Unit-5
NP hard and NP Complete problems

The Cook-Levin theorem

K. RAGHAVA RAO

Professor in CSE

KL University

krraocse@gmail.com

http://mcadaa.blog.com

Introduction
SAT

- Instance: A Boolean formula.
- Problem: To decide if the formula is satisfiable.

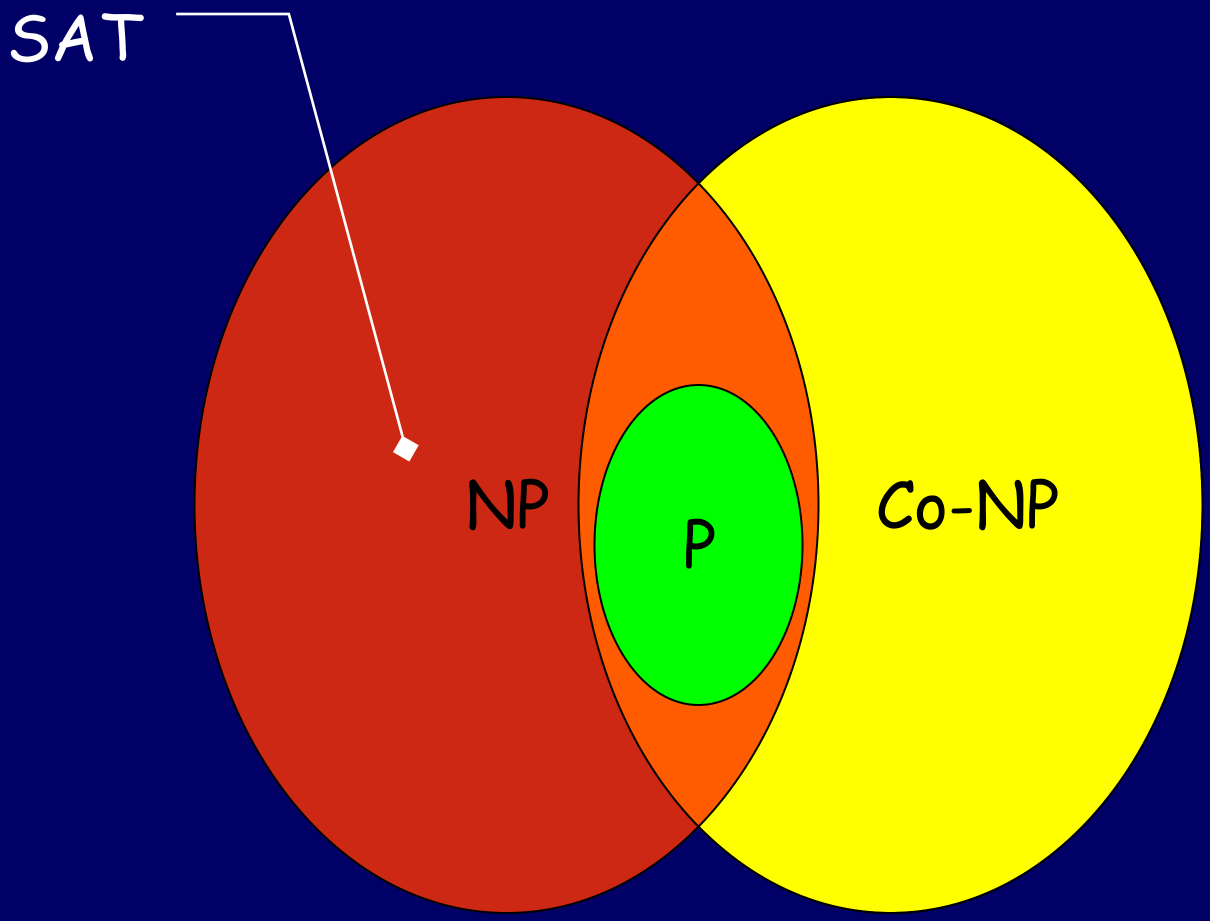
A satisfiable Boolean formula:

$$((F \lor T \lor \neg T) \land \neg F) \lor \neg(T \land T)$$

An unsatisfiable Boolean formula:

$$x_1 \land \neg x_1$$

To Which Time Complexity Class Does SAT Clearly Belong?



