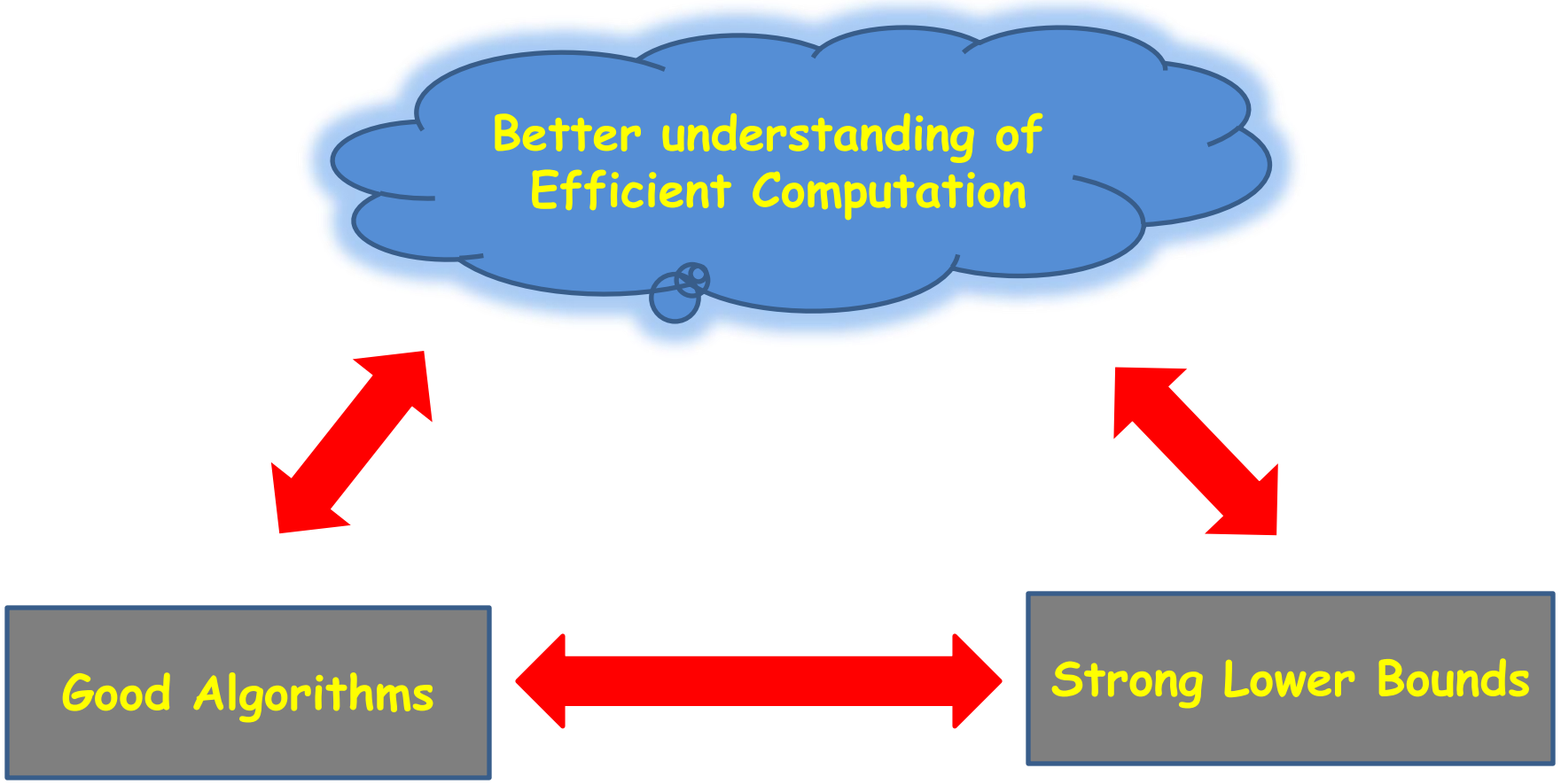
Meta-Algorithms vs. Circuit Lower Bounds

Valentine Kabanets

Tokyo Institute of Technology &

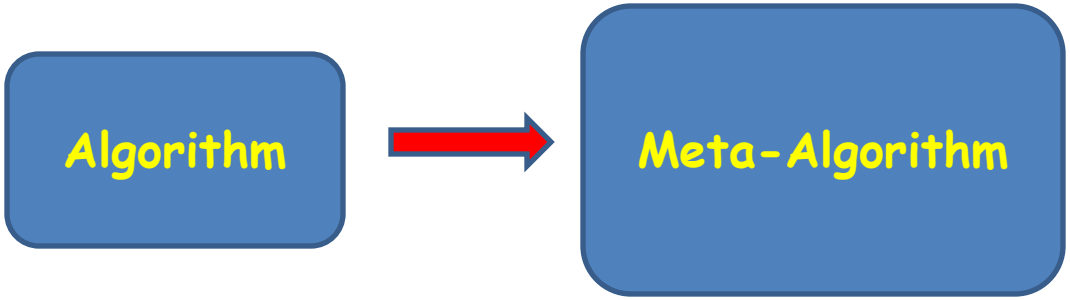
Simon Fraser University

Understanding Efficiency



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)

