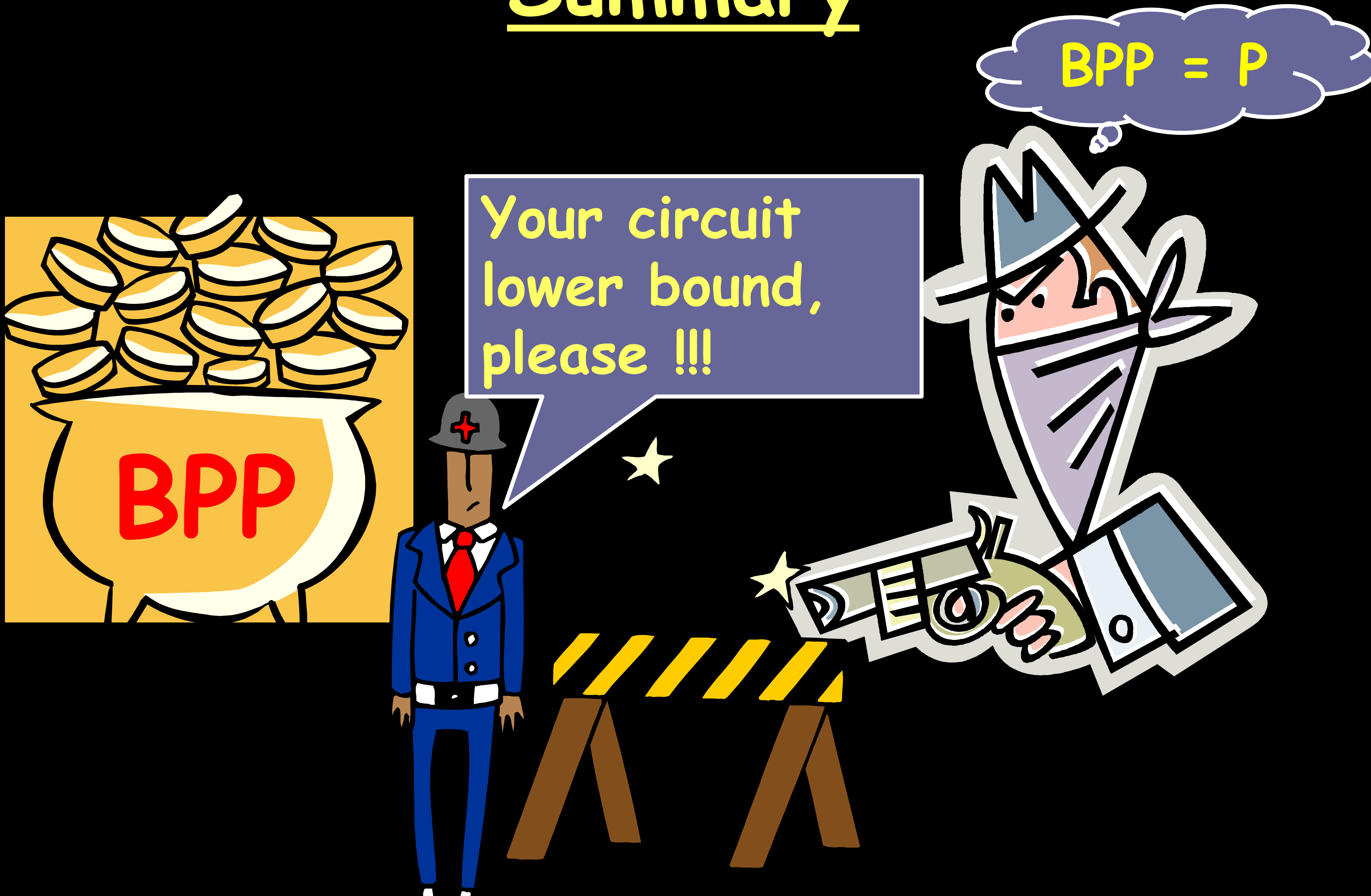
Derandomization vs. Circuit Lower Bounds

Valentine Kabanets (IAS & SFU)

Summary



Derandomization

Goal: Do efficiently deterministicaly what we can do efficiently probabilistically.

Impossible in general: - crypto, - comm. complexity, …

Our Focus:

- Randomized decision algorithms
- Randomized search algorithms

- **Decision:** BPP vs. P

Poly Id Testing:

Given arithmetic circuit C, decide if C computes identically zero polynomial.