SAT is in NP: Non-Deterministic Algorithm

- Guess an assignment to the variables.
- Check the assignment.

$$((F \lor T \lor \neg T) \land \neg F) \lor \neg(T \land T)$$

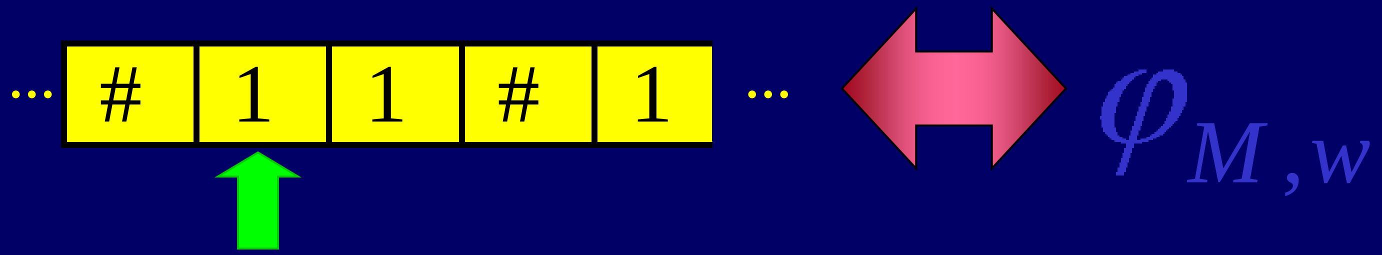
x_1	F
x_2	T
x_3	T

✓

The Cook-Levin Theorem: SAT is NP-Complete

Proof Idea:

For any NP machine M and any input string w, we construct a Boolean formula $\varphi_{M,w}$ which is satisfiable iff M accepts w.



