CS151
Complexity Theory

Lecture 11

May 8, 2017

Min-entropy

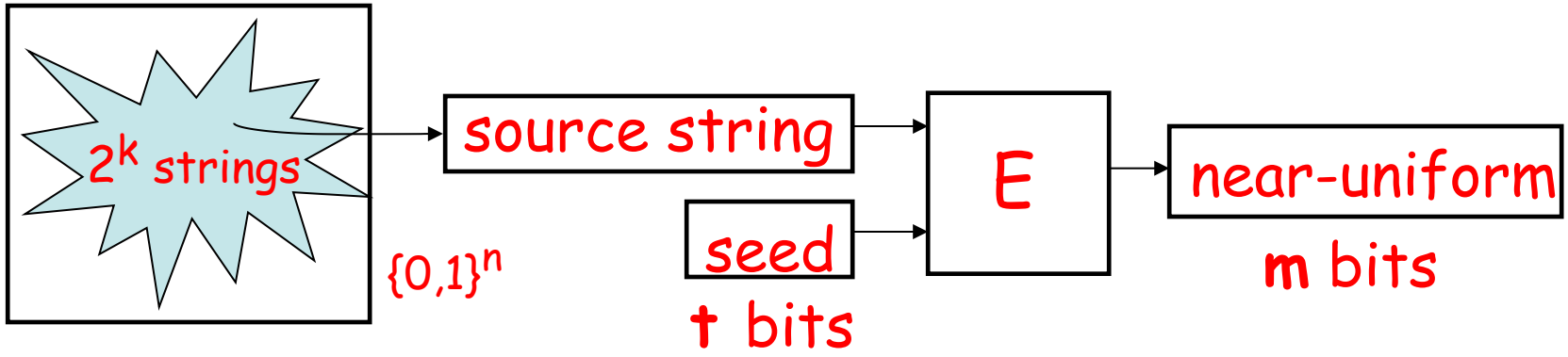
- General model of physical source w/ k < n bits of hidden randomness

Definition: random variable X on $\{0,1\}^n$ has **min-entropy** $\min_x -\log(\Pr[X = x])$

– min-entropy k implies no string has weight more than 2^{-k}

Extractor

- Extractor: universal procedure for "purifying" imperfect source:



  - E is efficiently computable
  - truly random seed as "catalyst"

Extractor

"(k, ε)-extractor" ⇒ for all X with min-entropy k:

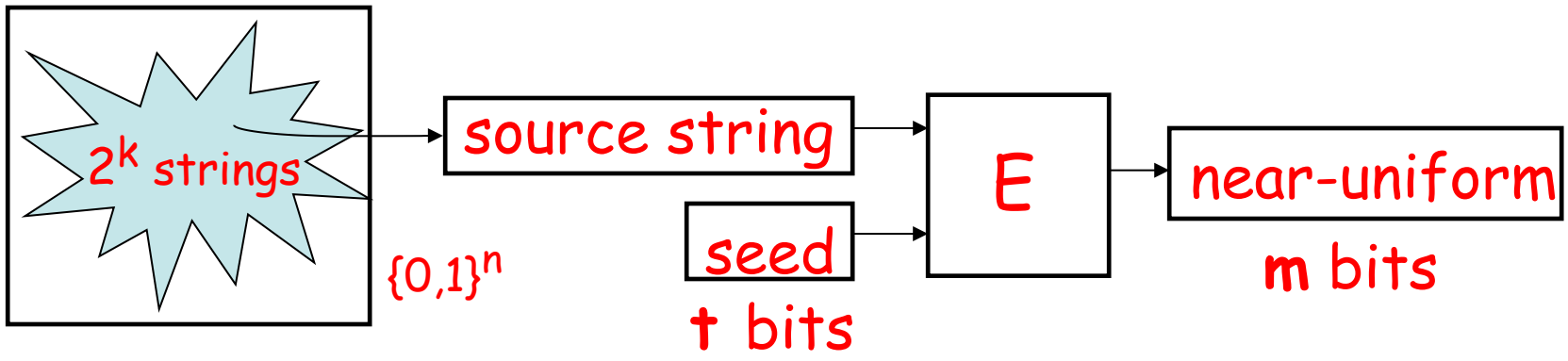
– output fools **all** circuits C:

$$|Pr_z[C(z) = 1] - Pr_{y,\, x \leftarrow X}[C(E(x, y)) = 1]| \leq \boldsymbol{\varepsilon}$$