- **Search:** Construct "random-like" combinatorial objects
  - expander graphs,
  - error-correcting codes,
  - truth tables of hard Boolean functions

- **Decision:** BPP vs. P

Poly Id Testing:

Given arithmetic circuit C, decide if C computes identically zero polynomial.

- **Search:** Construct "random-like" combinatorial objects
  - expander graphs,
  - error-correcting codes,
  - truth tables of hard Boolean functions

Constructing Hard Functions

$s(n)$ - Hardness Generator :

Given n, output a binary string of length 2^n which (as n-variate Boolean function) has circuit complexity at least $s(n)$. The running time should be $poly(2^n)$.

Trivial: Randomized $2^n/n$-Hardness Generator exists (may produce an easy string).

Hardness Generator (HG)

- Deterministic $s(n)$-Hardness Generator exists iff E has circuit complexity at least $s(n)$.

- Open: for $s(n) > \omega(n)$.

- Also open:

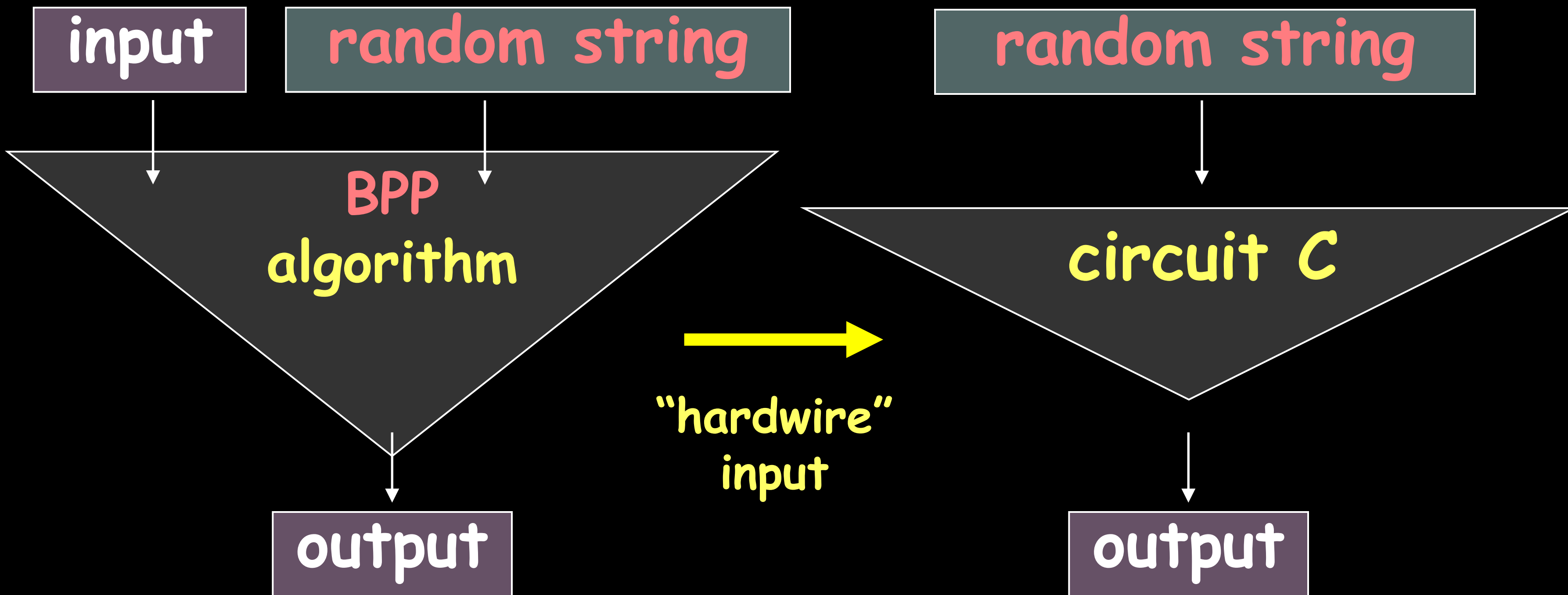
  - Nondeterministic HG,
  - Zero-Error Randomized HG, …

Hardness Generation

vs.

