Cook-Levin

HAM

NP-completeness

3SAT

CLIQUE

Reductions

Or

- How to link between problems' complexity, while not knowing what they are

Goal:

- Formalize the notion of "reductions"

Plan:

- Define Karp reductions
- Example: show HAMPATH $\leq_p$ HAMCYCLE
- Closeness under reductions
- Define Cook reductions
- Discuss Completeness

Reductions

an efficient procedure for *A* using

an efficient procedure for *B*

A cannot be radically harder than *B*

- In other words:

B is at least as hard as *A*

Notation

A ≤p B

SIP 250

Karp reductions -Definition

A is **polynomial-time reducible** to B
(denote $A \leq_p B$)

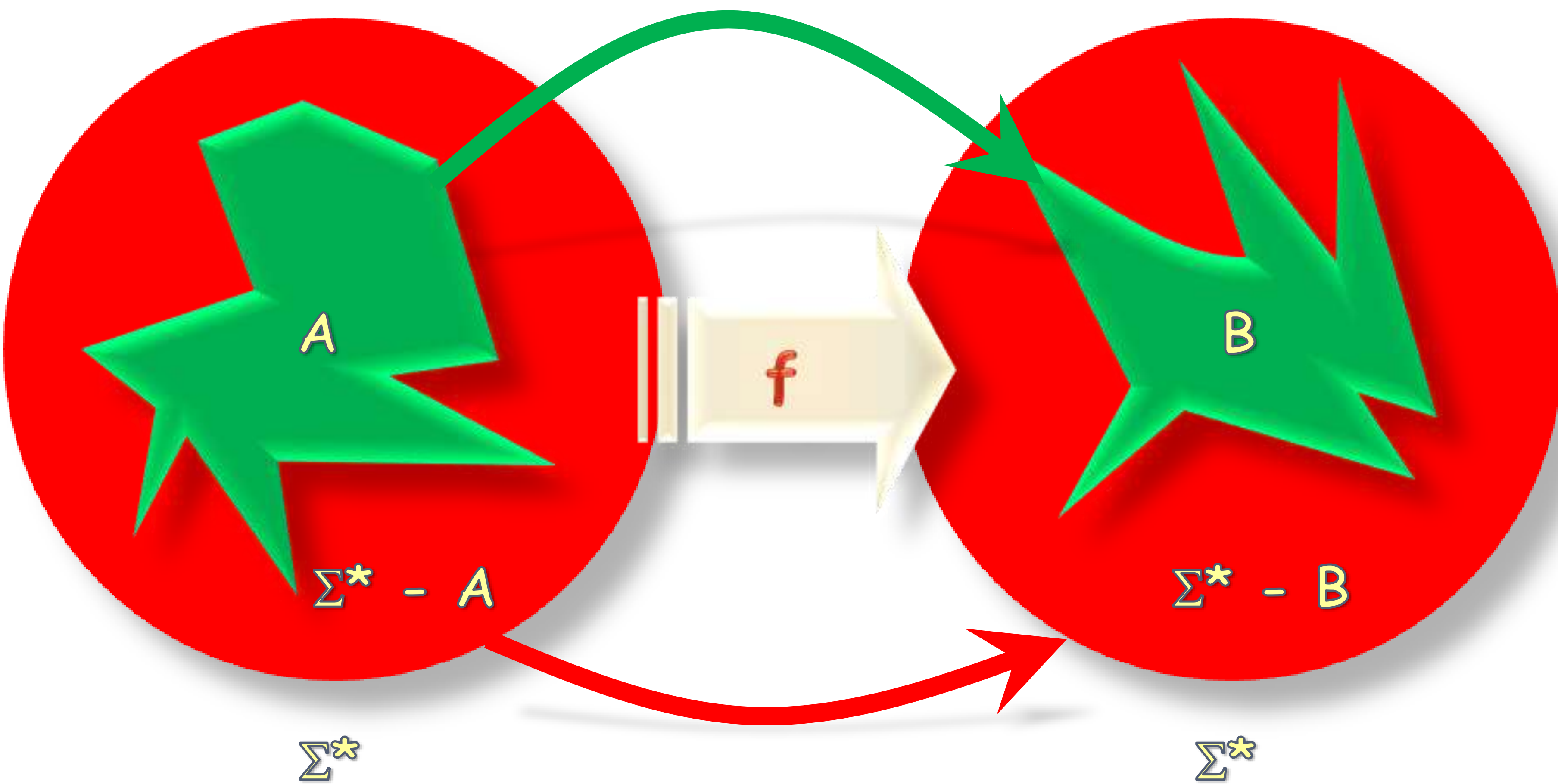
If there exists a poly-time-computable function $f:\Sigma^* \to \Sigma^*$