Self-Reference …

- Computability: impossibility results (via diagonalization)
- Complexity: efficient (non-trivial) meta-algorithms (for SAT , PIT) lead to circuit lower bounds

The rest of the talk …

part 1:

Meta-algorithms

Circuit lower bounds

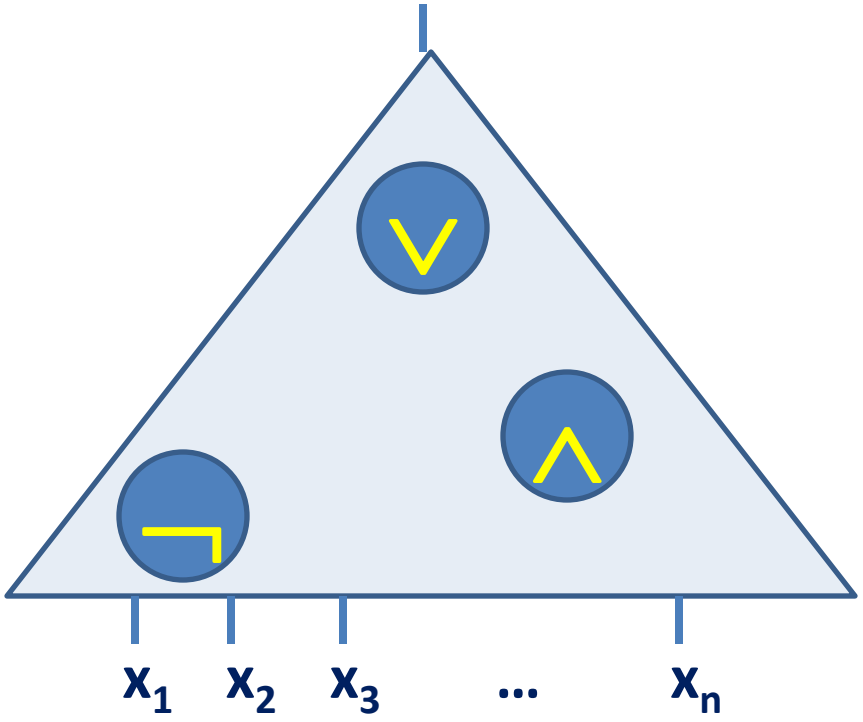
part 2:

Circuit lower bounds

Meta-algorithms

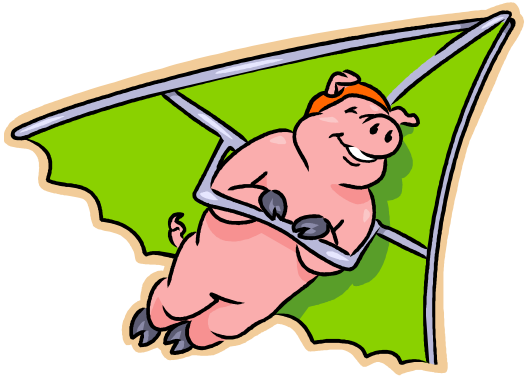
Circuit Lower Bounds from SAT-Algorithms

Circuit - SAT



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

Decide: Is C satisfiable ?



If pigs can fly …

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

If SAT in P, then

- $\exists$ L in $E = DTIME(2^{O(n)})$ of circuit size $> 2^n/n$
- $\forall$ constant c $\exists$ L_c in P of circuit size $> n^c$

[Kannan]

Kannan's proof

- **Brute-force (diagonalization) :** Define a hard language in a large complexity class C (using alternation)

- **Speed-up :** Collapse C to a smaller complexity class using $SAT \in P$ (removing alternation)

$P=NP \Rightarrow E$ requires size $2^n/n$

- E^{Σ_3} requires circuit size $2^n/n$:
  For each n, use $\exists,\forall$ to define the lex. first Boolean function $f_n : \{0,1\}^n \to \{0,1\}$ of circuit size $> 2^n/n$

Use of Self-Reference

- $P = NP \Rightarrow PH = P \Rightarrow E^{PH} = E$

… $\exists$ x $\forall$ y R(…,x,y) becomes … $\exists$ x R'(…,x)

