Computational Complexity

Harry Buhrman

(buhrman@cwi.nl)

Tom Bannink

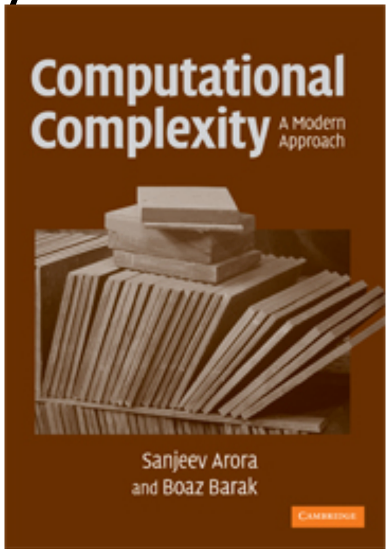
(bannink@cwi.nl)

Algorithms & Complexity group

QuSoft

Course requirements

- Computational Complexity: A Modern Approach by Arora & Barak (http://www.cs.princeton.edu/theory/complexity/)



- lectures (hoorcollege)
  - Tuesday 13:00-15:00, Thursday 13:00-15:00
- werkcollege:
  - Friday 9:00–11:00
  - http://turing-machine.nl/
- Compulsory: hand in exercises every week on Tuesday
- Final exam

Grade

- Hand in exercises before lecture on Tuesday the week after they were distributed
- Final grade exercises is average of obtained grades. We will drop the lowest grade
- Cooperation is allowed, always write down solutions on your own
- Final grade = average of grade final exam and exercises

Het is opgelost: het grootste en mooiste probleem uit de computerwetenschap. Dat zegt een onderzoeker. In zijn bewijs zitten nog veel gaten, zeggen anderen.

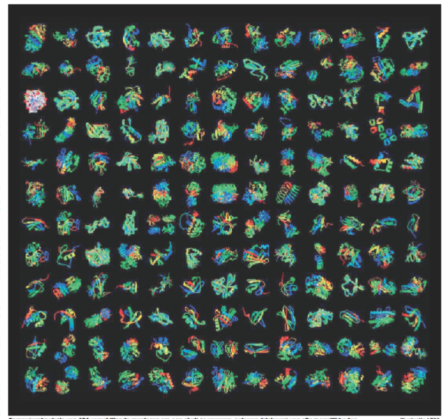
tekst Margriet van der Heijden

De oplossing van het onoplosbare

Monday, August 9th, 2010

Putting my money where my mouth isn't

A few days ago, Vinay Deolalikar of HP Labs started circulating a claimed proof of P≠NP. As anyone could predict, the alleged proof has already been Slashdotted (see also Lipton's blog and Bacon's blog), and my own inbox has been filling up faster than the Gulf of Mexico.



Bloggers slopen droombewijs

Wat is nog wel slim te berekenen en wat niet? Wie het weet, wint een miljoen. Deze zomer was er vals alarm.

Door Arnaut Jaspers

Hoe bewijs je in de wiskunde überhaupt dat iets niet bestaat?

Te berekenen of niet te berekenen, dat is de vraag

The New York Times

Step 1: Post Elusive Proof. Step 2: Watch Fireworks.

By John Markoff

Published: August 16, 2010

The potential of Internet-based collaboration was vividly demonstrated this month when complexity theorists used blogs and wikis to pounce on a claimed proof for one of the most profound and difficult problems facing mathematicians and computer scientists.

Het PAROOL
Vrij, Onverveerd

AMSTERDAM | SPORT | ECONOMIE | ETEN & DRINKEN | CU

MANO BOUZAMOUR | ROOS SCHLIKKER | ALBERT DE LANGE

'Megaloting zorgt niet voor betere match, maar voor minder klagende ouders'

05-06-15 19:00 uur - Bron: Het Parool

© anp

OPINIE

Niet iedereen is blij met het nieuwe plaatsingsmodel voor middelbare scholen. Volgens Krijn van Beek is het een onzichtbare megaloting en een ideaal middel om klagende ouders te omzeilen, zo schrijft hij in een opinie-artikel in Het Parool.

nrc.nl›

Nieuws | Weblogs | Columns | Cultuur | Carrière | In beeld

NU IN HET NIEUWS: GRIEKENLAND KIM JONG UN VLUCHTELINGEN

● 18 juni 2015, 10:42

A'damse ouders boos over uitgelote kinderen: scholenkoepel bedreigd

Barlaeus Gymnasium in Amsterdam, een van de populaire scholen. ANP / Kippa

door Mirjam Remie

ENLAND Een commissievergadering in het terdamse stadhuis zit vol met boze ouders, zo lt AT5. Ze willen hun onvrede uiten over het aar ingevoerde 'matchingsysteem', waarmee 7.500 achtstegroepers via een computer op middelbare school zijn geplaatst. Gisteren bleek dat de helpdesk van scholenkoepel naar een geheime locatie is verhuisd vanwege bedreigingen van ouders aan ewerkers.

ers zijn boos dat zij niets meer kunnen doen aan de uitslag van de matching. De meeste eren zijn geplaatst in de top-5 (99 procent) of top-3 (95 procent) van het lijstje van

plaatst cookies om een optimale gebruikerservaring te kunnen bieden. De cookies worden ingezet

NRC

Profiel Zoeken

Niet één middelbare school kiezen maar wel tien

8.000 kinderen kiezen dezer dagen een middelbare school in Amsterdam. Populaire scholen moesten vorig jaar 518 leerlingen uitloten. Nu is het systeem anders. „De pijn wordt beter verdeeld."
Mirjam Remie Bram Budel

Door **MIRJAM REMIE & BRAM BUDEL** 27 FEBRUARI 2015

P versus NP problem

P versus NP

- One of the seven millennium prize problems
- “In the case of the P versus NP problem and the Navier-Stokes problem, the SAB will consider the award of the Millennium Prize for deciding the question in either direction.”

- P not equal NP $\Rightarrow$ 1 million $
- P equal NP $\Rightarrow$ 6 million $

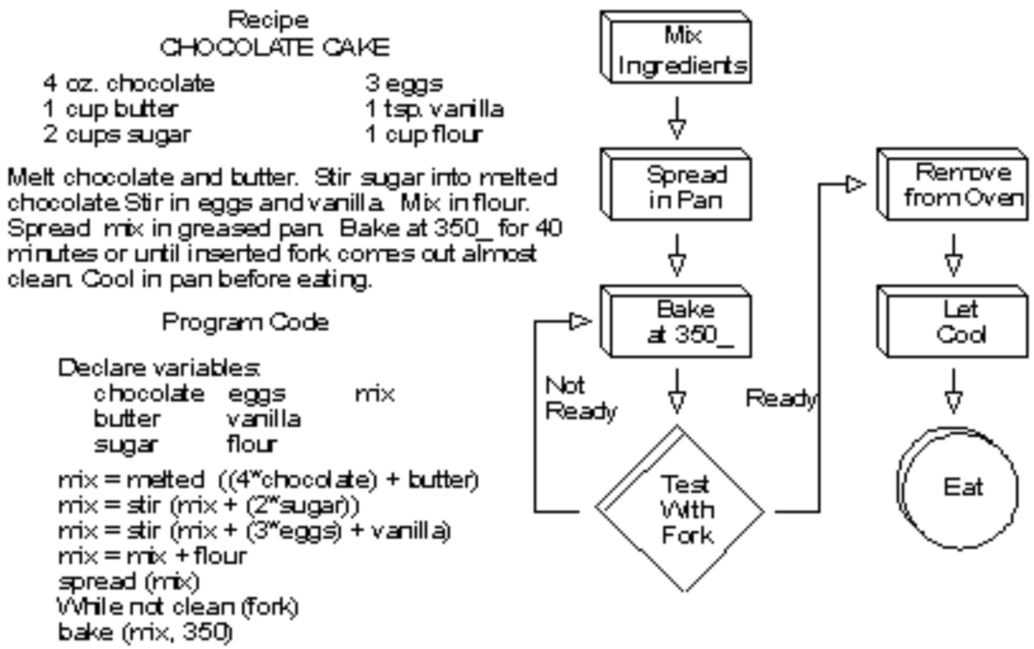
Main characters: Algorithms

Algorithm

- Algorithm is like a cooking recipe

Algorithm

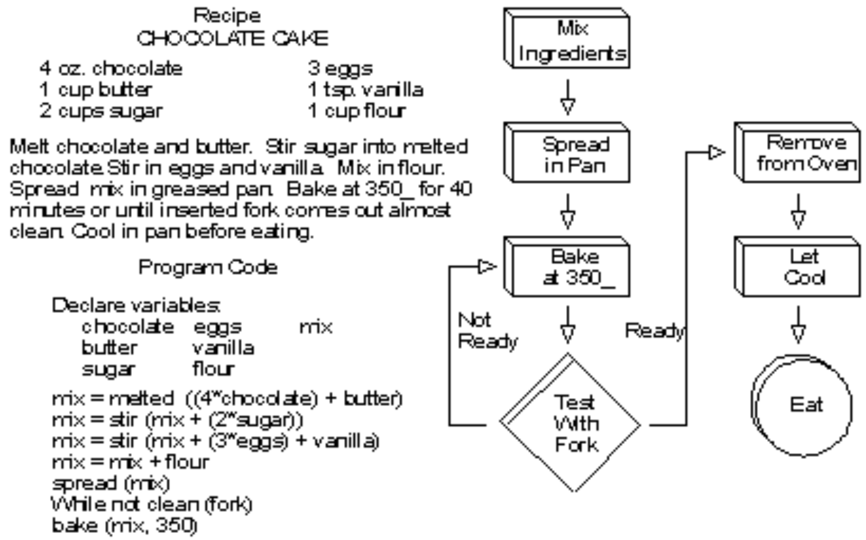
- Algorithm is like a cooking recipe



Algorithm

- algorithm is like a cooking recipe

- input
- computation
  - steps (1 time unit)
- output



Example

Greatest Common Divisor (GCD)

