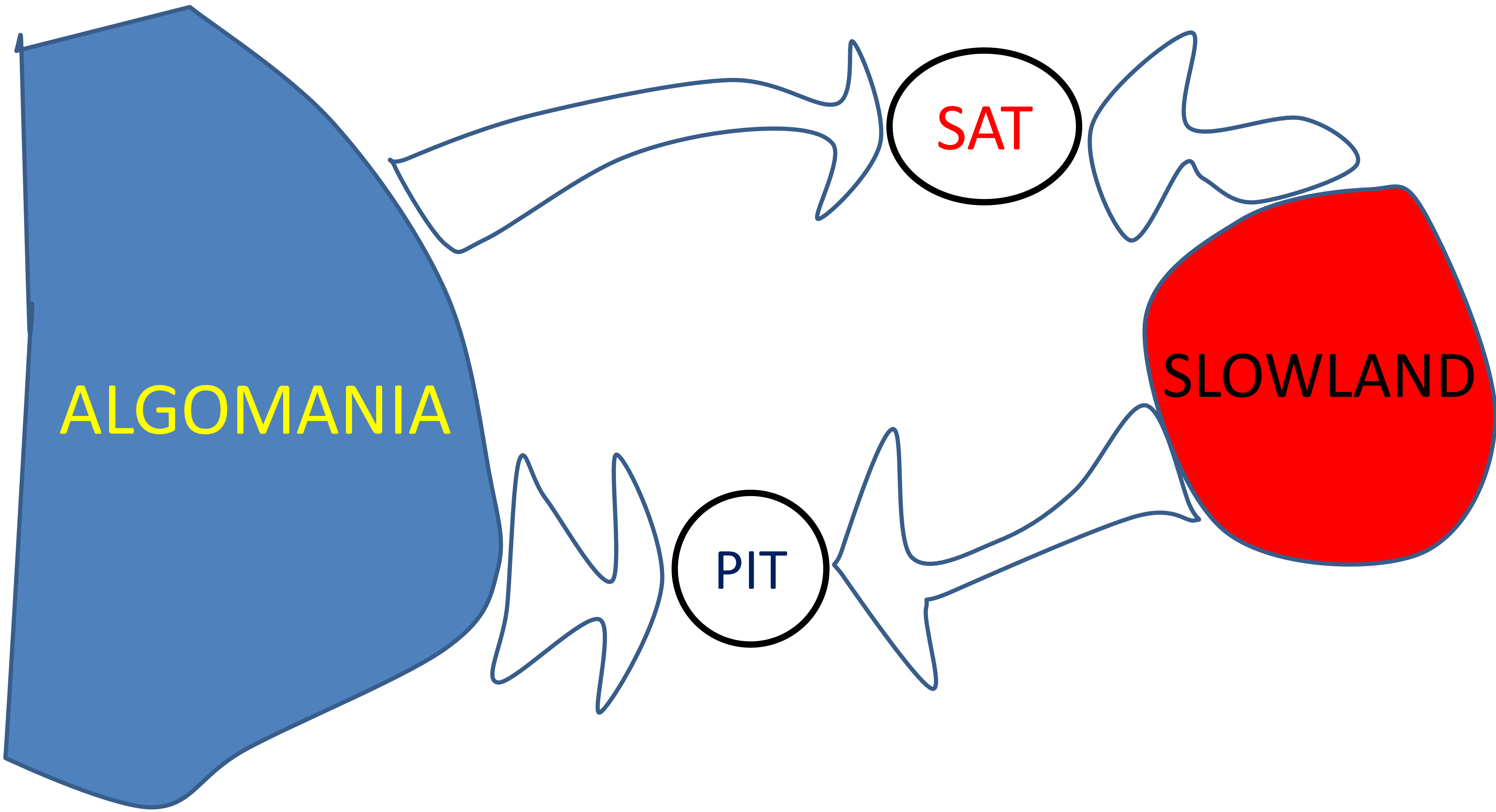
Meta-Algorithms vs. Circuit Lower Bounds

Valentine Kabanets

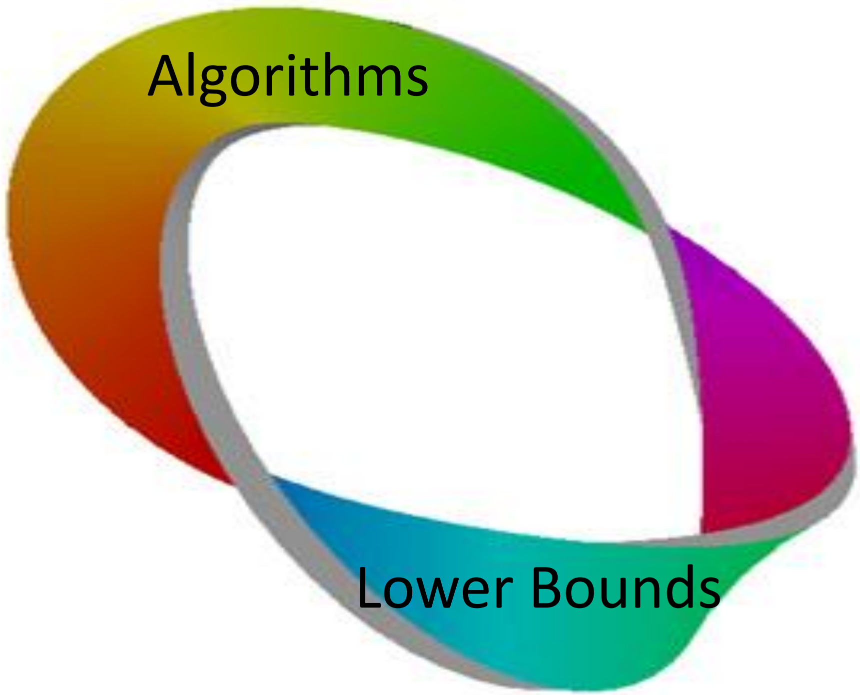
Simon Fraser University

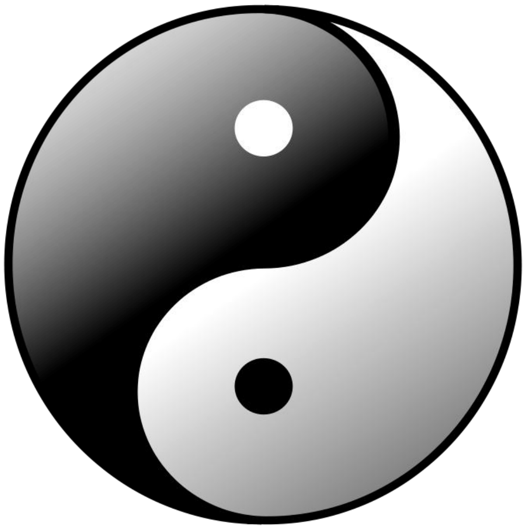
Vancouver, Canada

Algorithms vs Lower Bounds



Algorithms ⇔ Lower Bounds

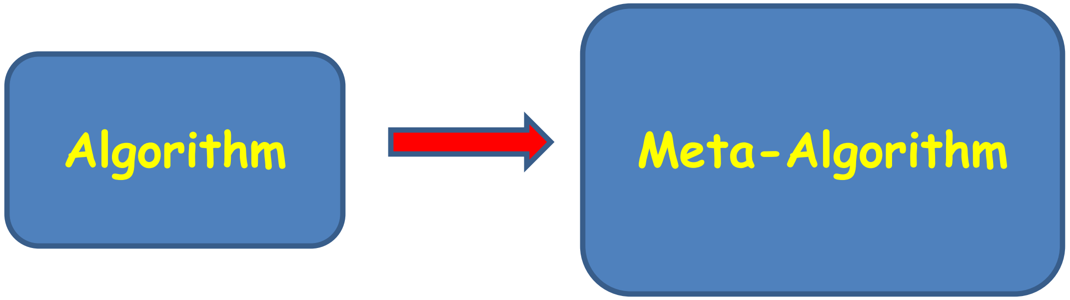




Algorithms yield lower bounds.
Lower bounds yield algorithms.