Derandomization of BPP

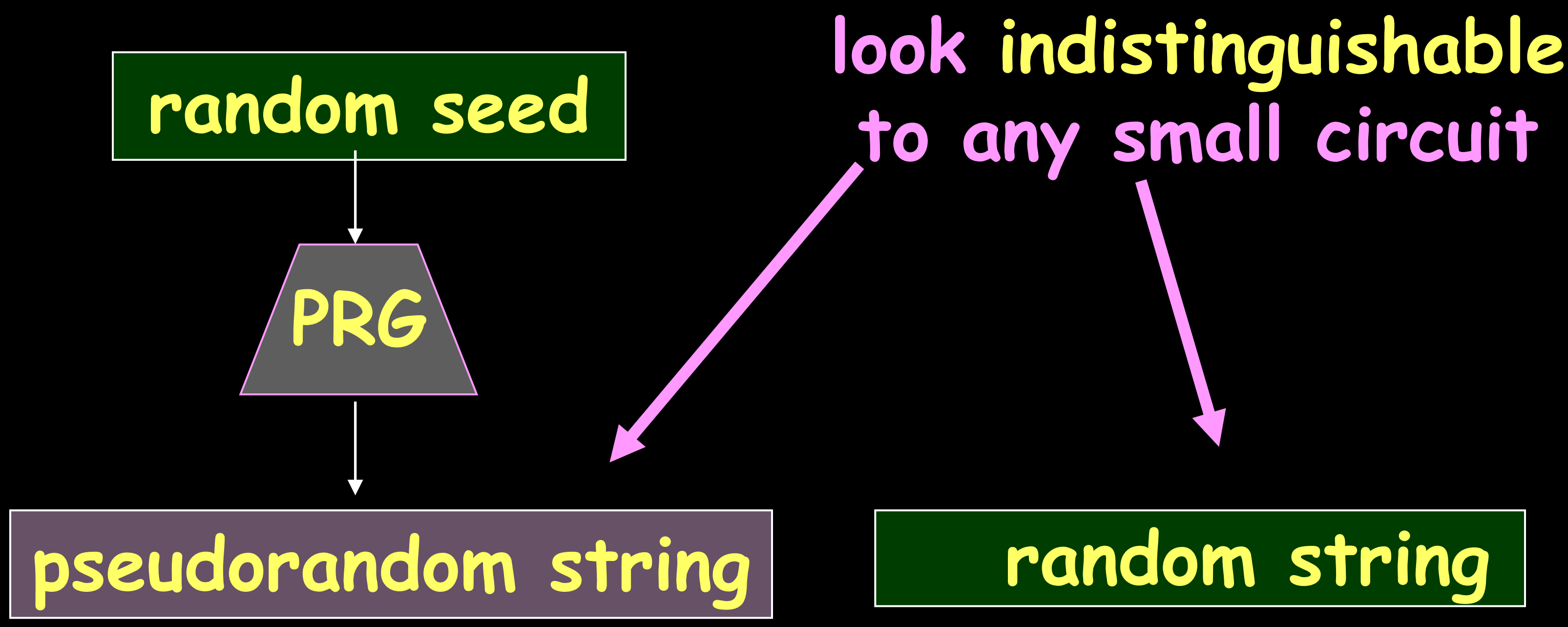
From BPP algorithms to Boolean circuits



Want to estimate the acceptance probability of circuit C (or find an accepted string, if C accepts many strings).

PseudoRandom Generator (PRG)

Definition [Nisan, Wigderson]: $s(n)$-PRG is a function $G: \{0,1\}^n \rightarrow \{0,1\}^{s(n)}$, with output distribution indistinguishable from uniform by any $s(n)$-size circuit; G is computable in time $2^{O(n)}$.

