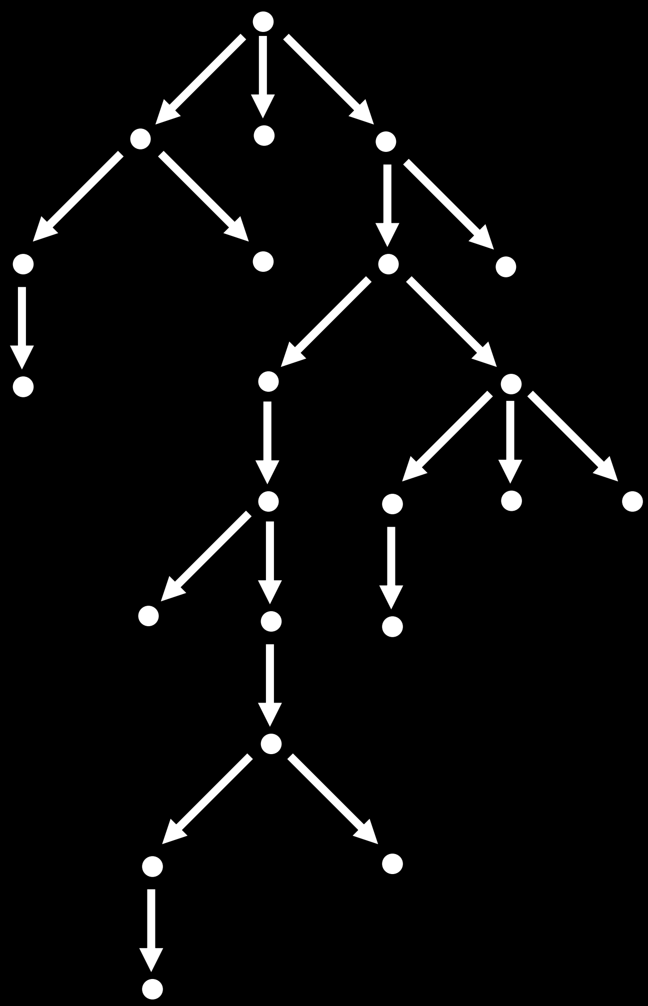
6.045

Lecture 18:
More Friends of NP,
Oracles in Complexity Theory

Definition: coNP = { L | ¬L ∈ NP }

What does a coNP computation look like?



In NP algorithms, we can use a "guess" instruction in pseudocode:

Guess string y of $|x|^k$ length…

and the machine **accepts** iff some y leads to an accept state

In coNP algorithms, we can use a "try all" instruction:

Try all strings y of $|x|^k$ length…

and the machine **accepts** iff every y leads to an accept state

Definition: A language B is coNP-complete if

1. B $\in$ coNP

2. For every A in coNP, there is a polynomial-time reduction from A to B
(B is coNP-hard)

Can use $A \leq_P B \iff \neg A \leq_P \neg B$
to turn NP-hardness into co-NP hardness

UNSAT = { ϕ | ϕ is a Boolean formula and *no* variable assignment satisfies ϕ }

Theorem: UNSAT is coNP-complete

TAUTOLOGY = { ϕ | ϕ is a Boolean formula and *every* variable assignment satisfies ϕ }
= $\{\phi \mid \neg\phi \in$ UNSAT$\}$

Theorem: TAUTOLOGY is coNP-complete

$NP \cap coNP = \{ L \mid L \text{ and } \neg L \in NP \}$

$L \in NP \cap coNP$ means that both $x \in L$ *and* $x \notin L$ have "nifty proofs"

Is $P = NP \cap coNP$?

THIS IS AN OPEN QUESTION!

An Interesting Problem in NP ∩ coNP

FACTORING
= { (m, n) | m > n > 1 are integers written in binary,
& there is a prime factor p of m where n ≤ p < m }

Theorem: FACTORING ∈ NP ∩ coNP

Theorem: If FACTORING ∈ P, then there is a polynomial-time algorithm which, given an integer n, outputs either "n is PRIME" or a prime factor of n.

PRIMES = {n | n is a prime number written in binary}

Theorem (Pratt '70s): PRIMES $\in$ NP $\cap$ coNP

PRIMES is in P

Manindra Agrawal, Neeraj Kayal and Nitin Saxena

Ann. of Math. Volume 160, Number 2 (2004), 781-793.