i.e., ∃ poly-time TM that outputs $f(w)$ on input w

s.t. for every w: $w \in A \Leftrightarrow f(w) \in B$

f is a poly-time reduction of A to B

Karp Reductions - Ilustrated



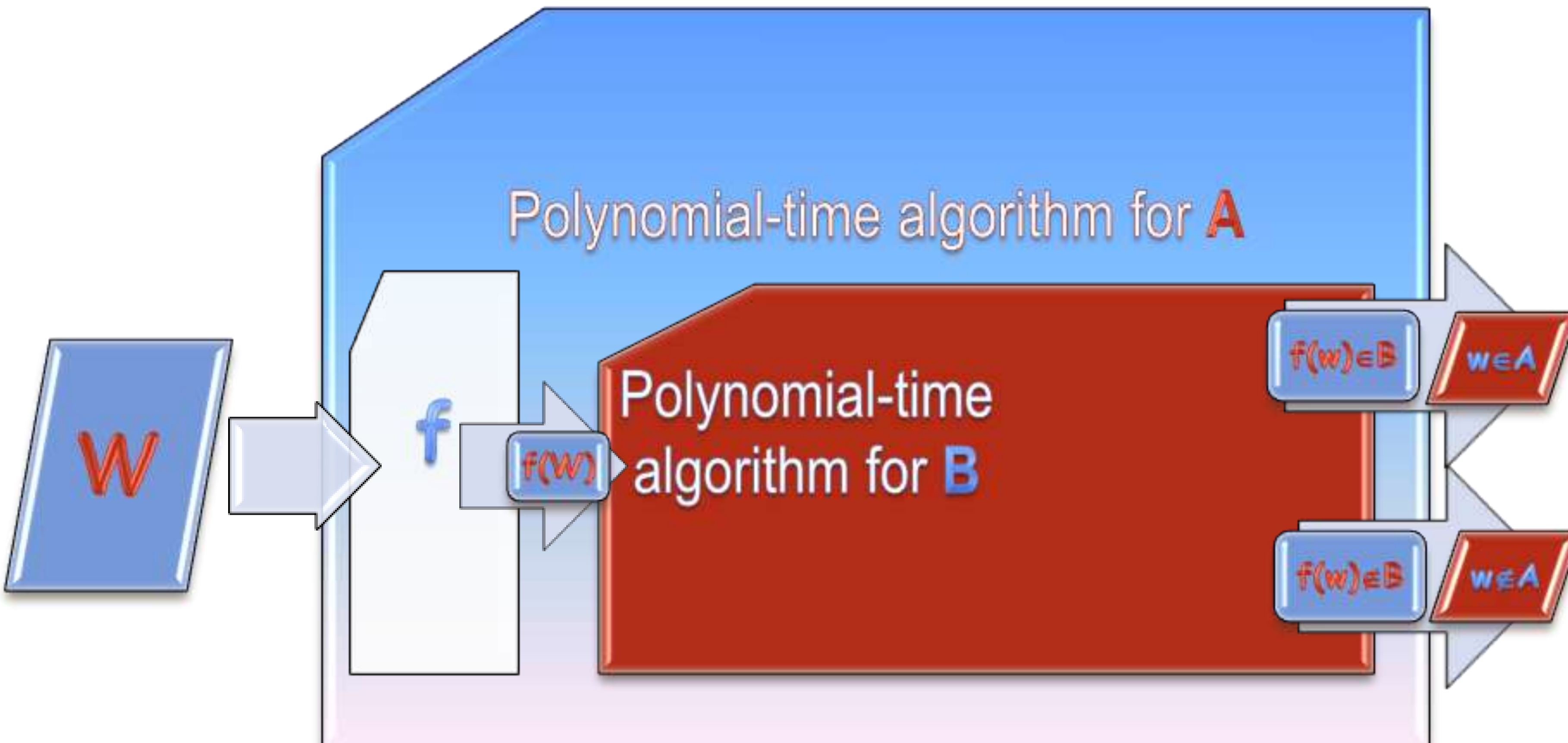
Reducing

To Do:

- Come up with a reduction-function f
- Show f is polynomial time computable
- Prove f is a reduction, i.e., show:
  - $w \in A \rightarrow f(w) \in B$
  - $w \in A \leftarrow f(w) \in B$

- We'll use reductions that, by default, would be of this type, which is called:
  - *Polynomial-time mapping* reduction
  - *Polynomial-time many-one* reduction
  - *Polynomial-time Karp* reduction

Reductions and Efficiency

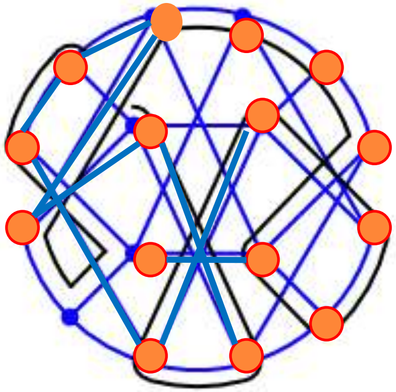


Hamiltonian Path Instance:

- A directed graph $G=(V,E)$

Decision Problem:

- Is there a path in G, which goes through every vertex exactly once?

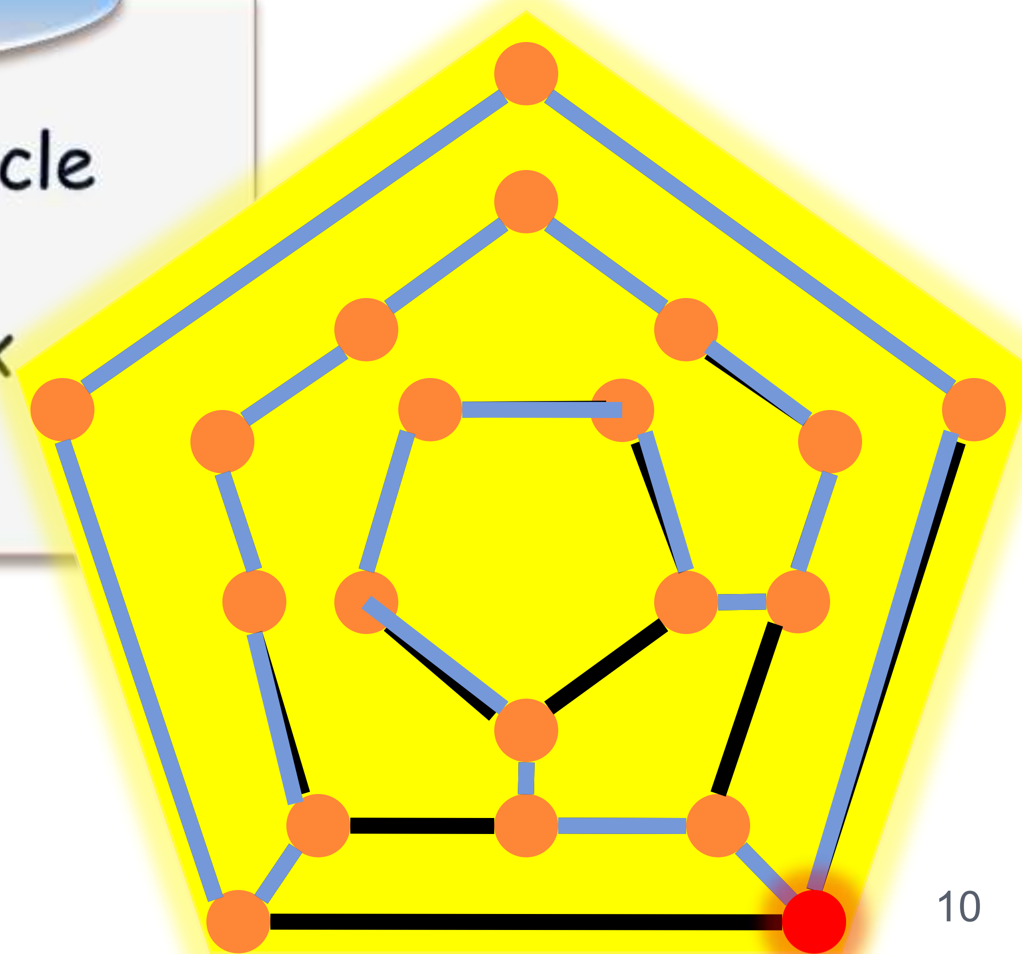


Hamiltonian Cycle Instance:

- a directed graph $G=(V,E)$.

