2013.3.18

Exploring the
Limits of
Computation

ELC Complexity Theory
Intro. Seminar Series

Algorithmic Approaches to Lower Bounds of Computational Complexity

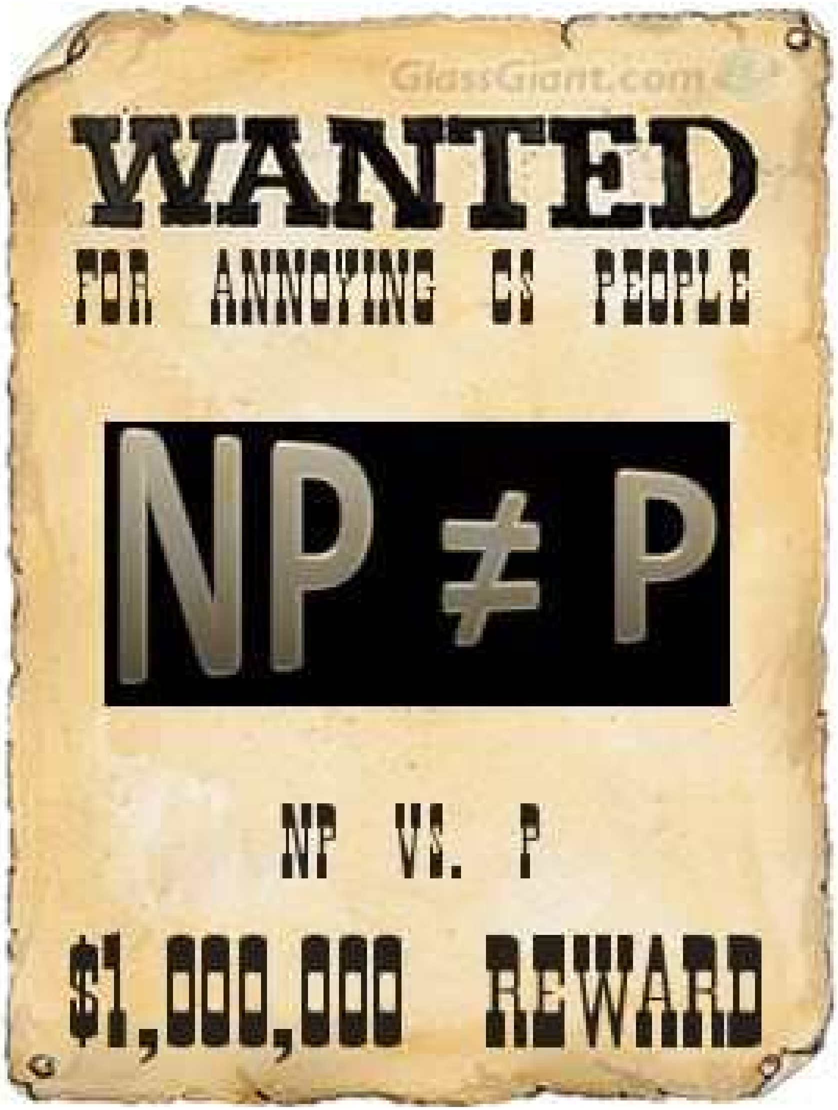
Akinori Kawachi
Dept. of Math and Comp. Sci.
Tokyo Institute of Technology

ELC is supported by Grant-in-Aid for Scientific Research on Innovative Areas, MEXT, Japan

ELC Tokyo Complexity Workshop
(Mar. 14-17, Shinagawa Prince Hotel)

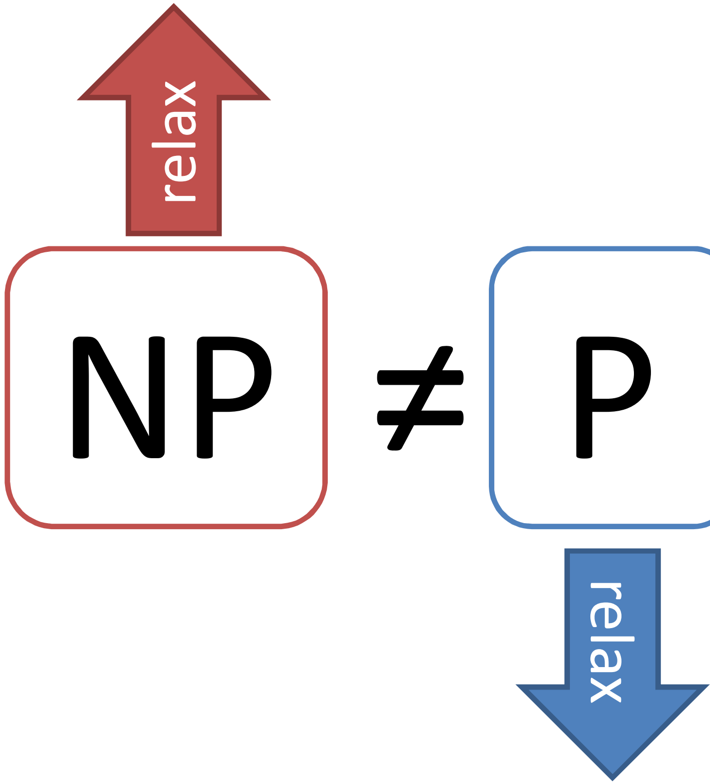
#participants > 150!!
Thank you for coming!

Today's Topic



Two Approaches

High-level approach: Discuss “Higher class vs. P”



Low-level approach: Discuss “NP vs. Lower class”

Circuit Complexity

Major Strategy in Two Approaches

Proving circuit complexity for classes:

No poly-size circuit can compute some NP problem

$$NP \neq P$$

$(NP \not\subset P/poly \Rightarrow NP \neq P)$

computable by
poly-size circuits
≈ class P

Circuits

Gate set = {$\wedge$, $\vee$, $\neg$}

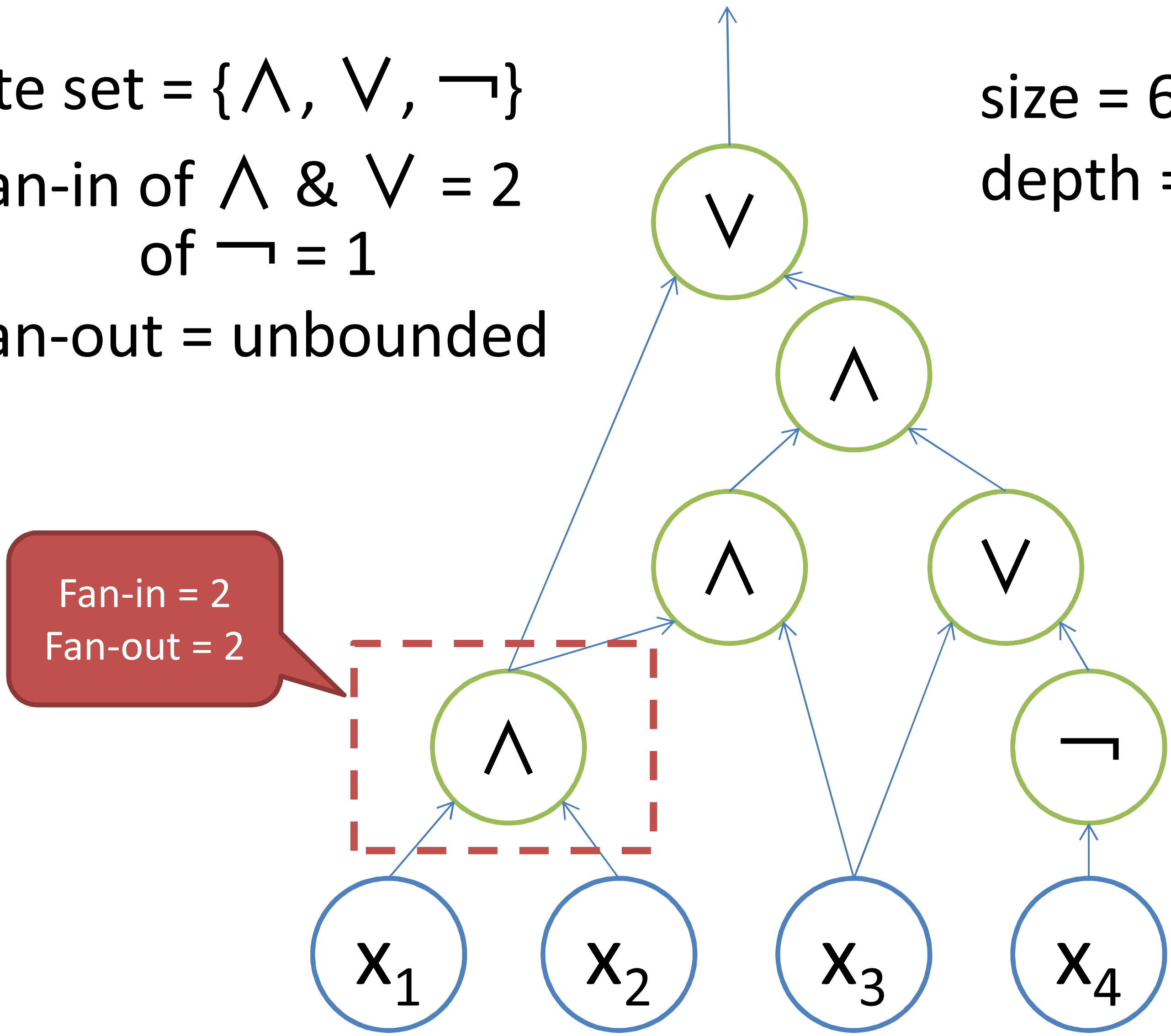
Fan-in of $\wedge$ & $\vee$ = 2

of $\neg$ = 1

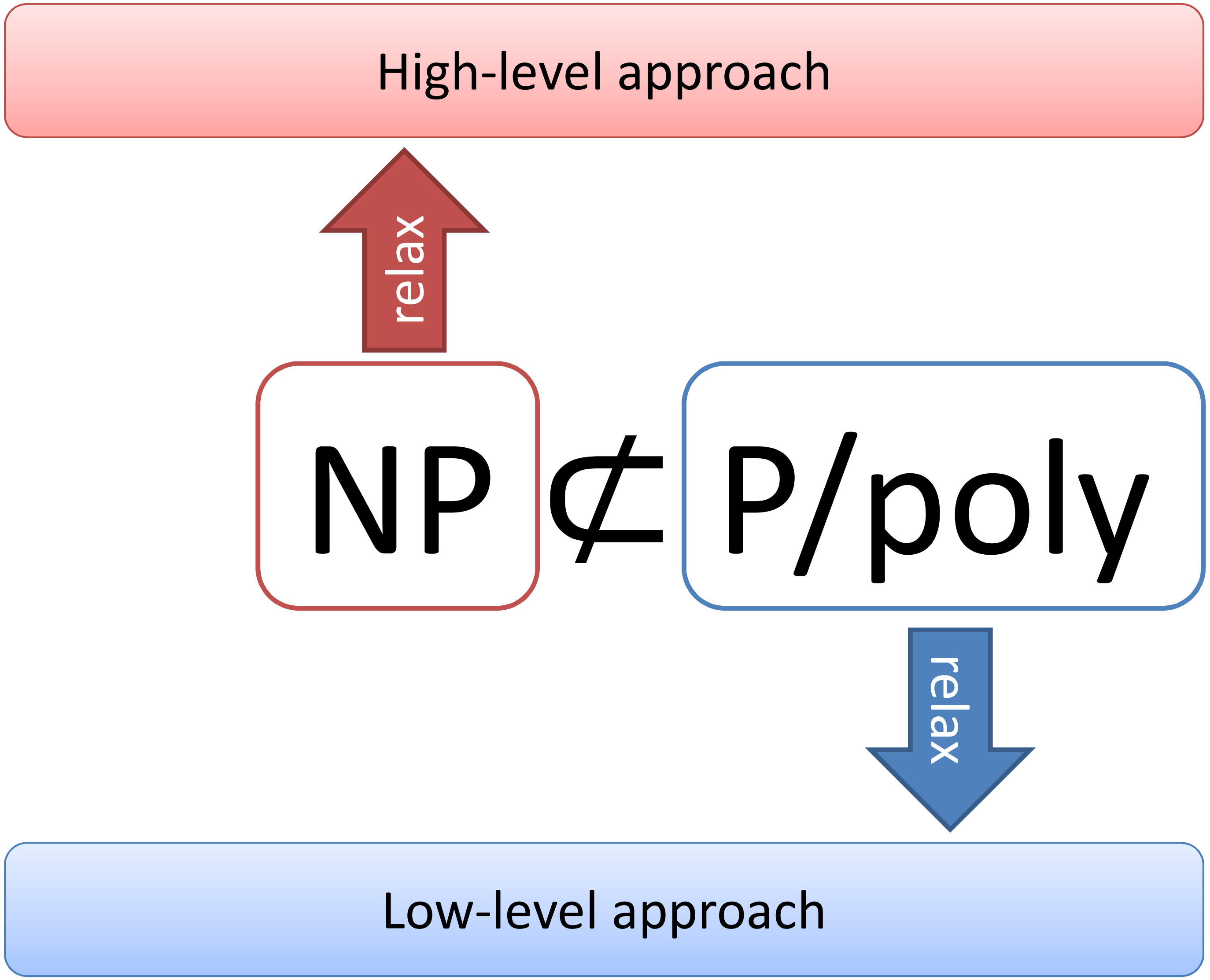
Fan-out = unbounded

size = 6

depth = 4



Why not close the gap?



From High Level

NP to higher complexity classes!

relax

$$\boxed{\mathrm{NP}} \not\subset \boxed{\mathrm{P/poly}}$$

Key Fact:
Almost all functions are hard!

Fact

$\exists$ $f:\{0,1\}^n \rightarrow \{0,1\}$ s.t. no $2^{0.1n}$-size circuit can compute f.

Furthermore,

$\Pr_f[$ No $2^{0.1n}$-size circuit can compute f $] \geq 1 - o(1)$.

($f:\{0,1\}^n \rightarrow \{0,1\}$ is uniformly at random.)

Proof is easy: $\#f = 2^{2^n} >> \#(2^{0.1n}\text{-size circuits}) = 2^{O(2^{0.1n})}$

Hard functions exist!
How find them near NP??