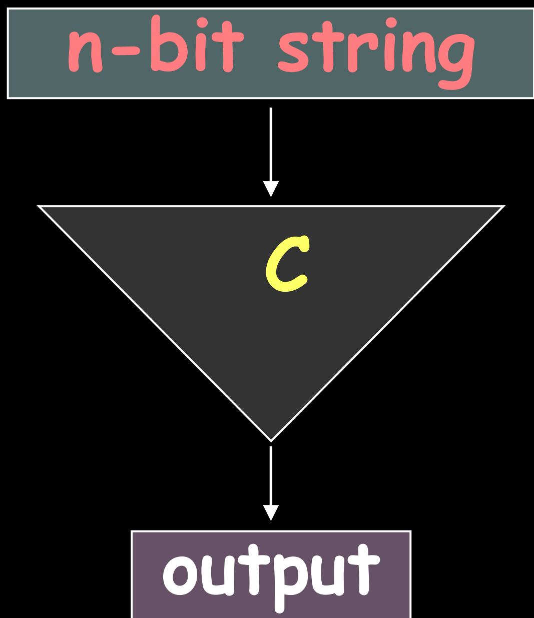


Hardness Generators

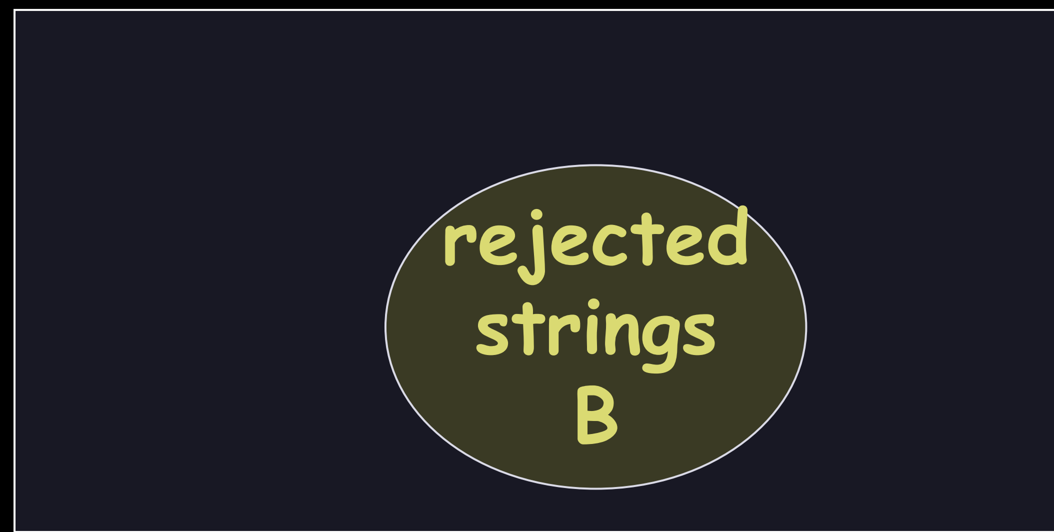
yield

Pseudorandom Generators

Incompressibility Argument



all n-bit strings



Each $r \in B$ has "small" description relative to C : $\log |B|$ bits specifying the rank of r in B, plus the description of C (common to all r in B)

Corollary: Any string incompressible relative to C is accepted by C.

Incompressibility Argument

Assume: $|B| < 2^n/n$, and $|C| < n$.
Let R be any incompressible n^2-bit string.
Partition R into n-bit strings $r_1, \ldots, r_n$.

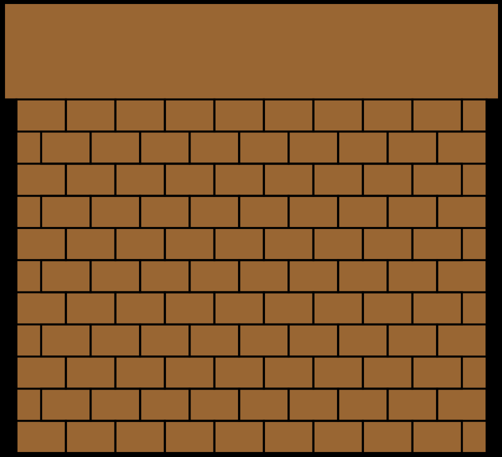
Claim: At least one r_i is accepted by C.

Proof: Else R is described by $< n\,(n - \log n) + n < n^2$ bits.

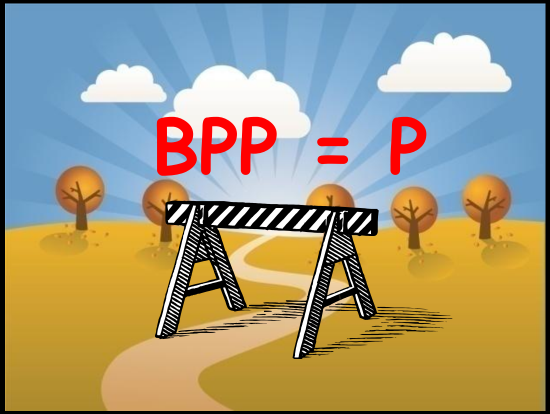
QED

Incompressibility Argument

Problem: Generating incompressible strings is algorithmically impossible !
(by definition)



[Nisan,Wigderson'88]: Enough to generate strings of **HIGH CIRCUIT COMPLEXITY** !!!



Hardness-Randomness Tradeoffs

[NisanWigderson,BabaiFortnowNisanWigderson,Impagliazzo,
ImpagliazzoWigderson,ImpagliazzoShaltielWigderson,
SudanTrevisanVadhan,ShaltielUmans,Umans,…]:

> Deterministic s(n)-Hardness Generator
> yields
> s(n)-Pseudorandom Generator,
>
> and vice versa.