Meta-Algorithms

Algorithms operating on algorithms



Examples of Meta-Algorithms

- Computability Theory :

  Virus checker ,

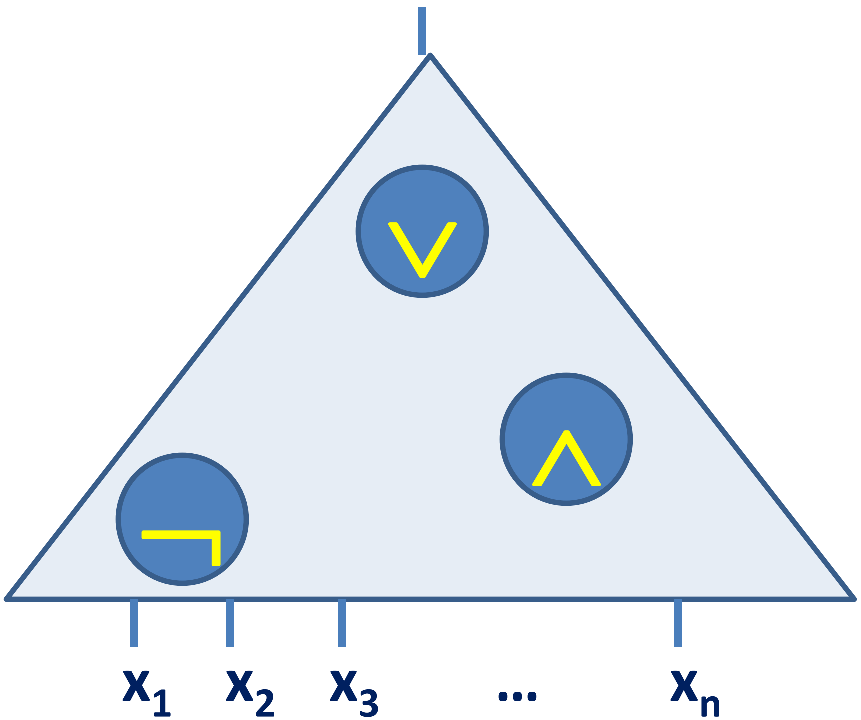
  Infinite-loop detector (aka Halting problem)

- Complexity Theory :

  SAT ,

  Polynomial Identity Testing (PIT)

Circuit - SAT

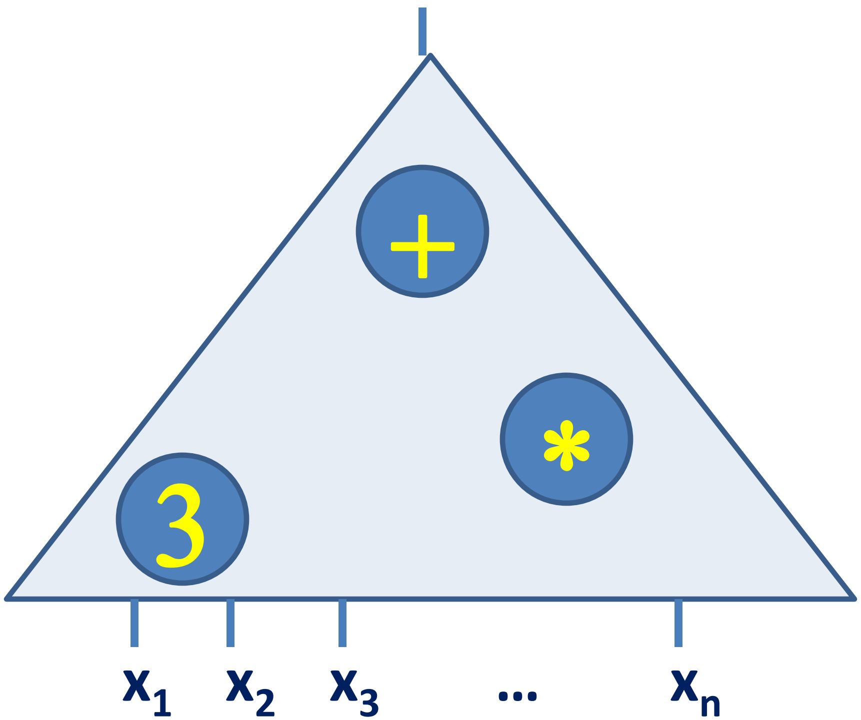


<u>Given:</u> poly(n)-size circuit C on n inputs.

<u>Decide:</u> Is C satisfiable ?

Canonical NP -complete problem. [Cook; Levin]

Polynomial Identity Testing

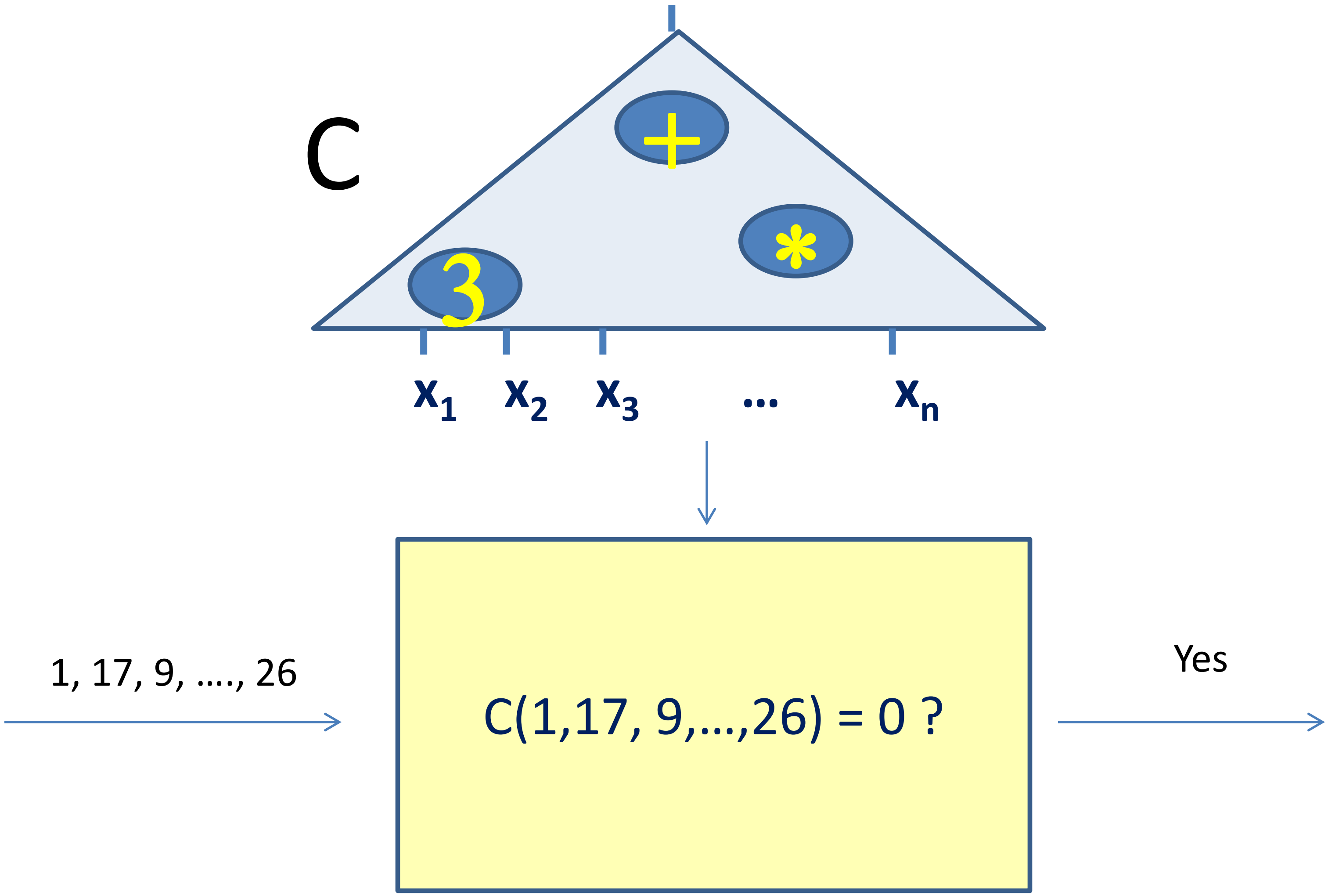


<u>Given:</u> poly(n)-size arithmetic circuit C on n inputs (over integers).

<u>Decide:</u> Is $C \equiv 0$?

Extra structure (C is a polynomial) makes PIT easier than UNSAT :
PIT in BPP [Schwartz, Zippel, DeMillo-Lipton, …]

Randomized PIT-Algorithm



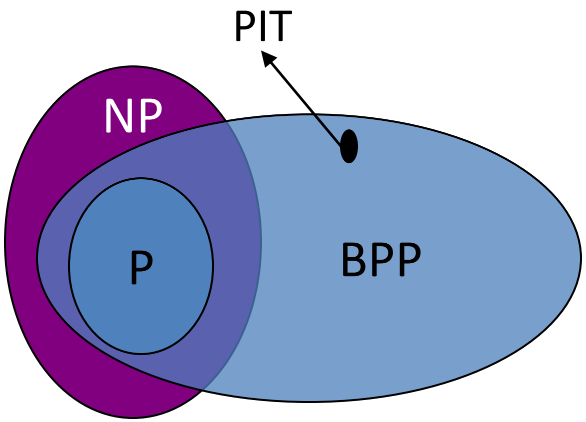
If $C \equiv 0$, then always correct. Else, correct with high probability. [Schwartz-Zippel]

Meta-Algorithms from Circuit Lower Bounds :

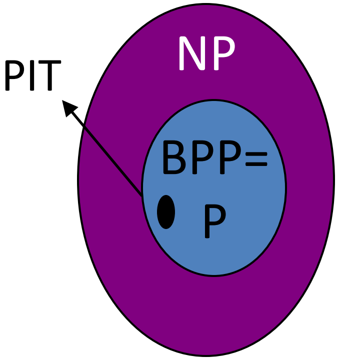
" Black-Box " use of lower bounds

Is randomness useful ?