Representing a Computation by a Configurations Table

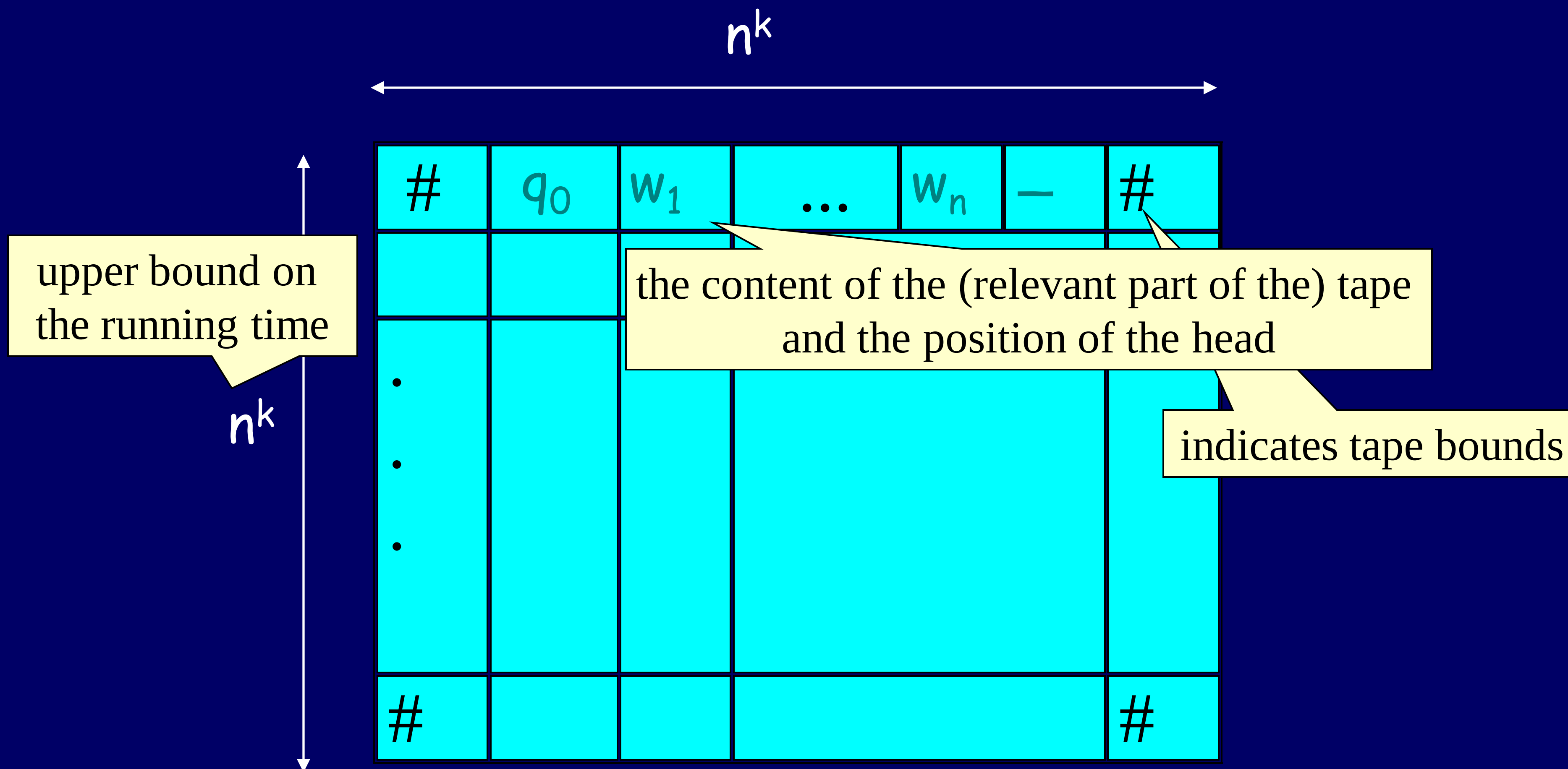


Tableau: Example

- TM:
  - $Q=\{q_0, q_{accept}, q_{reject}\}$
  - $\Sigma=\{1\}$