– distributions $E(X, U_t)$, U_m "**ε**-close" (L_1 dist ≤ 2**ε**)

- Notice similarity to PRGs
  - output of PRG fools all **efficient** tests
  - output of extractor fools **all** tests

Extractors



- Goals:

	good:	best:
short seed	$O(\log n)$	$\log n+O(1)$
long output	$m = k^{\Omega(1)}$	$m = k+t-O(1)$
many k's	$k = n^{\Omega(1)}$	any $k = k(n)$

Extractors

- random function for E achieves best !
  - but we need **explicit** constructions
  - many known; often complex + technical
  - optimal extractors still open
- Trevisan Extractor:
  - insight: use NW generator with source string in place of hard function
  - this works (!!)
  - proof slightly different than NW, easier

Trevisan Extractor

- Ingredients: ($\delta > 0$, m are parameters)
  - error-correcting code

$$C:\{0,1\}^n \rightarrow \{0,1\}^{n'}$$

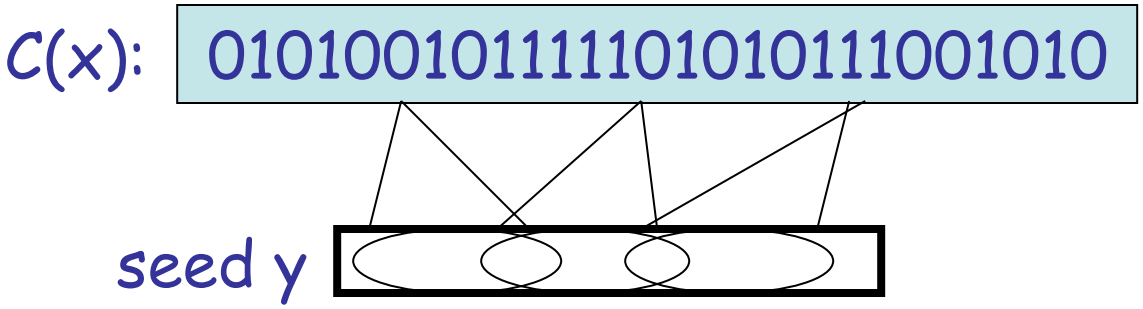
distance $(\frac{1}{2} - \frac{1}{4}m^{-4})n'$ blocklength $n' = poly(n)$

  - $(\log n', a = \delta \log n/3)$ design:
    $S_1, S_2, \ldots, S_m \subset \{1 \ldots t = O(\log n')\}$

$$E(x, y) = C(x)[y_{|S_1}] \circ C(x)[y_{|S_2}] \circ \ldots \circ C(x)[y_{|S_m}]$$

Trevisan Extractor

$$E(x, y)=C(x)[y_{|S_1}]\circ C(x)[y_{|S_2}]\circ\ldots\circ C(x)[y_{|S_m}]$$



Theorem (T): E is an extractor for min-entropy $k = n^{\delta}$, with

- output length $m = k^{1/3}$
- seed length $t = O(\log n)$
- error $\varepsilon \leq 1/m$

Trevisan Extractor

- Proof:
  - given $X \subseteq \{0,1\}^n$ of size 2^k
  - assume E fails to ε-pass statistical test C

    $|Pr_z[C(z) = 1] - Pr_{x \in X,\, y}[C(E(x, y)) = 1]| > \varepsilon$

  - **distinguisher** C $\Rightarrow$ **predictor** P:

    $Pr_{x \in X,\, y}[P(E(x, y)_1 \cdots_{i-1}) = E(x, y)_i] > ½ + \varepsilon/m$

Trevisan Extractor

- Proof (continued):
  - for at least ε/2 of $x \in X$ we have:

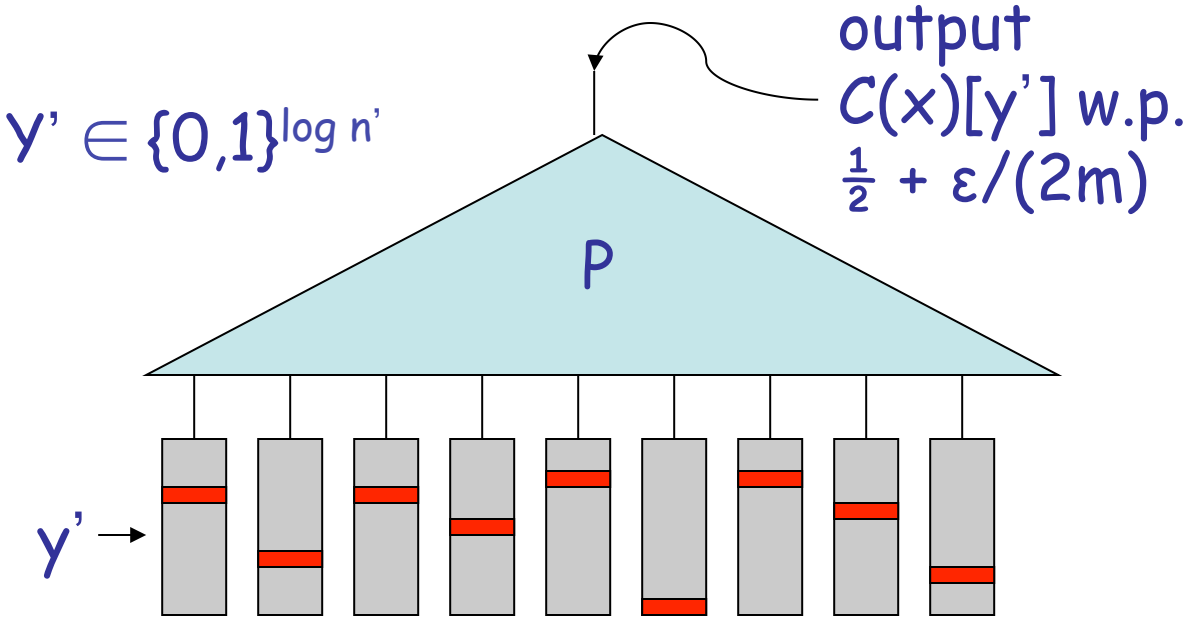
    $$Pr_y[P(E(x, y)_{1 \ldots i-1}) = E(x, y)_i] > \tfrac{1}{2} + \varepsilon/(2m)$$

  - fix bits α, β outside of S_i to preserve advantage

    $$Pr_{y'}[P(E(x; \alpha y' \beta)_{1 \ldots i-1}) = C(x)[y']\ ] > \tfrac{1}{2} + \varepsilon/(2m)$$

  - as vary y', for $j \neq i$, j-th bit of $E(x; \alpha y' \beta)$ varies over only 2^a values
  - (m-1) tables of 2^a values supply $E(x; \alpha y' \beta)_{1 \ldots i-1}$