Thm: s(n)-HG exists iff s(n)-PRG exists

Can we prove BPP = P without proving circuit lower bounds ?

BPP=P implies circuit lower bounds

Thm [K., Impagliazzo]:

If BPP = P, then

- either NEXP not in P/poly,
- or Permanent does not have polysize arithmetic circuits.

Why should a fast (deterministic) algorithm (for BPP) lead to any circuit lower bounds ?

Warm-up

Constructing a hard function from P = NP

Thm [Kannan]: There is a $2^n/n$ - Hardness Generator computable in $\mathrm{Time}(2^{O(n)})$, given Σ_3 oracle.

Proof Idea: Use alternating quantifiers to express " f is the first truth table not computable by any small circuit ". QED

Corollary: P=NP $\Rightarrow$ $\exists$ deterministic $2^n/n$-HG

($\Leftrightarrow$ E has language of $2^n/n$ circuit complexity)

How to construct a hard function

1. By diagonalization, construct a hard function in a “large“ complexity class.

2. Using an efficient meta-algorithm, collapse the “large“ class to a “smaller“ class.

How to construct a hard function: Application

1. By diagonalization, construct a hard function in a "large" complexity class.

$$E^{\Sigma_3} \text{ has } 2^n/n \text{ circuit complexity}$$

2. Using an efficient meta-algorithm, collapse the "large" class to a "smaller" class.

$$P = NP \Rightarrow E^{\Sigma_3} = E$$

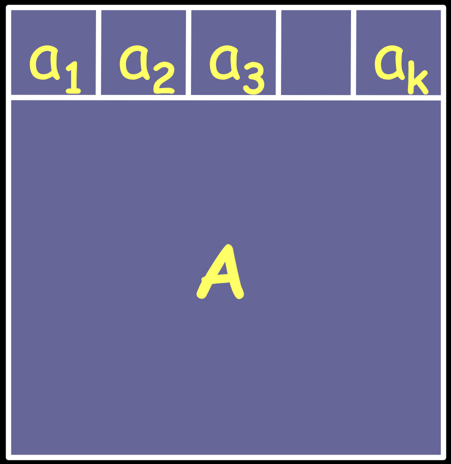
Constructing a hard function from BPP = P

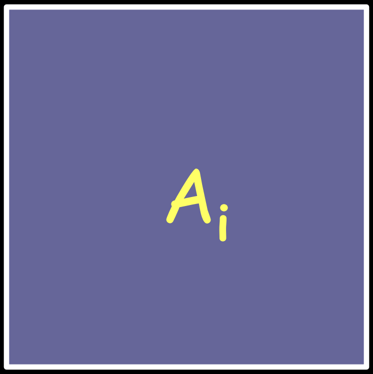
Meta-problem: Poly Identity Testing (PIT).

Main Observation: If PIT in P, then can test in P if a given arithmetic circuit computes Permanent.

(downward self-reducibility of Perm)

Downward self-reducibility of Perm





i^{th} minor of A along 1^{st} row

$$\mathrm{Perm}(A) = \sum_i a_i * \mathrm{Perm}(A_i)$$

Polynomial Identities for Perm

Let $C_1, \ldots, C_n$ be arithmetic circuits, where C_k has k^2 input variables.

The circuits $C_1, \ldots, C_n$ compute Permanent iff

- $C_1 (x) = x$, and
- $\forall \;\; 1 < k \leq n$, and $k \times k$ matrix $X = [x_{i,j}]$ of variables,

$$C_k (X) = \Sigma_{i=1..k} \;\; x_{1,i} * C_{k-1} (X_i),$$

where X_i is X without 1^{st} row and i^{th} column.

Main Observation

If PIT in P, then can test in P if a given arithmetic circuit computes Permanent.

Constructing a hard function from PIT in P

Assume PIT in P, and Perm has polysize arithmetic circuits. Then $P^{Perm} \subseteq NP$.

Corollary 1: $P^{\#P} \subseteq NP$. [Valiant]

Corollary 2: $PH \subseteq P^{\#P} \subseteq NP = coNP$. [Toda]

Corollary 3: $E^{PH} = NE = coNE$ requires $2^n/n$ circuit size.

Thanks [Aaronson, van Melkebeek].

Derandomization of PIT from Arithmetic Circuit Lower Bounds

Thm [K., Impagliazzo]: If Perm requires arithmetic circuits of size $2^{n^{\epsilon}}$ (over rationals), then PIT $\in$ DTIME ($n^{\text{polylog } n}$).