Decision Problem:

- Is there a simple cycle in G that paths through each vertex exactly once?



HAMPATH $\leq_p$ HAMCYCLE

$$f(\langle V, E \rangle) = (\langle V \cup \{u\}, E \cup V \times \{u\} \rangle)$$

Completeness:

- Given a Hamiltonian path $(v_0, \ldots, v_n)$ in G, $(v_0, \ldots, v_n, u)$ is a Hamiltonian cycle in G'

Soundness:

- Given a Hamiltonian cycle $(v_0, \ldots, v_n, u)$ in G', removing u yields a Hamiltonian path.

Check list

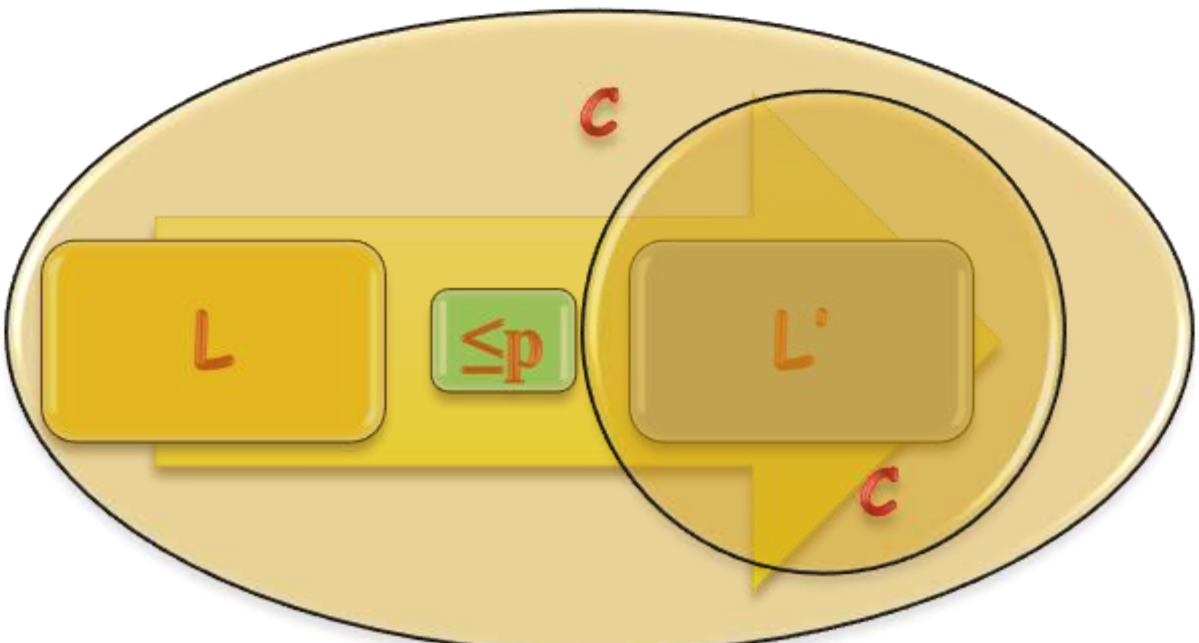
- Come up with a reduction-function f ✓
- Show f is polynomial time computable ? ✓
- Prove f is a reduction, i.e., show:
  - ✓ w∈HAMPATH → f(w)∈HAMCYCLE
  - ✓ w∈HAMPATH ← f(w)∈HAMCYCLE

Closeness Under Reductions: Definition

A complexity class C is **closed under poly-time reductions** if:

- L is reducible to L' and $L' \in C \Rightarrow$

  L is also in C.



Observation

Theorem:

- P, NP, PSPACE and EXPTIME are closed under polynomial-time Karp reductions

Proof:

- Do it yourself !!

Log-space Reductions

A is log-space reducible to B (denote $A \leq_L B$)

If there exists a log-space-computable function $f:\Sigma^* \to \Sigma^*$

i.e., ∃ log-space TM that outputs $f(w)$ on input w

s.t. for every w

$w \in A \Leftrightarrow f(w) \in B$

f is a log-space reduction of A to B

Theorem:

- L, NL, P, NP, PSPACE and EXPTIME are closed under log-space reductions.

Reductions: General

Cook Reduction:

- Assuming an efficient procedure that decides B, construct one for A.

an efficient procedure for *A* using

an efficient procedure for *B*

Karp is a special case of Cook reduction:

It allows only 1 call to B, whose outcome must be outputted as is

Cook red. : HAMCYCLE→HAMPATH.

1. Let $E' = E$
2. If $E' = \emptyset$ reject
3. choose (any) <u, v> in E'
4. If HAMPATH (<V+{w, z}, E'+{<w,u>, <v,z>}>) accept
5. $E' = E' - \{\langle u, v\rangle\}$
6. Go to step 2

Completeness

Definition: C-complete

- For a class C of decision problems and a language $L \in C$, L is **C-complete** if:

  $L' \in C \Rightarrow L'$ is reducible to L.