Slow Algorithm

a=21 b=13

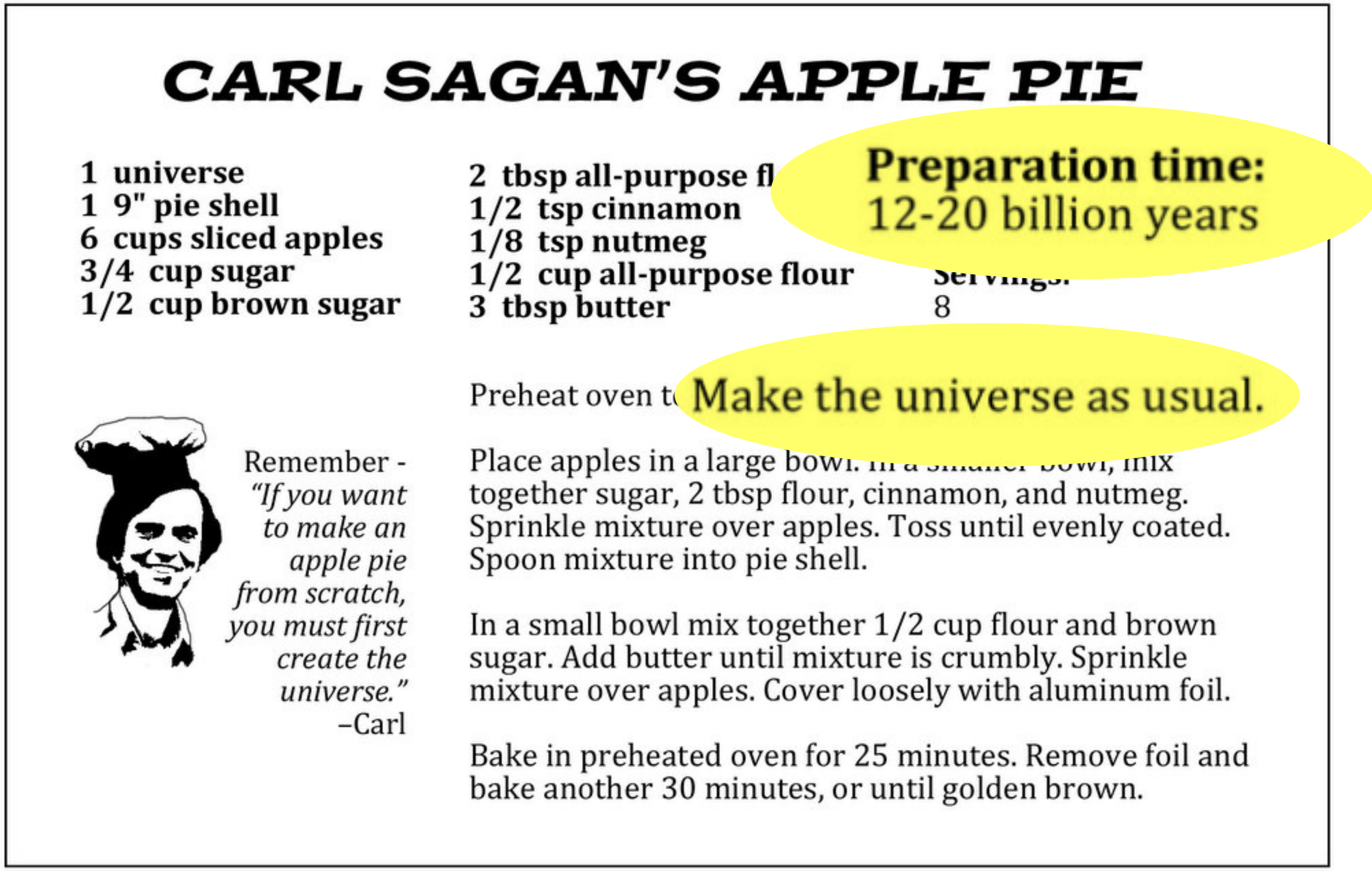
step 1	i=13	$13 \nmid 21$
step 2	i=12	$12 \nmid 21$
step 3	i=11	$11 \nmid 21$
step 4	i=10	$10 \nmid 21$
	⋮	
step 7	i=7	$7 \nmid 13$
	⋮	
step 13	i=1	

output 1

```
slow-gcd(a, b)
  i = min(a,b)
    while  i ∤ a  or  i ∤ b
      i := i -1
output  i
```

Analysis of Algorithm

Analysis of alg. is preparation time of recipe



Slow Algorithm

a=21 b=13

step 1	i=13	$13 \nmid 21$
step 2	i=12	$12 \nmid 21$
step 3	i=11	$11 \nmid 21$
step 4	i=10	$10 \nmid 21$
	⋮	
step 7	i=7	$7 \nmid 13$
	⋮	
step 13	i=1	

output 1

```
slow-gcd(a, b)
  i = min(a,b)
    while i ∤ a or i ∤ b
      i := i -1
output i
```

if gcd(a,b)=1 then

algorithm uses min(a,b) steps

Better Algorithm

Euclidean Algorithm

Greatest Common Divisor (GCD)

step 0	a=21	b=13
step 1	a=13	b= 21 mod 13 = 8
step 2	a=8	b= 13 mod 8 = 5
step 3	a=5	b= 8 mod 5 = 3
step 4	a=3	b= 5 mod 3 = 2
step 5	a=2	b= 3 mod 2 = 1
step 6	a=1	b= 2 mod 1 = 0

output 1

```
function gcd(a, b)
    while b ≠ 0
      t := b
      b := a mod b
      a := t
    output a
```

Analysis GCD-Algorithm

- worst case number of steps?

> Theorem
> alg. terminates in
> $2\log(m) + 1$ steps
> $m = \max(a,b)$

```
function gcd(a, b)
    while b ≠ 0
        t := b
        b := a mod b
        a := t
    output a
```

step 0	a=21	b=13
step 1	a=13	b= 21 mod 13 = 8
step 2	a=8	b= 13 mod 8 = 5
step 3	a=5	b= 8 mod 5 = 3
step 4	a=3	b= 5 mod 3 = 2
step 5	a=2	b= 3 mod 2 = 1
step 6	a=1	b= 2 mod 1 = 0

Proof:

every second step a is at least halved

Complexity

- Euclid: 2log (m) +1 m =max(a,b)
- Slow: m' m'= min(a,b)
- Length of the input: log(a) + log(b) = n

Euclid: $O(n)$ Slow: $2^{O(n)}$

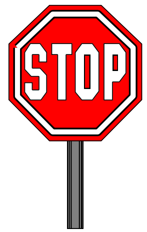
- Euclid exponentially faster than slow!
- Complexity of computational problem is running time of the best algorithm

Computation & Complexity

- Computational problem:
  - INPUT —computation→ OUTPUT
  - Example: a,b output gcd(a,b)
- Complexity:
  - Number of computation steps needed for "best" algorithm
  - function of the input size

Complexity

- Determine the complexity of a computational problem:
  - Upper bound: construct algorithm
  - Lower bound: **any** algorithm needs this many steps



- Ideally upper bound = lower bound

functions of the input size

Complexity of gcd problem

- Euclid's algorithm runs in O(n) steps
- Can we devise a faster algorithm?
- Not really: any algorithm has to read the whole input: requires n steps
  - Upper Bound: O(n)
  - Lower Bound: Ω(n)
- Complexity of gcd is linear.

Complexity Class P

Feasible Problems: P

- Feasible or efficient algorithms run in polynomial time: n^c (some c)
- Complexity Class **P :**
  - All the problems that have feasible algorithms
- Example:
  - Linear Programming
  - Network Flow Problems
  - Shortest Path

For these problems upper bound is "close" to lower bound: at most polynomial far off.