Class NP

Class NP

$L \in NP$

$\overset{\text{Def}}{\Longleftrightarrow}$

$x \in L \Longrightarrow \exists w\ V(x,w) = 1$

$x \notin L \Longrightarrow \forall w\ V(x,w) = 0$

$|w| = poly(|x|)$

V: poly-time comp.

e.g., $SAT \in NP$

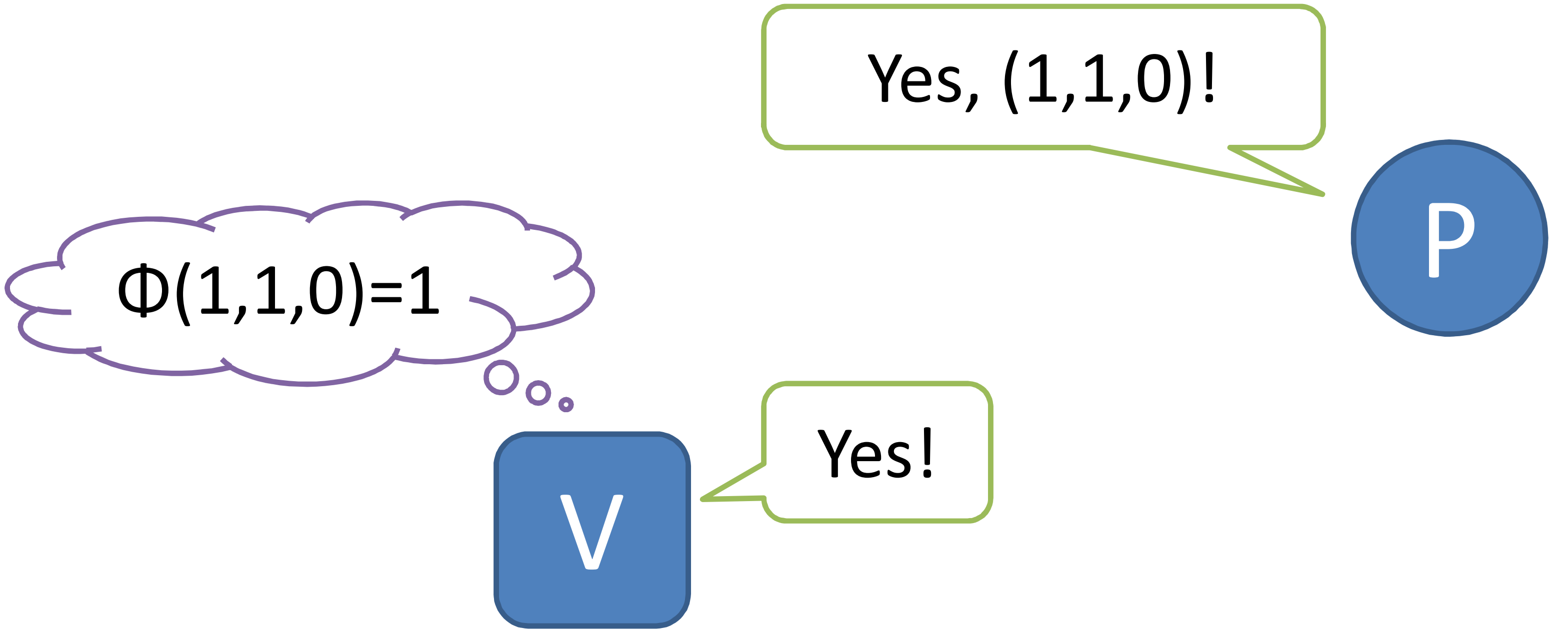
$\Phi(x_1,\ldots,x_n) \in SAT \Longleftrightarrow \exists a_1,\ldots,a_n\ \Phi(a_1,\ldots,a_n)=1$

$x_1 \wedge x_2 \wedge x_3 \in SAT$

$x_1 \wedge \neg x_1 \wedge x_3 \notin SAT$

Class NP

Input: $\Phi(x_1,x_2,x_3) = x_1 \land x_2 \land \neg x_3$



Generalization of NP

Class $\Sigma_2 P$

$L \in \Sigma_2 P$

Def

$x \in L \Rightarrow \exists w_1 \forall w_2 \; V(x, w_1, w_2) = 1$

$x \notin L \Rightarrow \forall w_1 \exists w_2 \; V(x, w_1, w_2) = 0$

$|w_1|, |w_2| = \mathrm{poly}(|x|)$

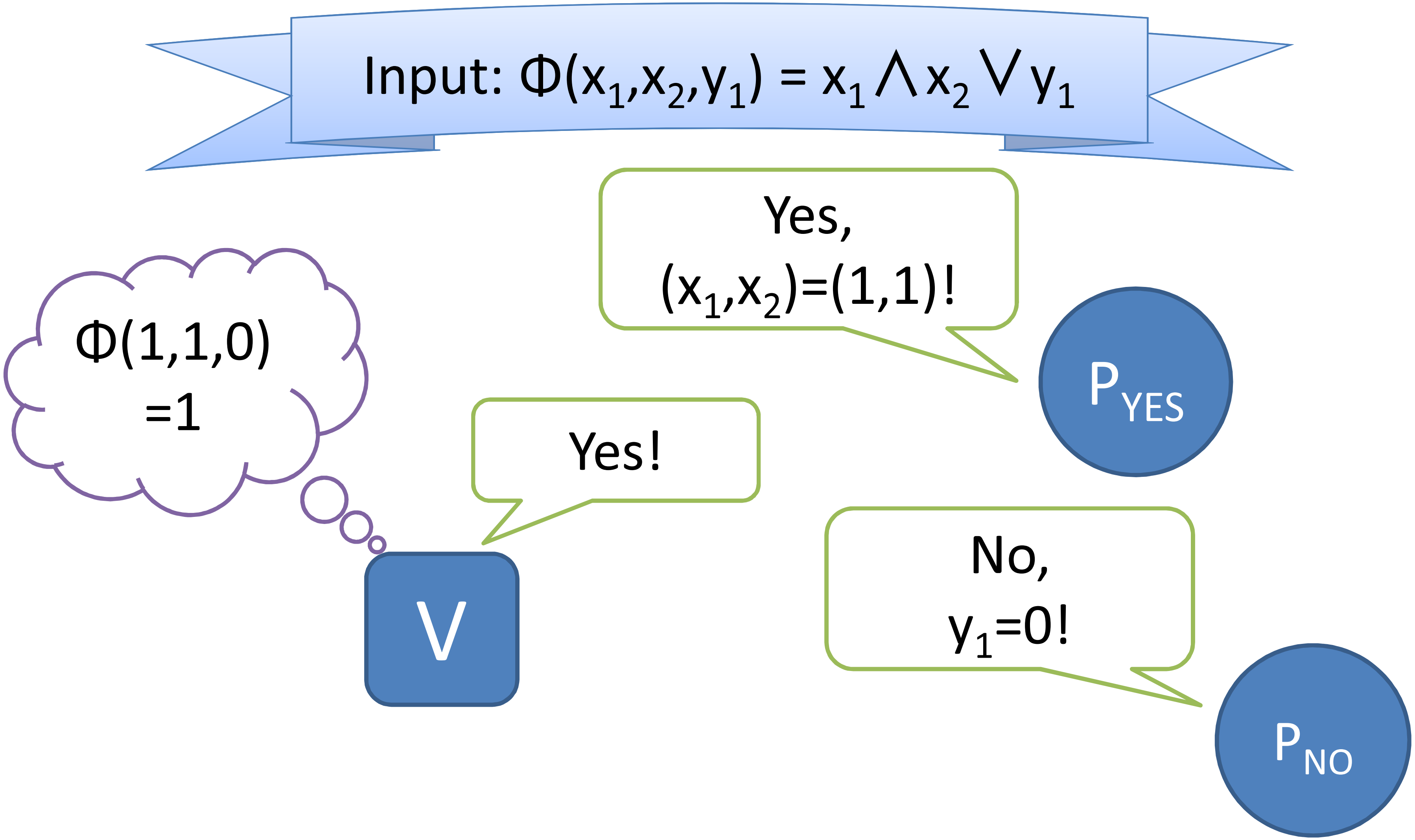
V: poly-time comp.

e.g., $\Sigma_2 SAT \in \Sigma_2 P$

$\Phi(x_1, \ldots, x_n, y_1, \ldots, y_m) \in \Sigma_2 SAT$

$\Leftrightarrow \exists a_1, \ldots, a_n, \forall b_1, \ldots, b_n \; \Phi(a_1, \ldots, a_n, b_1, \ldots, b_m) = 1$

Class $\Sigma_2 P$



Generalization of NP

Class $\Sigma_k P$

$L \in \Sigma_k P$

Def $\Leftrightarrow$ $x \in L \Rightarrow$

$$\exists w_1 \forall w_2 \ldots \exists w_k \; V(x, w_1, \ldots, w_k) = 1$$

$x \notin L \Rightarrow$

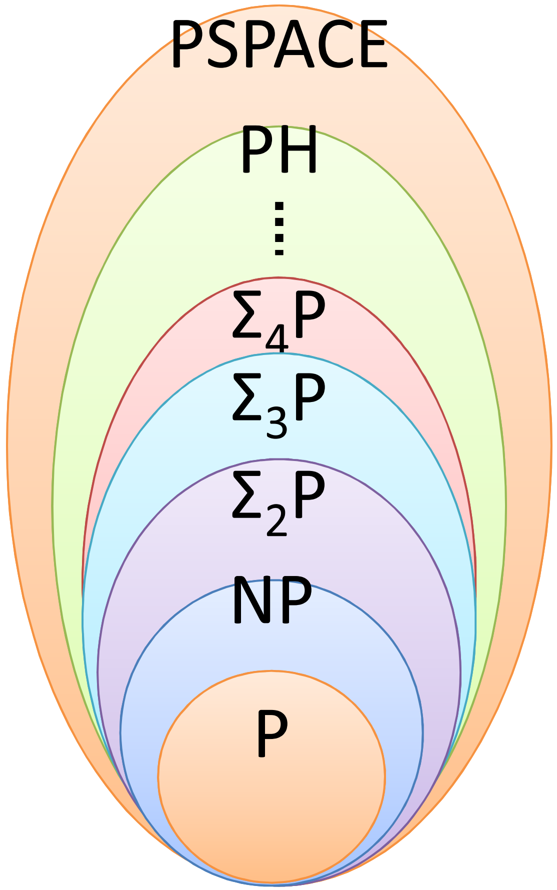
$$\forall w_1 \exists w_2 \ldots \forall w_k \; V(x, w_1, \ldots, w_k) = 0$$

$|w_1|, \ldots, |w_k| = \mathrm{poly}(|x|)$

V: poly-time comp.

Polynomial-time Hierarchy

$$PH = \bigcup_{k=1}^{\infty} \Sigma_k P$$



PH has a hard problem!

Theorem [Kannan, '82]

No n^{100}-size circuit can compute some $\Sigma_4 P$ problem.

Problem: HARD

Given: n-bit string x

Decide: $f_{HARD}(x) = 1$?

f_{HARD} is a Boolea … nction which **no n^{100}-size circuit can compute**.

$\forall C \in \{n^{100}\text{-size circuit}\}$

$\exists y \in \{0,1\}^n$

s.t. $C(y) \neq f_{HARD}(y)$

Definition of f_{HARD} (Sketch)

1. Computability

   f_{HARD} is computable by n^{200}-size circuits

2. Hardness

   f_{HARD} is not computable by n^{100}-size circuits

3. Uniqueness

   f_{HARD} is lex 1st func. satisfying above two

Definition of f_{HARD}

$f_{HARD}(x) = 1$

Def [1.] $\exists$ ① circuit C ($size(C)<n^{200}$) s.t. $C(x)=1$ and

[2.] $\forall$ ② circuit C' ($size(C')<n^{100}$)

$\exists$ ③ $z \in \{0,1\}^n$ s.t. $C(z) \neq C'(z)$ and

[3.] $\forall$ ② circuit C'' ($C''<C$ in lex order)

$\exists$ ③ circuit C''' ($size(C''')<n^{100}$)

$\forall$ ④ $z \in \{0,1\}^n$ $C''(z)=C'''(z)$

Improvement to lower class

Theorem [Kannan, '82]

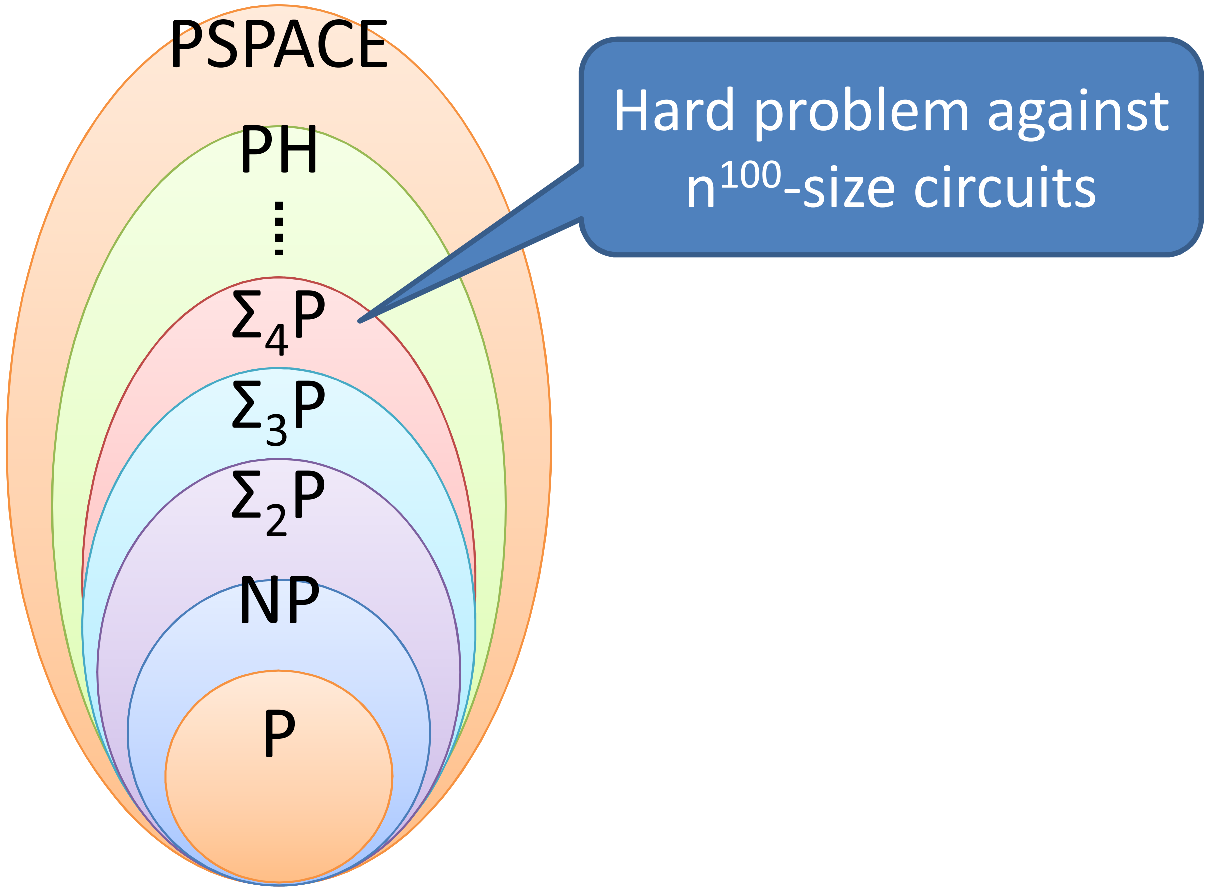
No n^{100}-size circuit can compute some $\Sigma_4 P$ problem.

Improvement

Theorem [Kannan, '82]

No n^{100}-size circuit can compute some $\Sigma_2 P$ problem.