Hitting set H for poly(n)-size n-variate arithmetic circuits (computing poly(n)-deg polynomials):

$$H = \{ (\text{Perm}(a_{i,1}), \ldots, \text{Perm} (a_{i,n})) :$$

$$a_{i,j} \in [n^c]^{d \log n} \text{ chosen using the NW design } \}$$

PIT is easy

iff

can prove circuit lower bounds

Meta-algorithms vs. Lower Bounds

Meta-algorithm = an algorithm that takes algorithms as input (e.g., SAT, Poly Id Test, …)

Zane's thesis: Progress on meta-algorithms is linked to progress on lower bounds.

LinialMansourNisan, PaturiPudlakSaksZane, RazborovRudich, NisanWigderson, Braverman, …

Constant Depth

PIT for constant-depth circuits

[DvirShpilkaYehudayoff]: Derandomization iff lower bounds (similar to [KI])

Depth-3 derandomization (bounded top fanin): [DvirShpilka, KayalSaxena, ArvindMukhopadyay, KarninShpilka, SaxenaSeshadri, KayalSaraf, …]

Challenge: Depth-3 circuits (unbounded fanin)

PIT from constant-depth lower bounds

[AgrawalVinay]:

Exponential depth-4 circuit lower bounds $\Rightarrow$ exponential (any depth) circuit lower bounds

$\Rightarrow$ general Circuit-PIT $\in$ $n^{\mathrm{polylog}\, n}$ time

[Raz'09]:

Exponential depth-3 formula lower bounds $\Rightarrow$ superpoly (any depth) formula lower bounds

$\Rightarrow$ general Formula-PIT $\in$??? time

Derandomization without circuit lower bounds ?

Weak

from weak

Derandomization ~~without~~
circuit lower bounds ?

Typically-correct derandomization

Relaxation: Allow derandomized algorithms to make mistakes on "few" inputs.

[Impagliazzo, Wigderson '01]:

A language L is in Heur-P if there is a determistic polytime algorithm A s.t.

$$\Pr_{x \leftarrow \mathcal{D}} [\, A(x) \neq L(x) \,] \text{ is "small"},$$

for any polytime-sampleable D.

[Goldreich, Wigderson ' 02]: D = Uniform

BPP vs. EXP

[Impagliazzo, Wigderson '01]:

EXP $\neq$ BPP $\Rightarrow$ BPP $\subseteq$ io- Heur-SUBEXP.

Cf. [BabaiFortnowNisanWigderson]:

EXP not in P/poly $\Rightarrow$ BPP $\subseteq$ io- SUBEXP

" EXP $\neq$ BPP " is not known to imply any circuit lower bounds ...

Cf. [IKW]: NEXP $\neq$ MA $\Leftrightarrow$ NEXP not in P/poly

Typically-correct derandomization

[IW'01, K'01, TV'07, GSTS'03, SU'07, GW'02, Zim'08, Sha'09, KMS'09]

[Kinne, Melkebeek, Shaltiel'09]: Under assumption (*), every BPP language has a P-algorithm that is correct on almost all inputs (of every length).

Assumption (*): P has a language that is average-hard for n^d-size circuits.

Typically-correct derandomization and circuit lower bounds

[Kinne, Melkebeek, Shaltiel'09]: If every BPP language has a SUBEXP-algorithm that is correct on all but subexp-many inputs, then

- either NEXP not in P/poly,
- or Perm is not computable by polysize arithmetic circuits.

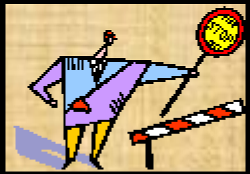
(extends [KI'04] to "typically-correct" setting.)

More on Hardness

Hardness Testing

Given a binary string x, test if x has "high" circuit complexity.

➢ Sound test accepting "many" strings is unlikely in P ("natural property" [RazborovRudich]).



➢ Sound test accepting "few" strings ?

[IKW'02] : $\exists$ sound test in NP $\Rightarrow$ NEXP not in P/poly.

That is, NP-constructivity $\Rightarrow$ lower bounds

Open Questions

- Strongly exponential arithmetic circuit lower bounds $\Rightarrow$ PIT $\in$ P ?

- Strong arithmetic formula lower bounds $\Rightarrow$ derandomization of Formula-PIT ?

- BPP $\neq$ EXP $\Rightarrow$ circuit lower bounds ?

  (extending [KMS'09] ???)