  - $\Gamma=\{1, \_\}$
  - $\delta(q_0, 1)=\{(q_0, \_, R)\}$
  - $\delta(q_0, \_)=\{(q_{accept}, L)\}$

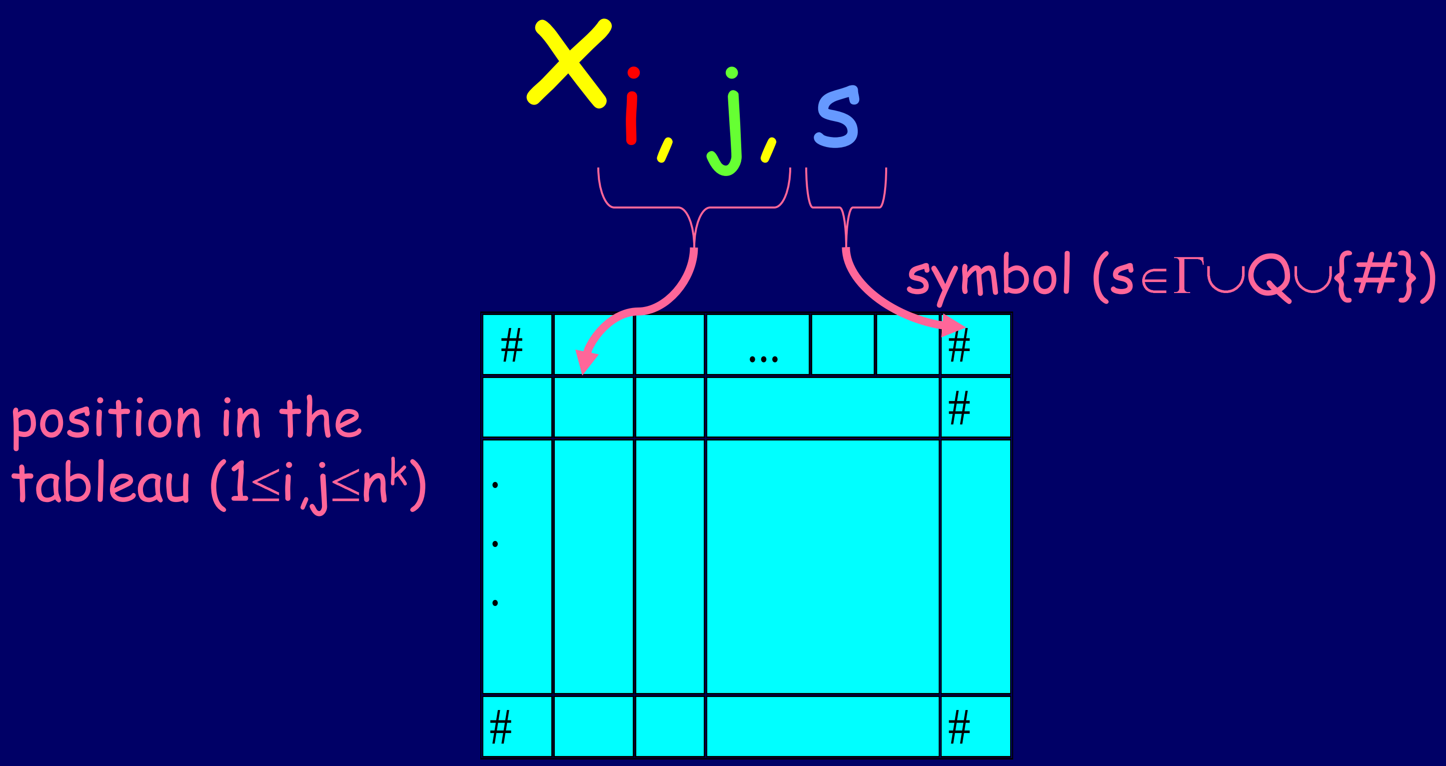
- tableau (input 11)