Circuit lower bound in $\Sigma_4 P \rightarrow \Sigma_2 P$



Proof Idea: Win-Win Strategy

- If n^{300}-size circuit can compute SAT

- If n^{300}-size circuit cannot compute SAT

Proof Idea: Win-Win Strategy

- If n^{300}-size circuit can compute SAT

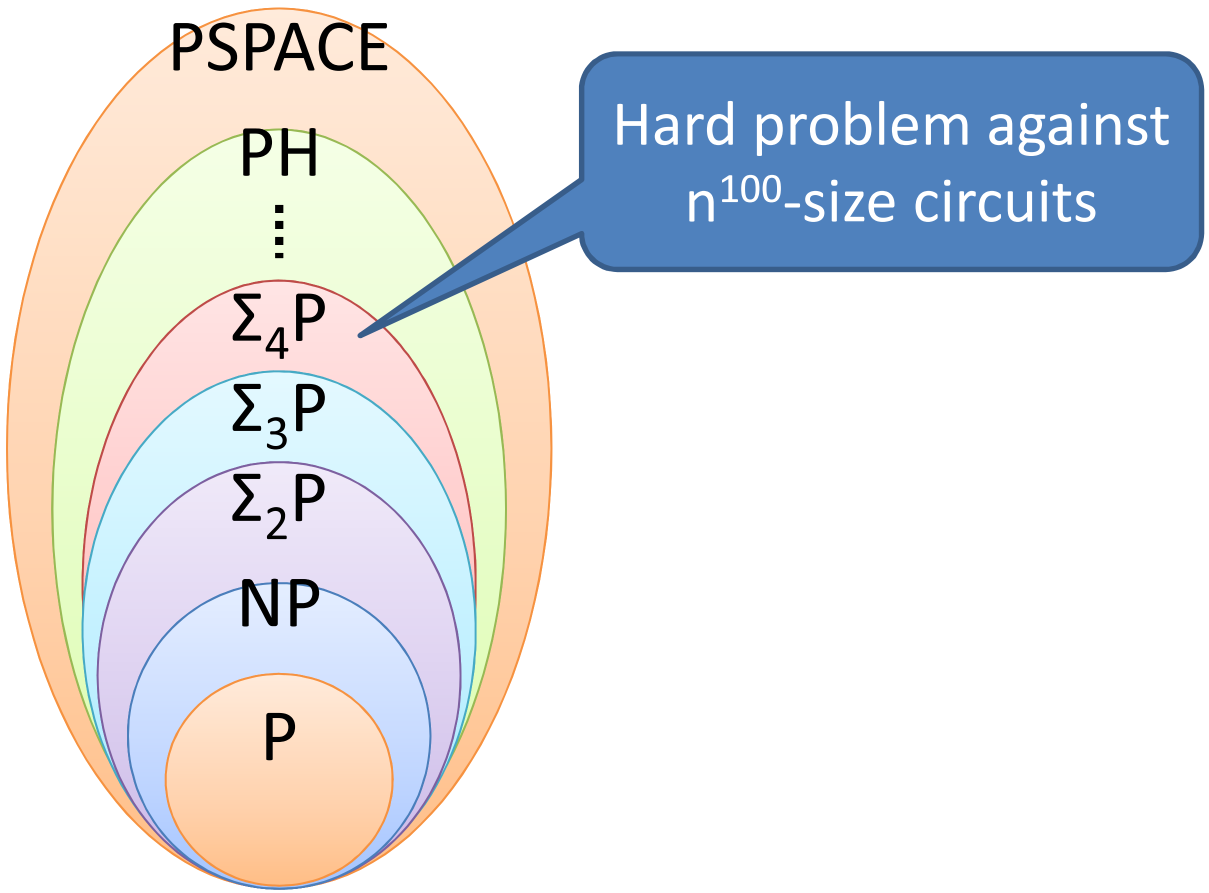
Key Tool: Collapse of PH

Theorem [Karp & Lipton, ‘82]

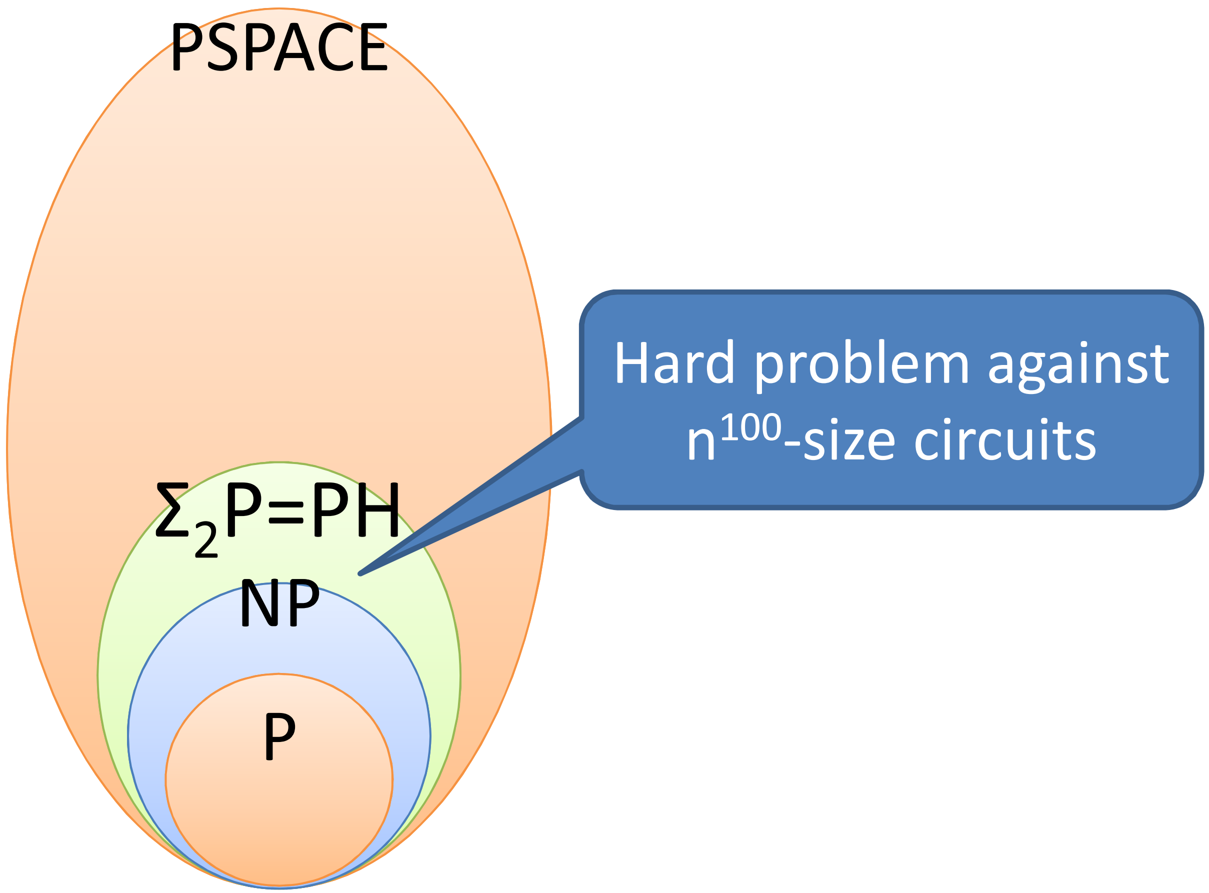
n^{300}-size circuit can compute SAT ➔ $PH = \Sigma_2 P$

(in fact, $PH = \Sigma_2 P \cap \Pi_2 P$)

If n^{300}-size circuit can compute SAT



If n^{300}-size circuit can compute SAT



Theorem [Karp & Lipton, '82]

n^{300}-size circuit C can compute SAT ➔ $PH = \Sigma_2 P$

(in fact, $PH = \Sigma_2 P \cap \Pi_2 P$)

Idea Circuit C for SAT can eliminate quantifiers!

If $L \in \Sigma_k P$

SAT!

$x \in L \Leftrightarrow \exists w \ \ \forall w \ \ \exists w \ \ V(x, w \ldots, w_{k-1}, w_k) = 1$

$\ldots, w_{k-1}) = 1$

Need to find the circuit C

to compute V_C' by **TM**!

$\Leftrightarrow V_C'(x) = 1$

$\Sigma_2 P$ is enough to find C!

Proof (circuit lower bound in $\Sigma_2 P$)

- If n^{300}-size circuit can compute SAT
  - $PH = \Sigma_4 P = \Sigma_2 P$ [Karp & Lipton '82]
  - $\Sigma_4 P$ has hard problem against $SIZE(n^{100})$
  - Thus, $\Sigma_2 P$ has, too.

- If n^{300}-size circuit cannot compute SAT
  - $SAT \in NP$
  - Thus, NP has hard problem against $SIZE(n^{300})$

$$\boxed{\Sigma_2 P \not\subset SIZE(n^{100}) \text{ or } NP \not\subset SIZE(n^{300})}$$

■

Summary: Kannan's argument

- Directly defines hard problem in $\Sigma_4 P$
  - By power of $\Sigma_4 P$
- Improves by Karp-Lipton collapse
  - SAT $\in$ SIZE(n^{300}) $\Rightarrow$ $\Sigma_4 P = \Sigma_2 P \not\subset$ SIZE(n^{100})
  - SAT $\notin$ SIZE(n^{300}) $\Rightarrow$ SAT $\in$ NP $\not\subset$ SIZE(n^{300})

- Improves further by deeper collapse
  - Requires algorithm finding the circuit C for SAT
    (in Karp-Lipton, $\Sigma_2 P$-algorithm works)

Further Improvements for Fixed Polynomial Lower Bounds

Theorem [Kannan, '82]

No n… …ompute some Σ^2P problem.

… Π^2P problem)

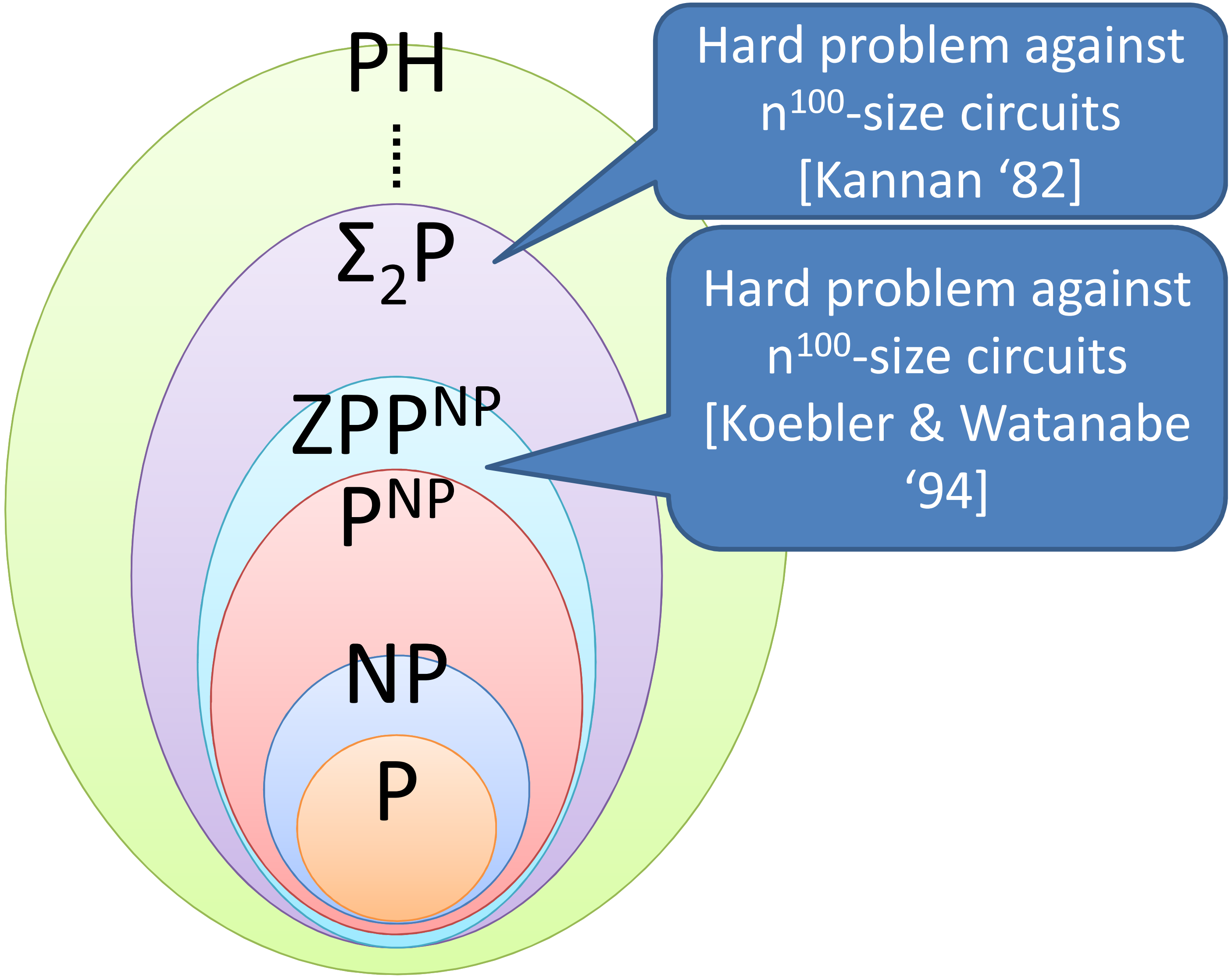
Our Leader!

Theorem [Koebler & Watanabe, '94]

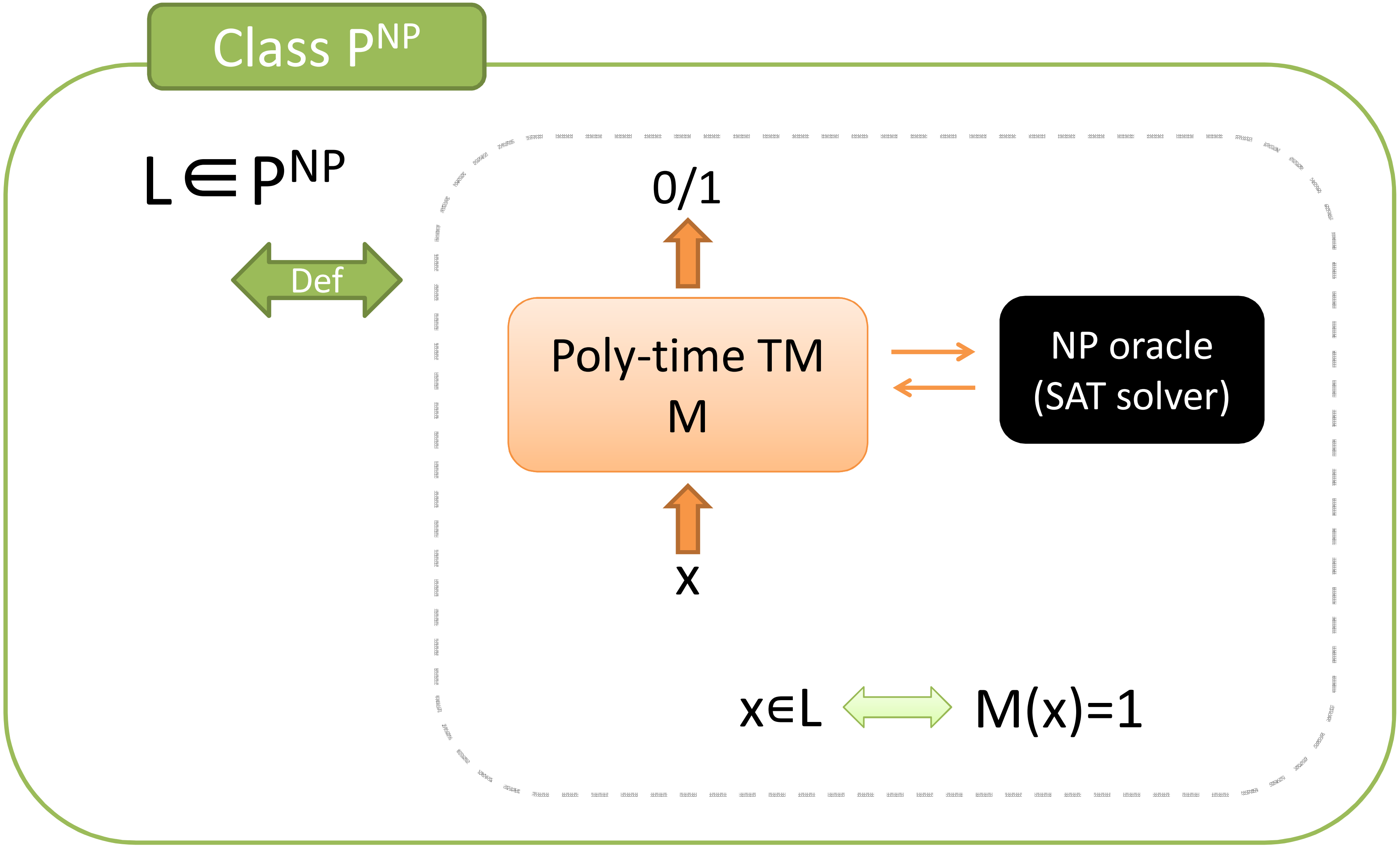
No n^{100}-size circuit can compute some ZPP^{NP} problem.