co-NP question

contains SAT-algorithm

Karp-Lipton's proof:
$P = NP \Rightarrow EXP$ not in PolySize

Local Checkability of Computation

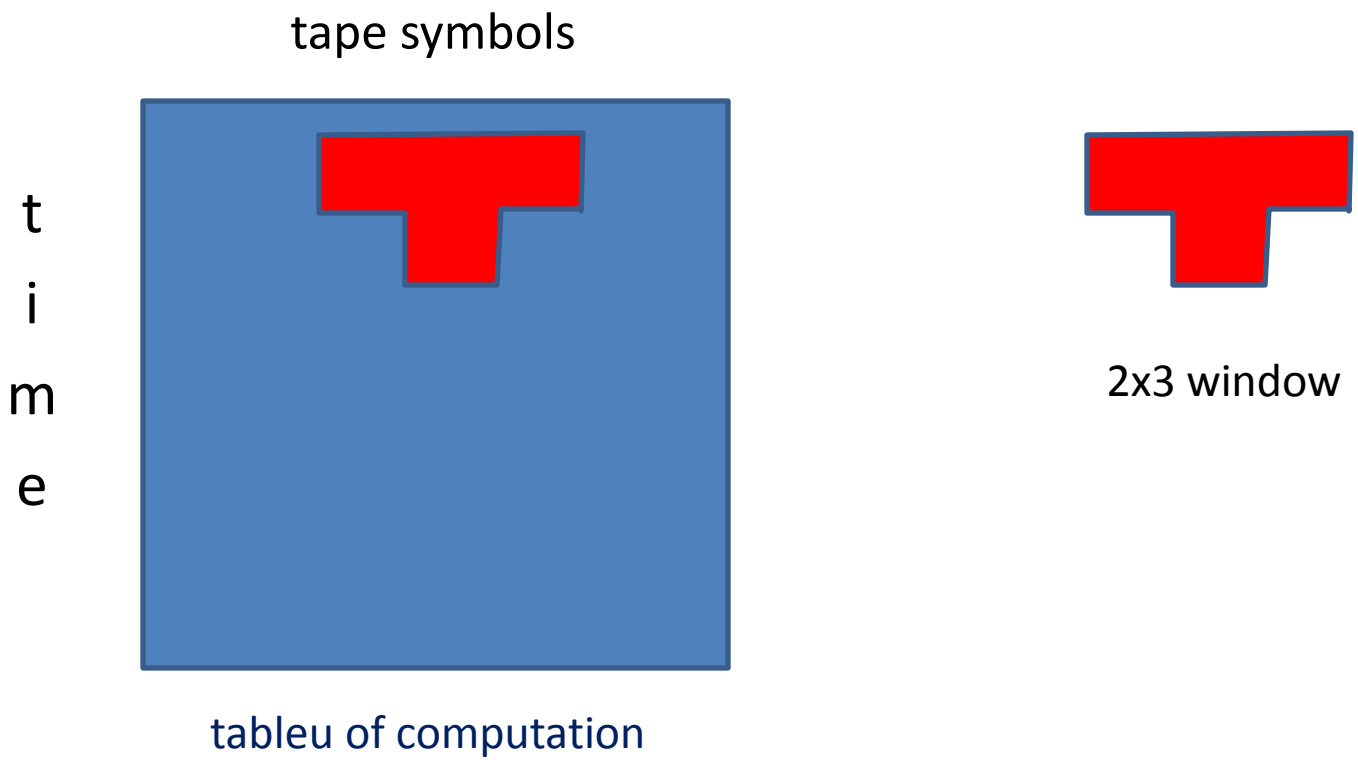
- EXP in PolySize $\Rightarrow$ $EXP = \Sigma_2$.
- $P = NP \Rightarrow \Sigma_2 = P$.

Self-Reference

- $EXP = P$. X

Diagonalization

Local Checkability of Computation



tableu of computation

TM's computation is correct iff all 2x3 "windows" are legal.

(Used by Cook & Levin to show 3-SAT is NP-complete)

More on local checkability in [Arora, Impagliazzo, Vazirani]

EXP in PolySize $\Rightarrow$ EXP $=\Sigma_2$

- Let $L \in$ EXP be decided by TM M
- $\text{Tableau}_M(x, i, j) \in$ EXP
- $\exists$ polysize circuit C computing Tableau_M
- $x \in L \Leftrightarrow \exists\, C\ \forall$ tableau windows defined by $C(x, . , . , .)$ are legal
  & the tableau is accepting

Karp-Lipton's proof:
$P = NP \Rightarrow EXP$ not in PolySize

Local Checkability of Computation

- EXP in PolySize $\Rightarrow$ $EXP = \Sigma_2$.
- $P = NP \Rightarrow \Sigma_2 = P$.

Self-Reference

- $EXP = P$. X

Diagonalization

Generalizing Karp-Lipton: co-NP in NSUBEXP $\Rightarrow$ NEXP not in PolySize

- NEXP in PolySize $\Rightarrow$ NEXP = Σ_2 [IKW]
- co-NP in NSUBEXP $\Rightarrow$ Σ_2 in NSUBEXP
- NEXP in NSUBEXP . X

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

[Williams] : Yes ! For ACC^0 – circuits
(constant depth, with mod-gates).

Williams' argument:

ACC^0-SAT algorithm $\Rightarrow$
NEXP not in ACC^0

Pushing Karp-Lipton further …

ACC^0-UNSAT in NTime($2^n / n^c$) is enough !