#	q_0	1	1	_	#
#	_	q_0	1	_	#
#	_	_	q_0	_	#
#	_	q_{acc}	_	_	#

Q: what does this machine compute?

The Variables of the Formula

stands for: "Is s the content of cell (i,j)?"



The Formula φ

$$\varphi_{M,w} = \phi_{cell} \wedge \phi_{start} \wedge \phi_{move} \wedge \phi_{accept}$$

cell content consistency

input consistency

transition legal

machine accepts

Ensuring Unique Cell Content

$$\varphi_{cell} = \bigwedge_{1 \le i,j \le n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{s \ne t \in C} \left(\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}} \right) \right) \right]$$

The (i,j) cell must contain some symbol

It shouldn't contain different symbols.

Note: the length of this formula is polynomial in n.

Ensuring Initial Configuration Corresponds to Input

Observe: we can explicitly state the desired configuration in the first step. Assuming the input string is $w_1w_2\ldots w_n$,

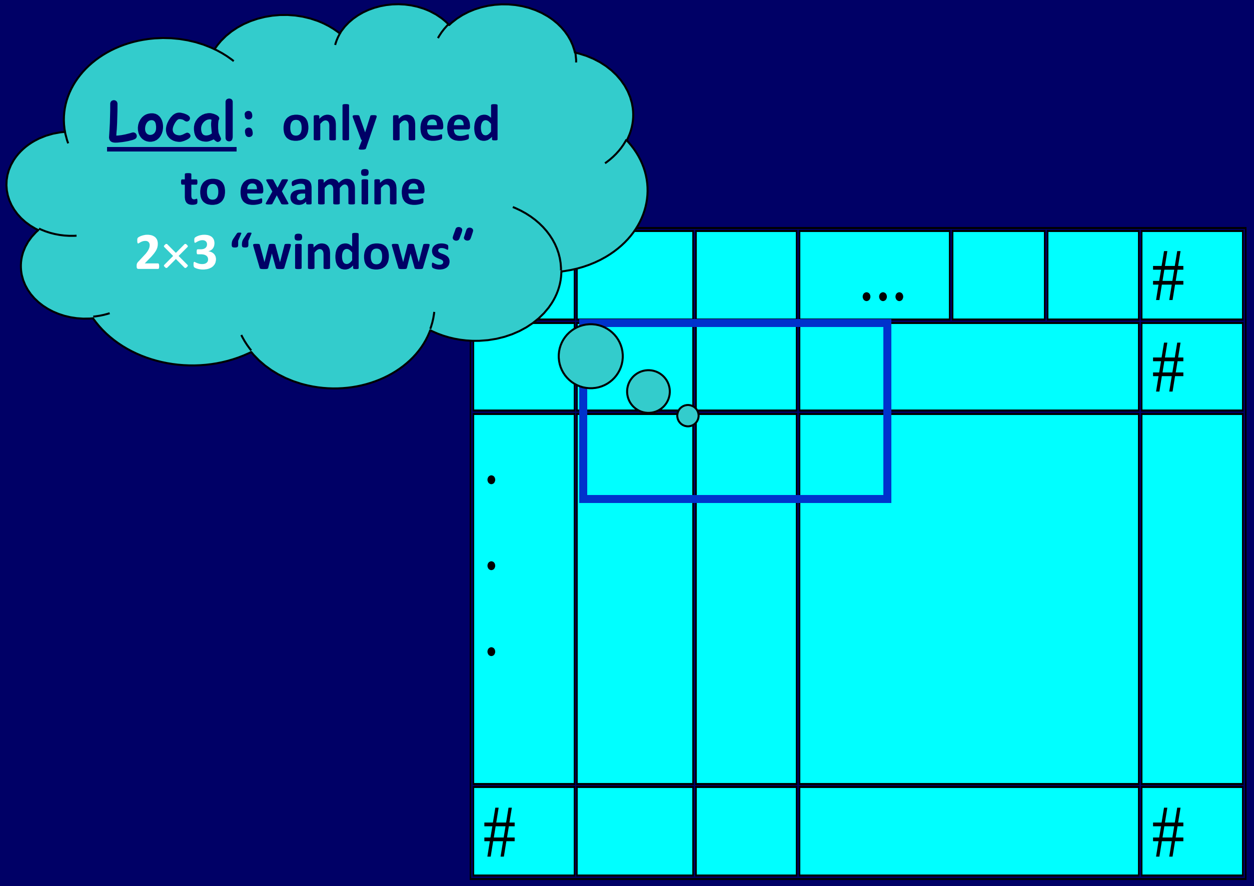
$$\varphi_{start} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,w_1} \wedge \ldots \wedge x_{1,n+3,\_} \wedge \ldots \wedge x_{1,n^k-1,\_} \wedge x_{1,n^k,\#}$$

Ensuring the Computation Accepts

The accepting state is visited during the computation.

$$\varphi_{accept} = \bigvee_{1 \le i, j \le n^k} x_{i,j,q_{accept}}$$

Ensuring Every Transition is Legal



Which Windows are Legal in the Following Example?

- TM:
  - $Q=\{q_0,q_{accept},q_{reject}\}$
  - $\Sigma=\{1\}$
  - $\Gamma=\{1,\_\}$
  - $\delta(q_0,1)=\{(q_0,\_,R)\}$
  - $\delta(q_0,\_)=\{(q_{accept},L)\}$

1	q_0	1
q_{acc}	_	_

_	q_0	1
_	_	q_0

1	q_0	1
1	_	q_0

#	q_0	1
#	_	q_0

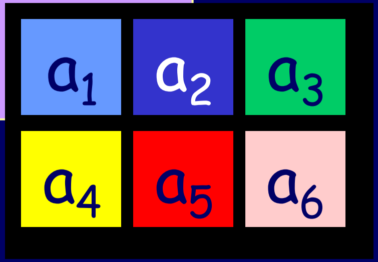
1	q_0	1
1	1	q_0

1	q_0	_
q_{acc}	_	_

Ensuring Every Transition is Legal

$$\varphi_{move} = \bigwedge_{1 \le i, j \le n^k} \bigvee_{a_1, \ldots, a_6} \left(x_{i-1, j, a_1} \wedge \ldots \wedge x_{i+1, j+1, a_6} \right)$$