Zero-error prob. poly-time with NP oracle

Circuit lower bound in $\Sigma_2 P$ ➔ ZPP^{NP}

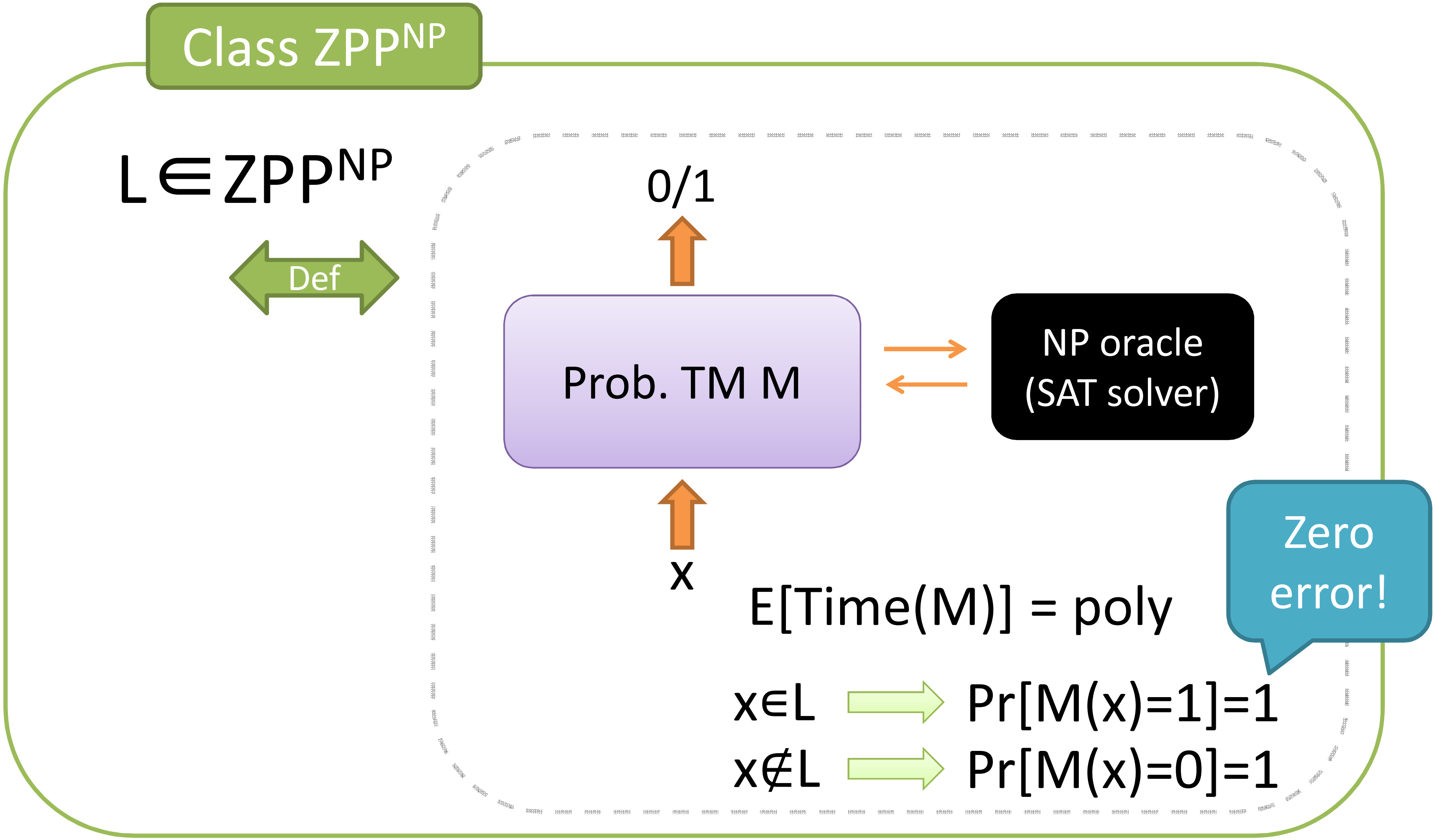


Class P^{NP}



Class P^{NP}
$L \in P^{NP}$
Def
0/1
Poly-time TM
M
NP oracle
(SAT solver)
x
$x \in L$
$M(x)=1$

Class ZPP^{NP}



Class ZPP^NP
L∈ZPP^NP
Def
0/1
Prob. TM M
NP oracle
(SAT solver)
x
E[Time(M)] = poly
Zero
error!
x∈L
Pr[M(x)=1]=1
x∉L
Pr[M(x)=0]=1

Koebler & Watanabe's argument

- If n^{300}-size circuit can compute SAT
  - PH = ZPP^{NP} (cf. Karp-Lipton: PH = Σ_2P)
    - Finding the circuit C computing SAT in ZPP^{NP}
  - Thus, $ZPP^{NP} \not\subset SIZE(n^{100})$

- If n^{300}-size circuit cannot compute SAT
  - SAT∈NP
  - Thus, NP $\not\subset SIZE(n^{300})$

$$\boxed{ZPP^{NP} \not\subset SIZE(n^{100}) \text{ or } NP \not\subset SIZE(n^{300})}$$

■

Koebler & Watanabe's argument ≈ Circuit Learning Algorithm

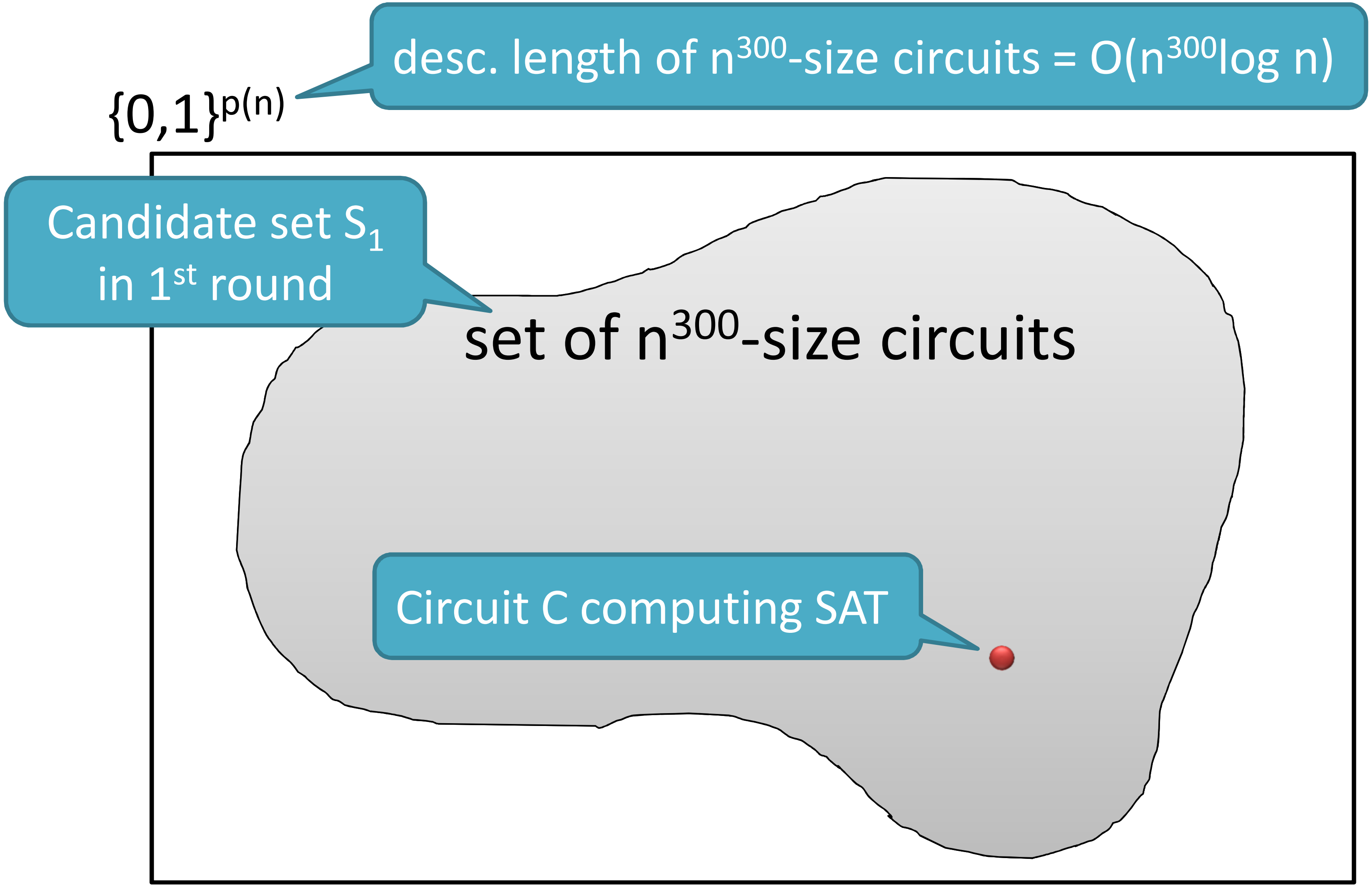
[Bshouty, Cleve, Gavalda, Kannan & Tamon '96]

- Assumption: $\exists\, n^{300}$-size circuit computing SAT
  - How find it by ZPP^{NP}-algorithm?

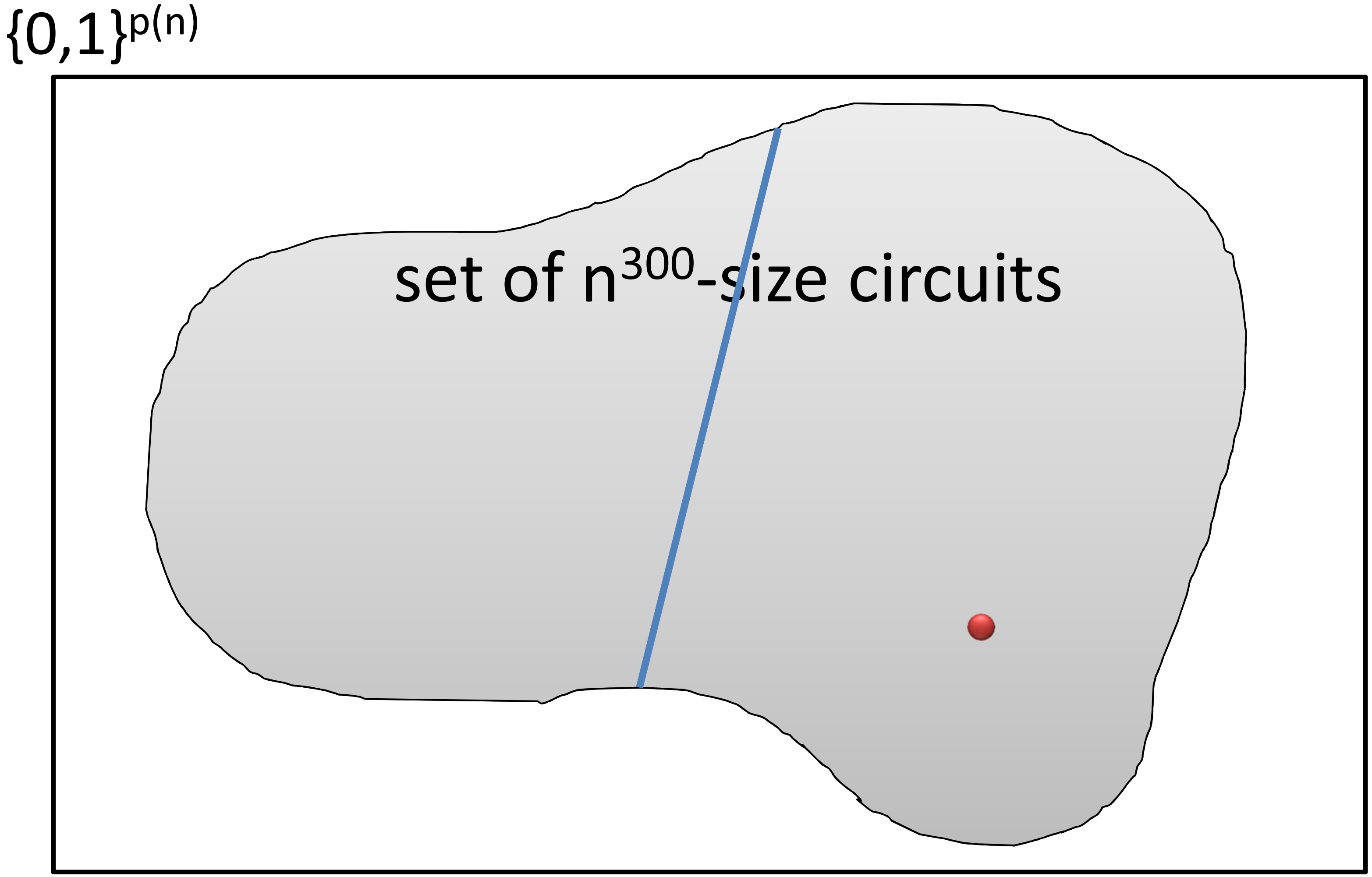
Idea

"Learn" it with power of NP oracle
by binary-search in set of n^{300}-size circuits