Another problem

Satisfiablity

Satisfiability

- variables $x_1 \ldots x_n$
- Clause $C_1 \ldots C_m$ $\quad C_l = (x_i \vee x_j \vee \overline{x_k})$
- formula $\phi(x_1 \ldots x_n) = C_1 \wedge \ldots \wedge C_m$
- exist $\alpha_1 \ldots \alpha_n$ $\quad \alpha_i \in \{T, F\}$
- such that $\phi(x_1 = \alpha_1 \ldots x_n = \alpha_n) = T$

Example

$$(\overset{F}{\overline{x_1}} \vee \overset{F}{\overline{x_2}}) \wedge (\overset{T}{x_1} \vee \overset{T}{x_3}) \wedge (\overset{T}{x_2} \vee \overset{F}{\overline{x_3}}) \wedge (\overset{F}{\overline{x_1}} \vee \overset{T}{x_3})$$

$$\Downarrow$$

$$F \quad \wedge \quad T \quad \wedge \quad T \quad \wedge \quad T$$

$$\Downarrow$$

$$F$$

$$x_1 = T \qquad x_2 = T \qquad x_3 = T$$

Example

$$(\overline{x_1} \vee \overline{x_2}) \wedge (x_1 \vee x_3) \wedge (x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_3)$$

$$T \wedge T \wedge \quad \wedge T$$

SATISFIABLE

$x_1 = F$ $\quad$ $x_2 = T$ $\quad$ $x_3 = T$

Satisfiability

- variables $x_1 \dots x_n$
- Clause $C_1 \dots C_m$ $\quad C_l = (x_i \vee x_j \vee \overline{x_k})$
- formula $\phi(x_1 \dots x_n) = C_1 \wedge \dots \wedge C_m$
- exist $\alpha_1 \dots \alpha_n$ $\quad \alpha_i \in \{T, F\}$
- such that $\phi(x_1 = \alpha_1 \dots x_n = \alpha_n) = T$

$$SAT = \{\phi \mid \phi \text{ is satisfiable}\}$$

simple algorithm: try all 2^n assignments

Unknown Complexity

- It is hard to determine the complexity of many problems
- Example:
  - Is this formula satisfiable? **SAT**
  - Traveling Salesman Problem. **TSP**
- Lower Bound: n
- Upper Bound: 2^n

Best Known!

Complexity Class NP

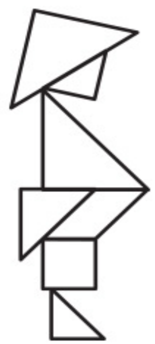
NP

- P = class of problems that are efficiently computable.
- NP = class of problems that have efficiently checkable solutions.
  - but solution may be hard to find!

Tangram

P: compute solution

NP: easy to check solution



solution

NP

- complexity class NP
  - polynomial time to check solution

- x in L: exists a y: P(x,y) = 1 (true)

polynomial time computable in length of x only

SAT in NP

φ is satisfiable

$\exists\, \alpha : \varphi(\alpha) = \text{True}$

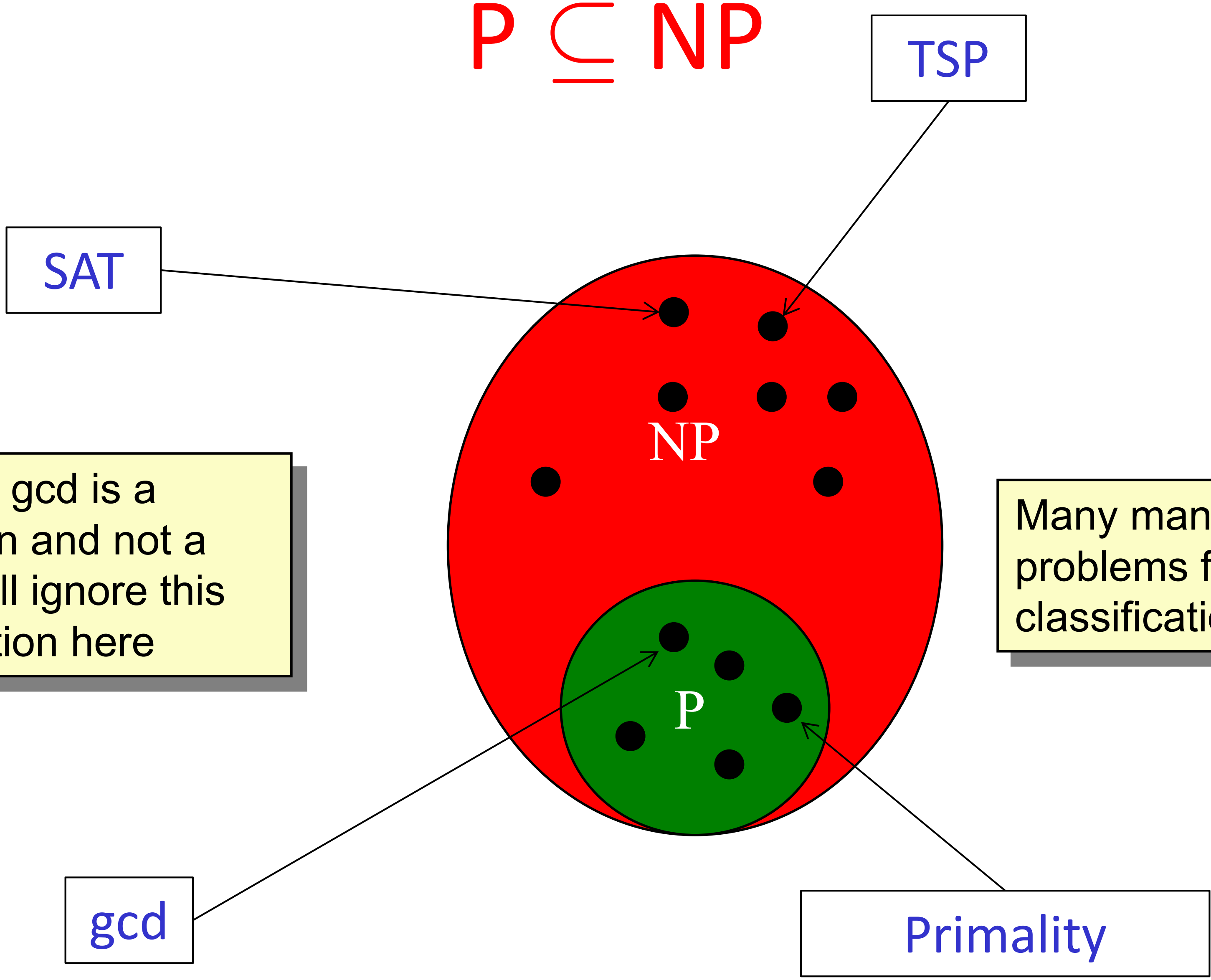
P & NP

- complexity class NP
  - easy to check solution
  - polynomial time check
  - easy to check assignment is satisfiable

versus

- complexity class P
  - easy to find solution
  - decide in polynomial time
  - compute in polynomial time gcd(a,b)

$P \subseteq NP$



strictly, gcd is a function and not a set. Will ignore this distinction here

Many many more problems fall into this classification

Reductions & Completeness

reduction

$$A \leq_T^p B$$

compute A in poly-time with B as free subroutine

"A is computationally not harder than B"

"if B in P then A in P"