Trevisan Extractor

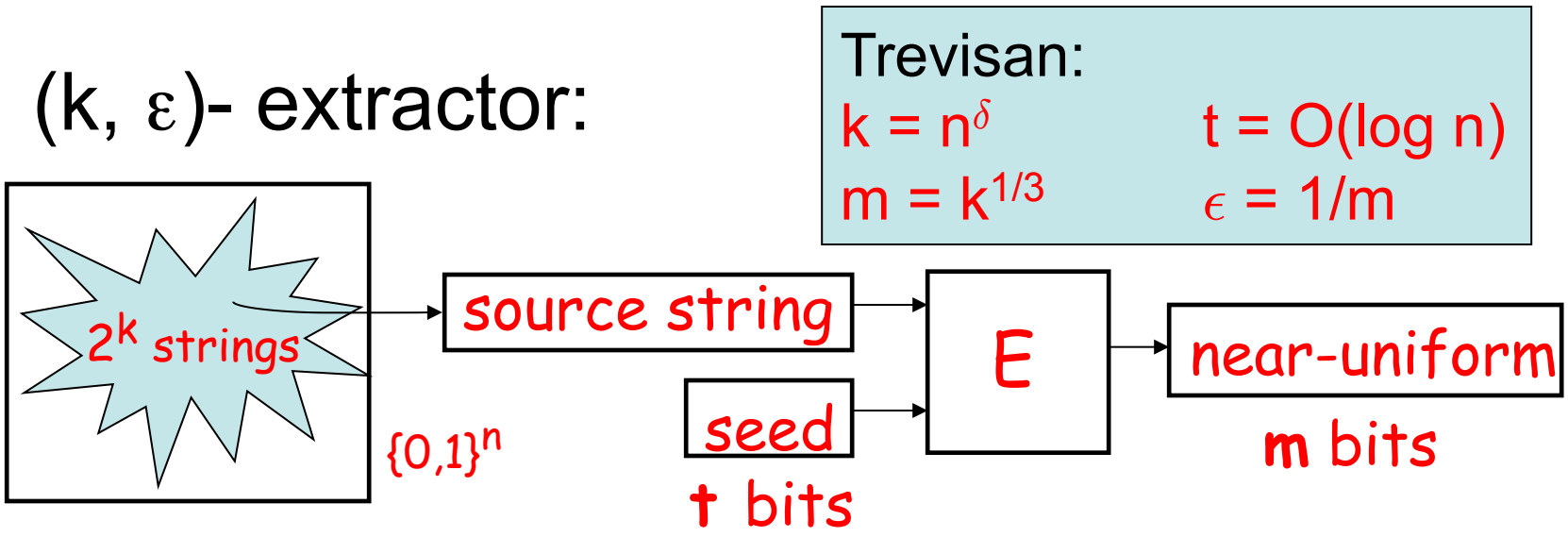


Trevisan Extractor

- Proof (continued):
  - (m-1) tables of size 2^a constitute a **description** of a string that has $\frac{1}{2} + \varepsilon/(2m)$ agreement with C(x)
  - # of strings x with such a description?
    - $\exp((m-1)2^a) = \exp(n^{\delta 2/3}) = \exp(k^{2/3})$ strings
    - Johnson Bound: each string accounts for at most $O(m^4)$ x's
    - total #: $O(m^4)\exp(k^{2/3}) << 2^k(\varepsilon/2)$
    - contradiction

Extractors

- (k, ε)- extractor:



  - E is efficiently computable
  - ∀ X with minentropy k, E fools **all** circuits C:

$$|Pr_z[C(z) = 1] - Pr_{y,\ x \leftarrow X}[C(E(x, y)) = 1]| \leq \boldsymbol{\varepsilon}$$

Strong error reduction

- $L \in$ **BPP** if there is a p.p.t. TM M:

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

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

- Want:

  $x \in L \Rightarrow Pr_y[M(x,y) \text{ accepts}] \geq 1 - 2^{-k}$

  $x \notin L \Rightarrow Pr_y[M(x,y) \text{ rejects}] \geq 1 - 2^{-k}$

- We saw: repeat O(k) times
  - n = O(k)·|y| random bits; **2^{n-k} bad strings**

Want to spend n = poly(|y|) random bits; achieve << $2^{n/3}$ bad strings

Strong error reduction

- Better:
  - E extractor for minentropy $k=|y|^3=n^\delta$, $\varepsilon < 1/6$
  - pick random $w \in \{0,1\}^n$, run $M(x, E(w, z))$ for all $z \in \{0,1\}^t$, take majority
  - call w "bad" if $maj_z M(x, E(w, z))$ incorrect

    $|Pr_z[M(x,E(w,z))=b] - Pr_y[M(x,y)=b]| \geq 1/6$

  - extractor property: at most 2^k bad w
  - n random bits; **2^{n^δ} bad strings**