for any $a_1, \ldots, a_6$ s.t. this is a legal window



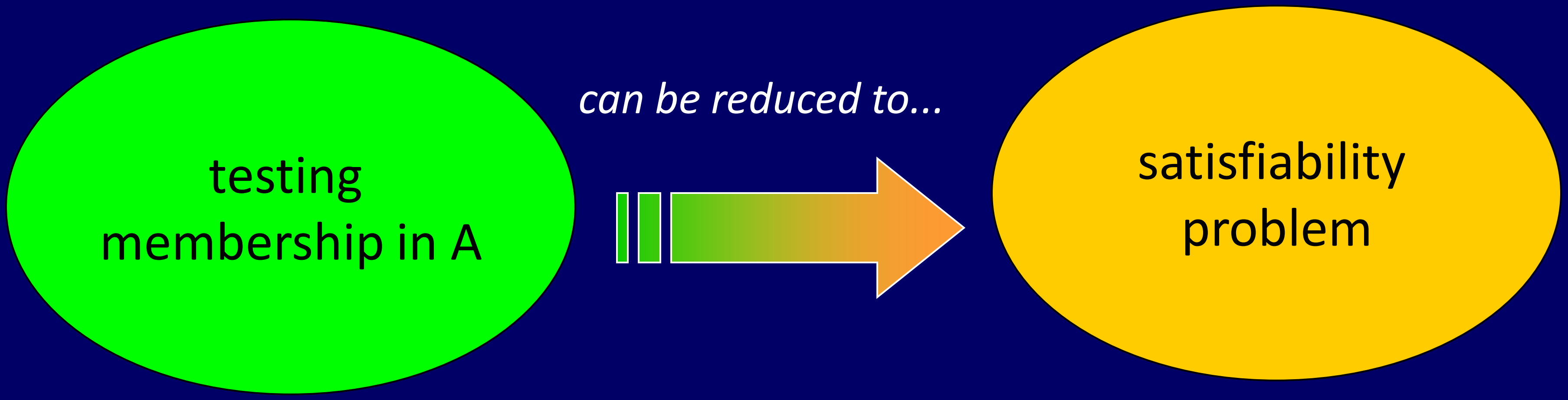
The Bottom Line

$$\varphi_{M,w} = \phi_{cell} \wedge \phi_{start} \wedge \phi_{move} \wedge \phi_{accept}$$

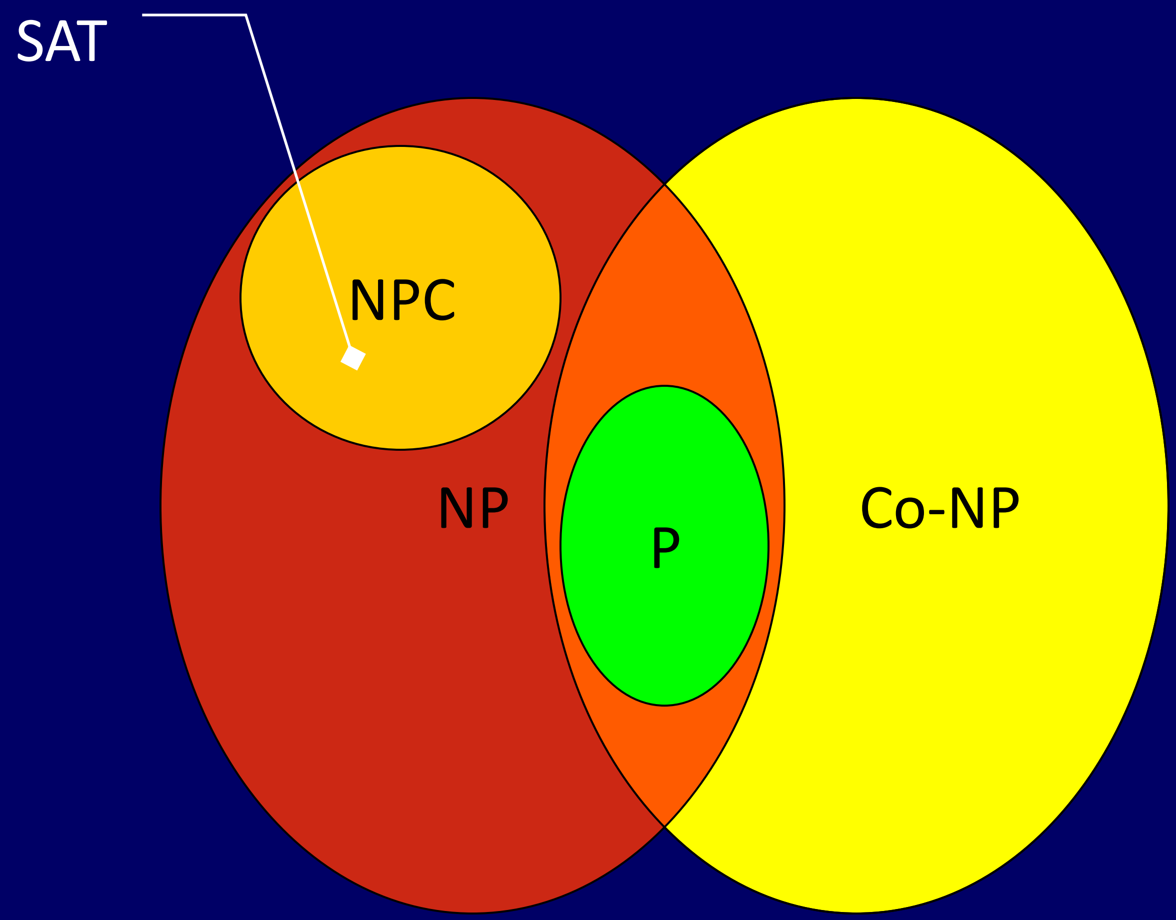
φ, which is of size polynomial in n - Check! - is satisfiable iff the TM accepts the input string.