Theorem:

- L is complete for classes C, $C' \Rightarrow C = C'$

Proof:

- All languages in C and in C' are reducible to L, which is in both. Since both are closed under reductions, they're the same■

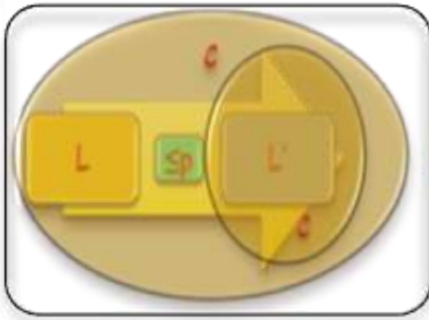
Theorem:

- Any $L \in NPC$, $L \in P \Rightarrow P = NP$

Summary

Discussed types of **reductions**:

- **Cook** vs. **Karp** reductions
- **Poly-time** vs. **log-space**



Defined:

- "completeness"

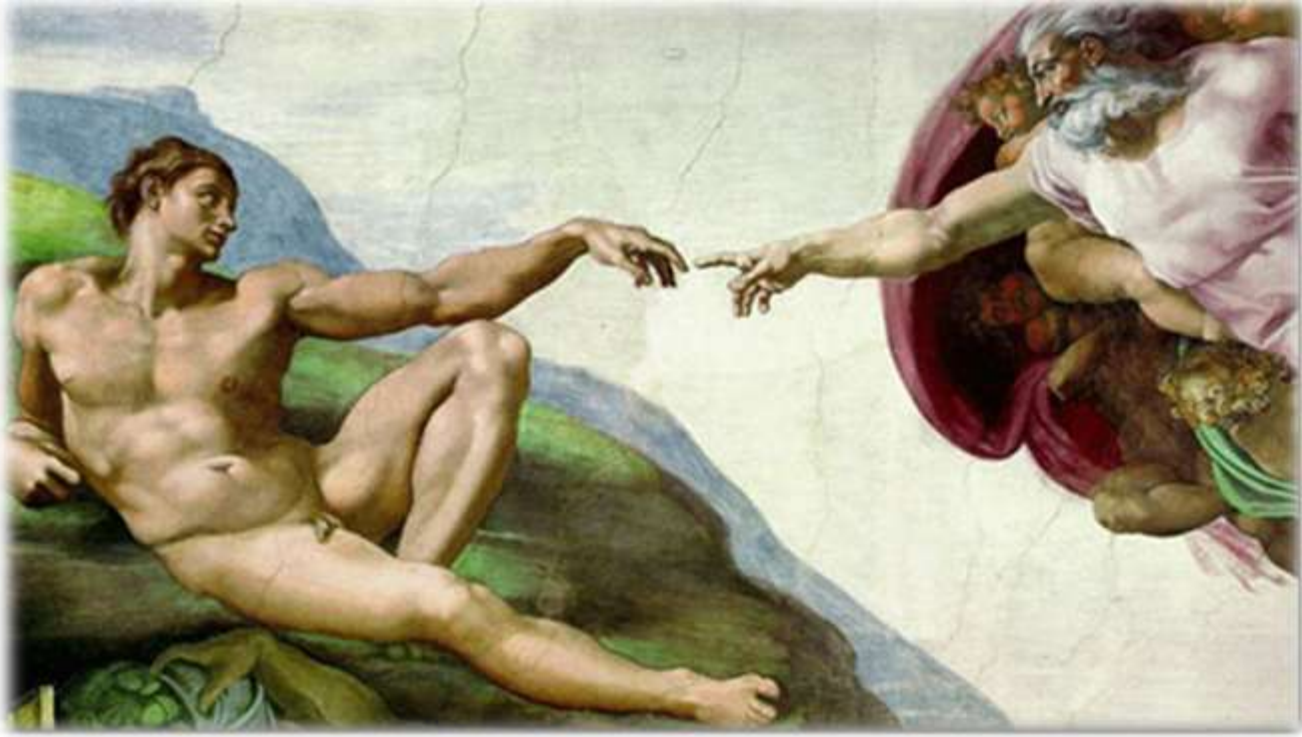
Find

L and **C** s.t. **L** is **C-complete**

Discussed a way to show:

equality between **complexity classes**

The Cook/ Levin theorem:



SAT is NP-Complete:

Goal:

- In the beginning… of **NP-Completeness**

Plan:

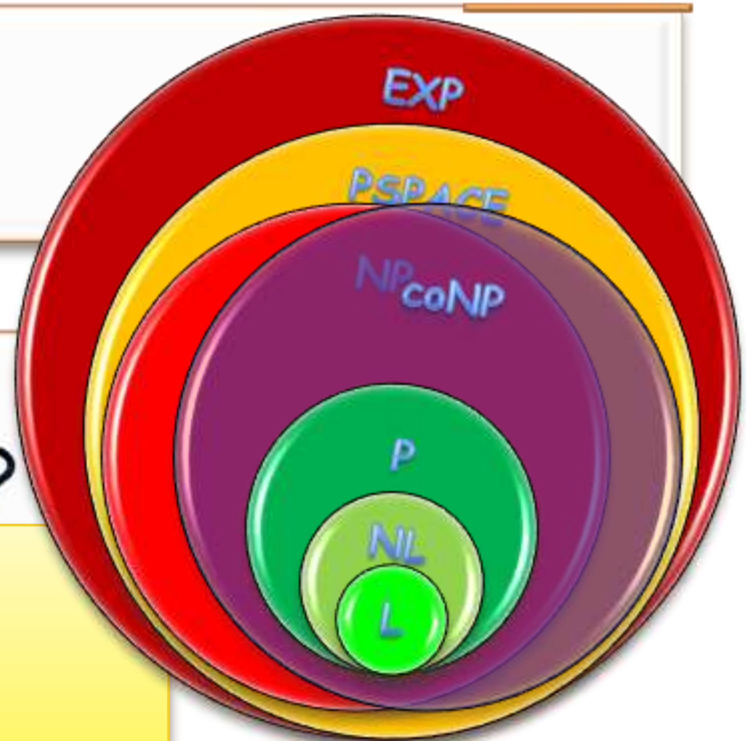
- **SAT** - definition and examples
- The **Cook-Levin Theorem**
- Look ahead

SAT Instance:

- A Boolean formula.

Decision Problem:

- Is the formula satisfiable?