C is NP-complete

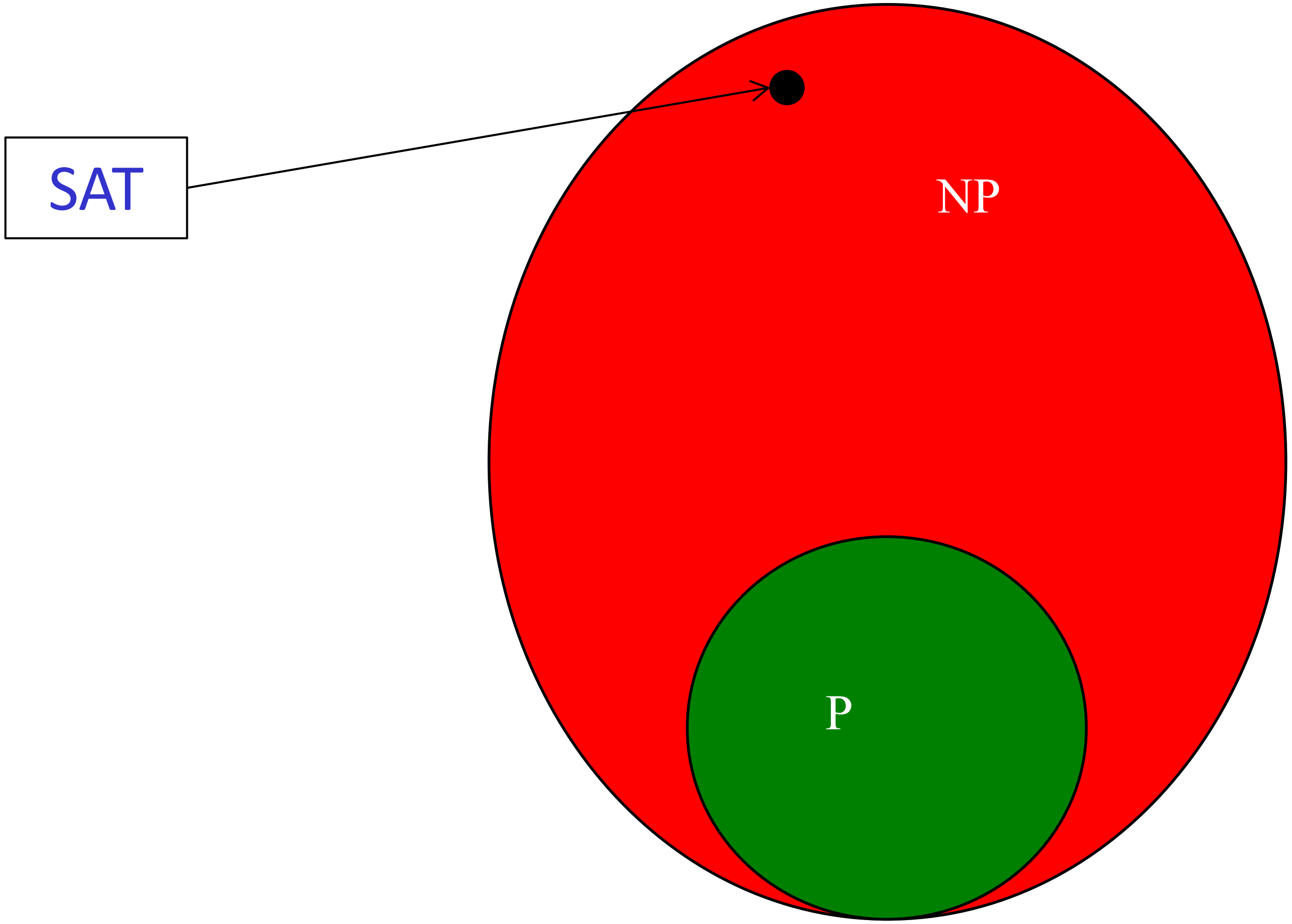
- $C \in NP$
- all $A \in NP$: $A \leq_T^p C$

Theorem

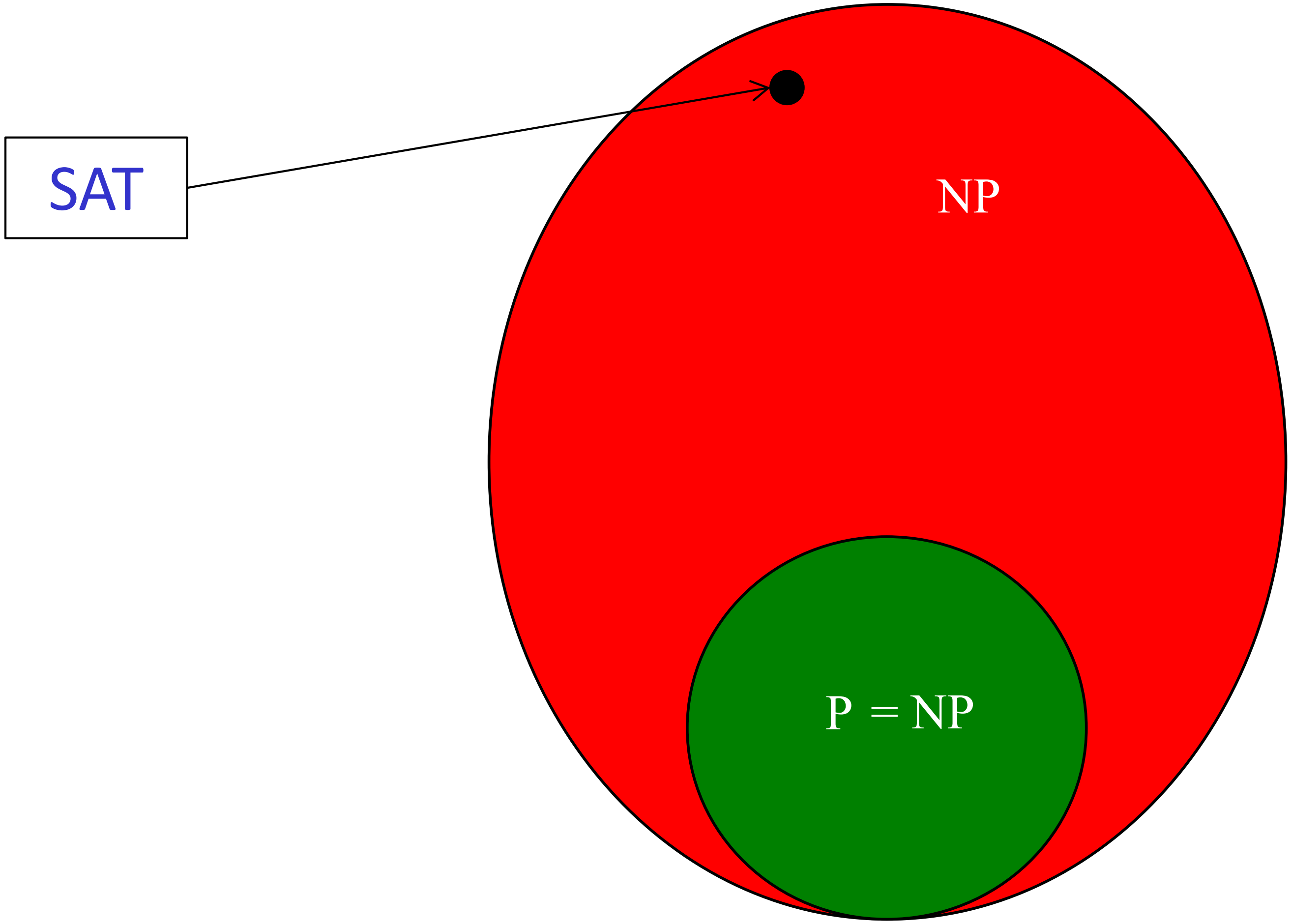
- SAT, TSP, many others NP-complete
- SAT in P $\Leftrightarrow$ P=NP

P versus NP

$P \neq NP$



P = NP



P versus NP Question

- P = NP?
- widely believed that $P \neq NP$
- how to show this is true?
  - Prove better lower bounds for existing problems like SAT
  - Construct problem in NP with super polynomial lower bound

Lower Bounds

- Construct D $\in$ NP
- no poly-time algorithm solves D
  - for every poly time algorithm M exists a string x such that:
    - M(x) = 1 & x $\notin$ D or
    - M(x) = 0 & x $\in$ D

$\Rightarrow$ D not in P

D $\leq_T^p$ SAT $\Rightarrow$ SAT not in P

Diagonalization

How big are the reals ?

- Cantor showed $\mathbb{R}$ not enumerable
- diagonalization
  - given an enumeration of the reals
  - construct real number d not in the enumeration

Diagonalization

reals in some enumeration

	1	2	3	4	5	6	7	8	→
r_1	0.8	1	0	7	7	4	1	5	
r_2	0.3	2	1	4	8	6	7	3	
r_3	0.5	3	9	7	7	9	4	1	
r_4	0.7	6	9	6	5	7	9	4	
r_5	0.8	3	6	8	9	5	1	4	
r_6	0.8	7	9	3	4	6	0	2	
r_7	0.9	8	5	2	5	3	1	3	
r_8	0.9	9	3	1	2	3	0	4	
↓									

d= 0.9 3 0 7 0 7 2 5 ⋯

i^{th} digit of d is i^{th} entry of diagonal +1

Diagonalizing out of P

Diagonalization (2)

polynomial time algorithms

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	→
M_1	0	1	0	0	0	0	1	1	
M_2	1	1	1	0	0	0	0	1	
M_3	0	1	1	0	0	0	1	1	
M_4	0	0	1	0	1	0	0	0	
M_5	1	0	0	1	1	0	1	0	
M_6	1	1	0	1	1	0	0	1	
M_7	0	1	0	1	0	0	1	1	
M_8	0	1	1	1	0	0	0	1	
↓									

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
$\in D$	$\notin D$	$\notin D$	$\in D$	$\notin D$	$\in D$	$\notin D$	$\notin D$

x_i in D if and only if $M_i(x_i)=0$

Diagonal Language

$$D = \{x_i \mid M_i(x_i) = 0\}$$

i^{th} poly-time algorithm/machine

D $\notin$ P, every poly-time machine errs on some input

D $\in$ NP ??   probably not, but

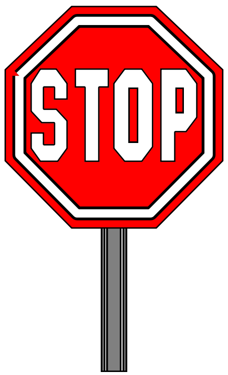
D $\in$ time($n^{\log n}$), quasi polynomial time

with more time can compute more

More Bad News

- Relativization (Oracles):
  - Exists oracle A: $\mathbf{P}^A = \mathbf{NP}^A$
  - (Exists oracle B: $\mathbf{P}^B \neq \mathbf{NP}^B$)