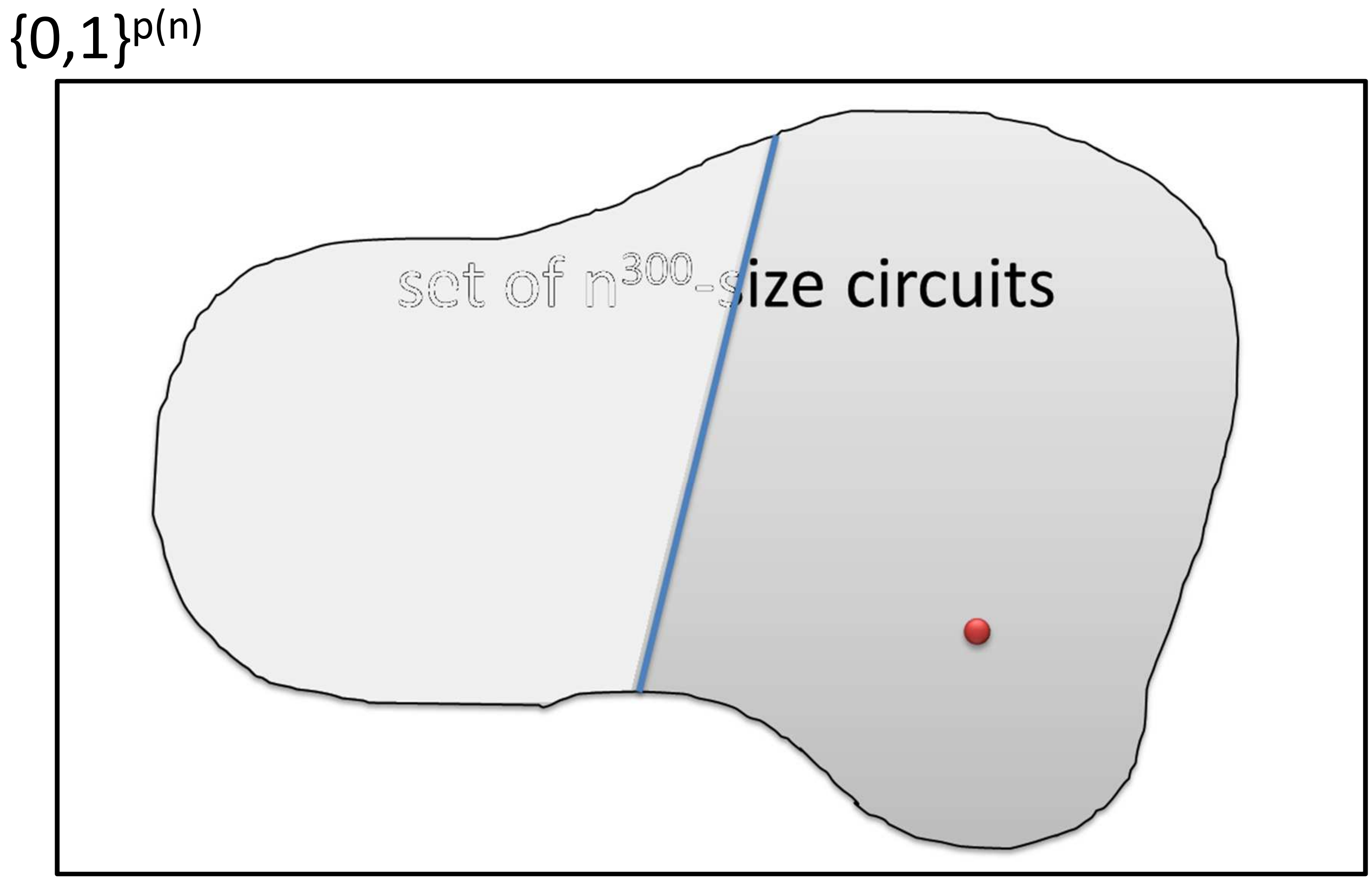
Search in set of circuits



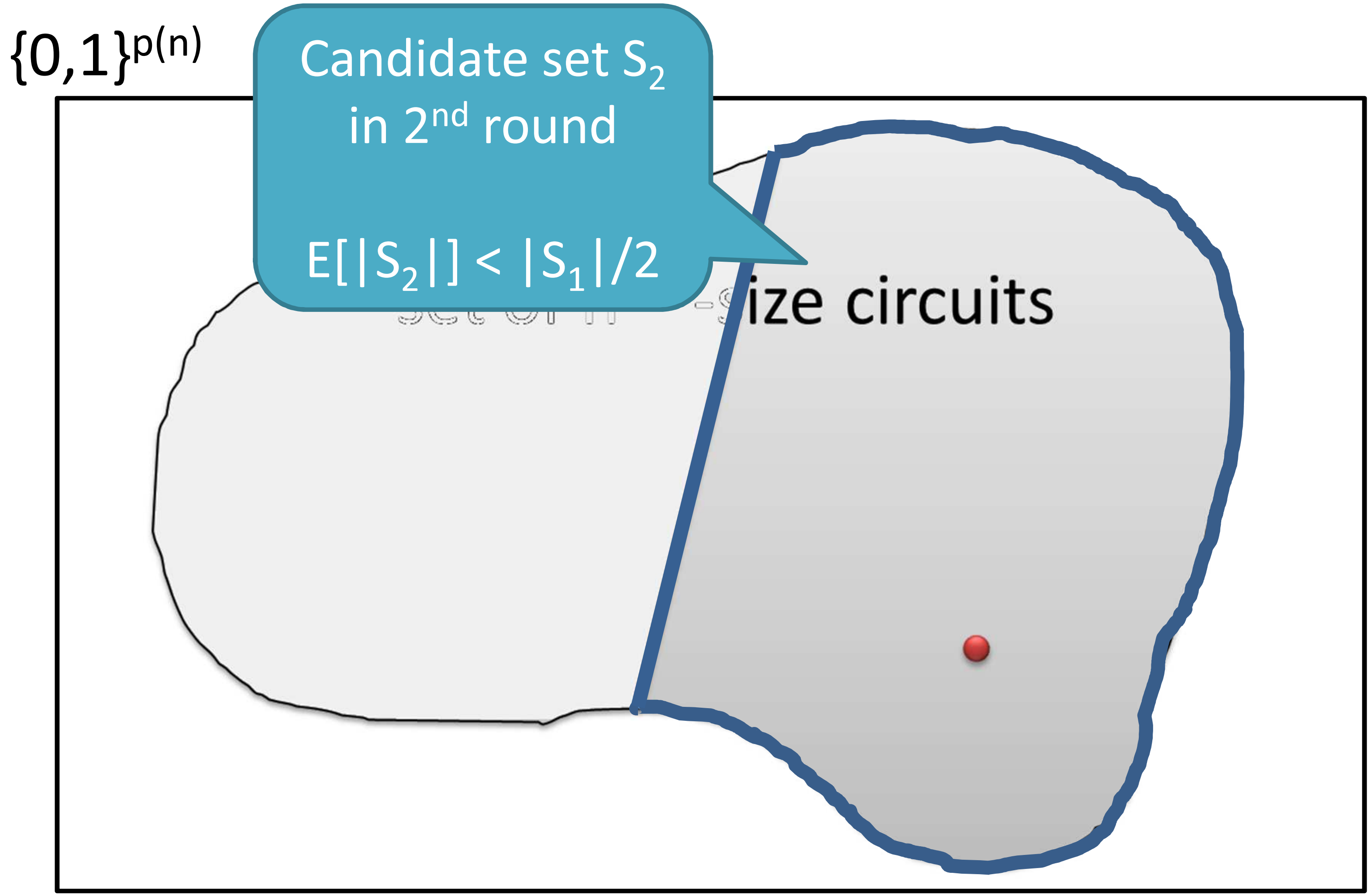
Search in set of circuits



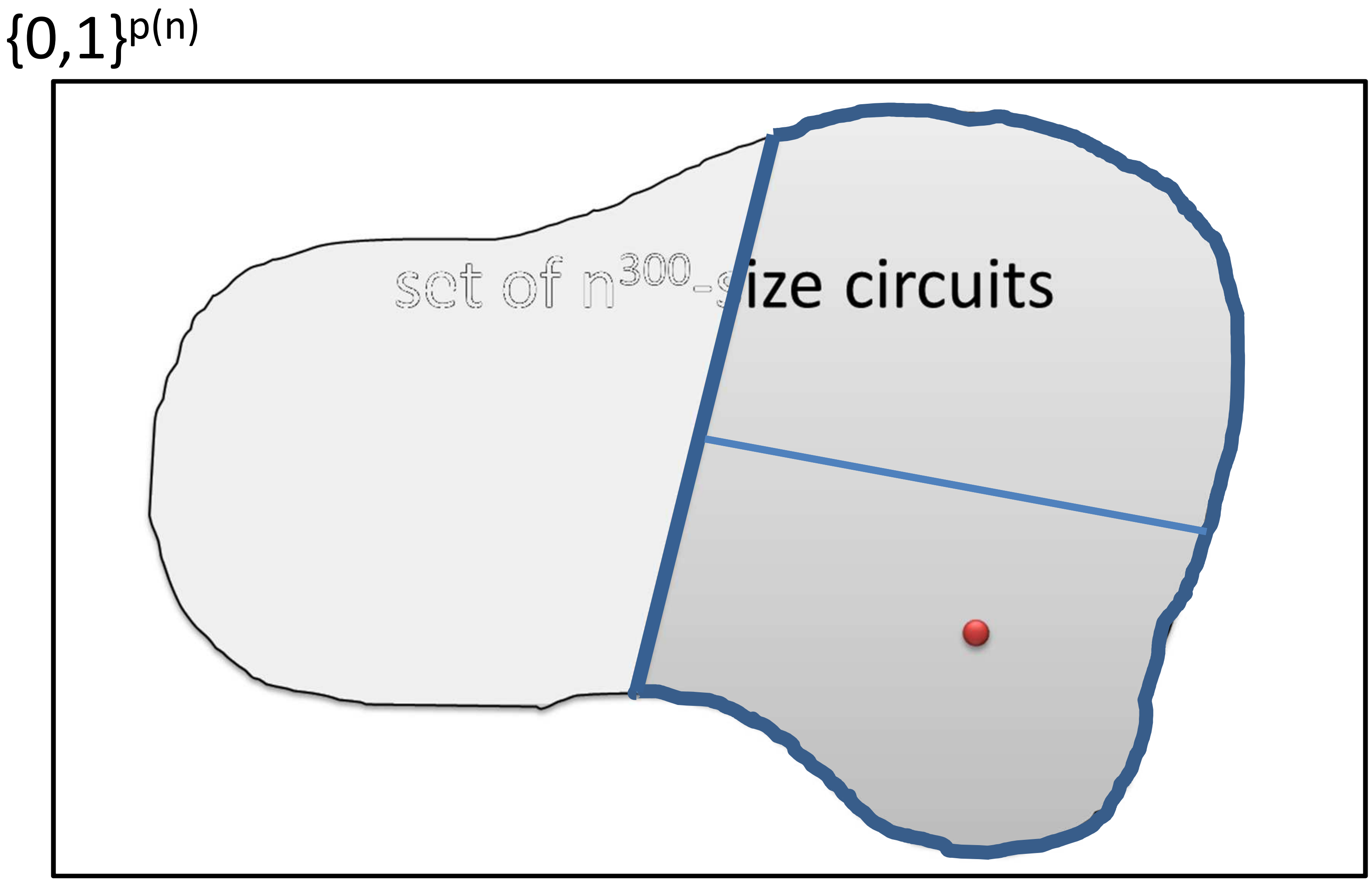
Search in set of circuits



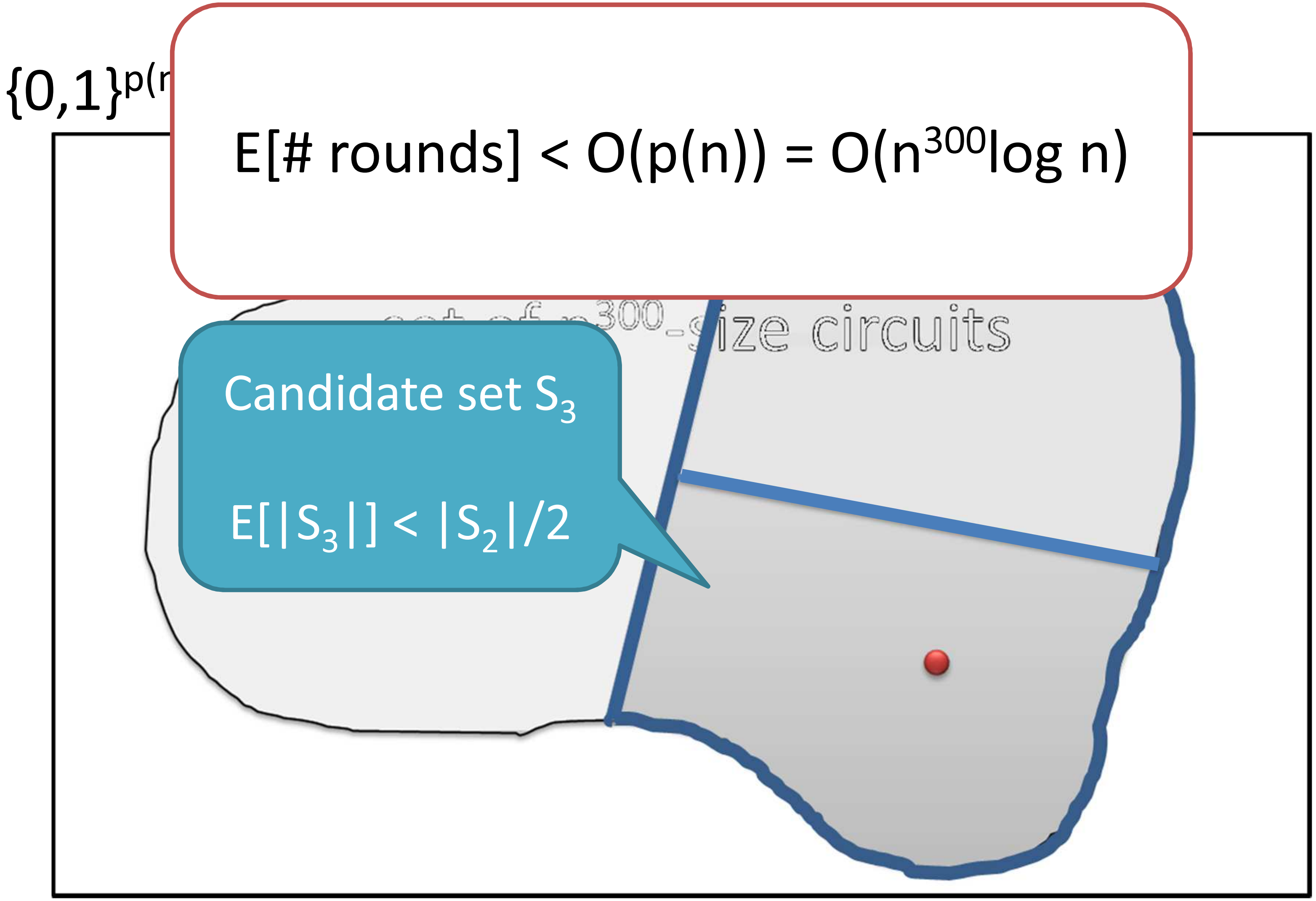
Search in set of circuits



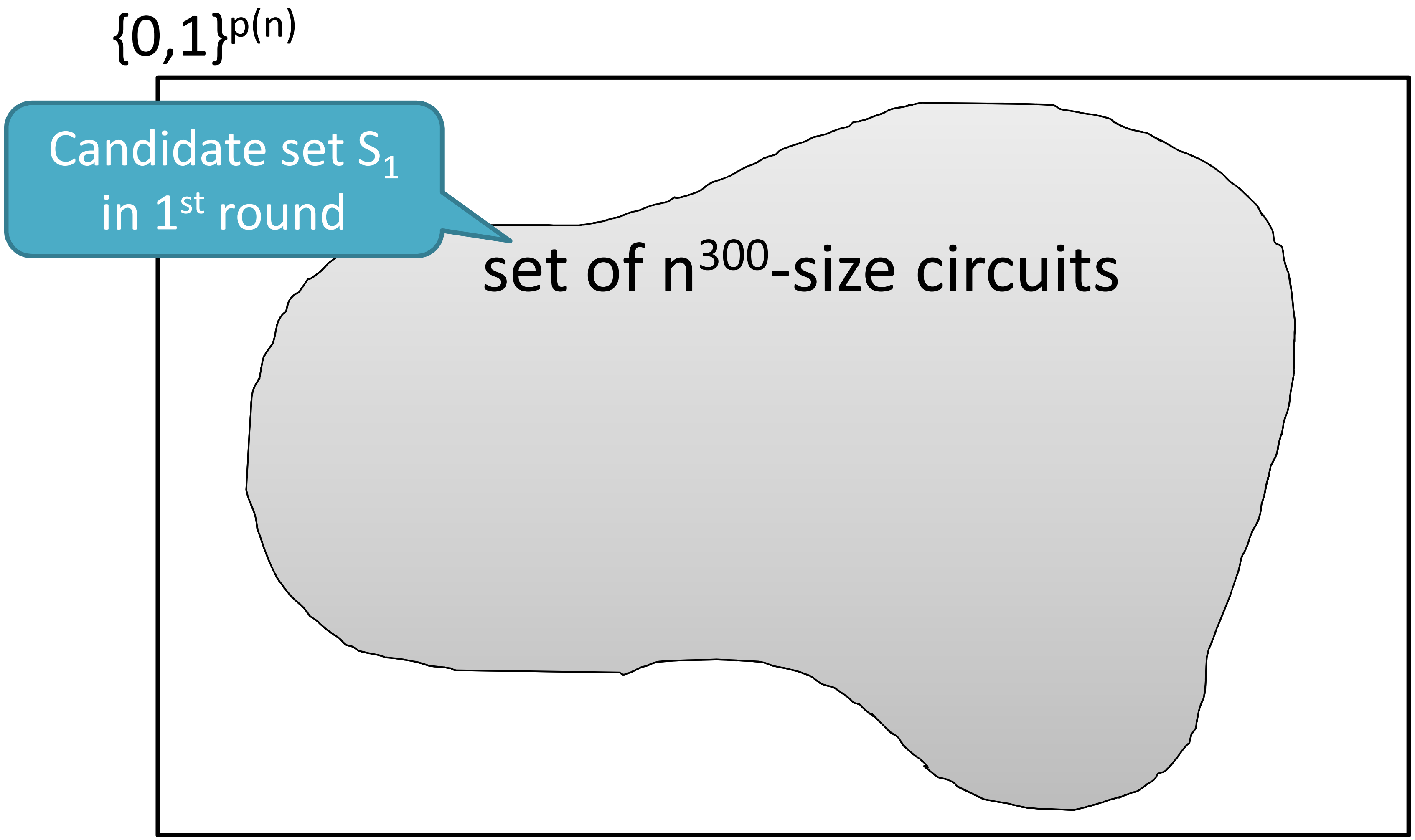