Can we remove the need for random coins in algorithms, without much slowdown ?



OR



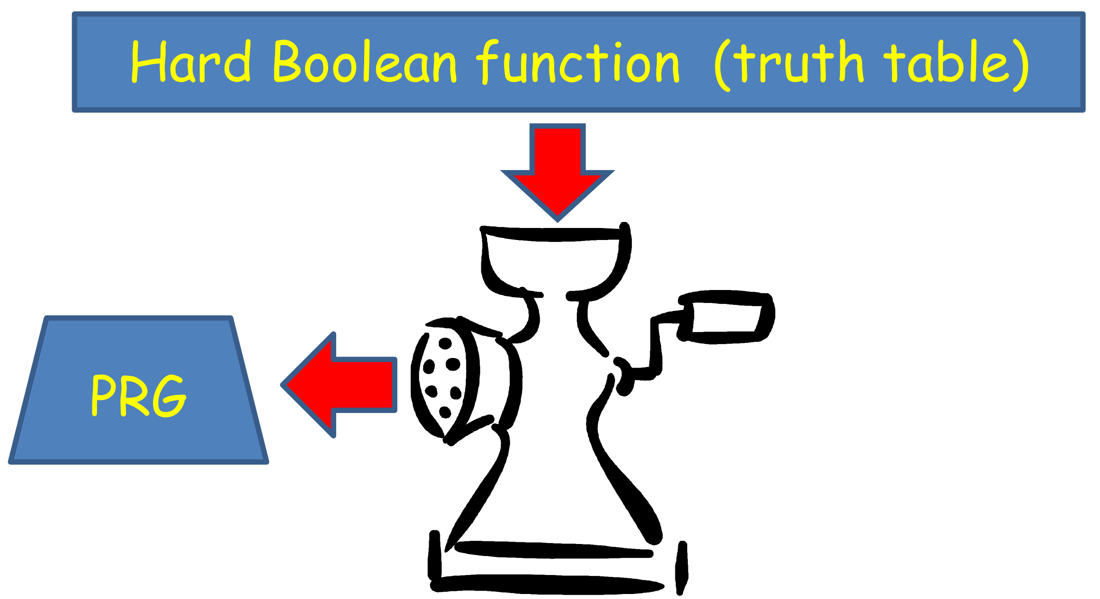
?

Derandomization

Computational Hardness $\Rightarrow$

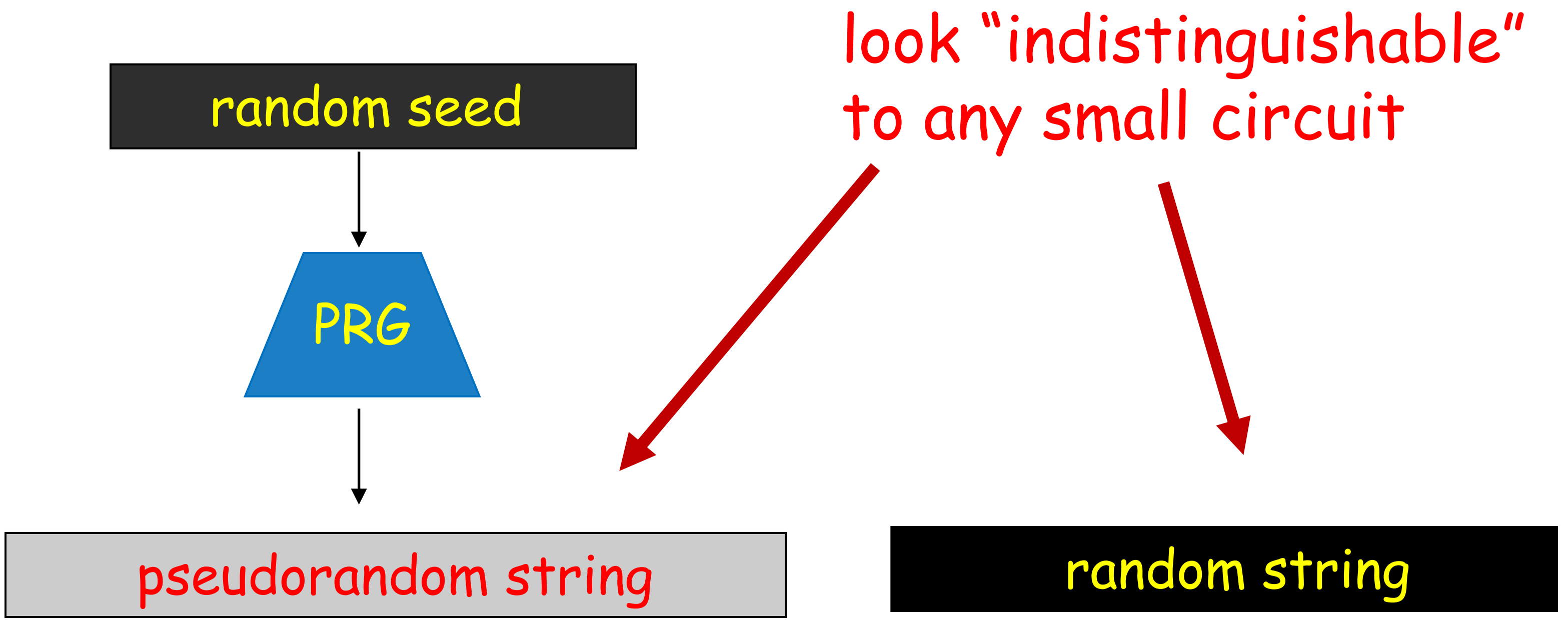
Computational Randomness (pseudorandomness)

[Blum, Micali, Yao; Nisan & Wigderson, Babai et al., …]

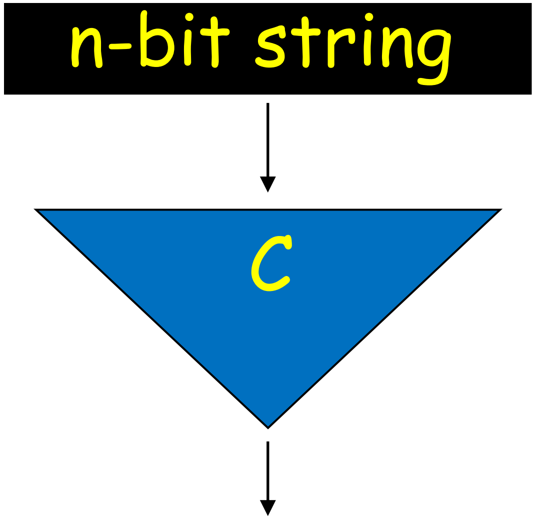


Hardness – into – Randomness Converter

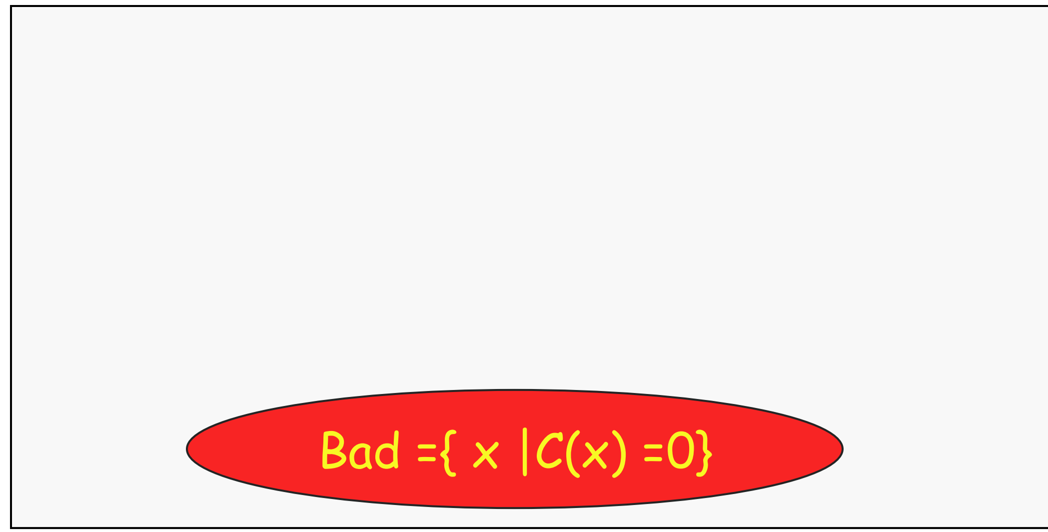
Pseudo-Random Generator (PRG)



