SAT

SAT: given Boolean formula φ (uses variables, and, or, not, ( )), determine if φ has a satisfying assignment (assignment of T/F to vars to make φ T)

SAT $\in$ NP: SAT-VERIFY(φ, A) (A: assignment of T/F to each variable)

```
substitute T/F into φ according to A        poly
result ← value of resulting expression      poly
if result = T then return YES
else           return NO
```

size = # variables in φ = polynomial in size of φ

If φ has a satisfying assignment A, then SAT-VERIFY(φ, A) = YES

If φ has no satisfying assignment, then SAT-VERIFY(φ, A) = NO for all A

SAT is NP-complete: later

CONTRADICTION

uses variables, and, or, not, ( )

SAT: given Boolean formula φ, determine if φ has a satisfying assignment

assignment of T/F to vars to make φ T

CONTRADICTION: given Boolean formula φ, determine if all truth assignments make φ false

Ex: $(x \vee y) \wedge (\sim x \wedge \sim y)$ is a contradiction

? unknown

CONTRADICTION $\in$ P $\longrightarrow$ SAT $\in$ P

(and hence P=NP)

SAT(φ)
return ~CONTRADICTION(φ)

CONTRADICTION

SAT: given Boolean formula φ (uses variables, and, or, not, ( )), determine if φ has a satisfying assignment (assignment of T/F to vars to make φ T)

CONTRADICTION: given Boolean formula φ, determine if all truth assignments make φ false

Ex: $(x \vee y) \wedge (\neg x \wedge \neg y)$ is a contradiction

$\text{CONTRADICTION} \overset{?}{\in} P \longrightarrow \text{SAT} \in P$ (unknown)

(and hence P=NP)

SAT(φ)
return ~CONTRADICTION(φ)

$\text{CONTRADICTION} \overset{?}{\in} NP$ (unknown)

x	y	$(x \vee y) \wedge (\neg x \wedge \neg y)$
T	T	F
T	F	F
F	T	F
F	F	F

2^n rows (not polynomial in size of φ)

But there is a polynomial-time verification algorithm for determining if CONTRADICTION(φ) = NO (if SAT(φ) = YES)

SAT is a function from Boolean formulas to {YES, NO} defined by

$$SAT(\varphi) = \begin{cases} YES & \text{if } \varphi \text{ has a satisfying assignment} \\ NO & \text{otherwise} \end{cases}$$

as a set $SAT = \{ \varphi \mid \varphi \text{ has a satisfying assignment} \}$

CONTRADICTION is a function from Boolean formulas to {YES, NO} defined by

$$CONTRADICTION(\varphi) = \begin{cases} NO & \text{if } \varphi \text{ has a satisfying assignment} \\ YES & \text{otherwise} \end{cases}$$

$$CONTRADICTION = \{ \varphi \mid \varphi \text{ has no satisfying assignment} \}$$

$$= \overline{SAT}$$

co-NP: Problems X such that $\overline{X} \in NP$ CONTRADICTION $\in$ co-NP

co-P: Problems X such that $\overline{X} \in P$

$P = co\text{-}P$

$co\text{-}P \subseteq P$: Let $X \in co\text{-}P$

Then $\overline{X} \in P$ (def co-P)

Create an algorithm for X: $X(x)$ return $\sim \overline{X}(x)$ poly time

So $X \in P$

$P \subseteq co\text{-}P$: Let $X \in P$

Create an algorithm for $\overline{X}$: $\overline{X}(x)$ return $\sim X(x)$ poly time

So $\overline{X} \in P$

Hence $\overline{\overline{X}} = X \in co\text{-}P$ (def co-P)

NP = co-NP? unknown!

$P \subseteq NP$ (from before)

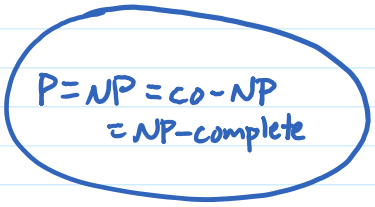
$P \subseteq$ co-NP: Let $X \in P$. Write $\overline{X}$-VERIFY(x,y)
return $\sim X(x)$ poly-time

$\overline{X}(x) = YES \rightarrow \overline{X}\text{-VERIFY}(x,y) = \sim X(x) = \sim NO = YES$ for all y
$\overline{X}(x) = NO \rightarrow \overline{X}\text{-VERIFY}(x,y) = \sim X(x) = \sim YES = NO$ for all y

so $\overline{X} \in NP$ and hence $X \in$ co-NP

If $P = NP$ then $NP =$ co-NP: Suppose $P = NP$
$P =$ co-P (earlier)
$NP =$ co-NP (substitute NP for P)

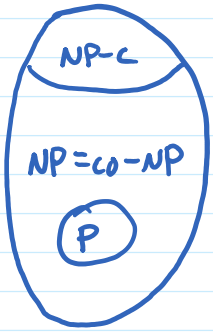
Possibilities

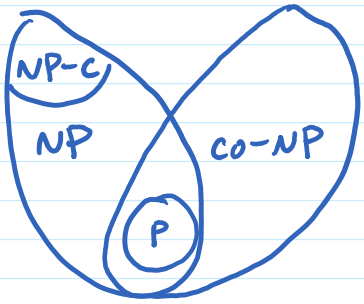


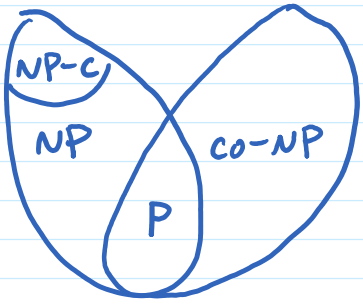
P=NP → All X∈P are NP-complete:

Suppose P=NP
Let X∈P. Then X∈NP. (assumption P=NP)
Let Y∈NP.
Then Y∈P and so some poly-time A solves Y. (P=NP; def. P)
Write algorithm for Y: Y(y)
return A(y) poly-time
+O calls to X

So $Y \leq_p X$ (def $\leq_p$ using algorithm above)
And X is NP-complete (def NP-complete)







No one knows which picture reflects reality!