Search in set of circuits



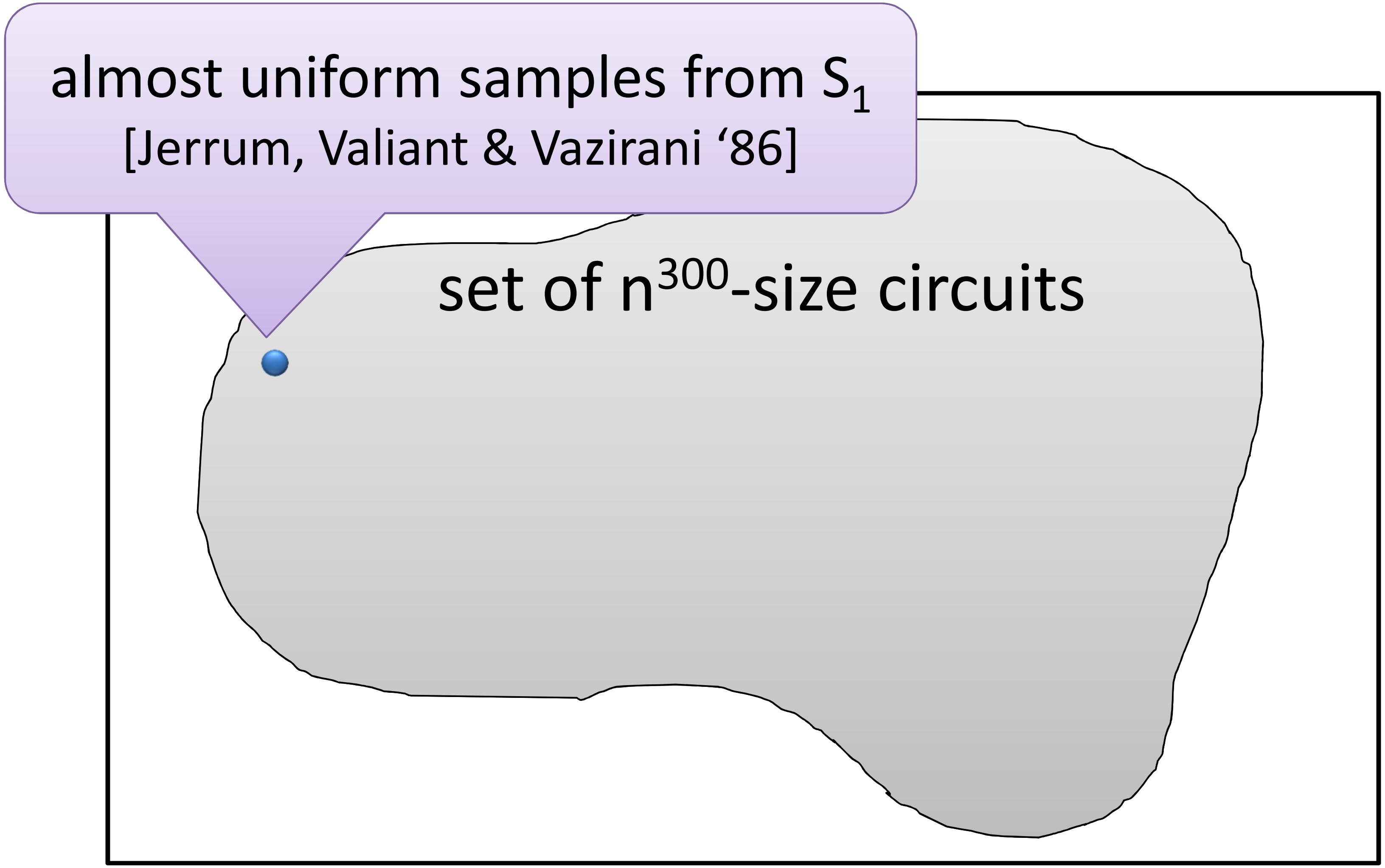
Search in set of circuits



How to Halve

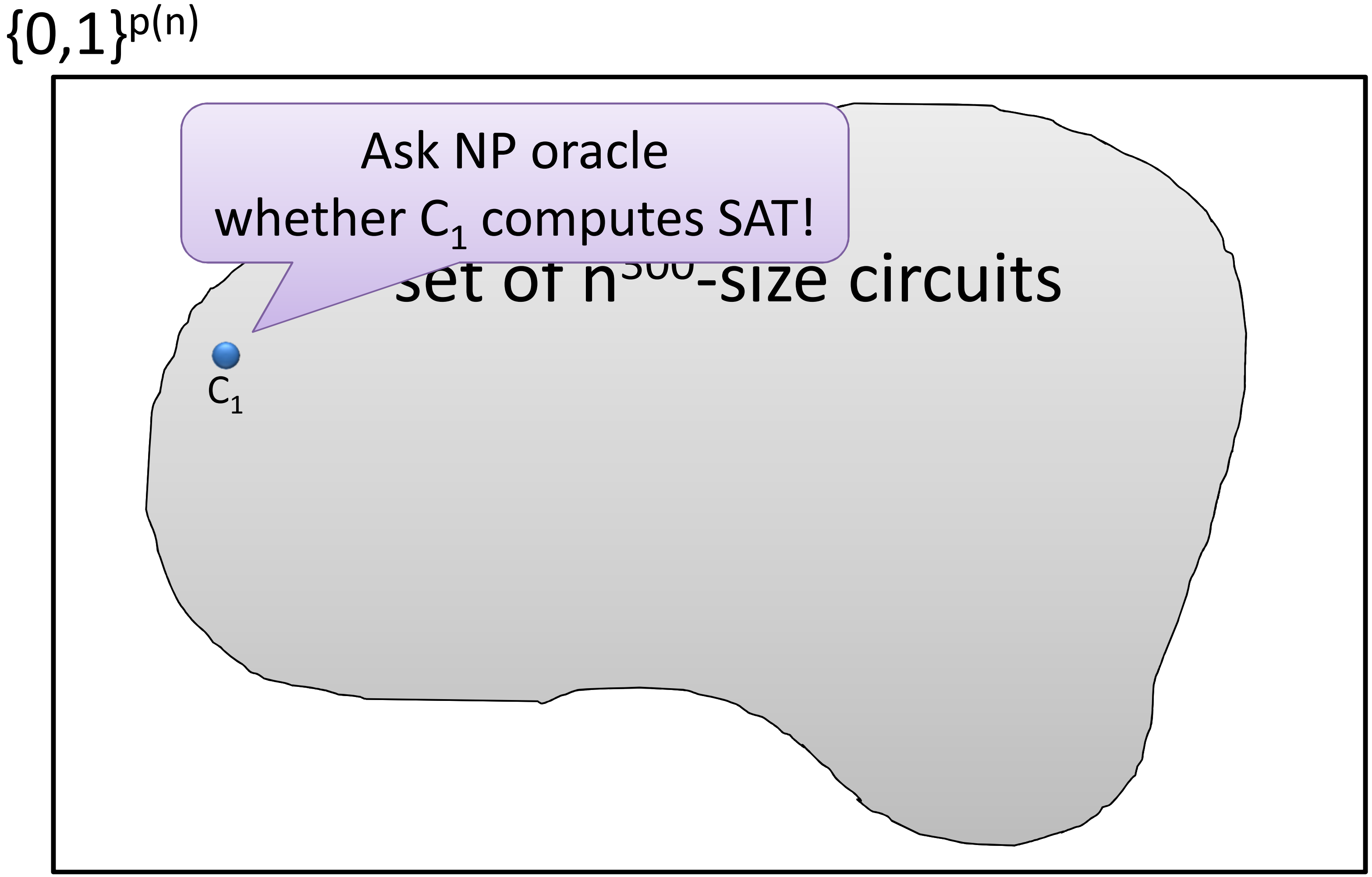


How to Halve

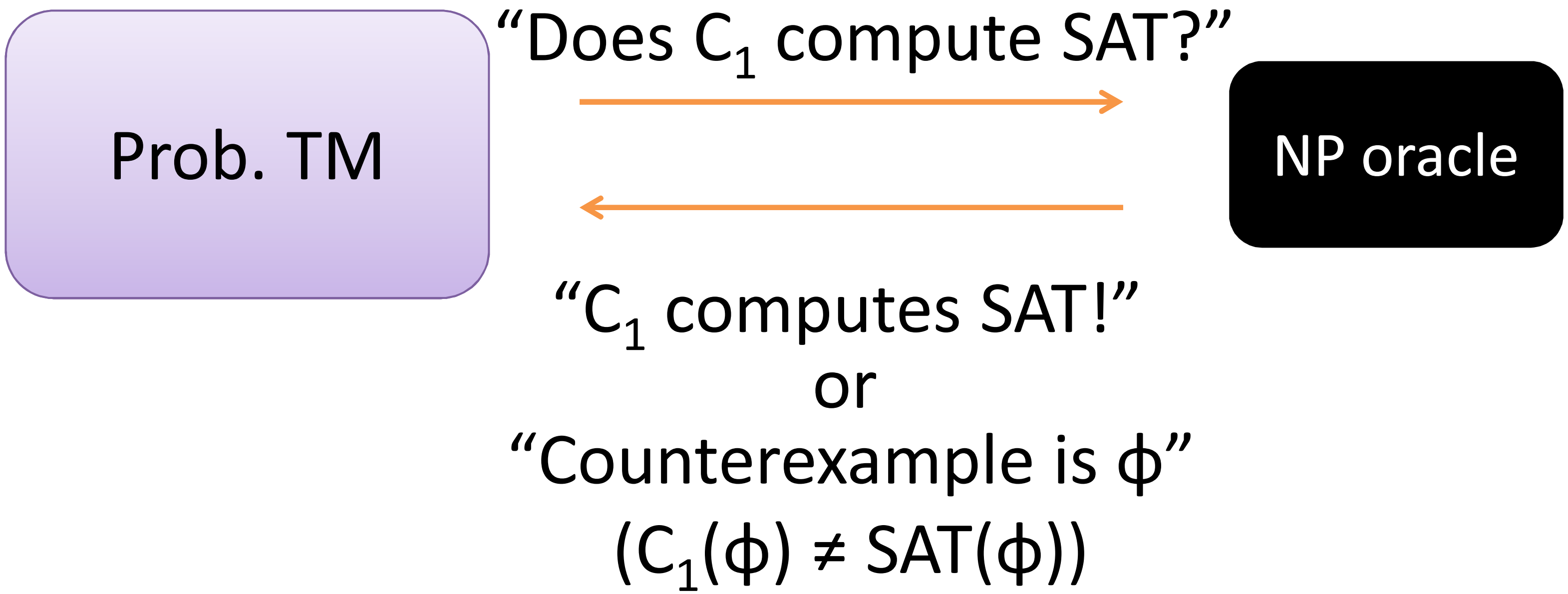


almost uniform samples from S_1
[Jerrum, Valiant & Vazirani '86]
set of n^{300}-size circuits