Incompressibility Argument



all n-bit strings

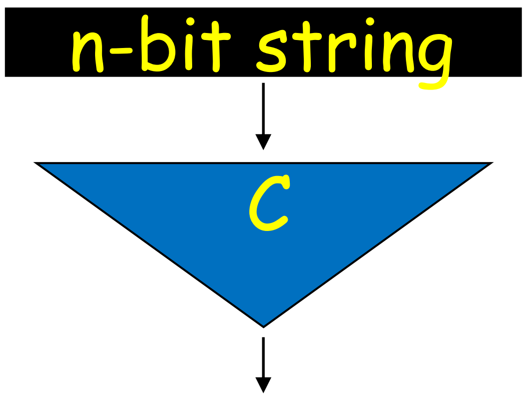


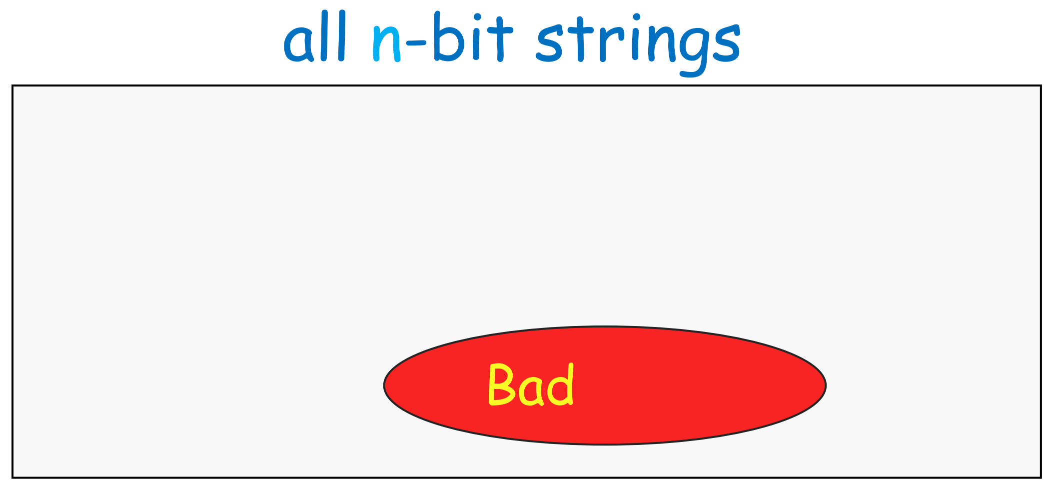
Each $x \in$ Bad has "small" description relative to C :

log |Bad| bits specifying the rank of x in Bad, plus the description of C

Any string incompressible relative to C is accepted by C.

Incompressibility Argument





Each $x \in$ Bad has "small" description relative to C

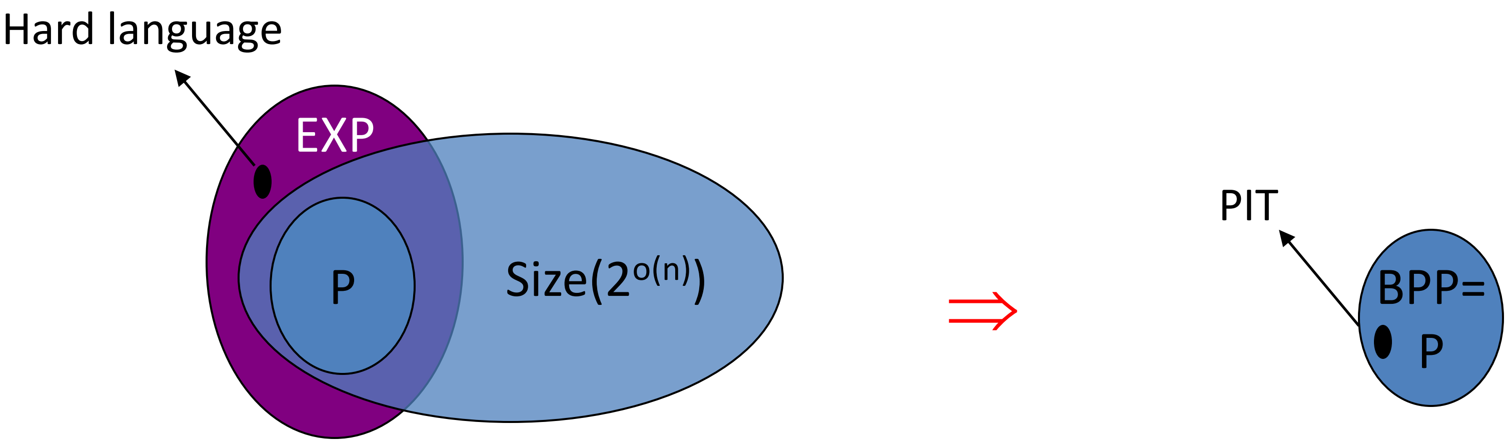
Any string incompressible relative to C is accepted by C.

High circuit complexity

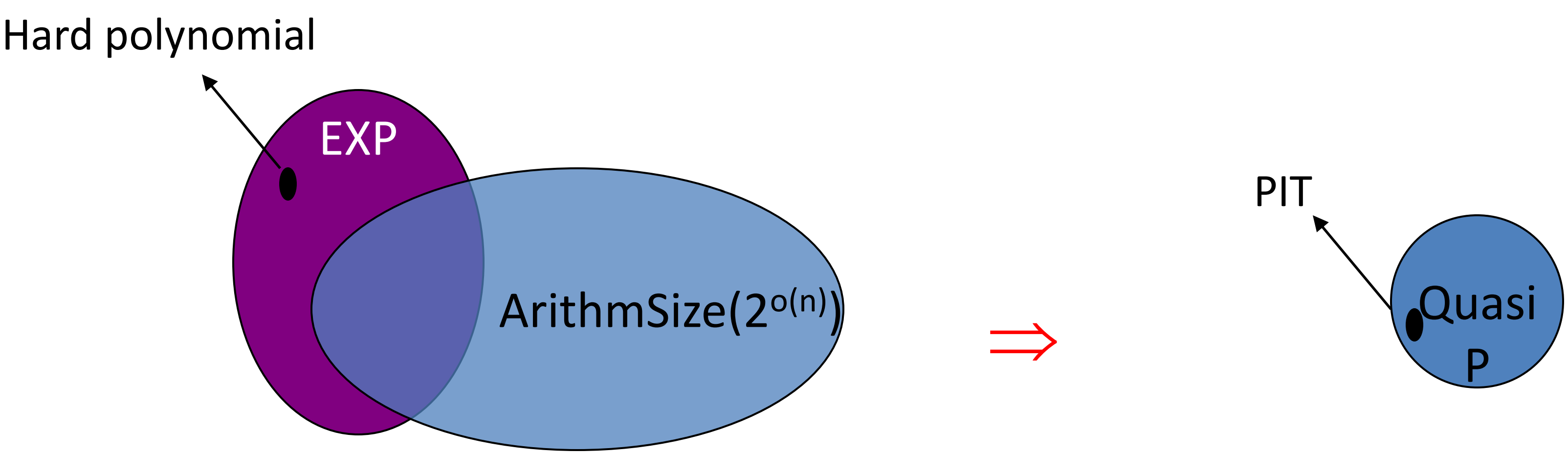
=

incompressibility relative to small circuits

Hardness into Randomness



EXP requires circuit size $2^{\Omega(n)}$ $\Rightarrow$ BPP = P [Impagliazzo & Wigderson]



EXP requires arithmetic circuit size $2^{\Omega(n)}$ $\Rightarrow$ PIT in Time($n^{\text{polylog } n}$) [K. & Impagliazzo]

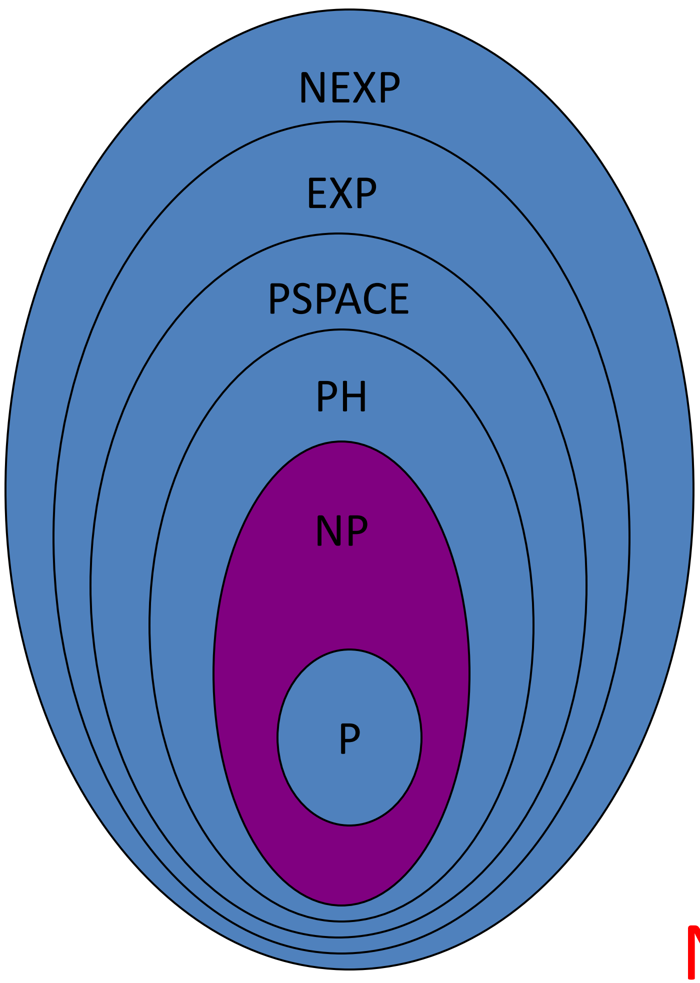
Non - " Black-Box " Use of Circuit Lower Bounds