Abstract

We present an unconditional deterministic polynomial-time algorithm that determines whether an input number is prime or composite.

FACTORING
$= \{ (n, k) \mid n > k > 1$ are integers written in binary,
there is a prime factor p of n where $k \le p < n \}$

Theorem: FACTORING $\in$ NP $\cap$ coNP

Proof: (1) FACTORING $\in$ NP

A prime factor p of n such that $p \ge k$ is a proof that (n, k) is in FACTORING
(can check primality in P*, can check* p *divides* n *in* P*)*

(2) FACTORING $\in$ coNP

The prime factorization $p_1^{E1} \dots p_m^{Em}$ of n is a proof that (n, k) is not in FACTORING:

Verify each p_i is prime in P, and that $p_1^{E1} \dots p_m^{Em} = n$

Verify that for all $i=1,\dots,m$ that $p_i < k$

FACTORING
= { (n, k) | n > k > 1 are integers written in binary,
there is a prime factor p of n where $k \leq p < n$ }

Theorem: If FACTORING $\in$ P, then there is a polynomial-time algorithm which, given an integer n, outputs either "n is PRIME" or a prime factor of n.

Idea: Binary search for the prime factor!

Given binary integer n, initialize an interval [2,n].

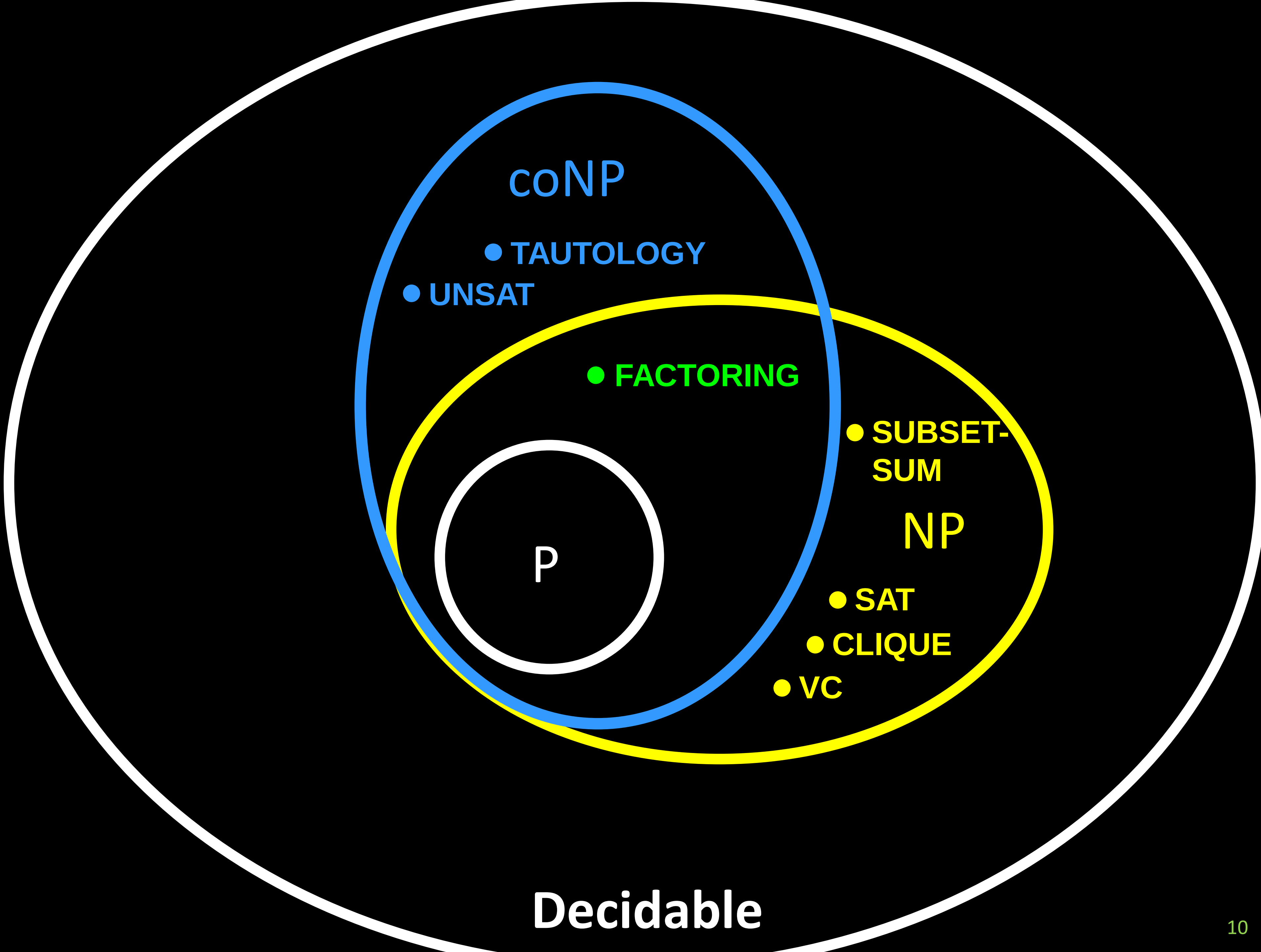
If (n, 2) is not in FACTORING then output "PRIME"