- NEXP in ACC^0 $\Rightarrow$ NEXP = Σ_2 [IKW]

ACC^0-UNSAT in NSUBEXP $\Rightarrow$ Σ_2 in NSUBEXP

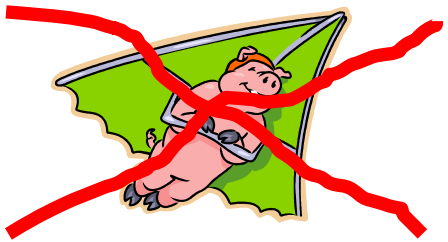
- NEXP in NSUBEXP . X

Relies on efficient oblivious simulation of TMs, & other ideas …

$\mathcal{C}$-Circuit Satisfiability

- 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.

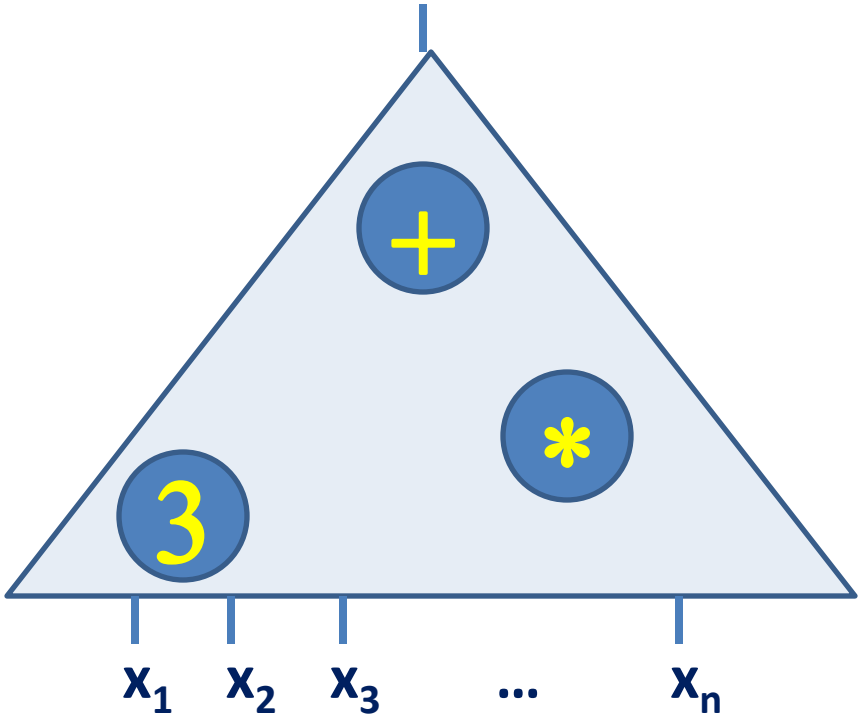
- ACC^0 - SAT is solvable in the required time. Hence, NEXP not in ACC^0.



[Williams]

Circuit Lower Bounds from PIT-Algorithms

Polynomial Identity Testing ("Arithmetic Circuit-UNSAT")



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

Decide: Is $C \equiv 0$?

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

Boolean or Arithmetic circuit lower bound

Suppose PIT in P.

Then

NEXP not in PolySize, OR

Permanent not in Arithmetic PolySize (over integers).

[K., Impagliazzo]

Important Polynomial Identities

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

$$\mathrm{Perm}_n(X) = \sum_{\pi} \prod x_{i,\pi(i)}$$

Defining Identities (expansion by minors):

$$\mathrm{Perm}_n(X) \equiv \sum x_{1,j} \, \mathrm{Perm}_{n-1}(X^{1,j})$$

$$\ldots$$

$$\mathrm{Perm}_1(x) \equiv x$$

Arithmetic circuits $C_1, \ldots, C_n$ compute $\mathrm{Perm}_1, \ldots, \mathrm{Perm}_n$ iff $C_1, \ldots, C_n$ satisfy the identities.

If PIT is easy …

Permanent-Check:

Given an arithmetic circuit C,

decide if $C \equiv \mathrm{Perm}_n$.

$$\mathrm{PIT} \in \mathrm{P} \Rightarrow \text{Permanent-Check} \in \mathrm{P}$$

Importance of Permanent

- #P- completeness :

$$P^{\#P} = P^{Perm}$$ [Valiant]

- can simulate alternating $\exists, \forall$:

$$PH \subset P^{\#P} = P^{Perm}$$ [Toda]

Concluding the proof of [KI]

Suppose PIT is in P.

Suppose Perm is in Arithmetic PolySize.

Then can " guess & verify " a polysize circuit for $Perm_n$. So,

$$PH \subset P^{Perm} \subset NP.$$

By padding,

$$E^{PH} \subset NE.$$

By diagonalization, E^{PH} not in PolySize. QED

Pseudo-Random Generator (PRG) for PIT $\Rightarrow$ Circuit Lower Bounds

- PRG fools every arithmetic circuit of size s.

- Define a non-zero polynomial p that is zero on all outputs of PRG.

- Then p requires arithmetic circuit size $> s$. Also, p is computable in E.

[Heintz, Schnorr]

Non-trivial algorithms for

SAT or PIT

yield circuit lower bounds.