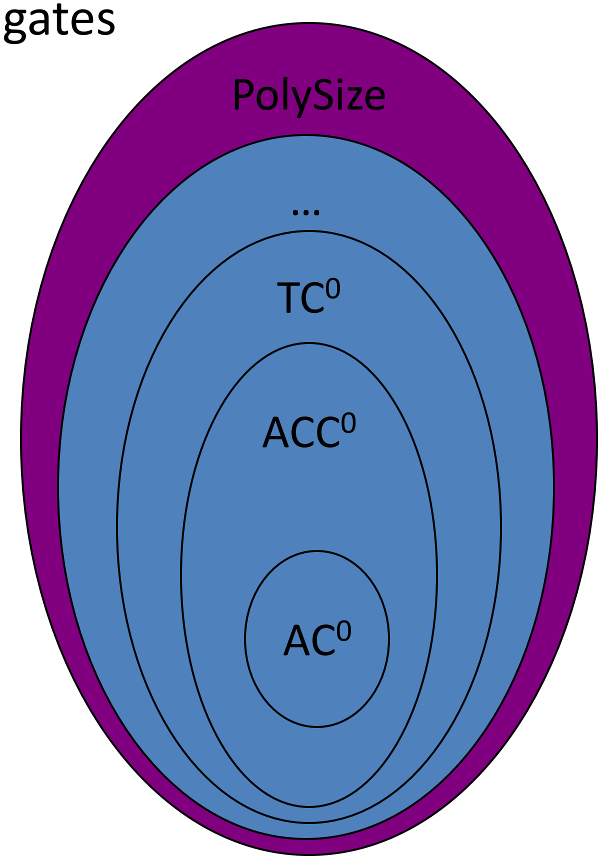
Elusive Circuit Lower Bounds

AC^0: Constant depth, unbounded fan-in, poly-size

ACC^0: **AC^0** with MOD m gates

TC^0: **AC^0** with MAJ gates





NEXP in PolySize ?

Natural Proofs [Razborov, Rudich]

A combinatorial property T of n-variable Boolean functions is natural against a class *C* if it is

- Constructive: "f in T" is decidable in poly(2^n) time
- Large: $|T| > 1/\text{poly}(2^n)$ of all n-variable fns
- Useful against *C*: f in T $\Rightarrow$ f not in *C*

Natural Proofs $\Rightarrow$ No Crypto

A natural proof of a circuit lower bound =
a proof using a natural property .

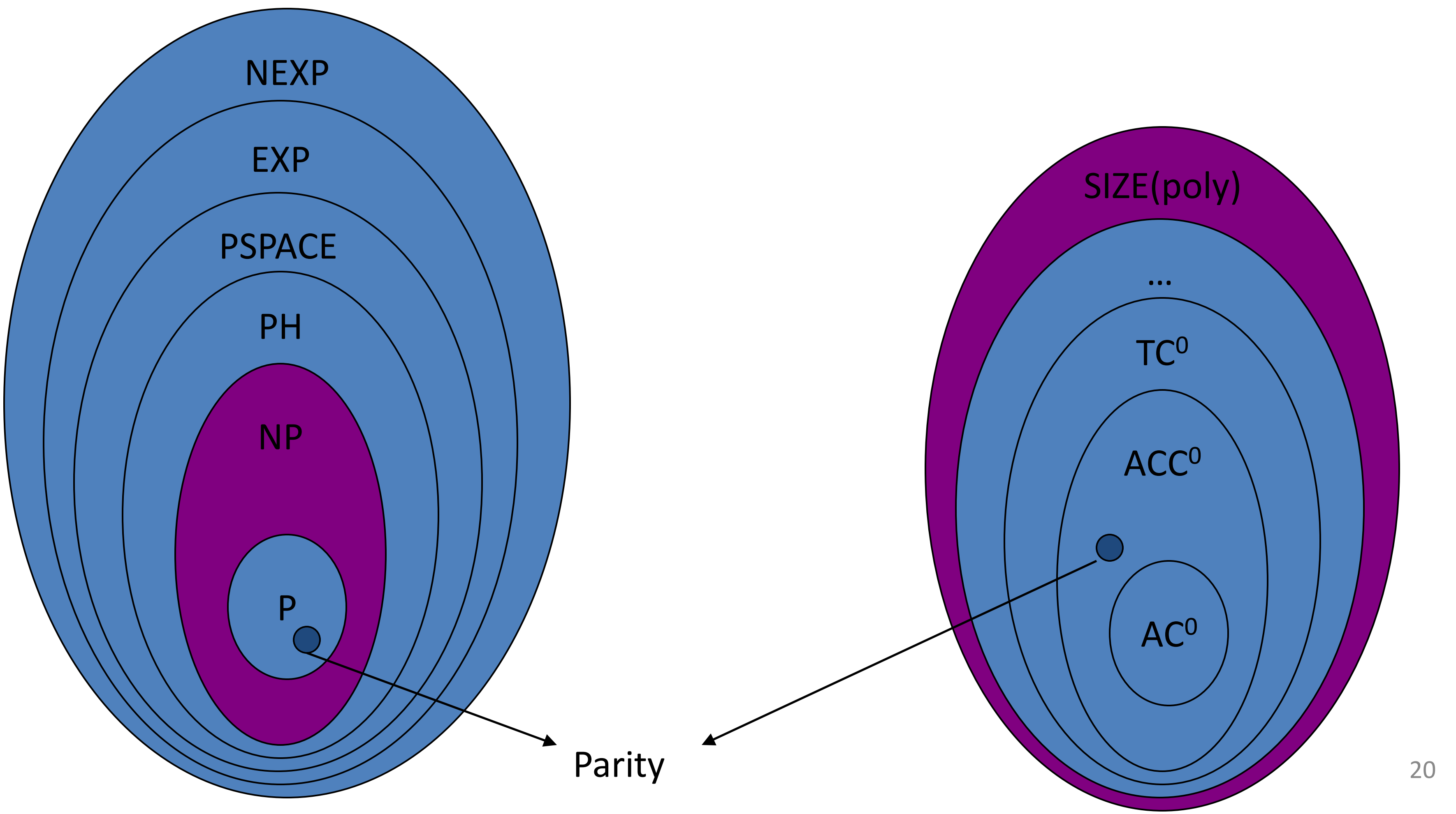
Theorem [Razborov, Rudich]:

A natural proof of a circuit lower bound against a class C $\Rightarrow$
algorithm breaking every candidate PRG implemented in C

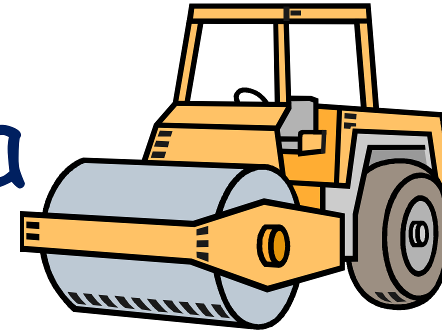
(i.e., class C cannot compute a strong PRG)

Parity is not in AC^0 [FSS+84, Yao85, Hastad86]

AC^0: Constant depth, unbounded fan-in, poly-size



Switching Lemma



Given an AC^0 circuit $C(x_1,\dots,x_n)$,

- Choose a random subset of variables,
- Assign them to 0 or 1 randomly.

Very likely, the circuit becomes shallow.

[Hastad]

C not too large $\Rightarrow$ can make it a constant function, with some variables still free.

So, C can't compute PARITY.