Proof technique should not relativize



Diagonalization and most other techniques we know relativize

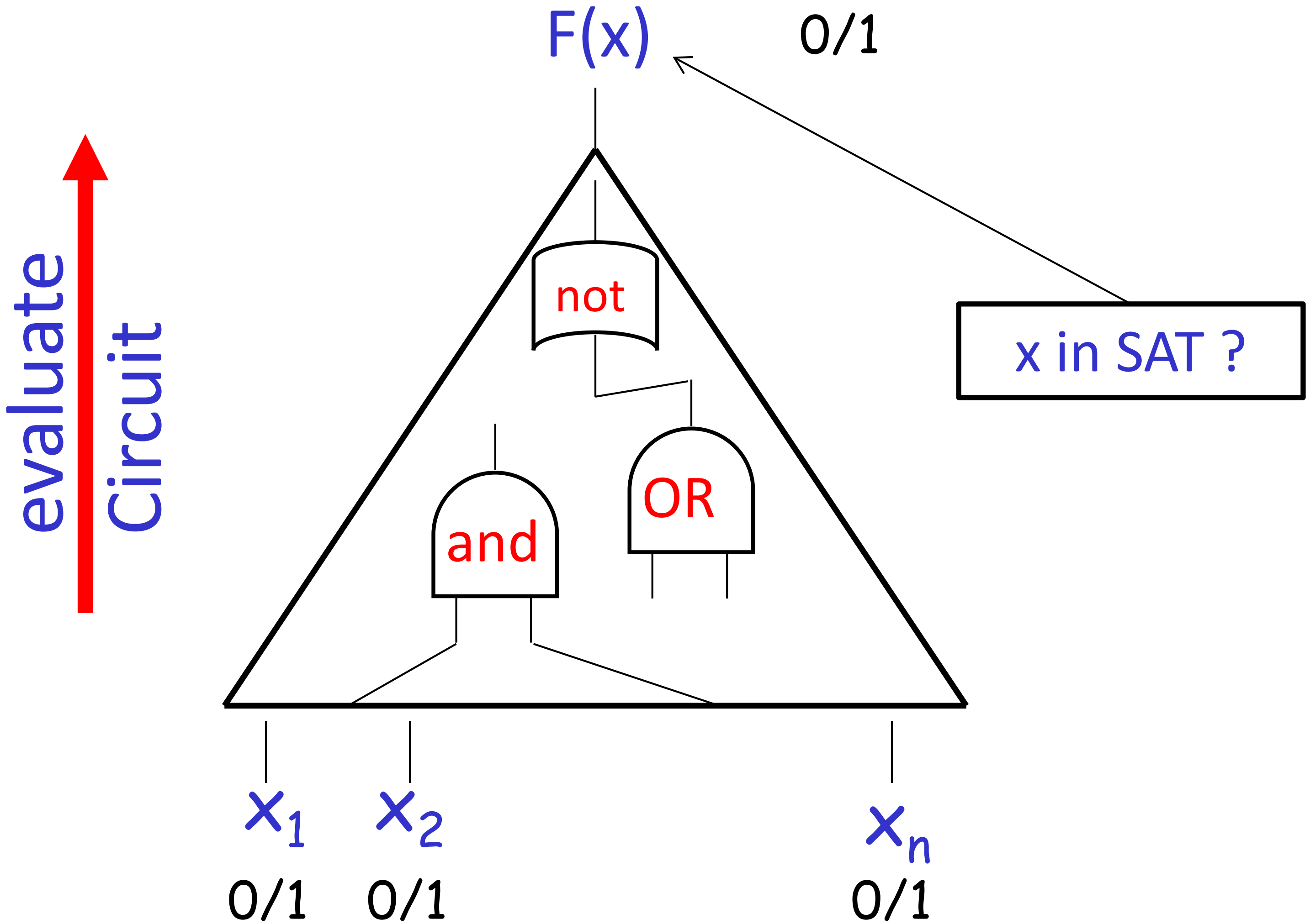
Try something easier

- Study weaker models of computation and develop new lower bound techniques
  - Circuits with small depth
  - Monotone circuits
  - Decision Trees
  - Branching Programs
- The weaker the model the better the lower bounds!

Simple model:

Circuits

Circuit Model of Computation



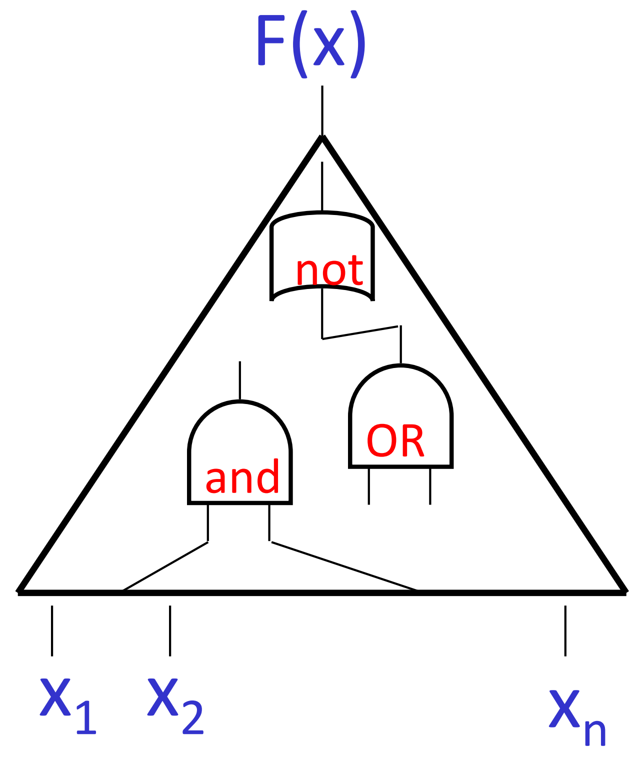
Size of the Circuit

1. most important:

   number of gates

2. Depth of the circuit

   Parallel time of computation



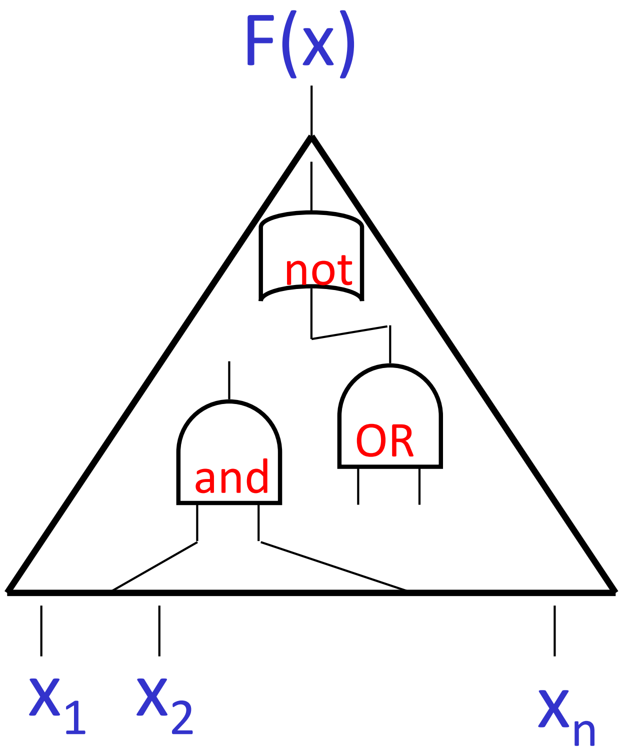
Constant Depth

depth is constant
size is polynomial

AC^0

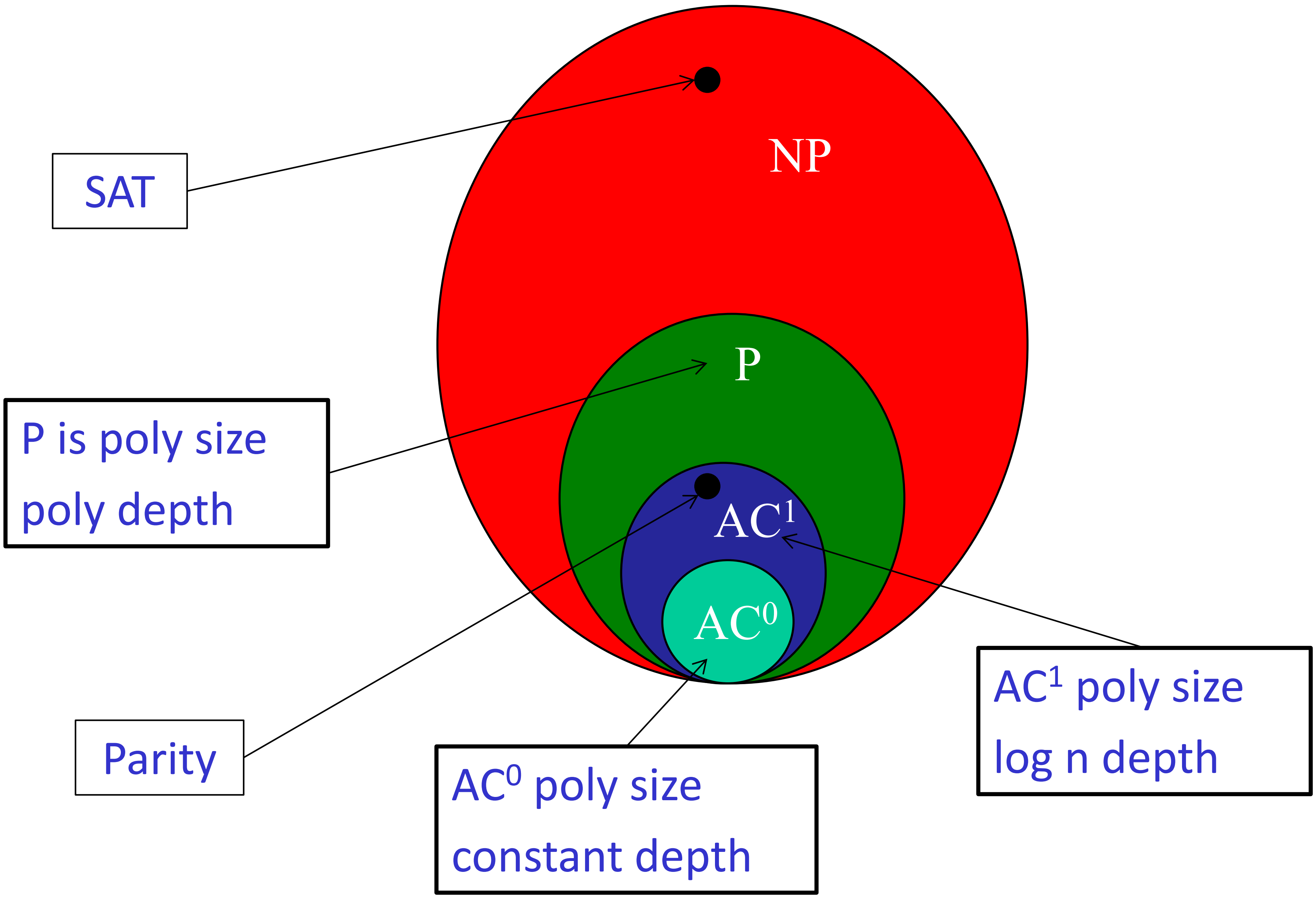
compute parity:
$F(x) = x_1 + x_2 + \cdots + x_n \bmod 2$

Theorem
parity requires $2^{n^{1/d}}$ size circuits
of depth d



Note: d = log n bound is meaningless

NP = AC^1 ?



SAT
NP
P
P is poly size
poly depth
AC1
AC0
Parity
AC0 poly size
constant depth
AC1 poly size
log n depth

natural proofs another hurdle?

- proof technique that shows parity not in AC^0 likely won't work to separate P from NP
- these proofs fit in a framework called natural proofs

> Theorem
>
> if one-way functions exist then
>
> natural proofs can't separate P and NP

Approaches

- Structural approach using eg. autoreducibility
- Combinatorial approach
- Algebraic, degrees of multivariate polynomials
- Geometric Complexity
  - algebraic geometry
  - representation theory
- Communication complexity

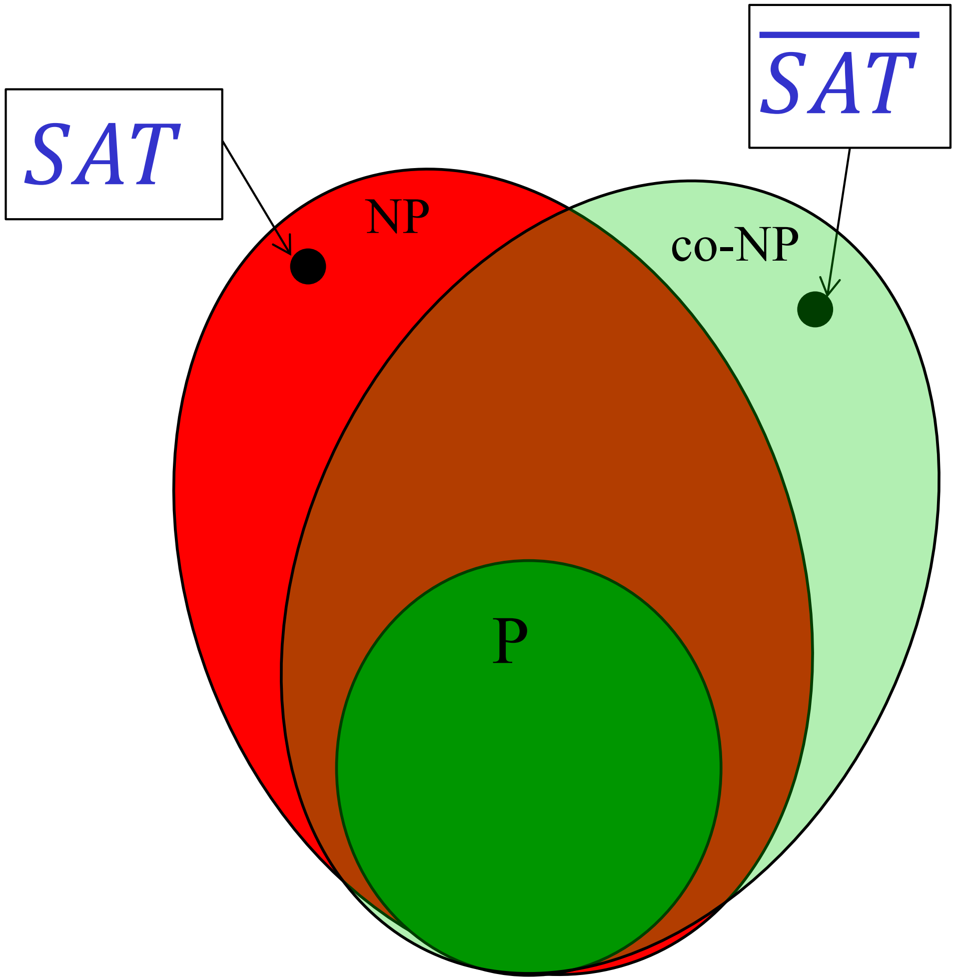
P vs NP & Cryptography

- computational hardness guarantees security of cryptographic protocols
  - factoring, discrete logarithm
  - lattice problems
  - learning problems
- one-way functions
  - compute f(x) quickly
  - hard to invert
- if P=NP then no cryptography

efficient on quantum computer

Beyond NP

coNP



$L \in coNP \leftrightarrow \overline{L} \in NP$

$x \in L: \forall y\ P(x, y) = 0$

$\phi \in \overline{SAT}$: $\phi(x)$ has no satisfying assignment

Tangram:
puzzle has no solution

Polynomial Time Hierarchy

first level

$L \in NP$:

$x \in L: \exists y\, P(x, y) = 1$

Σ_1^p

$L \in coNP$:

$x \in L: \forall\, P(x, y) = 0$

Π_1^p

second level

$L \in \Sigma_2^p$:

$x \in L: \exists y \forall z\, P(x, y, z) = 1$

$L \in \Pi_2^p$:

$x \in L: \forall y \exists z\, P(x, y, z) = 0$

Circuit Minimization (CM):

given circuit A $\exists$ circuit $B < A$ $\forall x: A(x) = B(x)$

CM is Σ_2^p-complete

Polynomial Time Hierarchy

first level

$L \in NP$:

$x \in L: \exists y\, P(x, y) = 1$

Σ_1^p

$L \in coNP$:

$x \in L: \forall\, P(x, y) = 0$

Π_1^p

second level

$L \in \Sigma_2^p$:

$x \in L: \exists y \forall z\, P(x, y, z) = 1$

$L \in \Pi_2^p$:

$x \in L: \forall y \exists z\, P(x, y, z) = 0$

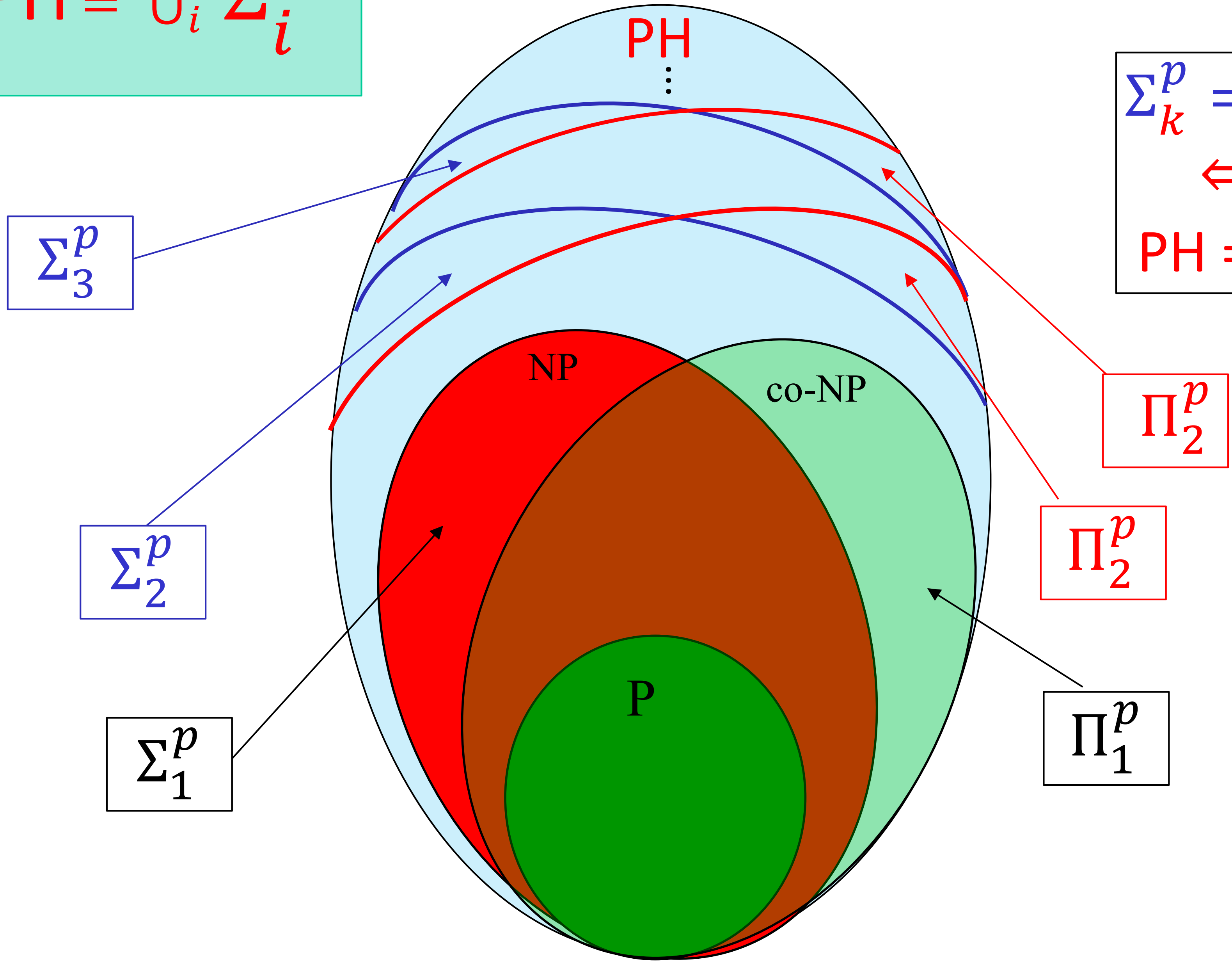
$L \in \Sigma_3^p$:

$x \in L: \exists y_1 \forall y_2 \exists y_3\, P(x, y_1, y_2, y_3) = 1$

PH$= \cup_i \Sigma_i^p$

$$PH = \bigcup_i \Sigma_i^p$$

Believe: PH is infinite

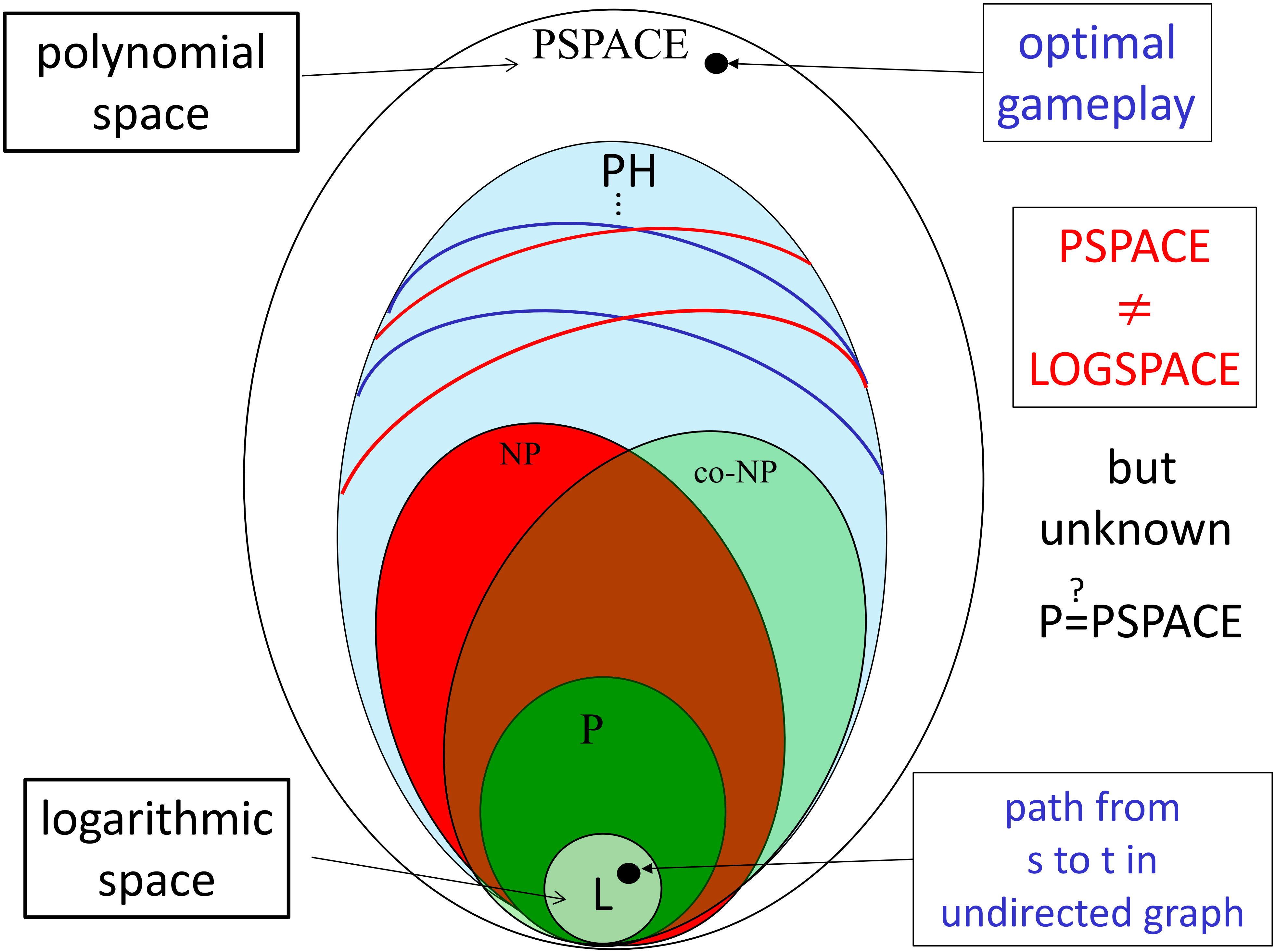


$$\Sigma_k^p = \Pi_k^p \Leftrightarrow PH = \Sigma_k^p$$


Space Complexity
to boldly go where no man has gone before

Space Complexity

- Time of a computation not only resource that matters
- Space or memory the computer uses
- L: logarithmic space usage
  - models web applications
- PSPACE: polynomial space usage
  - natural class with natural complete problems

