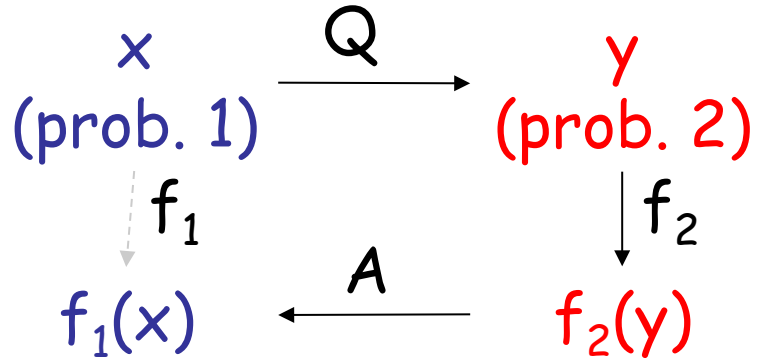
Reductions

- 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

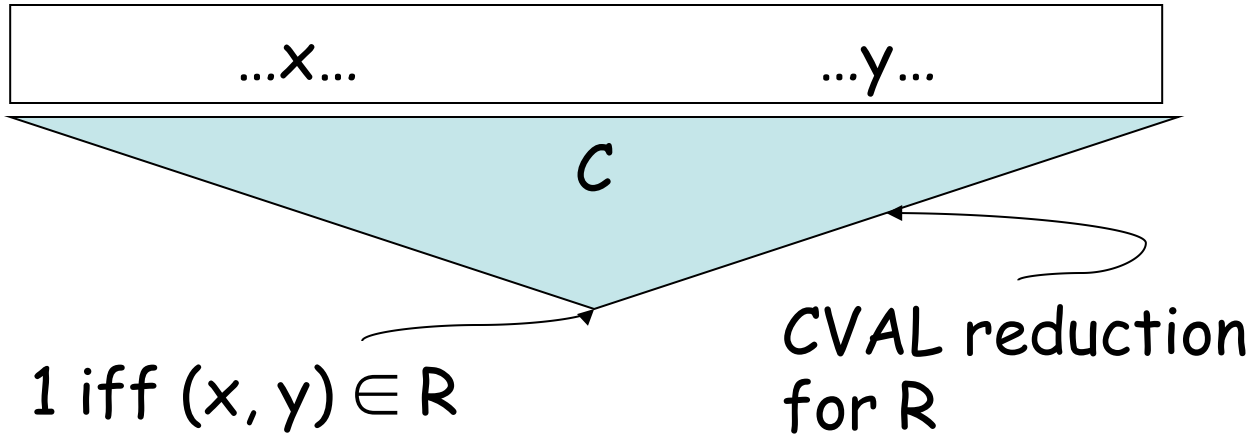
#SAT: given 3-CNF φ how many satisfying assignments are there?

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

- Proof:
  - clearly in **#P:** $(\varphi, A) \in R \Leftrightarrow A$ satisfies φ
  - take any $f \in$ **#P** defined by $R \in$ **P**

#SAT

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



– add new variables z, produce φ such that

$$\exists z\ \varphi(x, y, z) = 1 \Leftrightarrow C(x, y) = 1$$

– for (x, y) such that C(x, y) = 1 this z is *unique*

– hardwire x

– # satisfying assignments = $|\{y : (x, y) \in R\}|$

Relationship to other classes

- To compare to classes of decision problems, usually consider

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

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

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

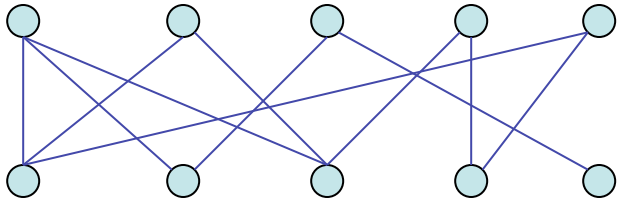
Relationship to other classes

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

…or is *counting* difficult by itself?

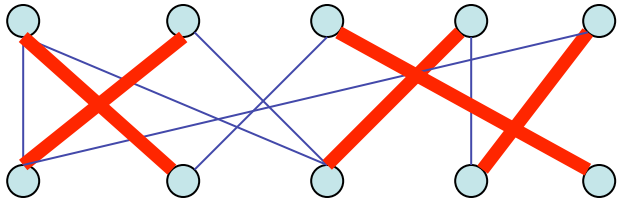
Bipartite Matchings

- Definition:
  - G = (U, V, E) bipartite graph with |U| = |V|
  - a **perfect matching** in G is a subset M ⊂ E that touches every node, and no two edges in M share an endpoint



Bipartite Matchings

- Definition:
  - G = (U, V, E) bipartite graph with |U| = |V|
  - a **perfect matching** in G is a subset M ⊂ E that touches every node, and no two edges in M share an endpoint



Bipartite Matchings

- #MATCHING: given a bipartite graph G = (U, V, E) how many perfect matchings does it have?

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

- But… can *find* a perfect matching in polynomial time!
  - counting itself must be difficult