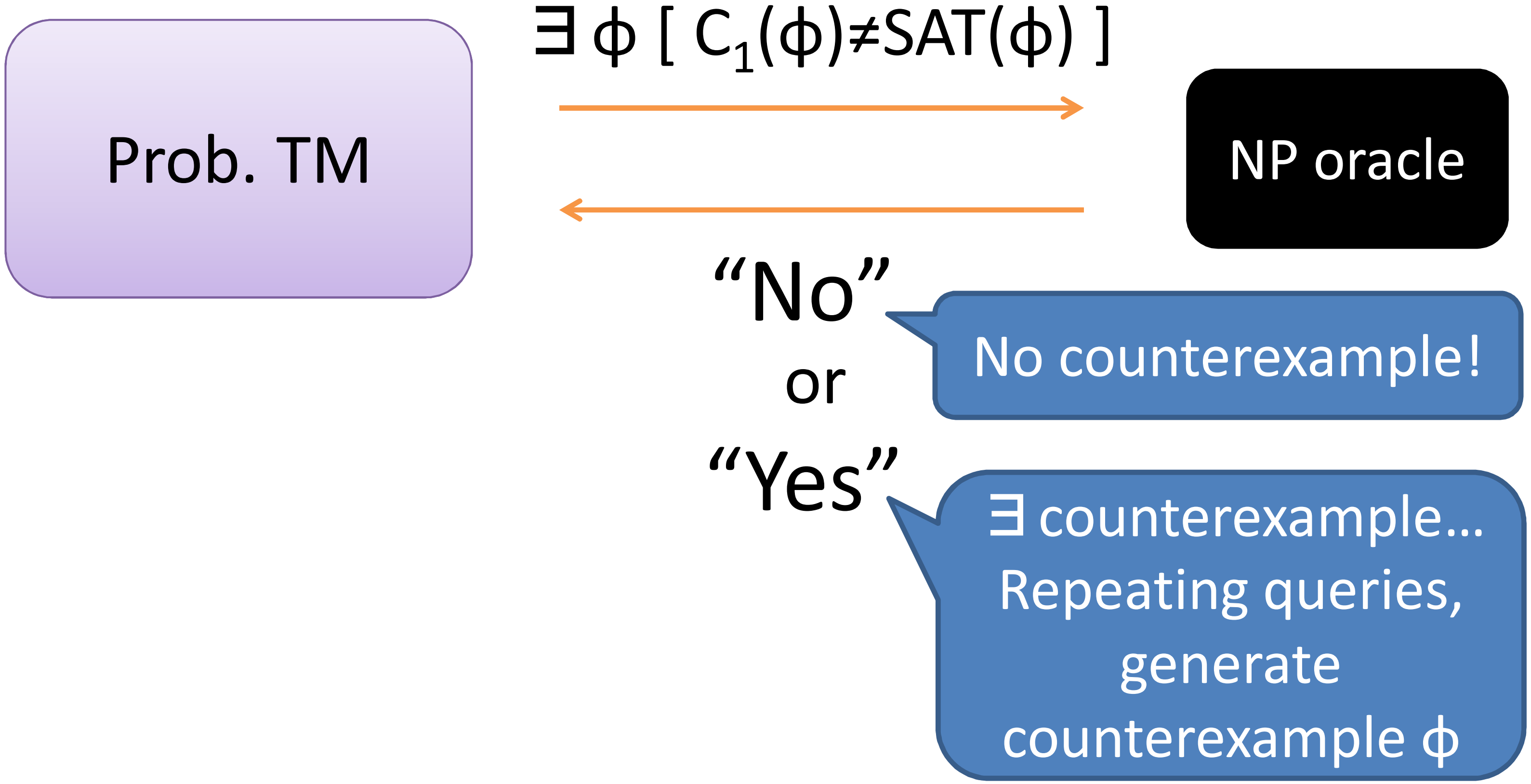
How to Halve


{0,1}^p(n)
Ask NP oracle
whether C1 computes SAT!
set of n -size circuits
C1

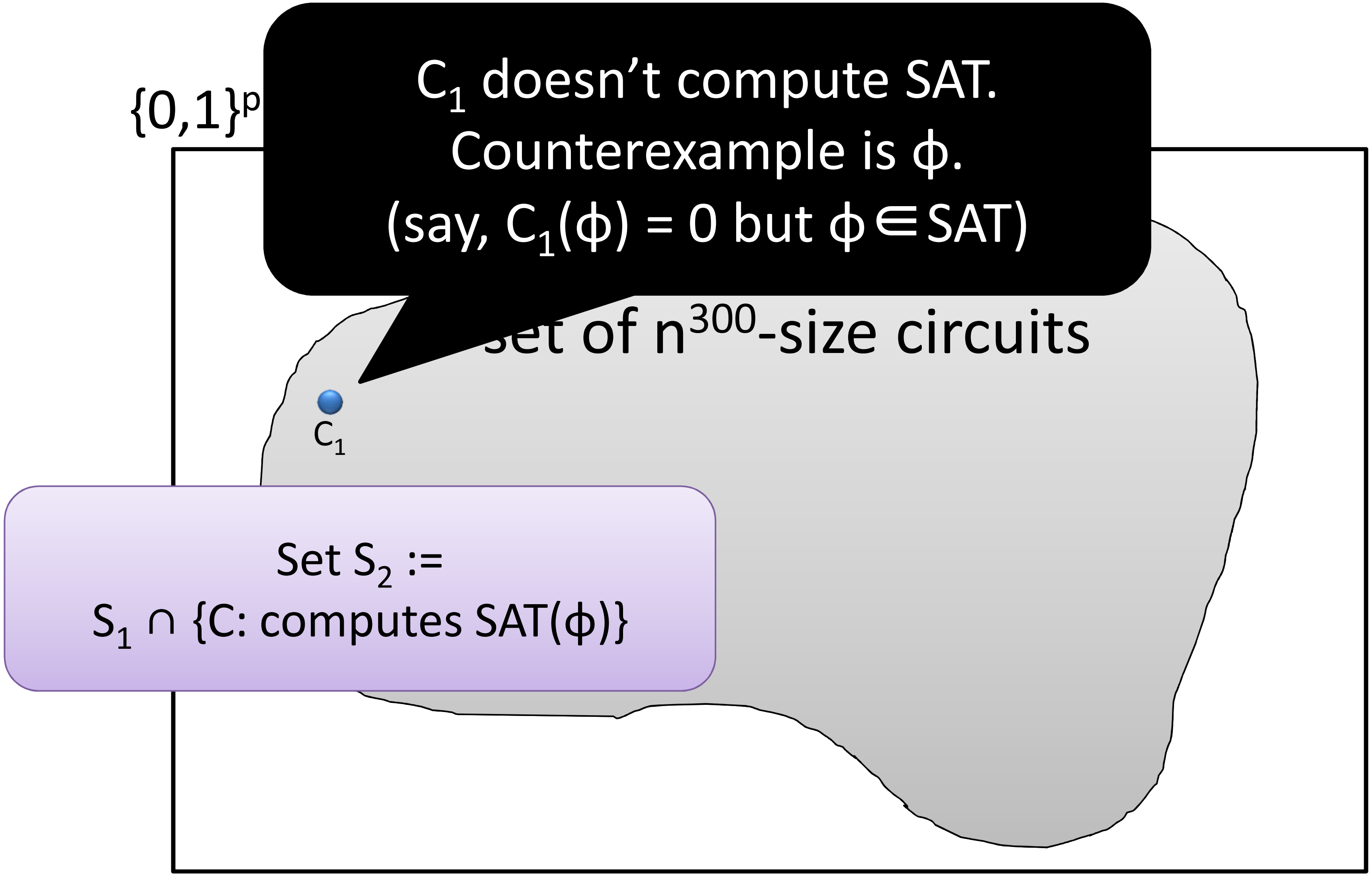
Query to NP oracle



Query to NP oracle

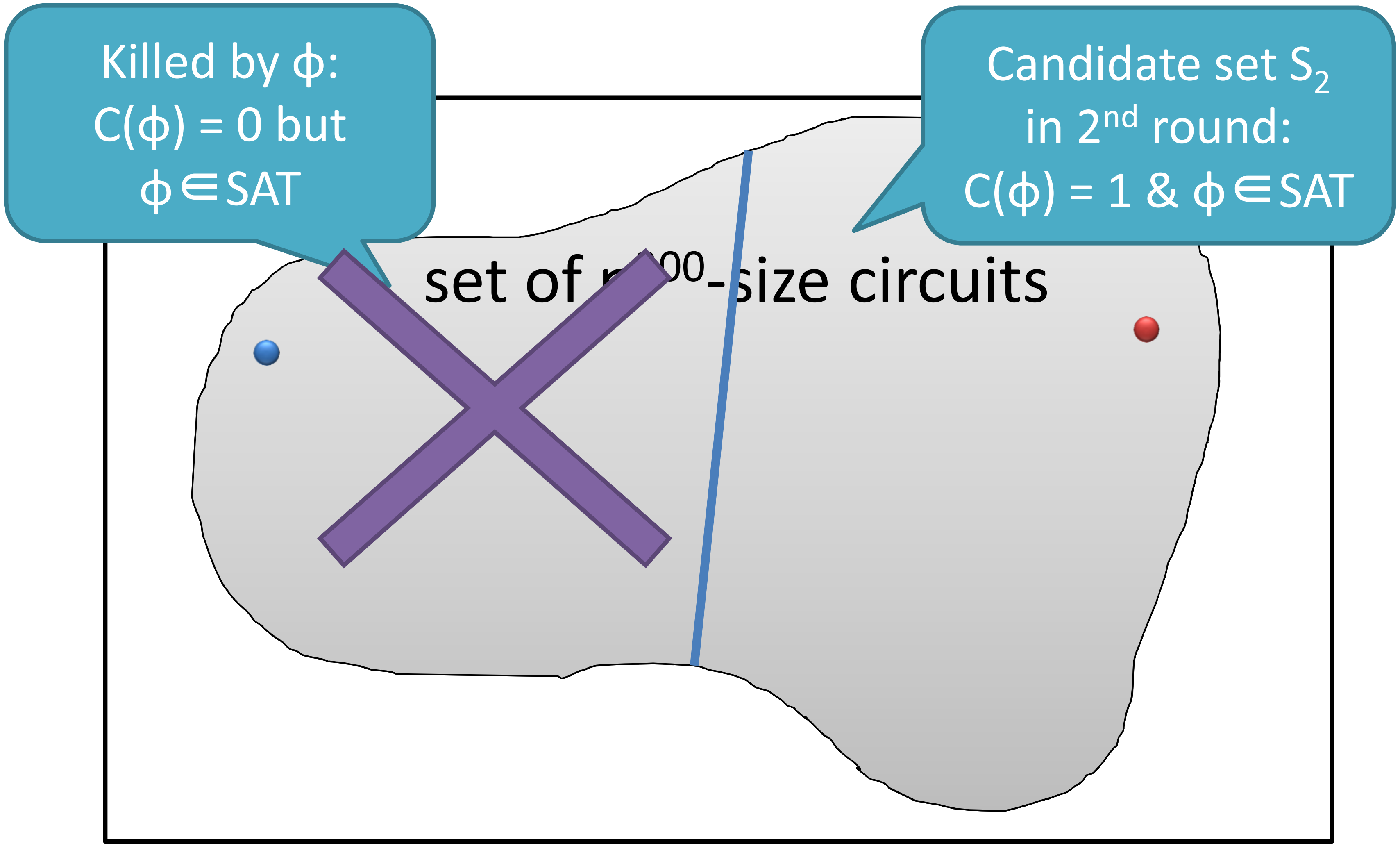


How to Halve

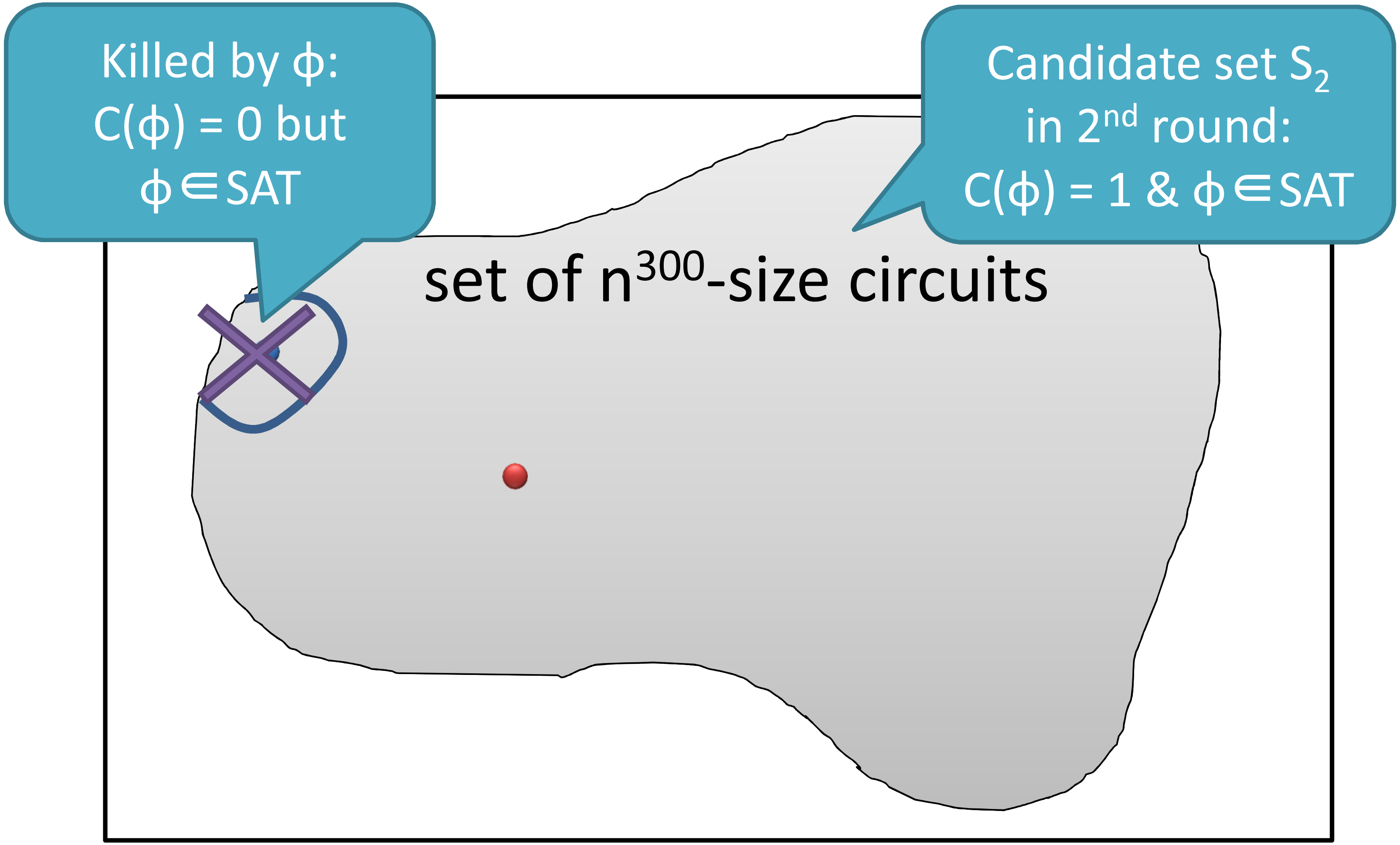


C_1 doesn't compute SAT.
Counterexample is ϕ.
(say, $C_1(\phi) = 0$ but $\phi \in$ SAT)
$\{0,1\}^p$
set of n^{300}-size circuits
C_1
Set S_2 :=
$S_1 \cap$ {C: computes SAT(ϕ)}

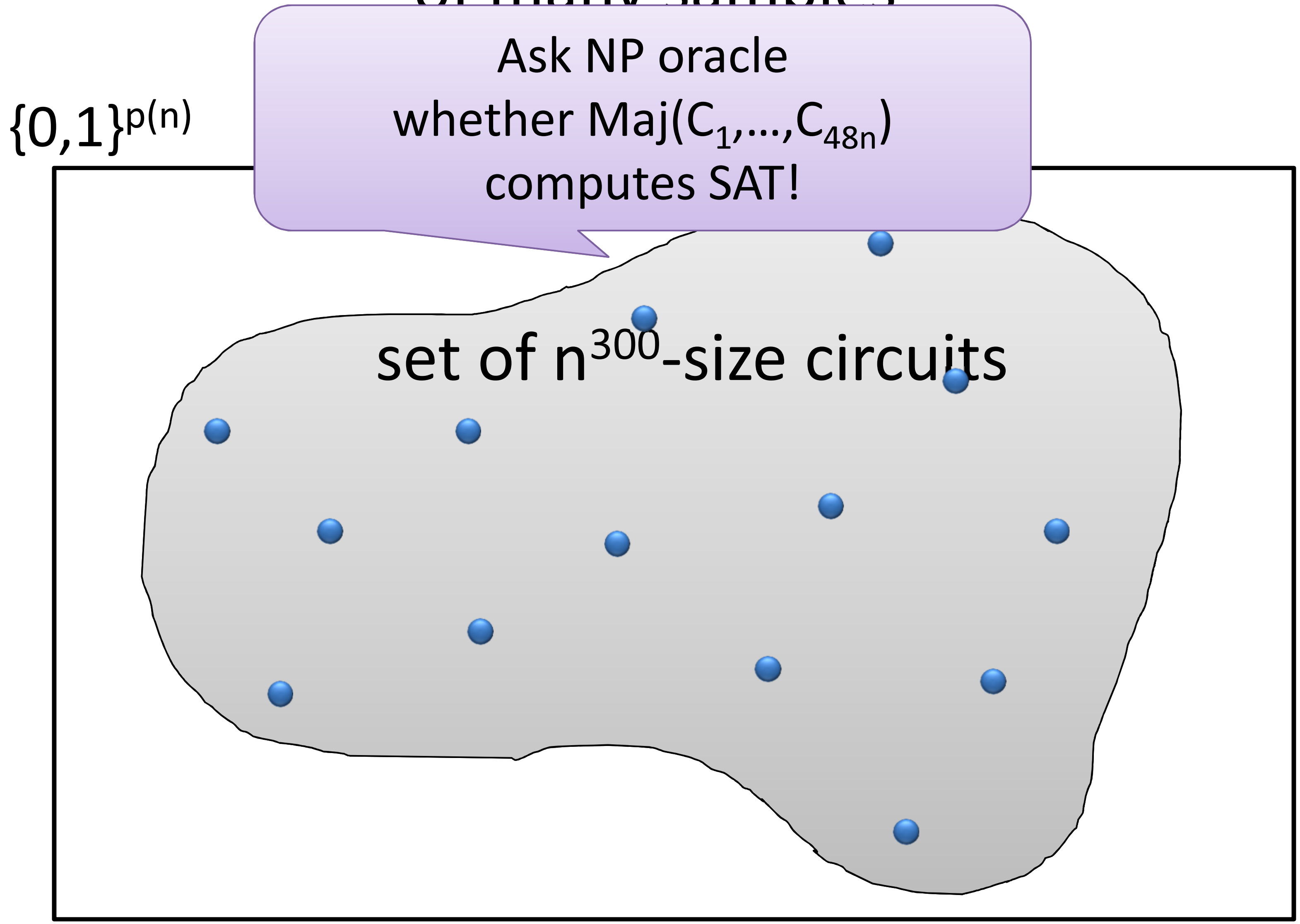
Hopefully…


Killed by ϕ:
C(ϕ) = 0 but
ϕ∈SAT
Candidate set S2
in 2nd round:
C(ϕ) = 1 & ϕ∈SAT
set of
00-size circuits

But, could be…


Killed by ϕ:
C(ϕ) = 0 but
ϕ∈SAT
Candidate set S_2
in 2nd round:
C(ϕ) = 1 & ϕ∈SAT
set of n^{300}-size circuits

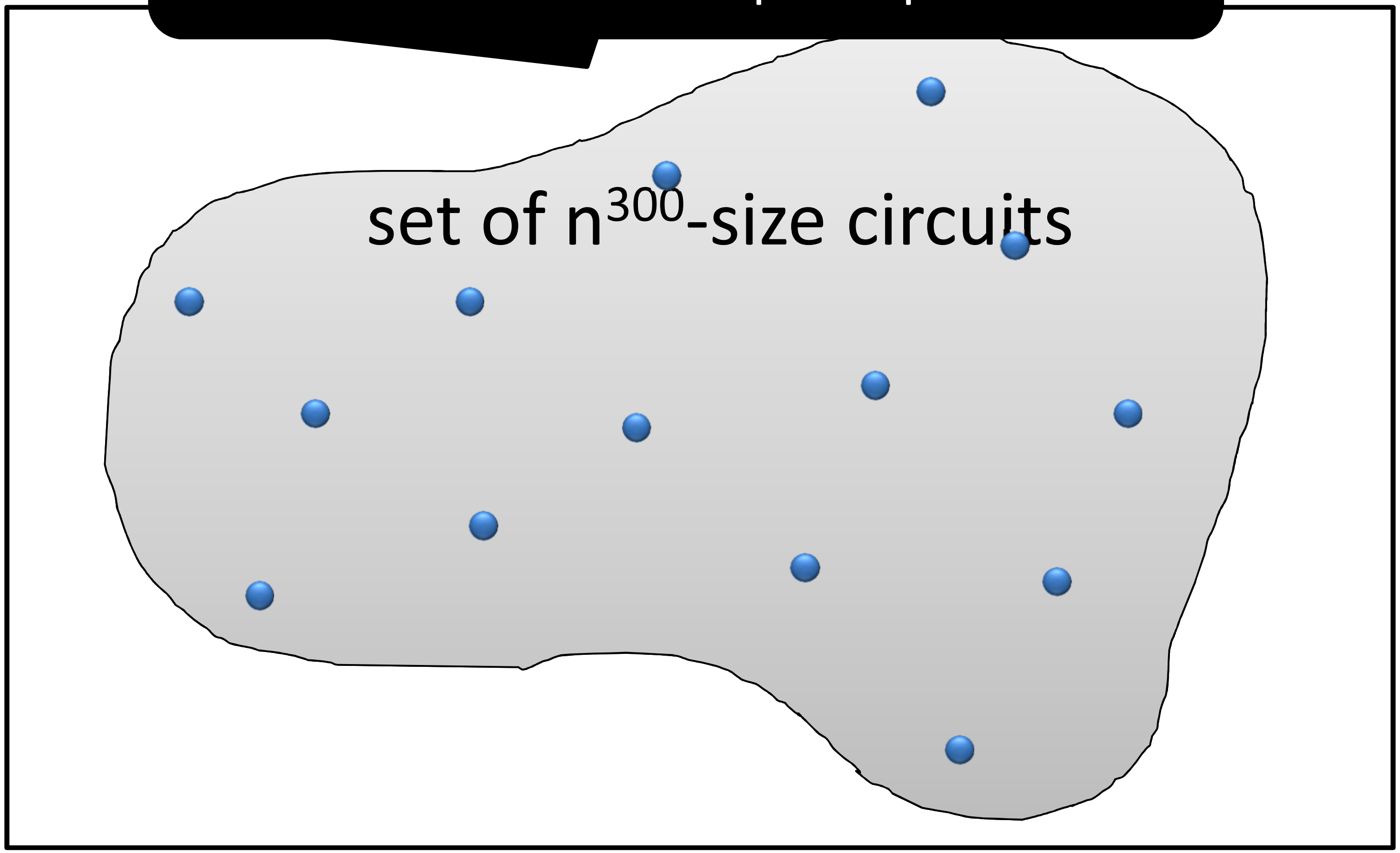
Idea: generate ϕ against majority of many samples