SAT or unSAT?

$$x_1 \wedge \neg x_1$$

$$((x_1 \vee x_2 \vee \neg x_3) \wedge \neg x_1) \vee \neg(x_3 \wedge x_2)$$

Theorem:

- SAT is in NP

Proof:

- Can verify an ass. efficiently

SIP 254-259

Cook/Levin

Theorem:

- SAT is NP-Complete

Proof Outline:

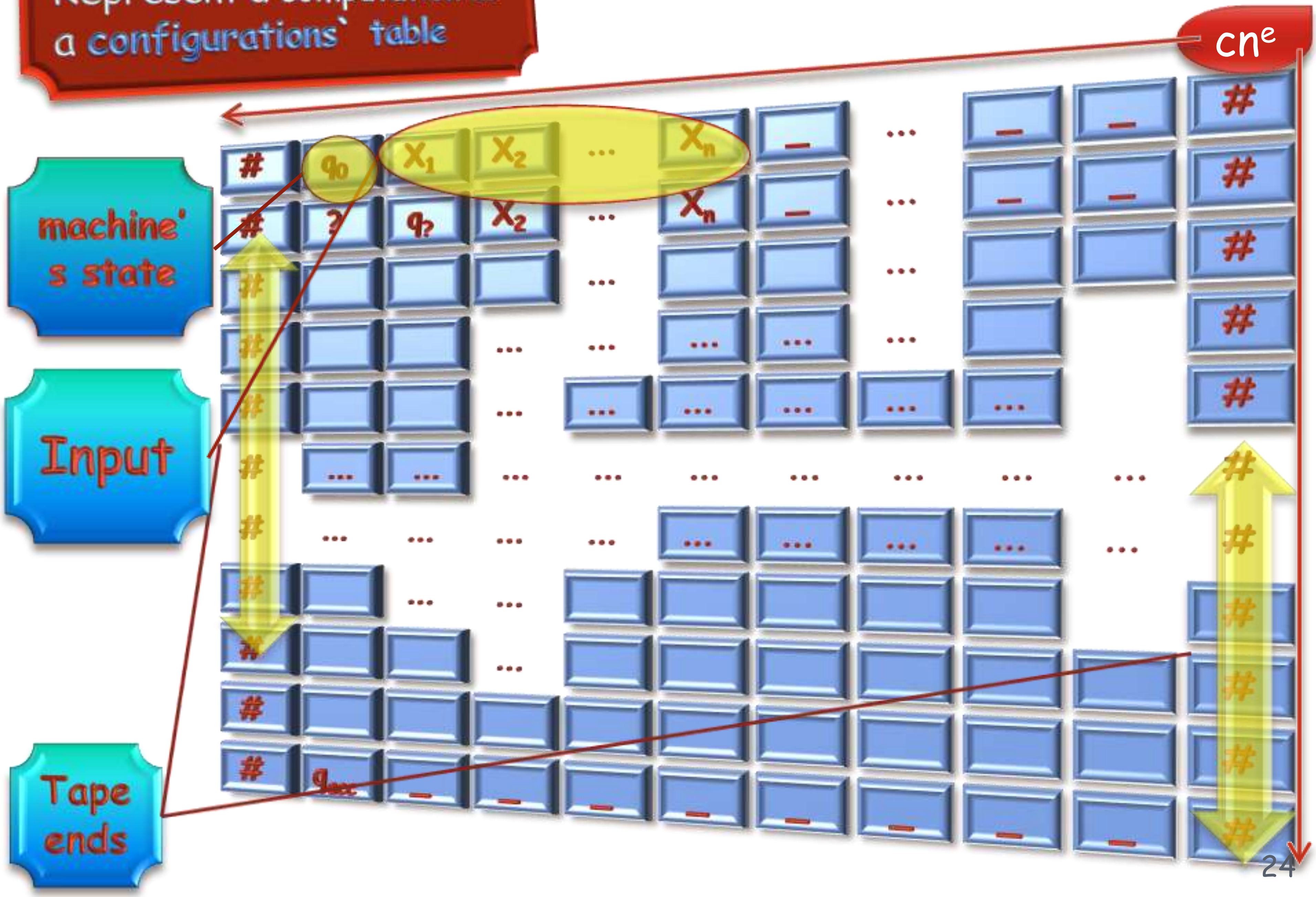
- Given an NP machine M and an input w, construct a *Boolean* formula $\varphi_{M,w}$
  $\varphi_{M,w}$ satisfiable $\Leftrightarrow$ M accepts w.

$\in L_M$

$\varphi_{M,w}$
$\in SAT$

Tableaux

Represent a computation as a configurations` table



Example

Q $= \{q_0, q_1, q_{accept}, q_{reject}\}$

Σ $= \{0, 1\}$

Γ $= \{0, 1, \_\}$

δ

$\delta(q_0,1)=\{(q_1,\_,R)\}$

$\delta(q_1,1)=\{(q_0,\_,R)\}$

$\delta(q_0,0)=\{(q_0,\_,R)\}$

$\delta(q_1,0)=\{(q_1,\_,R)\}$

$\delta(q_0,\_)=\{(q_{acc},\_,L)\}$

$\delta(q_1,\_)=\{(q_{rej},\_,L)\}$

#	q_0	0	1	1	1	#
#	_	q_0	1	1	1	#
#	_	_	q_1	1	1	#
#	_	_	_	q_0	1	#
#	_	_	_	_	q_1	#
#	_	_	_	q_{rej}	_	#

$Q = \{q_0, q_1, q_{accept}, q_{reject}\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, \_\}$

δ:

$\delta(q_0, 1) = \{(q_1, \_, R)\}$

$\delta(q_1, 1) = \{(q_0, \_, R)\}$

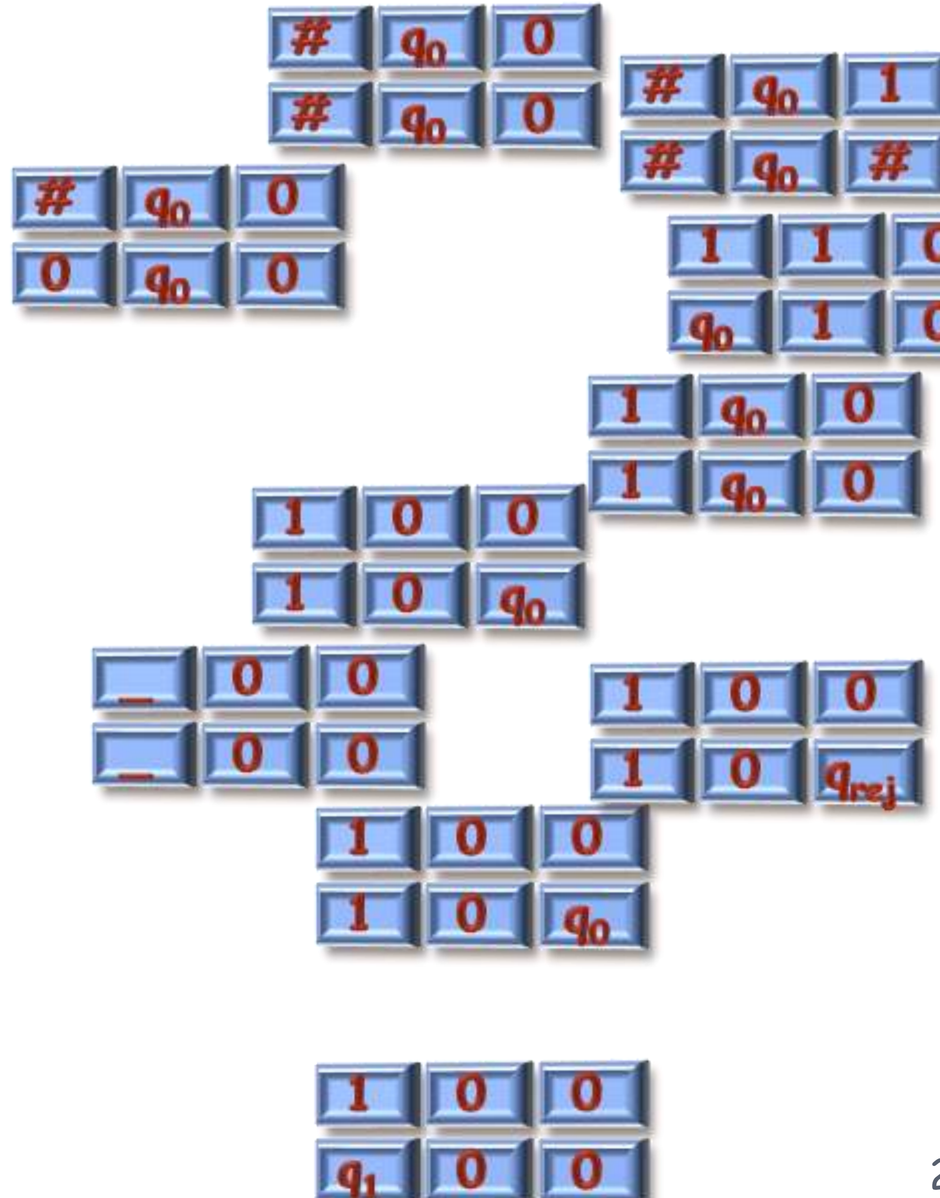
$\delta(q_0, 0) = \{(q_0, \_, R)\}$

$\delta(q_1, 0) = \{(q_1, \_, R)\}$

$\delta(q_0, \_) = \{(q_{acc}, \_, L)\}$

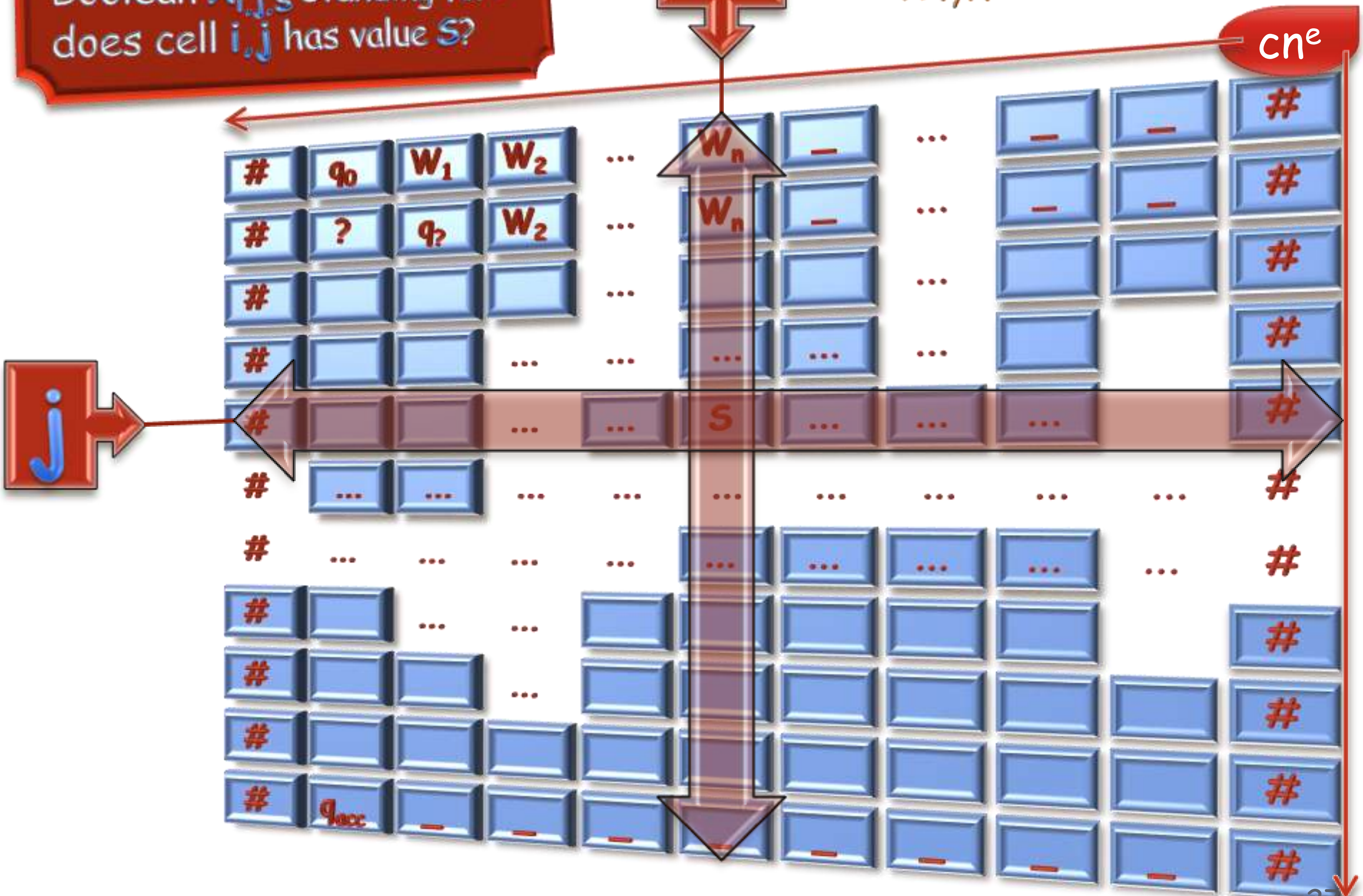
$\delta(q_1, \_) = \{(q_{rej}, \_, L)\}$