If $(n, \lceil n/2 \rceil)$ is in FACTORING then

shrink interval to $[\lceil n/2 \rceil, n]$ (set $k := \lceil 3n/4 \rceil$)

else, shrink interval to $[2, \lceil n/2 \rceil]$ (set $k := \lceil n/4 \rceil$)

Keep picking k to halve the interval after each (n,k) call to FACTORING. Takes O(log n) calls to FACTORING!


coNP
TAUTOLOGY
UNSAT
FACTORING
SUBSET-SUM
NP
P
SAT
CLIQUE
VC
Decidable

NP-complete problems:

SAT, 3SAT, CLIQUE, VC, SUBSET-SUM, …

coNP-complete problems:

UNSAT, TAUTOLOGY, NOHAMPATH, …

(NP ∩ coNP)-complete problems:

Nobody knows if they exist!

P, NP, coNP can be defined in terms of specific machine models, and for every possible machine we can give a simple encoding of it.

NP ∩ coNP is *not* known to have a corresponding machine model!

Polynomial Time With Oracles

***We do not condone smoking. Don't do it. It's bad. Kthxbye**

Oracle Turing Machines

Polynomial time

q_{YES}

Is formula F in SAT?

yes

NP

A	N	P	U	T					

INFINITE TAPE

Oracle Turing Machines

An oracle Turing machine M^B is equipped with a set $B \subseteq \Gamma^*$ to which a TM M may ask membership queries on a special "oracle tape"
[Formally, M^B enters a special state $q_?$]

and the TM receives a query answer in one step
[Formally, the transition function on $q_?$ is defined in terms of the *entire oracle tape*:
if the string y written on the oracle tape is in B, then state $q_?$ is changed to q_{YES}, otherwise q_{NO}]

This notion makes sense even when M runs in *polynomial time* and B is *not* in P!

How to Think about Oracles?

Think in terms of Turing Machine pseudocode!

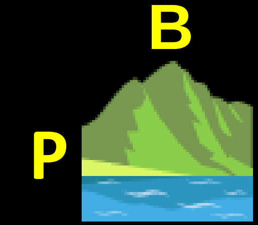
An oracle Turing machine M with oracle $B \subseteq \Gamma^*$ lets you include the following kind of branching instructions:

"if (z in B) then <do something>
else <do something else>"

where z is some string defined earlier in pseudocode. By definition, the oracle TM can always check the condition (z in B) in one step

Some Complexity Classes With Oracles

Let B be a language.



P^B = { L | L can be decided by some *polynomial-time* TM with an oracle for B }

P^{SAT} = the class of languages decidable in polynomial time with an oracle for SAT

P^{NP} = the class of languages decidable by *some* polynomial-time oracle TM with an oracle for *some* B in NP

Is $P^{SAT} \subseteq P^{NP}$?

Yes! By definition...

Is $P^{NP} \subseteq P^{SAT}$?

Yes!

Every NP language can be reduced to SAT!