Meta-Algorithms from Circuit Lower Bounds

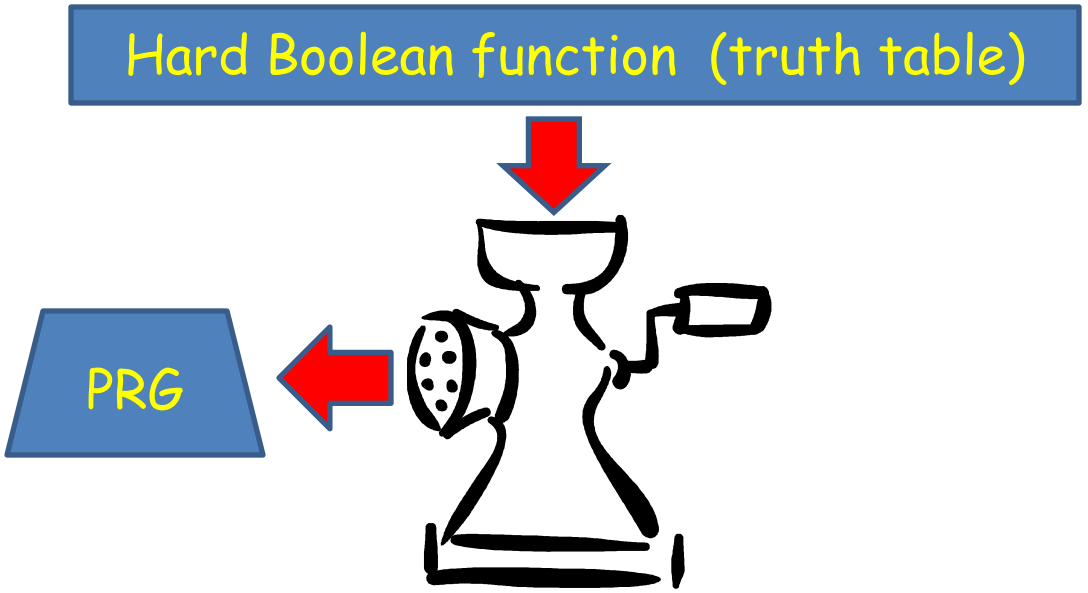
" Black-Box " Use of Circuit Lower Bounds

Derandomization

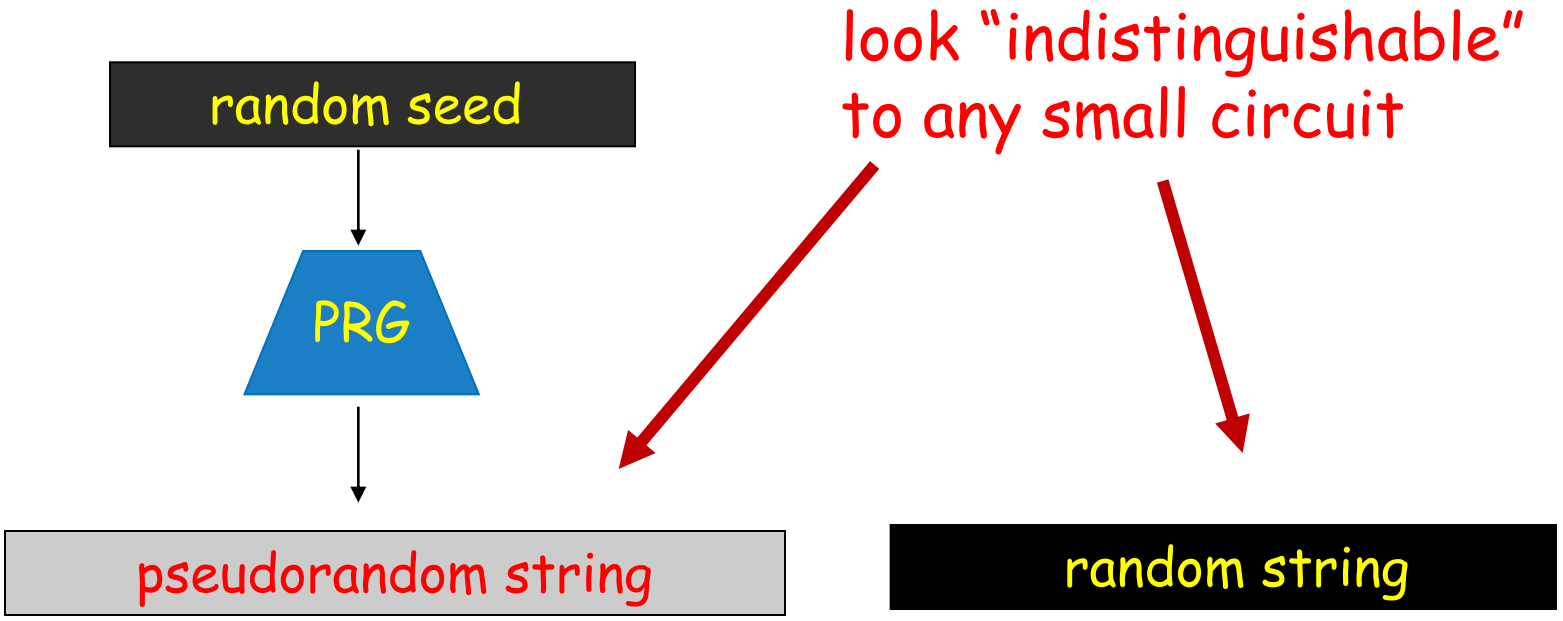
Computational Hardness $\Rightarrow$

Computational Randomness (pseudorandomness)