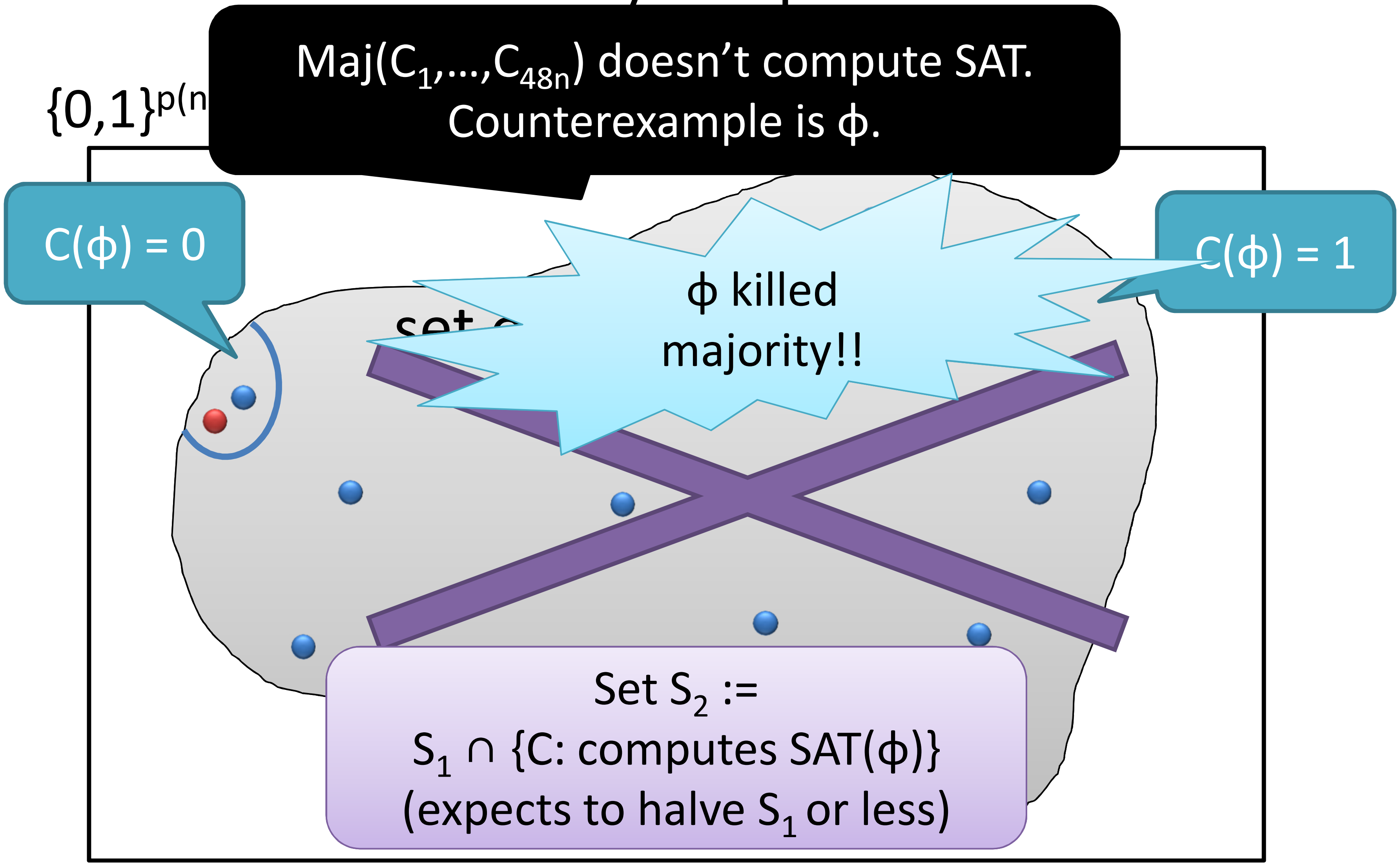
Idea: generate ϕ against majority of many samples

Maj(C_1,…,C_{48n}) doesn't compute SAT.
Counterexample is ϕ.

$\{0,1\}^{p(n}$



Idea: generate ϕ against majority of many samples



Koebler-Watanabe argument

- If n^{300}-size circuit can compute SAT
  - $PH = ZPP^{NP}$ (cf. Karp-Lipton: $PH = \Sigma_2 P$)
    - Finding the circuit C computing SAT in ZPP^{NP}
  - Thus, $ZPP^{NP} \not\subset SIZE(n^{100})$

- If n^{300}-size circuit cannot compute SAT
  - $SAT \in NP$
  - Thus, $NP \not\subset SIZE(n^{300})$

$$\boxed{ZPP^{NP} \not\subset SIZE(n^{100}) \text{ or } NP \not\subset SIZE(n^{300})} \quad \blacksquare$$

Summary

- Koebler & Watanabe's argument

  ≈ Circuit learning algorithm in $\mathrm{ZPP}^{\mathrm{NP}}$
  - Lower-class algorithms improve the result!
  - Learning approach is useful [cf. Gutfreund & K. 2010]

- Open Problem: P^{NP}-learning algorithm?
  - cf. Conjecture: $\mathrm{ZPP}^{\mathrm{NP}} = \mathrm{P}^{\mathrm{NP}}$
  - $\mathrm{ZPP}^{\mathrm{NP}}$-algorithm with pallalel queries ($\mathrm{ZPP}_{||}^{\mathrm{NP}}$)?
  - Relativizable argument doesn't work

    [Aaronson '06].

Recent Breakthroughs

Theorem [Williams '11]

No ACC^0 circuit can compute some NEXP problem

ACC^0 = constant-depth poly-size circuit with 'counter'
Gate set = $\{\wedge, \vee, \neg, Mod_m\}$ for any fixed m
with unbounded fan-in

NEXP = nondet. exp-time comp.
(cf. NP = nondet. poly-time comp.)

New technique:
Fast algorithm computing CKT-SAT implies circuit LBs!

$\mathcal{C}$ CKT-SAT (for circuit class $\mathcal{C}$)

- Given: n-input circuit C: $\{0,1\}^n \to \{0,1\}$
  of class $\mathcal{C}$ (e.g. P/poly, ACC^0)
- Decide: $\exists$ x s.t. C(x)=1

- brute-force algorithm needs $O(m \cdot 2^n)$ time
  – m = circuit size |C|

Overvie Suppose $\mathcal{C}$ = P/poly ument

1st step

∃ Fast (exp-time) algorithm for $\mathcal{C}$ CKT-SAT
➔ NEXP $\not\subset \mathcal{C}$

2nd step

∃ Fast (exp-time) algorithm for ACC^0 CKT-SAT

Proof Overview:

Fast CKT-SAT algorithm ➔ NEXP lower bounds

Assumption

NEXP $\subset$ P/poly & $\exists$ fast CKT-SAT algorithm

$\mathrm{NTIME}[2^n] \subsetneq \mathrm{NTIME}[2^n/n]$

Goal

$\mathrm{NTIME}[2^n] \subseteq \mathrm{NTIME}[2^n/n^8]$,

contradicts the Nondet. Hierarchy Theorem!

Ingredients

1. efficient & local reduction to 3SAT [Tourlakis '00, Fortnow, Lipton, van Melkebeek, & Viglas '05]
2. witness circuits for NEXP problem [Impagliazzo, Kabanets & Wigderson '02]

Efficient & Local Reduction to 3SAT

Theorem [Tourlakis '00, Fortnow, Lipton, van Melkebeek & Viglas '05]

$\exists$ $(2^n \cdot \mathrm{poly}(n))$-time reduction R s.t. $\forall L \in \mathrm{NTIME}[2^n]$,

$\underbrace{\boxed{x}}_{n} \rightarrow \boxed{R} \rightarrow \underbrace{\boxed{\text{3CNF: } \phi_x = C_1 \wedge C_2 \wedge \ldots}}_{2^n \cdot \mathrm{poly}(n)}$

$x \in L \Leftrightarrow R(x) = \phi_x \in \mathrm{SAT}$

$\exists$ **poly(n)**-time algorithm M s.t.

$\overbrace{\boxed{x}}^{n} \rightarrow \boxed{M} \rightarrow \boxed{\text{i-th clause } C_i}$

$\underbrace{\boxed{i}}_{n+O(\log n)} \rightarrow \boxed{M}$

Witness Circuit for NEXP

Theorem [Impagliazzo, Kabanets & Wigderson ’02]

NEXP $\subset$ P/poly ➔ NEXP has poly-size witness circuit

Class NEXP

$L \in$ NEXP

Def

$x \in L \Longrightarrow \exists w\ R(x,w) = 1$

$x \notin L \Longrightarrow \forall w\ R(x,w) = 0$

$|w| = 2^{poly(|x|)}$

Exponentially long witness!

Witness Circuit for NEXP

Theorem [Impagliazzo, Kabanets & Wigderson '02]

NEXP $\subset$ P/poly ➔ NEXP has poly-size witness circuit

Class NEXP

poly-size witness circuit

$L \in \text{NEXP}$

Def

$x \in L \Rightarrow \exists\, W_x\ R(x, W_x(0\ldots0)\ldots W_x(1\ldots1)) = 1$

$x \notin L \Rightarrow \forall\, W_x\ R(x, W_x(0\ldots0)\ldots W_x(1\ldots1)) = 0$

$|W| = \textbf{poly}(|x|)$

Fast Algorithm for $\forall L \in NTIME[2^n]$

Algoritm: Hierarchy Breaker

Input: $x \in \{0,1\}^n$

1. Nondet.ly guess witness circuit W_x

2. Construct a circuit D_{Wx}: $\{0,1\}^{n+O(\log n)} \to \{0,1\}$
 - s.t. $\exists i, D_{Wx}(i) = 1 \Leftrightarrow x \notin L$ (next slide for details)

3. Apply CKT-SAT algorithm A to $A(D_{Wx})$
 - Output "Yes" $\Leftrightarrow A(D_{Wx}) = 0$ ($\Leftrightarrow \forall i, D_{Wx}(i) = 0$)

Running Time = $O(2^n/n^8)$

➔ Contradiction with Nondet. Hierachy Theorem!

2. Construct a circuit $D_{Wx}: \{0,1\}^{n+O(\log n)} \to \{0,1\}$

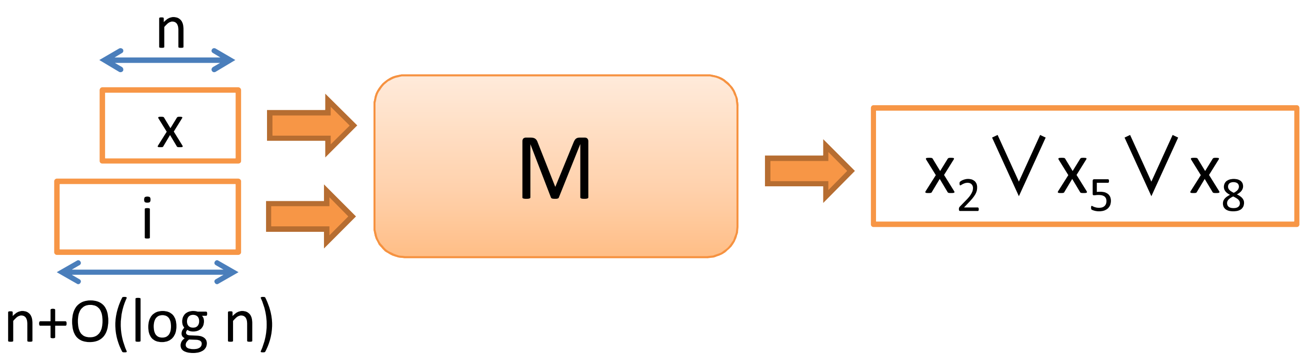
s.t. $\exists\, i,\ D_{Wx}(i) = 1 \Leftrightarrow x \notin L$

Circuit D_{Wx}

$\phi_x \in SAT \Leftrightarrow x \in L$

Input: $i \in \{0,1\}^{n+O(\log n)}$

1. Print i-th clause C_i of ϕ_x by M



2. Check if C_i is NOT satisfied by W_x

3. Output 1 $\Leftrightarrow$ C_i is NOT satisfied

What's D_{Wx} doing?

Case: ϕ_x is NOT satisfiable by any W_x

UNSAT!

Sat. by W_x

Not Sat. by W_x!

Sat. by W_x

$\phi_x = \quad \wedge \; \neg x_1 \vee x_6 \vee x_{11} \; \wedge \; x_3 \vee x_5 \vee x_8 \; \wedge \; x_1 \vee \neg x_6 \vee x_{11} \; \wedge \; \dots$

W_x is inconsistent = D_{Wx} is SAT

SAT!

Sat. by W_x

Sat. by W_x

Sat. by W_x

$\phi_x = \quad \wedge \quad \wedge \quad \wedge \quad \wedge \; \dots$

$\forall$ clause C_i sat. $\Leftrightarrow$ $\forall i$, $D_{Wx}(i) = 0$

Fast Algorithm for $\forall L \in NTIME[2^n]$

Algoritm: Hierarchy Breaker

Input: $x \in \{0,1\}^n$

1. Nondet.ly guess witness circuit W_x

2. Construct a circuit $D_{Wx}: \{0,1\}^{n+O(\log n)} \rightarrow \{0,1\}$
 - s.t. $\exists$ i, $D_{Wx}(i) = 1 \Leftrightarrow x \notin L$

3. Apply CKT-SAT algorithm A to $A(D_{Wx})$;
 - Output "Yes" $\Leftrightarrow A(D_{Wx}) = 0$ $(\Leftrightarrow \forall i, D_{Wx}(i) = 0)$

Running Time = $O(2^n/n^8)$

➔ Contradiction with Nondet. Hierachy Theorem!

Summary

- Williams' argument

  ≈ fast nondet. algorithm from CKT-SAT

- Open Problem: Fast CKT-SAT algorithms?
  - NC^1, or P/poly?
  - Algebrization barrier in NEXP vs. P/poly

    [Aaronson & Wigderson '08].

Concluding Remarks

- High-level approach involves algorithms
  (in bizarre computing models)
  – Koebler-Watanabe: n^{100}-size lower bound in ZPP^{NP}
    - ZPP^{NP} algorithm for circuit learning
  – Williams: superpoly-size ACC^0 lower bound in NEXP
    - Fast non-det. algorithm from CKT-SAT

- "Hardness" is not enough, must put it into NP!
  – Algorithms!