AC^0-functions are sparse

small AC^0-circuit

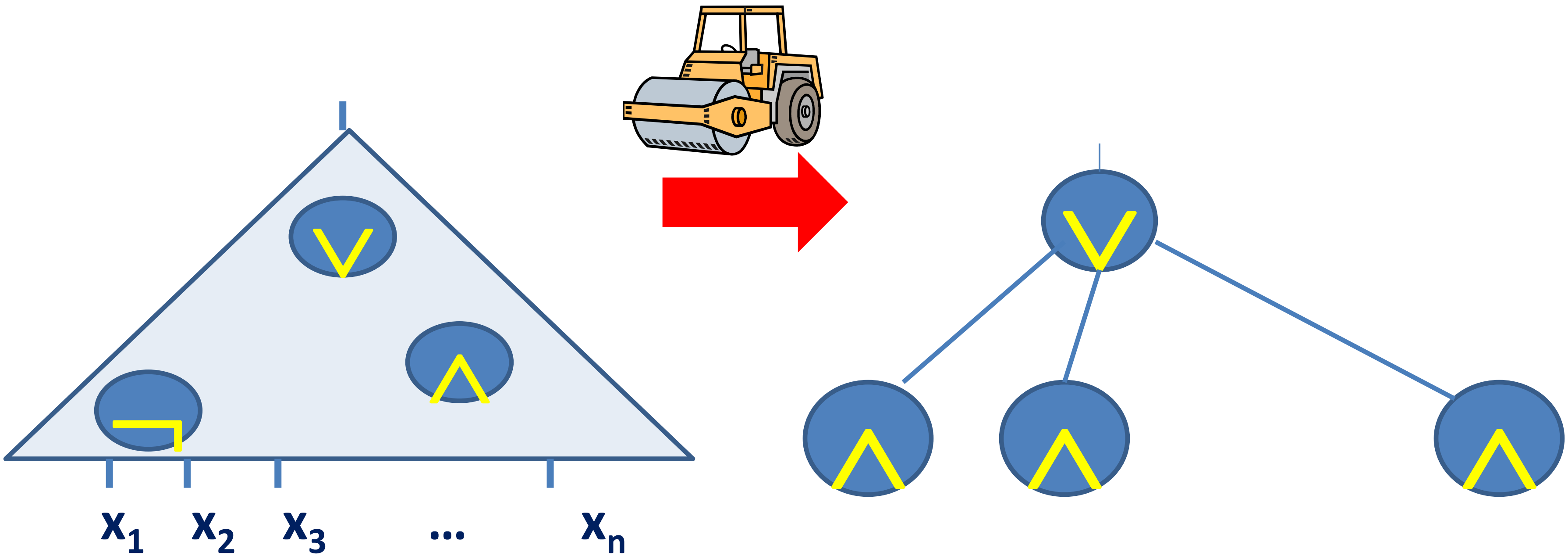
$\approx$

sparse Fourier representation

[Linial, Mansour, Nisan]

Can approximately learn AC^0 -computable functions.

AC^0-SAT faster than "brute-force"



AC^0 circuit:
size cn, depth d

DNF : $\leq 2^{n(1-\mu)}$ ANDs, with
$\mu = 1/(\log c + d \log d)^{d-1}$

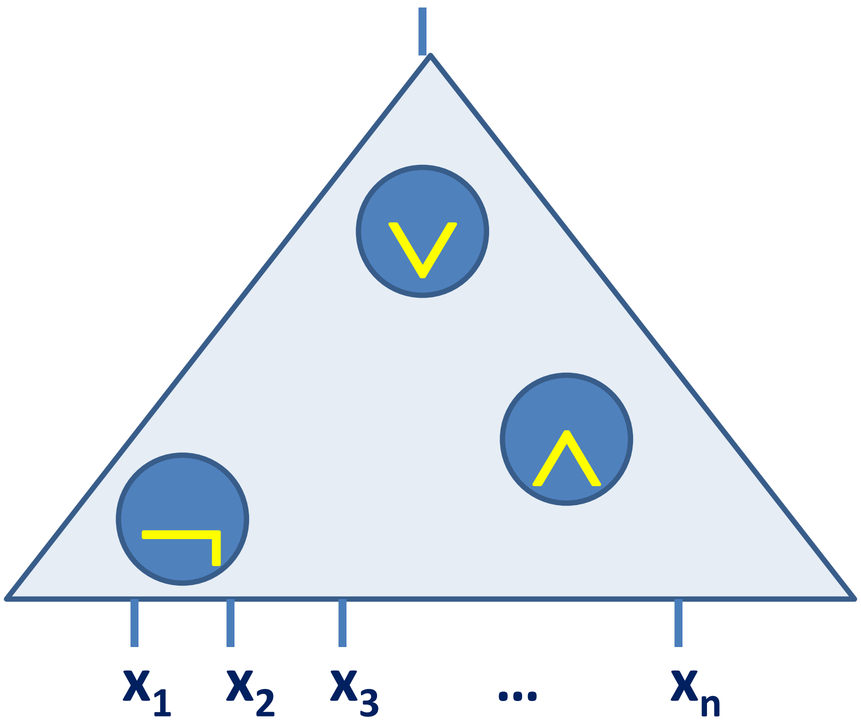
SAT for such DNFs is easy !

ANDs have disjoint sets of satisfying assignments

[Impagliazzo, Mathews, Paturi]

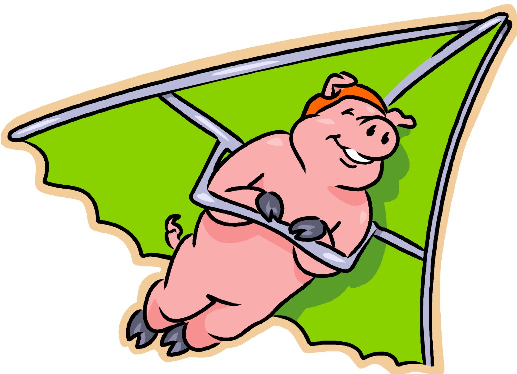
Circuit Lower Bounds from Meta-Algorithms

Circuit - SAT



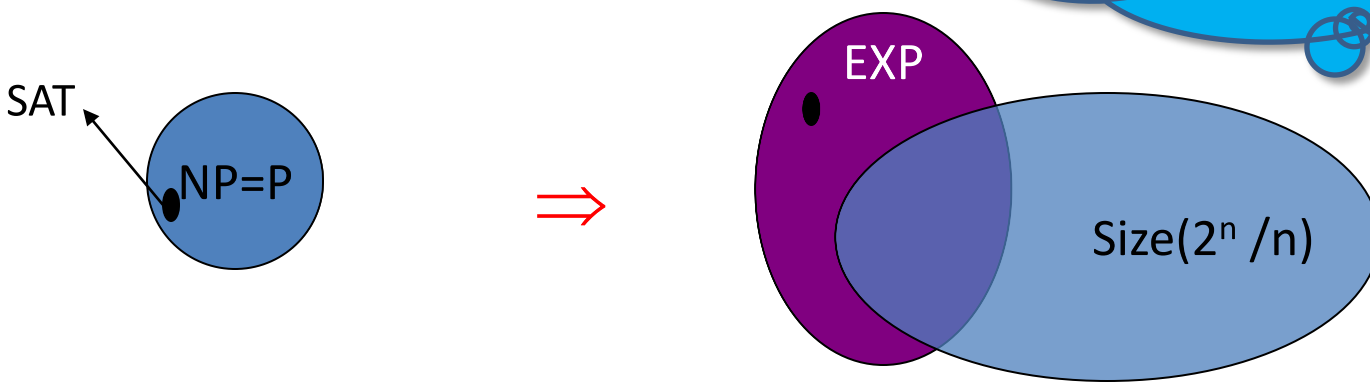
Given: poly(n)-size circuit C on n inputs.

Decide: Is C satisfiable ?



If pigs can fly …

$EXP \subset SIZE(2^n/n)$
$\Rightarrow \; P \neq NP$



If SAT in P, then EXP requires circuit size $> 2^n/n$ [Kannan]

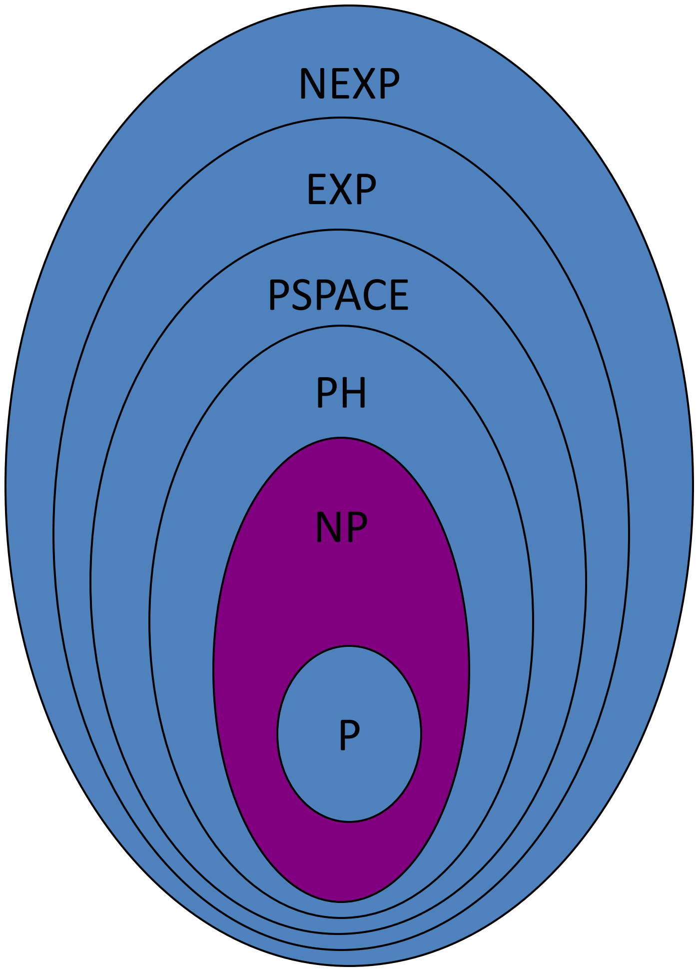
Facts:

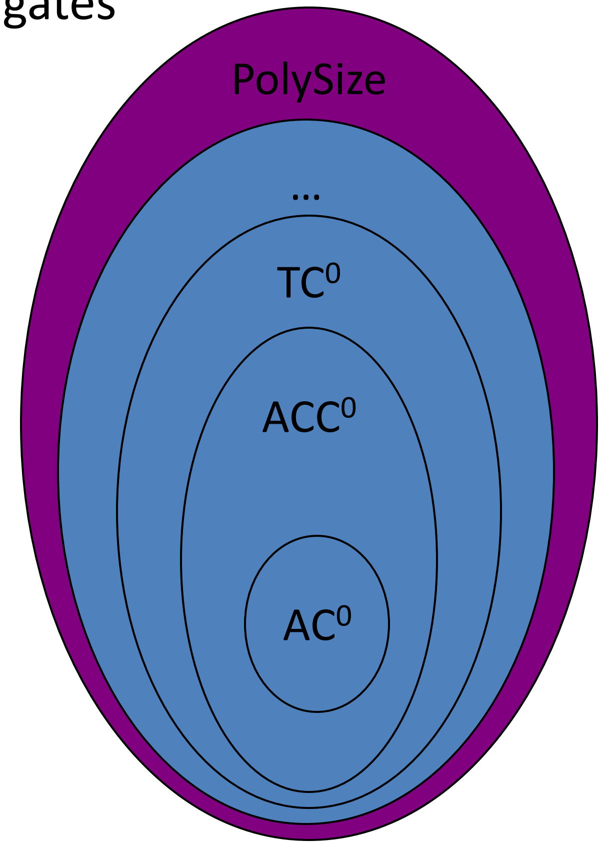
- Almost all Boolean functions $f(x_1,\ldots,x_n)$ require circuit size $> 2^n/n$.
- But, open if NEXP $\subset$ PolySize .

Can we use this approach to get
any actual
circuit lower bounds ???

Elusive Circuit Lower Bounds

AC^0: Constant depth, unbounded fan-in, poly-size
ACC^0: **AC^0** with MOD m gates
TC^0: **AC^0** with MAJ gates