RL

- Recall: probabilistic Turing Machine
  - deterministic TM with extra tape for "coin flips"
- **RL** (Random Logspace)
  - L ∈ **RL** if there is a probabilistic logspace TM M:

$$x \in L \Rightarrow \Pr_y[M(x,y) \text{ accepts}] \geq \tfrac{1}{2}$$

$$x \notin L \Rightarrow \Pr_y[M(x,y) \text{ rejects}] = 1$$

  - important detail #1: only allow one-way access to coin-flip tape
  - important detail #2: explicitly require to run in polynomial time

RL

- **L** $\subseteq$ **RL** $\subseteq$ **NL** $\subseteq$ **SPACE**($\log^2 n$)
- Theorem (SZ) : **RL** $\subseteq$ **SPACE**($\log^{3/2} n$)

- Belief: L = RL (open problem)

RL

L ⊆ RL ⊆ NL

- Natural problem:

  Undirected STCONN: given an **undirected** graph G = (V, E), nodes s, t, is there a path from s → t?

Theorem: USTCONN ∈ **RL.**

(Recall: STCONN is **NL**-complete)

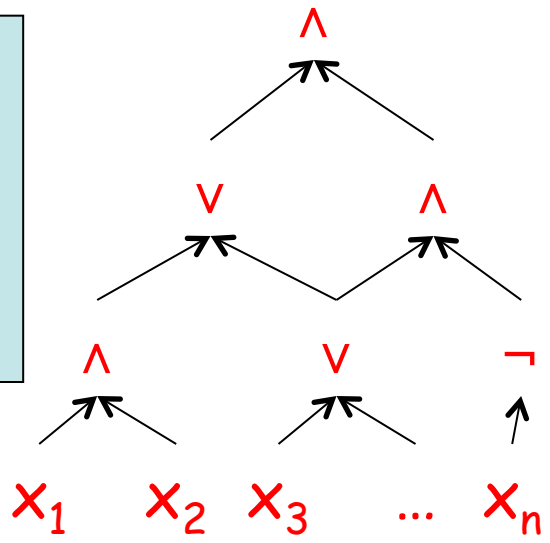
Undirected STCONN

- 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 $s=v_1, v_2, \ldots, v_n=t$ 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 $v_n \leq |V||E|$
  - Pr[fail reach t in 2|V||E| steps] ≤ ½
- Reingold 2005: USTCONN ∈ **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?
  - **NP**-hard? in **NP**? in **coNP**? in **PSPACE**?
  - complete for any of these classes?

Oracle Turing Machines

- Oracle Turing Machine (OTM):
  - multitape TM M with special "query" tape
  - special states $q_?$, q_{yes}, q_{no}
  - on input x, with oracle language A
  - M^A runs as usual, except…
  - when M^A enters state $q_?$:
    - y = contents of query tape
    - $y \in A \Rightarrow$ transition to q_{yes}
    - $y \notin A \Rightarrow$ transition to 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

  e.g.: given $\varphi_1, \varphi_2, \ldots, \varphi_n$ are even # satisfiable?
  - poly-time OTM solves with SAT oracle

Oracle Turing Machines

Shorthand #1:

- applying oracles to entire complexity classes:
  - complexity class **C**
  - language A

  $\mathbf{C}^A$ = {L decided by OTM M with oracle A with M "in" **C**}

  - example: $\mathbf{P}^{SAT}$

Oracle Turing Machines

Shorthand #2:

- using complexity classes as oracles:
  - OTM M
  - complexity class **C**
  - $M^{\mathbf{C}}$ decides language L if for some language $A \in \mathbf{C}$, M^A decides L