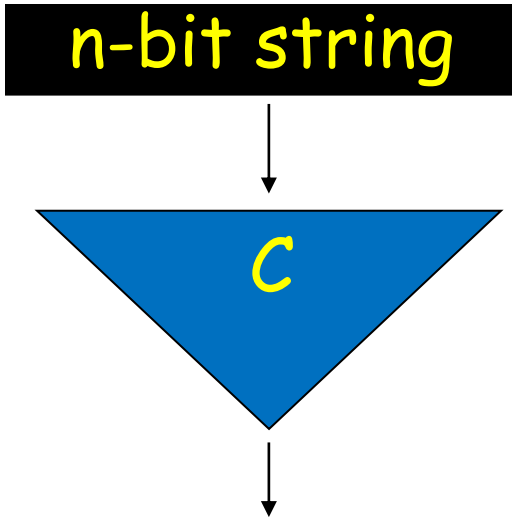
[Blum, Micali, Yao; Nisan & Wigderson, Babai et al., …]

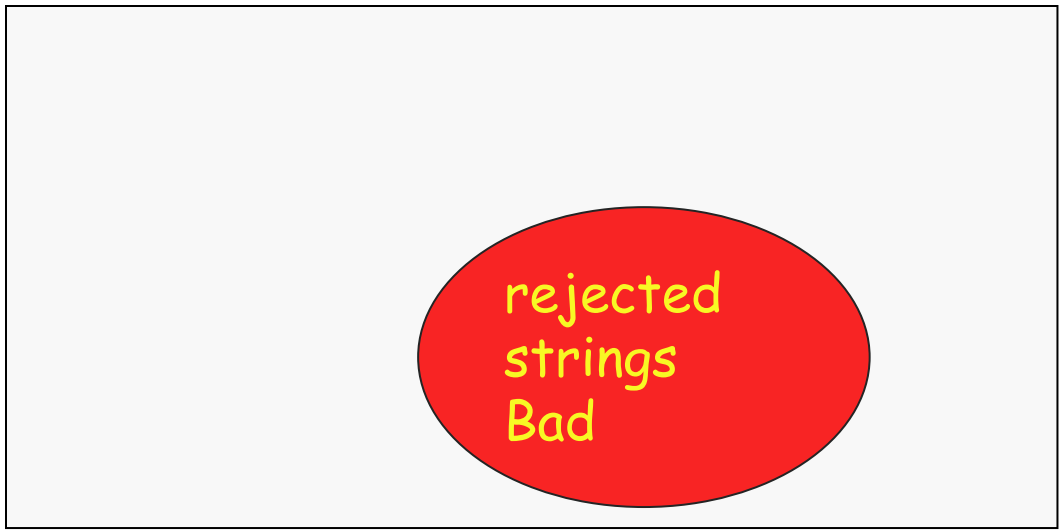


Pseudo-Random Generator (PRG)



Incompressibility Argument

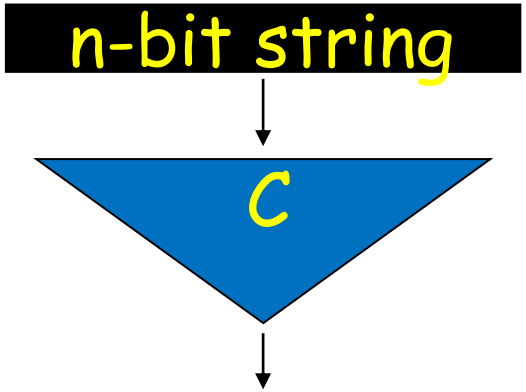


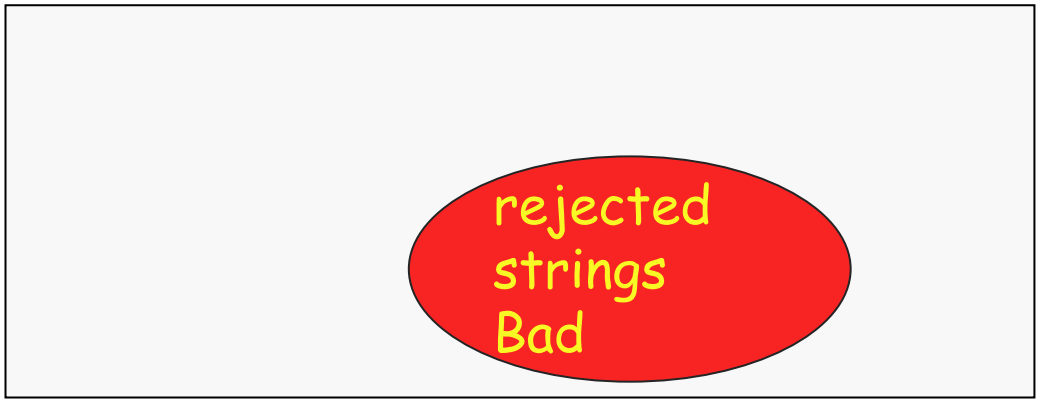


Each $r \in Bad$ has "small" description relative to C :
$\log |B|$ bits specifying the rank of r in B, plus the description of C

Any string incompressible relative to C is accepted by C.

