NP complete:

1) in NP

2) <u>every</u> problem in NP reducible to it.

Transitivity of p-time reduction implies

NP complete iff

1) in NP

2') <u>some</u> NP-complete problem reducible to it.

We need one NP-complete problem to get started.

Sat $\in NP$: proof = satisfying assignment

Graph coloring $\in NP$: proof = coloring

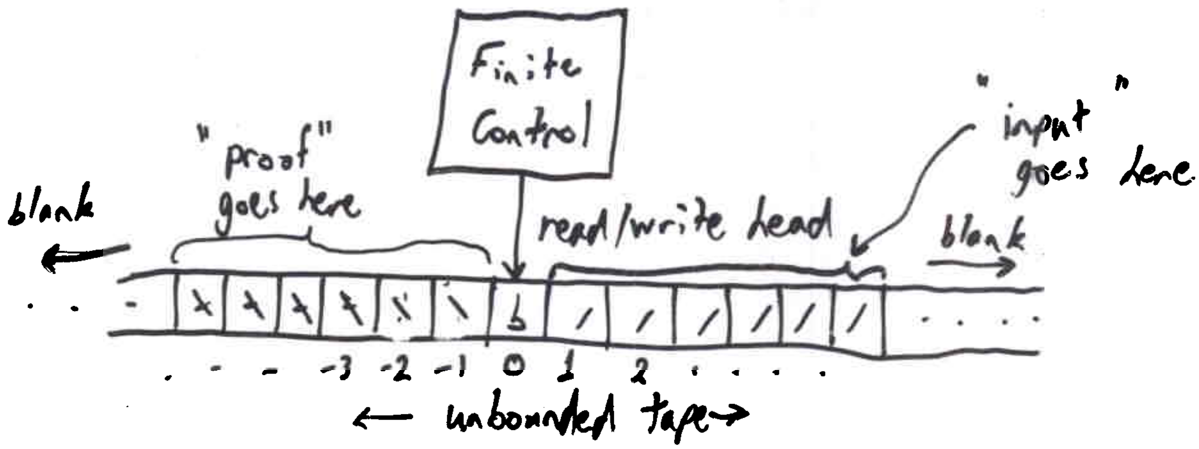
k-clique, k-vertex cover, k-independent set $\in NP$

Tautology $\in$ co-NP ("no" instances have a proof)

Cook-Levin Theorem: Sat is NP-complete.

Given any p-time verifier, construct (in p-time) an instance of Sat s.t. verifier answers "yes" iff formula is satisfiable.

Verifier: Turing Machine



In one step, machine can write a symbol, move head one position, change state.

What to do is based on state, symbol read.

Fixed # of states, fixed # of tape symbols, including blank; start state, "yes" state, ("no" state)

Explicitly given polynomial time bound $p(n)$.

Input (of size n) is a "yes" instance iff
for some "proof" and given input, the machine reaches "yes" state within $p(n)$ steps from start state.

Must construct a formula that is satisfiable iff this happens.

Note: input is specified, proof is not (nondeterministic part)

Proof can't exceed length $p(n)$: machine cant get farther in $p(n)$ steps.

Can assume machine loops in "yes" state: if ever in "yes", will be in "yes" at step $p(n)$.

States: $1, \ldots, y$ $\quad$ 1 = start, y = yes

Symbols: $1, \ldots, z$ $\quad$ 1 = blank

Tape cells, $-p(n), \ldots, 0, \ldots, p(n)$

Time: $0, 1, \ldots, p(n)$

Variables for formula:

h_{it}: true if head on tape cell i at time t
$-p(n) \le i \le p(n)$, $0 \le t \le p(n)$

s_{jt}: true if state j at time t
$1 \le j \le y$, $0 \le t \le p(n)$

c_{ikt}: true if tape cell i holds symbol k at time t
$-p(n) \le i \le p(n)$, $1 \le k \le z$, $0 \le t \le p(n)$

What does the formula need to say?

At most one state, head position, and symbol per cell at each time:

$(\bar{h}_{it} \vee \bar{h}_{i't})$ $i \neq i'$, all t

$(\bar{s}_{jt} \vee \bar{s}_{j't})$ $j \neq j'$, all t

$\bar{c}_{ikt} \vee \bar{c}_{ik't})$ $k \neq k'$, all i, all t

Correct initial state, head position, and tape contents:

$$h_{00} \wedge s_{10} \wedge c_{010} \wedge c_{1k_1 0} \wedge c_{2k_2 0} \wedge \cdots \wedge c_{nk_n 0}$$

$$\wedge\, c_{(n+1)10} \wedge \cdots \wedge c_{p(n)10}$$

Input is $k_1 k_2 \ldots k_n$, rest of right side of tape is blank

Correct final state: $s_{y\,p(n)}$

Correct transitions:

E.g. if machine in state j reads k, it then writes k', moves head right, and changes to state j':

$$s_{jt} \wedge h_{it} \wedge c_{ikt} \supset s_{j'\,t+1} \wedge h_{i+1\,t+1} \wedge c_{ik'\,t+1}$$

($\supset$ = "implies")   (for each i, t)

$$h_{it} \wedge c_{i'kt} \supset c_{i'k\,t+1} \quad (\text{for } i \neq i', \text{ each } k, t)$$

(unread tape cells are unaffected)

CNF?

$$(x \wedge y \wedge z) \supset (a \wedge b \wedge c)$$

$$\Rightarrow$$

$$((x \wedge y \wedge z) \supset a)$$
$$\wedge$$
$$((x \wedge y \wedge z) \supset b)$$
$$\wedge$$
$$((x \wedge y \wedge z) \supset c)$$

$$\Rightarrow$$

$$(\bar{x} \vee \bar{y} \vee \bar{z} \vee a)$$
$$\wedge$$
$$(\bar{x} \vee \bar{y} \vee \bar{z} \vee b)$$
$$\wedge$$
$$(\bar{x} \vee \bar{y} \vee \bar{z} \vee c)$$

Any proof that gives a "yes" execution gives a satisfying assignment, and vice-versa.

Conclusion: SAT is NP-complete

(and k-coloring, k-clique, k-independent set, k-vertex cover)

Subset Sum is NP-complete

Given n integers, and a target k, is there a subset that sums to exactly k?

$\{2, 5, 6, 8, 9, 12\}$ $k = 31$

yes: 5, 6, 8, 12

(no for $k = 30$)

In NP: subset is proof (verifiable in p-time)

Some NPC problem reducible to subset sum

reduce 3-CNF sat to subset sum

Write numbers base 5

$$\underbrace{(x \vee \bar{y} \vee \bar{z})}_{C_1} \wedge \underbrace{(\bar{x} \vee y \vee z)}_{C_2} \wedge \underbrace{(y \vee \bar{z})}_{C_3}$$

		x	y	z	C_1	C_2	C_3	
	x	1	0	0	1	0	0	
	$\bar{x}$	1	0	0	0	1	0	
	y	0	1	0	0	1	1	← y makes C_2, C_3 true
	$\bar{y}$	0	1	0	1	0	0	
	z	0	0	1	0	1	0	
	$\bar{z}$	0	0	1	1	0	1	← $\bar{z}$ makes C_1, C_2 true
Dummies to get clause colums to sum to 4		0	0	0	1	0	0	
		0	0	0	2	0	0	
		0	0	0	0	1	0	
		0	0	0	0	2	0	
		0	0	0	0	0	1	
		0	0	0	0	0	2	
		1	1	1	4	4	4	$= k$ Required sum

Interpret each row as a base-5 (or base-10) number.

Subset sum has a solution iff formula is satisfiable.