Both together: $\mathbf{C}^{\mathbf{D}}$ = languages decided by OTM "in" **C** with oracle language from **D**

exercise: show $\mathbf{P}^{\mathrm{SAT}} = \mathbf{P}^{\mathbf{NP}}$

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

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

$\Delta_1 = \mathrm{P}^{\mathrm{P}}$	$\Sigma_1 = \mathbf{NP}$	$\Pi_1 = \mathbf{coNP}$
$\Delta_2 = \mathrm{P}^{\mathrm{NP}}$	$\Sigma_2 = \mathrm{NP}^{\mathrm{NP}}$	$\Pi_2 = \mathrm{coNP}^{\mathrm{NP}}$
$\Delta_{i+1} = \mathrm{P}^{\Sigma_i}$	$\Sigma_{i+i} = \mathrm{NP}^{\Sigma_i}$	$\Pi_{i+1} = \mathrm{coNP}^{\Sigma_i}$

Polynomial Hierarchy $\mathbf{PH} = \bigcup_i \Sigma_i$

The Polynomial-Time Hierarchy

$$\Sigma_0 = \Pi_0 = P$$

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

- 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

$$\Sigma_0 = \Pi_0 = P$$

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

- 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 $\in \Delta_2$

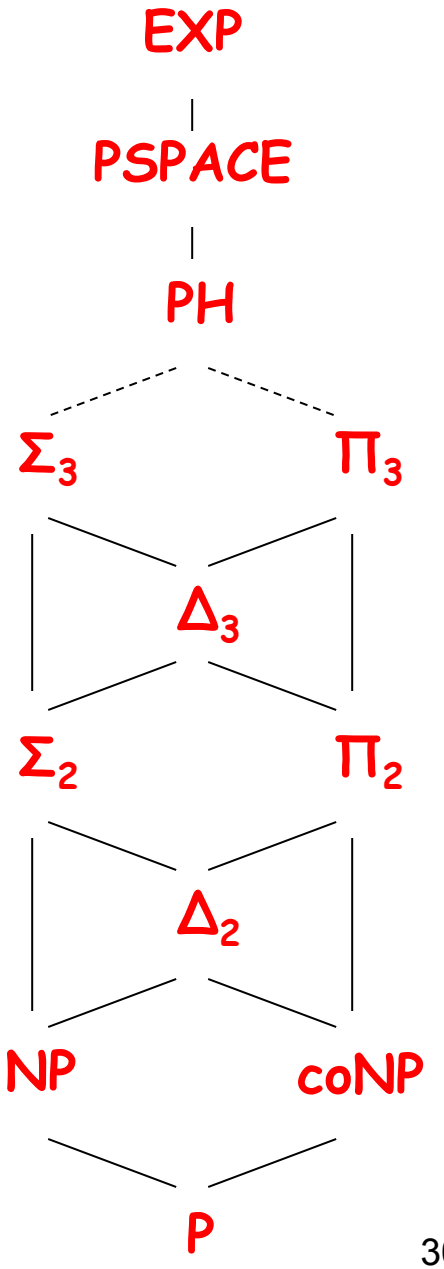
The PH

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: $L \in$ **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**.

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

Useful characterization

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

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

- Proof of Theorem:
  - induction on i
  - base case (i =1) on previous slide

  ($\Leftarrow$)
  - we know $\Sigma_i = NP^{\Sigma_{i-1}} = NP^{\Pi_{i-1}}$
  - guess y, ask oracle if $(x, y) \in R$

Useful characterization

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

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

($\Rightarrow$)

– given $L \in \Sigma_i = NP^{\Sigma_{i-1}}$ 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: $L \in \mathbf{\Sigma_i}$ iff expressible as

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

- 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 $(A \in \mathbf{\Sigma_{i-1}})$
- verify "no" queries in $\mathbf{\Pi_{i-1}}$:

  e.g: $z_1 \notin A \wedge z_3 \notin A \wedge \ldots \wedge z_8 \notin A$
- for each "yes" query z_j: $\exists w_j, |w_j| \leq |z_j|^k$ with $(z_j, w_j) \in R'$ for some $R' \in \mathbf{\Pi_{i-2}}$ by induction.
- for each "yes" query z_j put w_j in description of path y

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

– single language R in $\mathbf{\Pi_{i-1}}$:

$(x, y) \in R$

$\Leftrightarrow$

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 $\mathbf{\Pi_{i-1}}$ predicates is in $\mathbf{\Pi_{i-1}}$.

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 $R \in \mathbf{P}$