Incompressibility Argument





Each $r \in Bad$ 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

- Boolean circuit hardness :

E requires circuit size $2^{\Omega(n)}$ $\Rightarrow$ PIT $\in$ P

[Impagliazzo & Wigderson]

- Arithmetic circuit hardness :

E has a multilinear polynomial of circuit size $2^{\Omega(n)}$ $\Rightarrow$ PIT $\in$ Quasi-P (Time($n^{\text{polylog } n}$))

[K. & Impagliazzo]

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

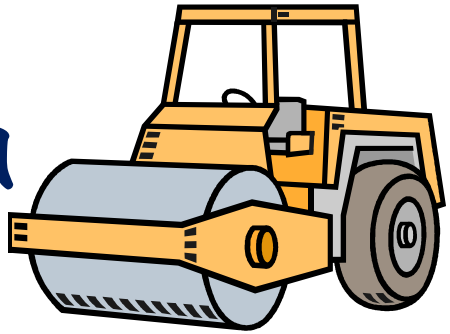
AC^0 Circuits

PARITY $(x_1, \ldots, x_n)$ requires depth d AC^0 circuits of size $> 2^{n^{1/(d-1)}}$

[Furst, Saxe, Sipser; Yao; Hastad]

Main Tool : Switching Lemma

Switching Lemma



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

- 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 Lower Bound Strategy

1. Simplify a given small circuit (by random restriction).

Useful for meta-algorithms

2. Argue that the resulting simple circuit cannot possibly compute the restriction of the original function.

AC^0-functions are sparse

small AC^0-circuit

$\approx$

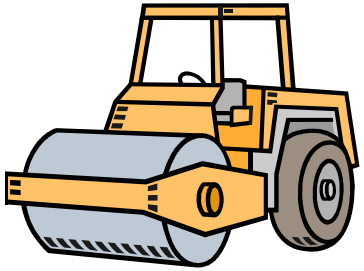
sparse polynomial (Fourier)

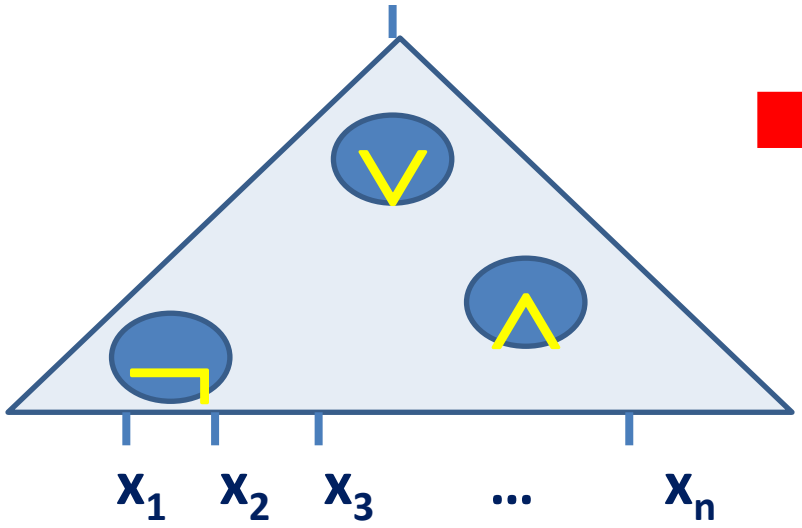
[Linial, Mansour, Nisan]