Let M^B be a poly-time TM with oracle $B \in NP$.
We define N^{SAT} that simulates M^B step for step.
When the sim of M^B makes query w to oracle B,
N^{SAT} reduces w to a formula ϕ_w in poly-time,
then calls its oracle for SAT on ϕ_w

Is $NP \subseteq P^{NP}$?

Yes!

Just ask the oracle for the answer!

For every $L \in NP$ define an oracle TM M^L which asks the oracle if the input is in L, then outputs the answer.

Is coNP $\subseteq$ P^{NP}?

Yes!

Again, just ask the oracle for the answer!

For every L $\in$ coNP we have $\neg$L $\in$ NP

Define an oracle TM M$^{\neg L}$ which asks the oracle if the input is in $\neg$L

accept if the answer is no,

reject if the answer is yes

In general, P^{NP} = P^{coNP} and P^{SAT} = P^{UNSAT}

P^B = { L | L can be decided by a polynomial-time TM with an oracle for B }

Suppose B is in P.

Is $P^B \subseteq P$?

Yes!

For every poly-time TM M with oracle $B \in P$, we can simulate each query z to oracle B by simply running a polynomial-time decider for B.

The resulting machine runs in polynomial time!

P^{NP} = the class of languages decidable by some polynomial-time oracle TM M^B for some B in NP

Informally: P^{NP} is the class of problems you can solve in polynomial time, assuming a SAT solver which gives you answers quickly

P^{NP} = the class of languages decidable by some polynomial-time oracle TM M^B for some B in NP

Informally, P^{NP} is the class of problems you can solve in polynomial time, if SAT solvers work

A problem in P^{NP} that looks harder than SAT or TAUT:

FIRST-SAT = { (ϕ, i) | $\phi \in$ SAT and the i-th bit of the lexicographically first SAT assignment of ϕ is 1}

Using polynomially many calls to SAT, we can compute the lex. first satisfying assignment

Theorem FIRST-SAT is P^{NP}-complete

NP^B = { L | L can be decided by a polynomial-time nondeterministic TM with an oracle for B }

$coNP^B$ = { L | L can be decided by a poly-time co-nondeterministic TM with an oracle for B }

Is $NP = NP^{NP}$?

Is $coNP^{NP} = NP^{NP}$?

THESE ARE OPEN QUESTIONS!

It is believed the answers are NO …

Logic Minimization is in $coNP^{NP}$

Two Boolean formulas ϕ and ψ over the variables $x_1,\ldots,x_n$ are equivalent if they have the same value on every assignment to the variables

Are x and $x \vee x$ equivalent? Yes

Are x and $x \vee \neg x$ equivalent? No

Are $(x \vee \neg y) \wedge \neg(\neg x \wedge y)$ and $x \vee \neg y$ equivalent? Yes

A Boolean formula ϕ is minimal if
no *smaller* formula is equivalent to ϕ
(count number of $\vee$, $\wedge$, $\neg$, and variable occurrences)

MIN-FORMULA = $\{ \phi \mid \phi \text{ is minimal} \}$

Theorem: MIN-FORMULA $\in$ coNP^{NP}

Proof:

Define NEQUIV = { (ϕ, ψ) | ϕ and ψ are not equivalent }

Observation: NEQUIV $\in$ NP (Why?)

Here is a coNP^{NEQUIV} machine for MIN-FORMULA:

Given a formula ϕ,

Try all formulas ψ such that ψ is smaller than ϕ.