Conclusion: SAT is NP-Complete

For any language A in NP,

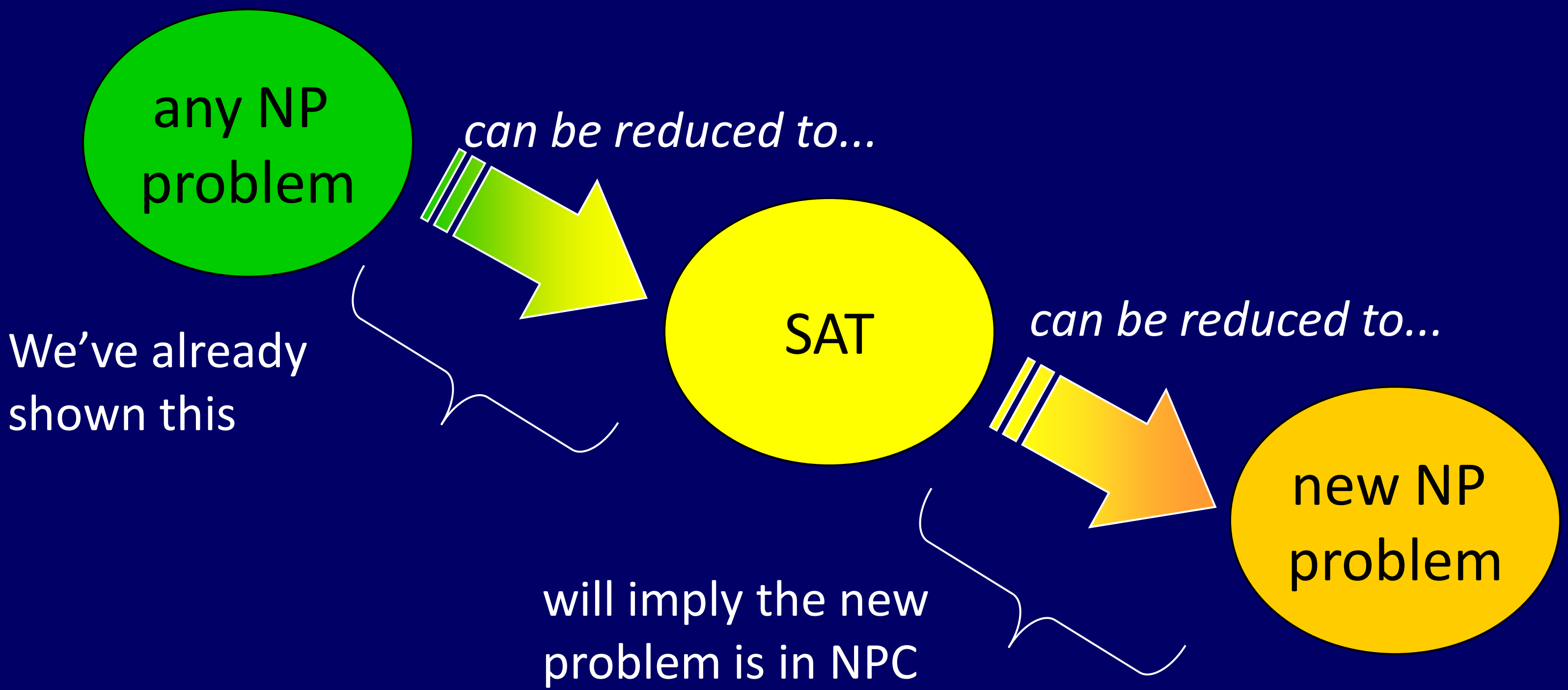


Revisiting the Map



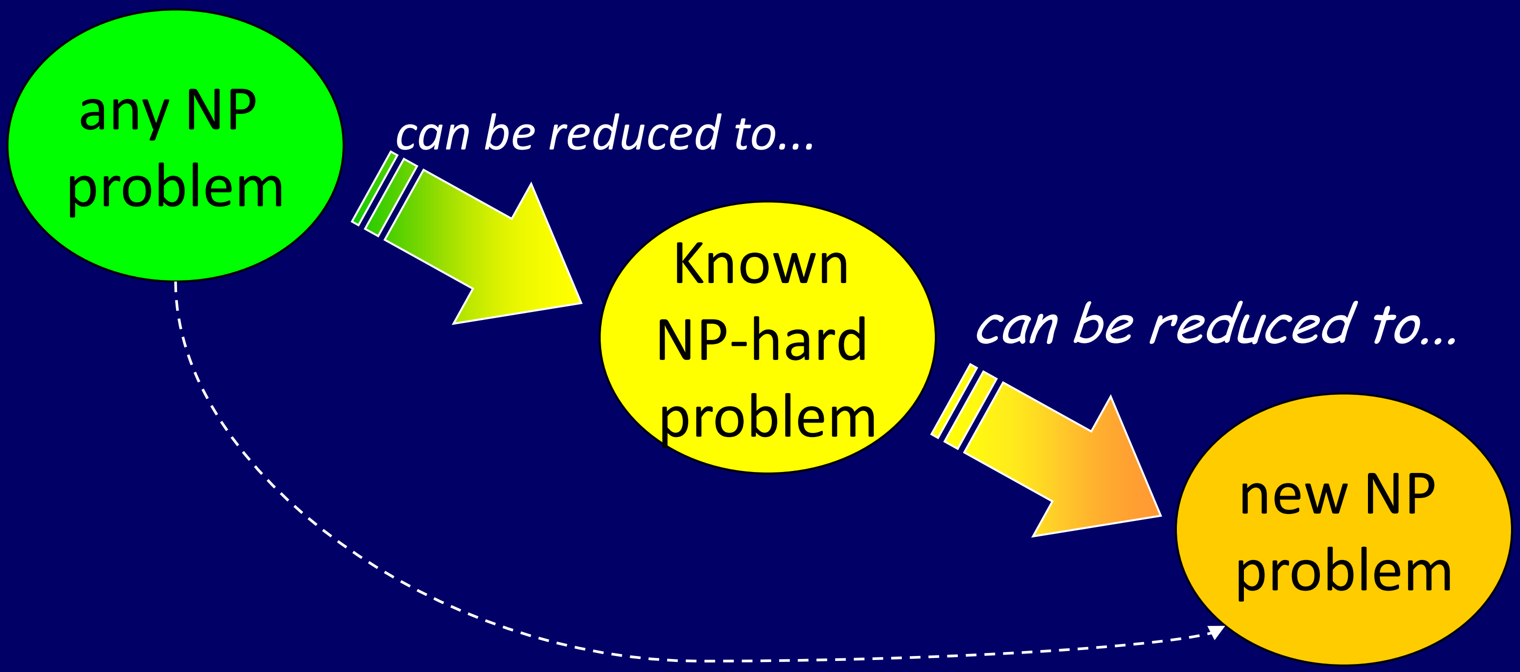
Looking Forward

From now on, in order to show some NP problem is NP-Complete, we merely need to reduce SAT to it.



and Beyond!

Moreover, every NP-Complete problem we discover, provides us with a new way for showing problems to be NP-Complete.



Summary

- We've proved SAT is NP-Complete.
- We've also described a general method for showing other problems are NP-Complete too.