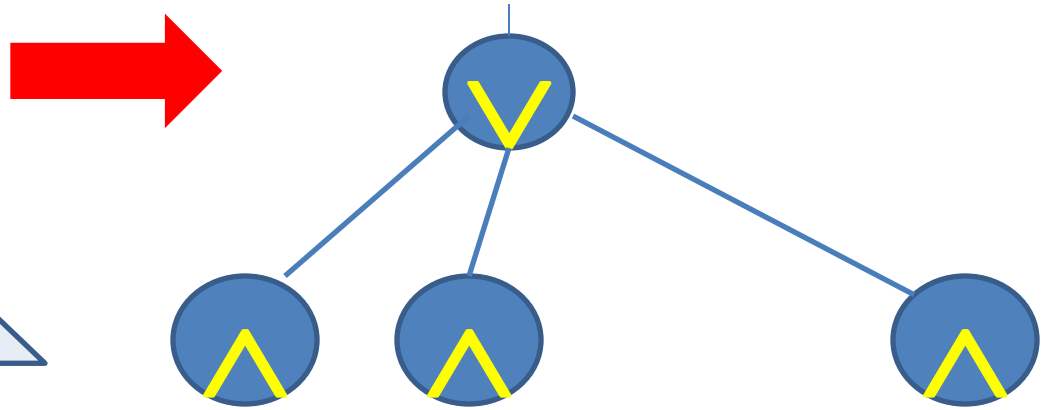
Learning AC^0 -circuits

1. Small AC^0-circuit C is likely a constant function under a random restriction
2. C can't crucially depend on many variables
3. C is well-approximated by a linear combination of simple functions of very few variables (most energy on low frequencies)
4. Good approximation to C can be efficiently learned (by learning few Fourier coefficients)

AC^0 into DNF







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]

Flattening AC^0-circuits

Given an AC^0 circuit $C(x_1,\ldots,x_n)$ of depth d and size $c\,n$.

Using Switching Lemma, find a set of restrictions $\{\rho_i\}$ that partition $\{0,1\}^n$ so that each $C|\rho_i$ is constant.

Whp, # restrictions $< 2^{n(1-\mu)}$, where $\mu > \Omega(1/(\log c + d \log d)^{d-1})$. The running time is poly(n) $|C|$ (# restrictions).

AC^0 – #SAT Algorithm [IMP]

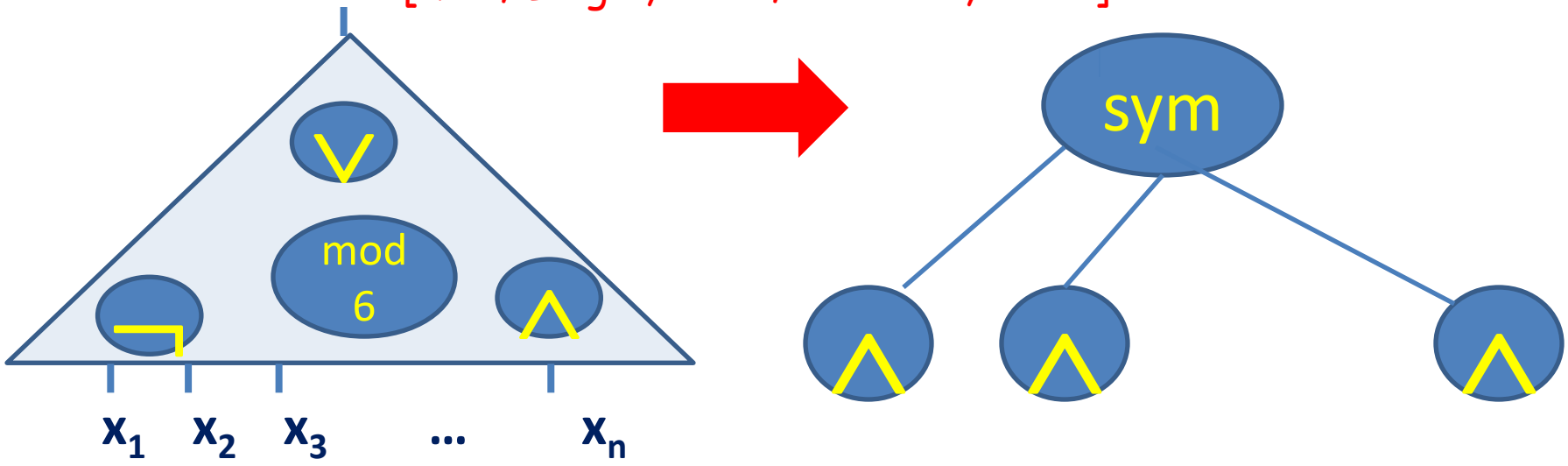
- Convert a given small AC^0 – circuit to an equivalent DNF = OR of not too many ANDs, with disjoint sets of satisfying assignment.

- Each AND on t variables has 2^{n-t} satisfying assignments.

- Add up over all ANDs.

ACC^0 – #SAT Algorithm [Williams]

[Yao; Beigel, Tarui; Allender, Gore]



ACC^0 circuit:
size s, depth d

Rather than Time
2^n poly(S)

SYM-AND : $S \leq s^{(\log s)^m}$

ANDs, with $m \leq 2^{O(d)}$

SYM : # true ANDs $\rightarrow$ { 0,1 }

#SAT for SYM-AND : Time $\cdot$ (2^n + poly(S)) poly(n)

via Dynamic Programming (z-transform)

More "LB → Algorithms"

- [Santhanam] : Formula-SAT in time $|\phi| \, 2^{n\,(1-\,1/\mathrm{poly}(c))}$, for formulas ϕ of size $c\,n$.

(random restrictions)

- [Braverman] : k-wise independence fools AC^0 – circuits.

(Linial,Mansour,Nisan; Razborov–Smolensky)

- MORE TO COME ? Yes, I think so !

Summary

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

  strong circuit lower bounds

- Can use this connection to 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 ?]