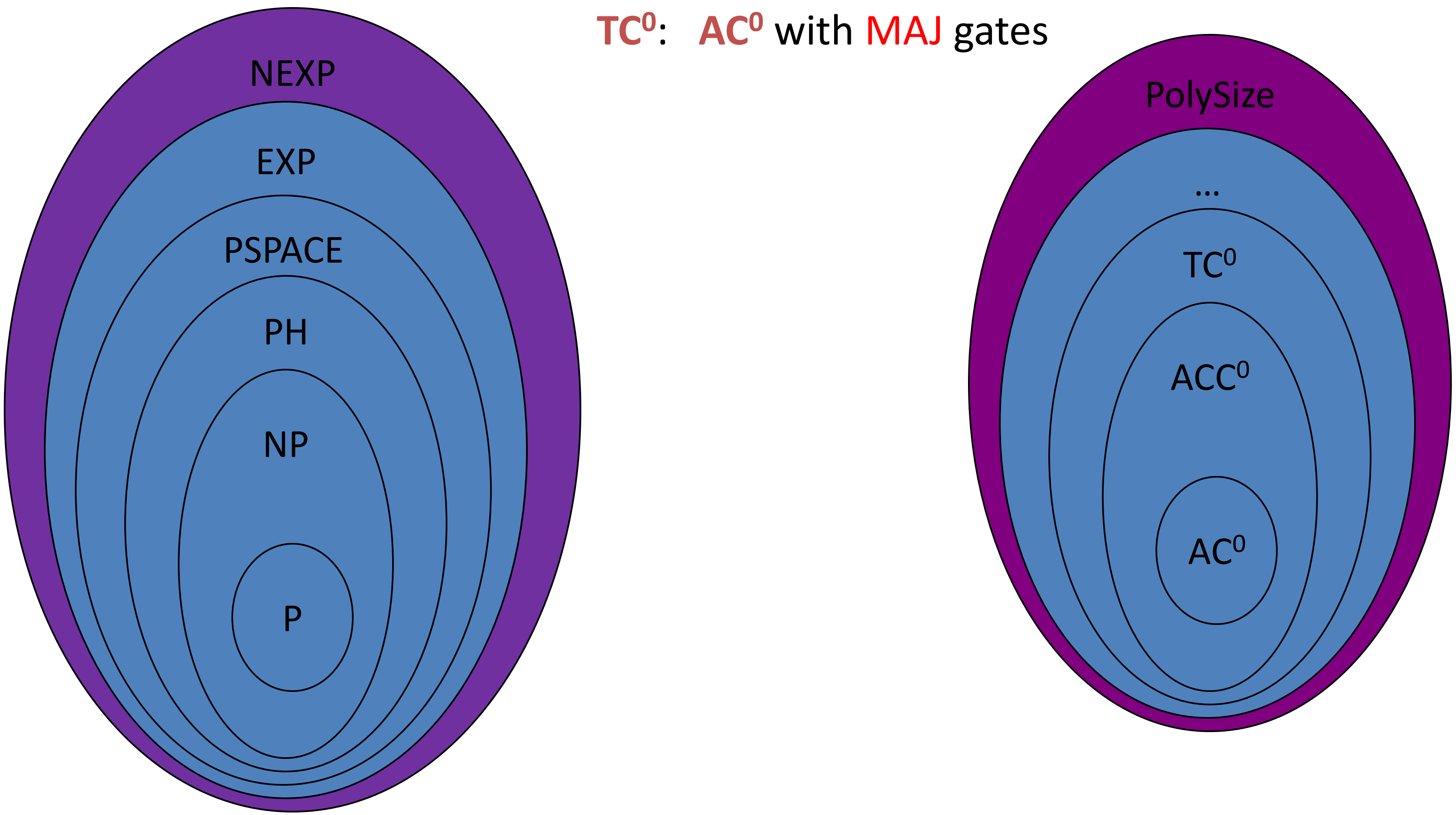


Elusive Circuit Lower Bounds

AC^0: Constant depth, unbounded fan-in, poly-size

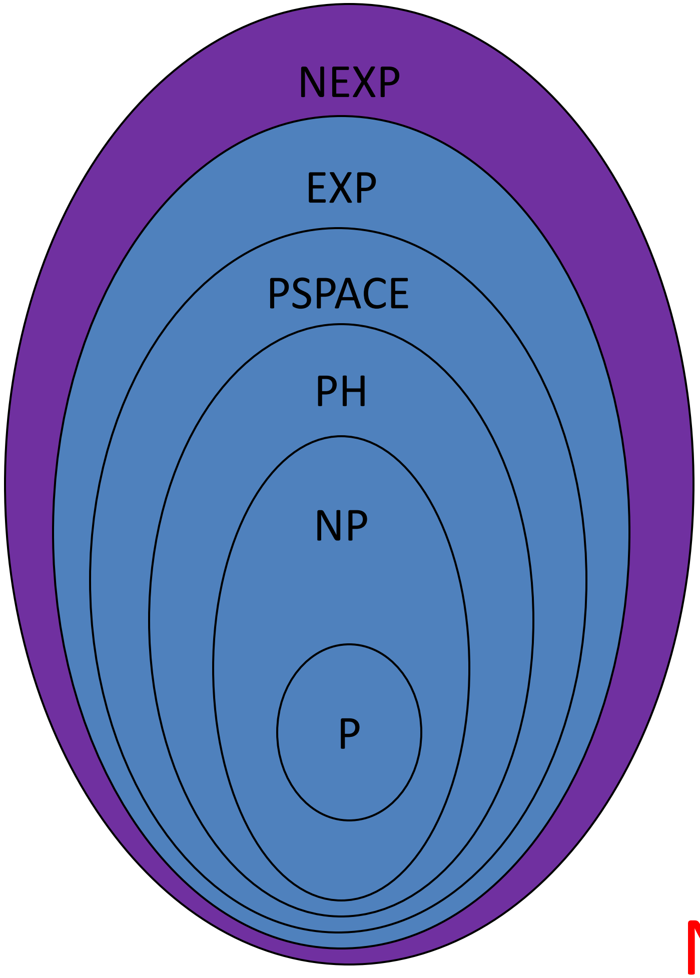
ACC^0: AC^0 with MOD m gates

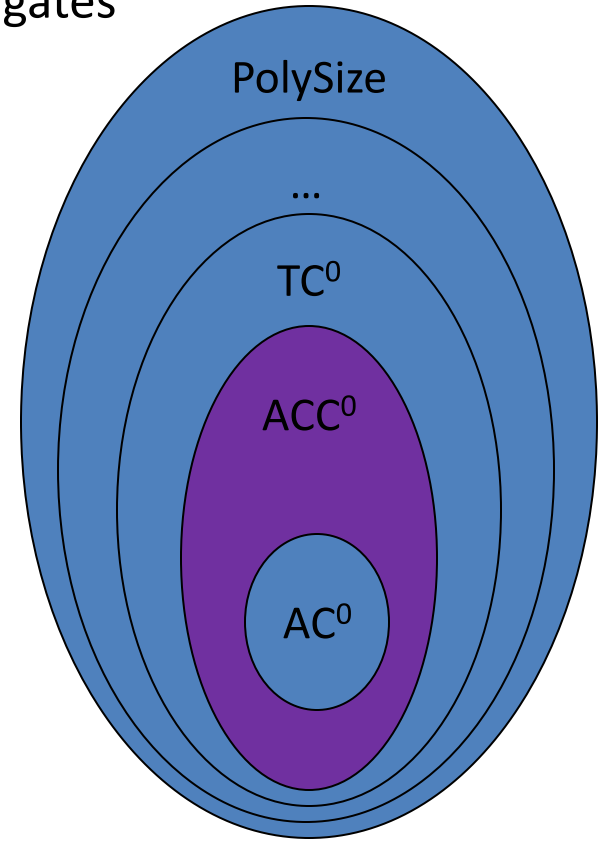
TC^0: AC^0 with MAJ gates



Elusive Circuit Lower Bounds

AC^0: Constant depth, unbounded fan-in, poly-size
ACC^0: **AC^0** with MOD m gates
TC^0: **AC^0** with MAJ gates





NEXP in ACC^0 ?

Williams' Circuit Lower Bound

Theorem 1: If can solve $\mathcal{C}$ - SAT slightly better than "brute-force", then NEXP not in $\mathcal{C}$ - PolySize.

Theorem 2: ACC^0 - SAT can be solved faster than "brute-force".

Corollary: NEXP not in ACC^0.

$\mathcal{C}$ - Circuit Satisfiability

Theorem 1. There is $k>0$ such that :

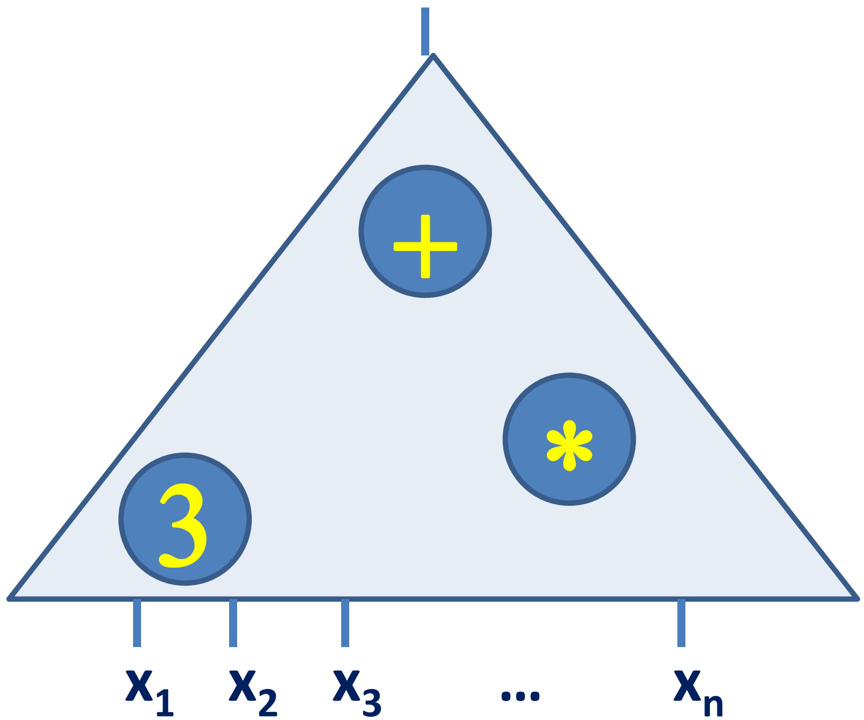
If $\mathcal{C}$ - SAT for n^c-size n-input circuits is in time $O(2^n/n^k)$ for every c,

then $NTime(2^n)$ is not in $\mathcal{C}$- PolySize.

Contrast: "Brute-force" $\mathcal{C}$-SAT algorithm is in time $2^n\, poly(n^c)$.

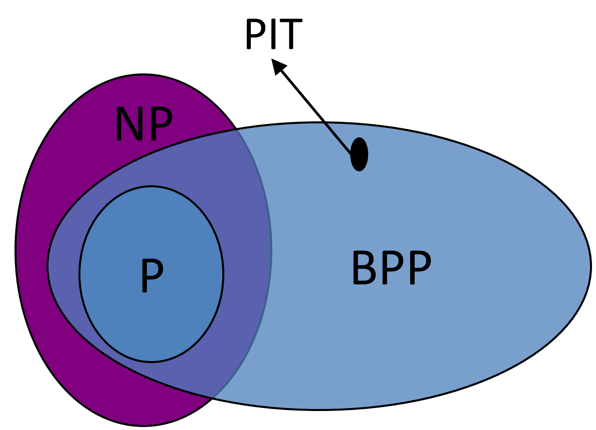
Circuit Lower Bounds from PIT-Algorithms

Polynomial Identity Testing (PIT)

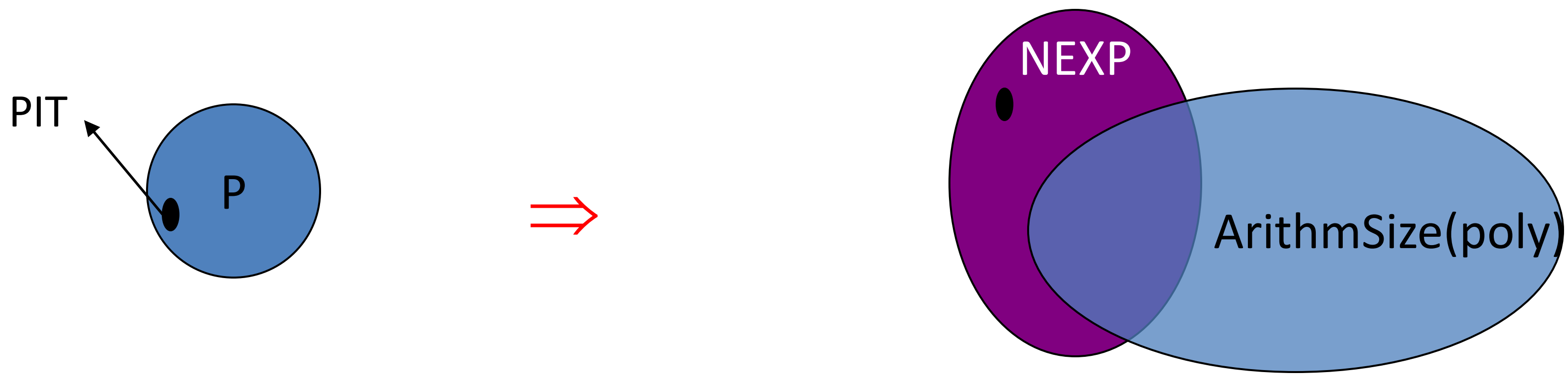


Given: poly(n)-size arithmetic circuit C on n inputs (over integers).

Decide: Is $C \equiv 0$?



PIT easy $\Rightarrow$ NEXP hard



PIT in P $\Rightarrow$ NEXP requires superpoly-size arithmetic circuits

(i.e., NEXP not in PolySize, OR Permanent not in Arithmetic PolySize) [K., Impagliazzo]

Important Polynomial Identities

Permanent: For $X = (x_{i,j})_{n \times n}$

$$Perm_n(X) = \sum_{\pi} \prod x_{i,\pi(i)}$$

Defining Identities (expansion by minors):

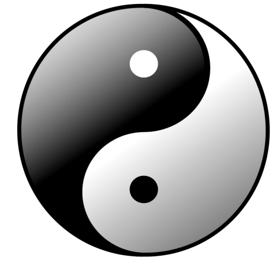
$$Perm_n(X) \equiv \sum x_{1,j} \, Perm_{n-1}(X^{1,j})$$

$$\cdots$$

$$Perm_1(x) \equiv x$$

PIT easy $\Rightarrow$ efficient program-checker for Permanent

Summary



- Good meta-algorithms (SAT, PIT, Learning)

strong circuit lower bounds

- Can get unconditional results on each side.

Some Challenges

- Non-black-box "SAT-Algorithms $\Rightarrow$ Circuit Lower Bounds" conversions ?

[BPP = P $\Rightarrow$ circuit l.b. for NEXP, but PRG $\Rightarrow$ circuit l.b. for EXP]

- Better circuit lower bounds for AC^0 $\Rightarrow$ better SAT-algorithms ?

[beyond the Switching Lemma ?]