If $((\phi, \psi) \in$ NEQUIV) then *accept* else *reject*

MIN-FORMULA is not known to be in coNP or NP^{NP}

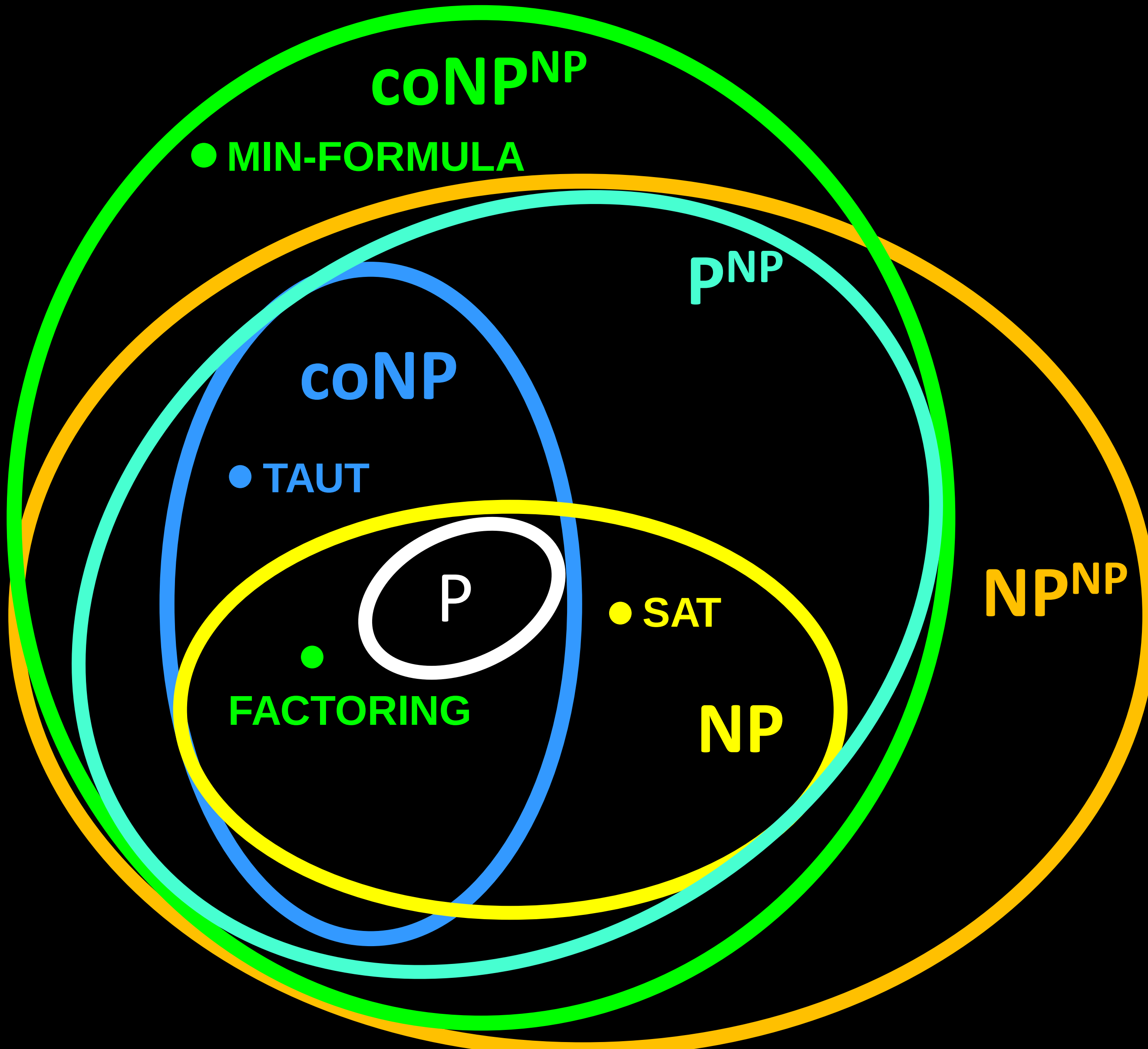
The Difficulty of Formula Minimization

MIN-CNF-FORMULA = { ϕ | ϕ is CNF and is minimal }

Theorem: MIN-CNF-FORMULA is $\mathrm{coNP}^{\mathrm{NP}}$-complete

Proof: Beyond the scope of this course…

Note: We don't know if MIN-FORMULA is $\mathrm{coNP}^{\mathrm{NP}}$ complete!



coNP^NP
MIN-FORMULA
P^NP
coNP
TAUT
P
SAT
FACTORING
NP
NP^NP

Oracles and P vs NP

Everything about TMs we have proved
in this class also works for TMs with arbitrary oracles.

Theorem [Baker, Gill, Solovay '75]:

(1) There is an oracle B where $P^B = NP^B$

(2) There is an oracle A where $P^A \neq NP^A$

See Sipser 9.2

Moral: Any proof technique that also works to
Turing Machines with arbitrary oracles
won't be able to resolve P versus NP!

THE "RELATIVIZATION BARRIER"