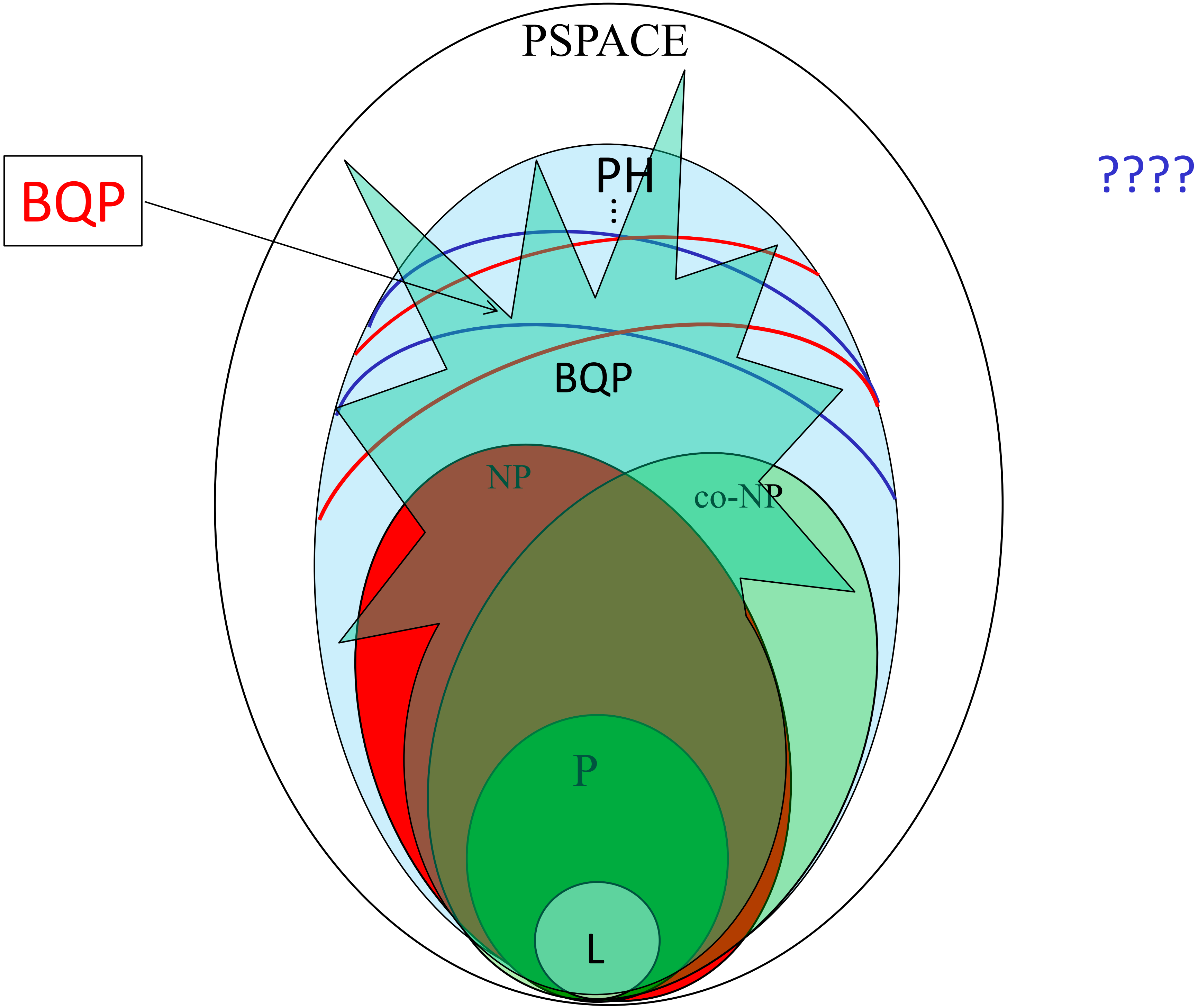

polynomial space
PSPACE
optimal gameplay
PH
PSPACE ≠ LOGSPACE
but unknown
P ≟ PSPACE
NP
co-NP
P
L
logarithmic space
path from s to t in undirected graph

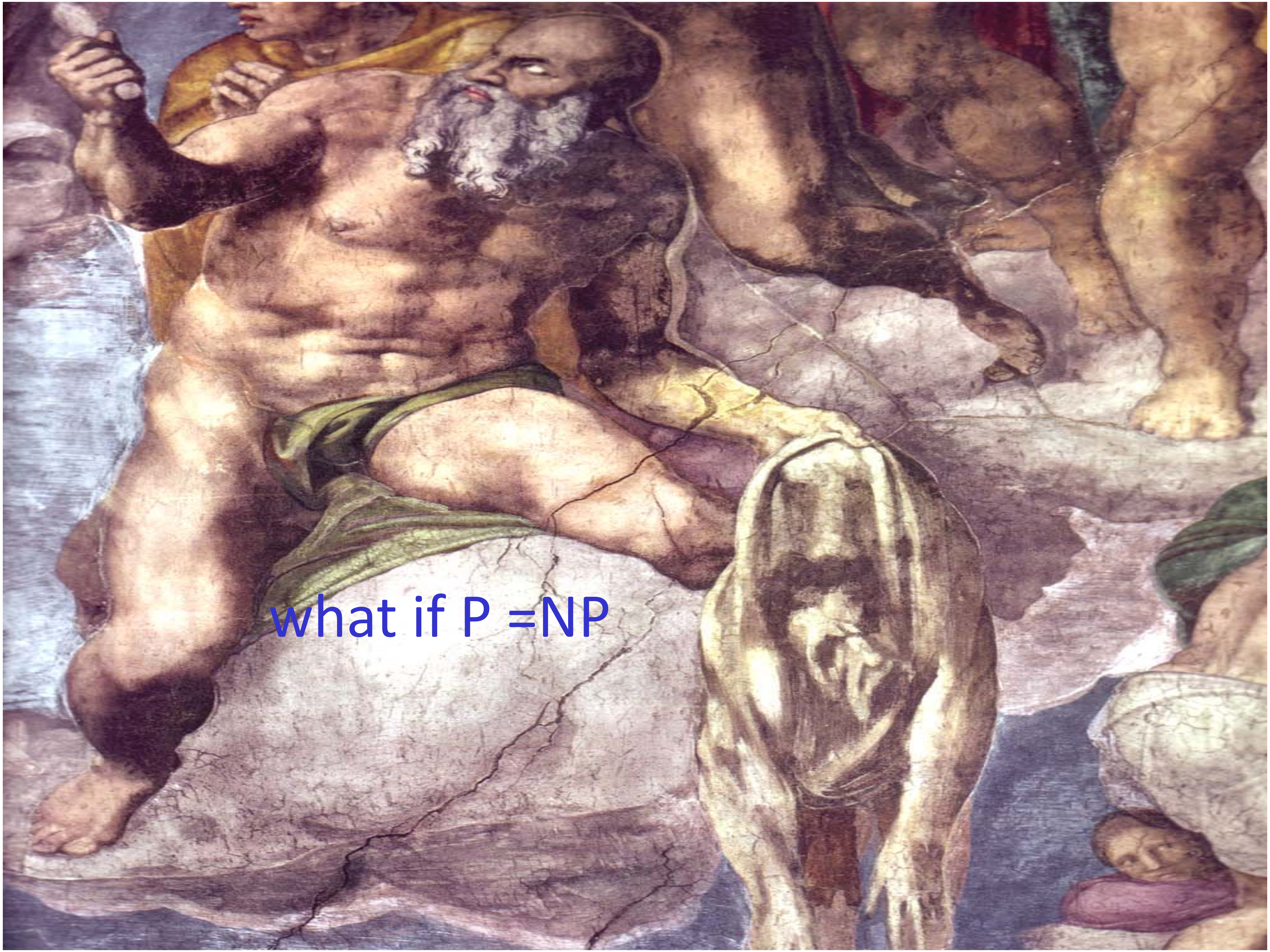
Quantum Polynomial Time

- New Complexity Class
- Problems that can be efficiently computed on a quantum computer

BQP

- Where does BQP sit in the complexity landscape?


BQP
PSPACE
PH
⋮
BQP
NP
co-NP
P
L
????


what if P =NP

P=NP

- P=NP, but the proof does not give us an algorithm
- P=NP, but algorithm for SAT runs in time $n^{1000000}$
- P=NP, but algorithm for SAT runs in time $2^{100}n$
- P=NP, and algorithm for SAT runs in time n^2

n^2 algorithm for SAT

- Wonderful!!!
  - computing ground states of Hamiltonians
  - protein folding problem solved
  - artificial Intelligence takes really off
  - optimal scheduling
  - computational learning theory
  - weather prediction improves

n^2 algorithm for SAT

- For mathematics
  - can find proofs to theorems, provided they have short proofs
  - can simply ask computer whether theorem/conjecture is true/false
  - mathematics will change dramatically
  - quickly solve the other 5 remaining Clay problems

Summary

- P versus NP central, not just in mathematics and computer science but also in physics, biology, chemistry, cryptography etc.
- Not clear how to attack it, several obstacles: relativization, natural proofs, algebraization
- Much simpler questions are still way out of reach
- If P=NP, the world would drastically change, with lots of fantastic application, but no privacy (cryptography).

Schedule

2) P, NP, reductions, co-NP
3) Cook-Levin Thm:3-SAT is NP-complete, Decision vs Search
4) Diagonalization, time hierarchies
5) Relativization
6) Space complexity, PSPACE, L, NL
7) The polynomial hierarchy
8) Circuit complexity, the Karp-Lipton Theorem
9) Parity not on AC^0
10) Probabilistic algorithms
11) BPP, circuits and polynomial hierarchy
12) Interactive proofs, Graph-Isomorphism problem
13) IP = PSPACE

https://exploration.open.wolframcloud.com/objects/exploration/Turing.nb