$$\Delta_M \subset (\Gamma \cup Q \cup \{\#\})^6$$



$\varphi_{M,w}$ variables

Boolean $X_{i,j,s}$ standing for:
does cell i,j has value s?



$\varphi_{M,w} =$

$\varphi_{M,w}$

≥1 value ∧ <2 values — **1 value to each entry**

$$\bigwedge_{1\le i,j\le cn^e}\left[\left(\bigvee_{s\in(\Gamma\cup Q\cup\{\#\})} x_{i,j,s}\right)\wedge\left(\bigwedge_{s\neq t\in(\Gamma\cup Q\cup\{\#\})}(\overline{x_{i,j,s}}\vee\overline{x_{i,j,t}})\right)\right]$$

start · input · blanks — **input consistent**

$$\wedge X_{0,0,\#}\wedge X_{0,1,q_0}\wedge X_{0,2,w_1}\wedge\cdots\wedge X_{0,n+1,w_n}\wedge X_{0,n+2,\_}\wedge\cdots\wedge X_{0,cn^e,\_}\wedge X_{0,cn^e+1,\#}$$

locally legal transition — **transitions legal**

$$\bigwedge_{0\le i,j\le cn^e}\left[\bigvee_{\langle s_{0,0},s_{0,1},s_{1,0},s_{1,1},s_{2,0},s_{2,1}\rangle\in\Delta_M}\left[\bigwedge_{i'=0,1,2,\,j'=0,1}\left[X_{i+i',j+j',s_{i',j'}}\right]\right]\right]$$

∃ accepting configuration — **machine accepts**

$$\bigwedge\bigvee_{0\le i,j\le cn^e} X_{i,j,q_{acc}}$$

Q.E.D.

$$\varphi_{M,w} = \bigwedge_{1\le i,j\le cn^e}\left[\left(\bigvee_{s\in(\Gamma\cup Q\cup\{\#\})} x_{i,j,s}\right)\wedge\left(\bigwedge_{s\neq t\in(\Gamma\cup Q\cup\{\#\})}(\overline{x_{i,j,s}}\vee\overline{x_{i,j,t}})\right)\right]$$

$$\wedge X_{0,0,\#}\wedge X_{0,1,q_0}\wedge X_{0,2,w_1}\wedge\cdots\wedge X_{0,n+1,w_n}\wedge X_{0,n+2,\_}\wedge\cdots\wedge X_{0,cn^e,\_}\wedge X_{0,cn^e+1,\#}$$

$$\bigwedge_{0\le i,j\le cn^e}\left[\bigvee_{\langle s_{0,0},s_{0,1},s_{1,0},s_{1,1},s_{2,0},s_{2,1}\rangle\in\Delta_M}\left[\bigwedge_{i'=0,1,2,\,j'=0,1}\left[X_{i+i',j+j',s_{i',j'}}\right]\right]\right]$$

$$\wedge\bigvee_{0\le i,j\le cn^e} X_{i,j,q_{acc}}$$

Claim:

- $\forall i,j$ transition is locally legal $\Leftrightarrow$ tableau legal

Corollary:

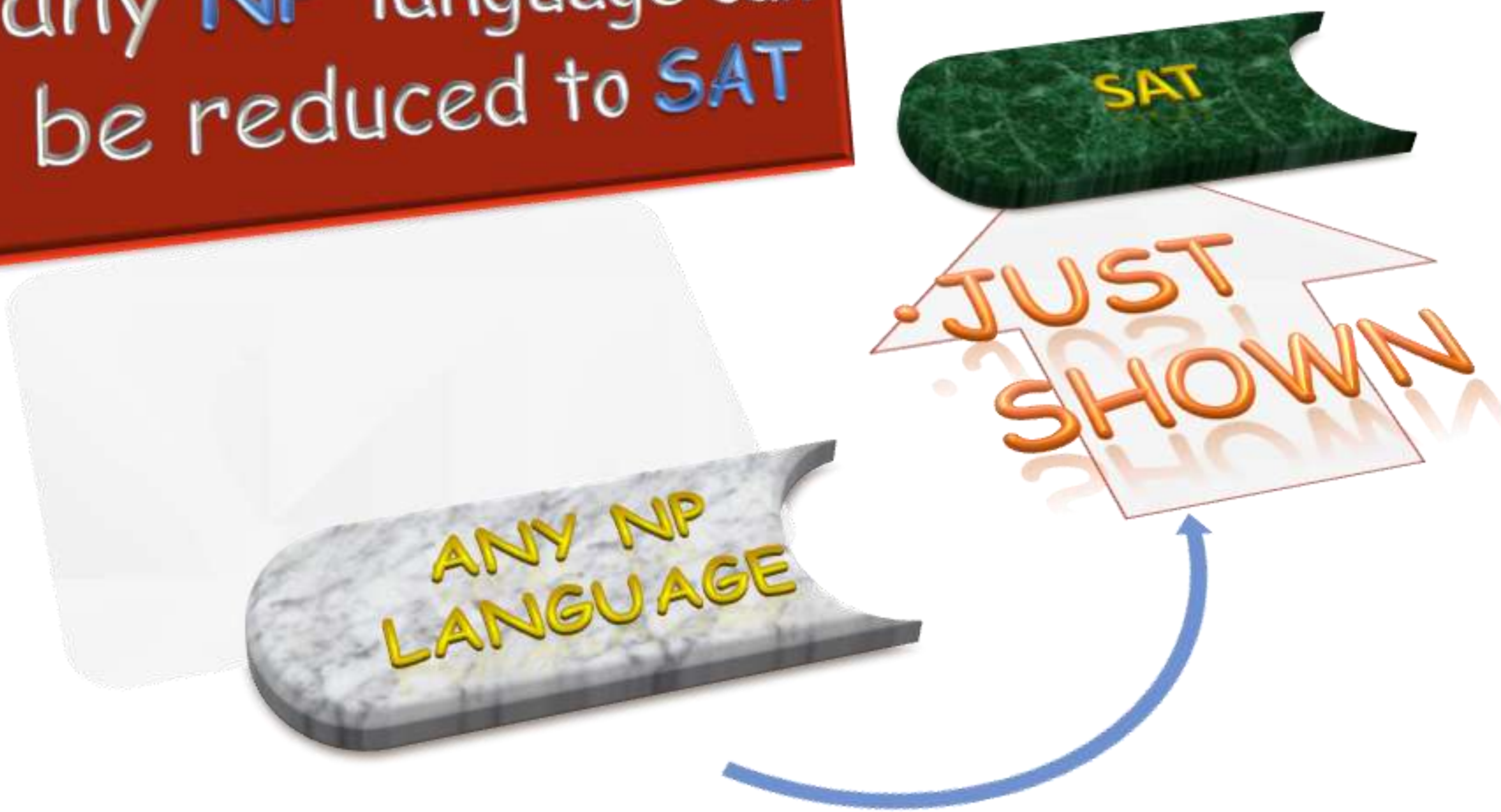
- $\varphi_{M,w}$ Satisfiable $\Leftrightarrow$ $W\in L_M$

Claim:

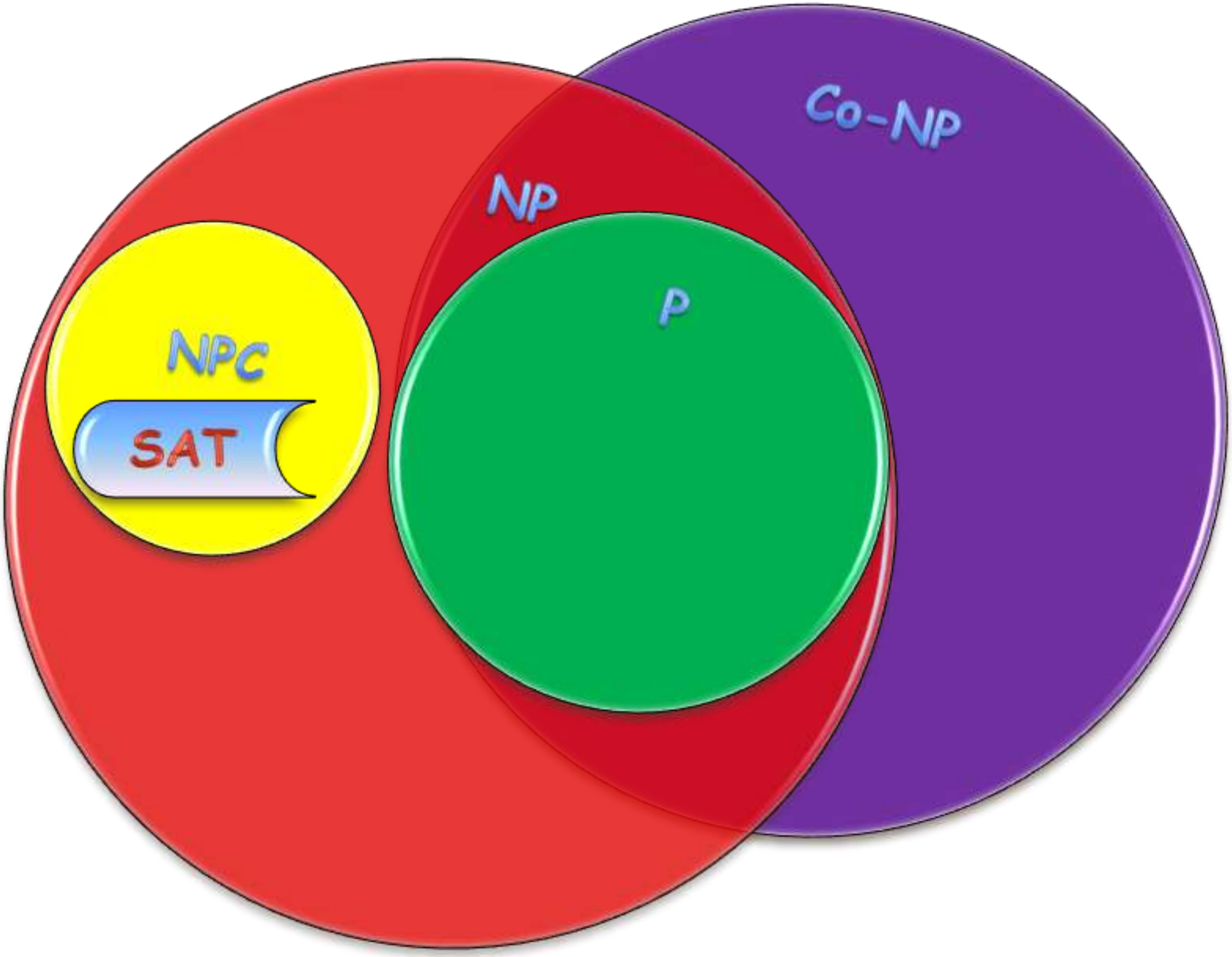
- Size of $\varphi_{M,w}$ polynomial in $|W|$

SAT is NPC

We have just shown **SAT** is **NP-hard**, as any **NP** language can be reduced to **SAT**

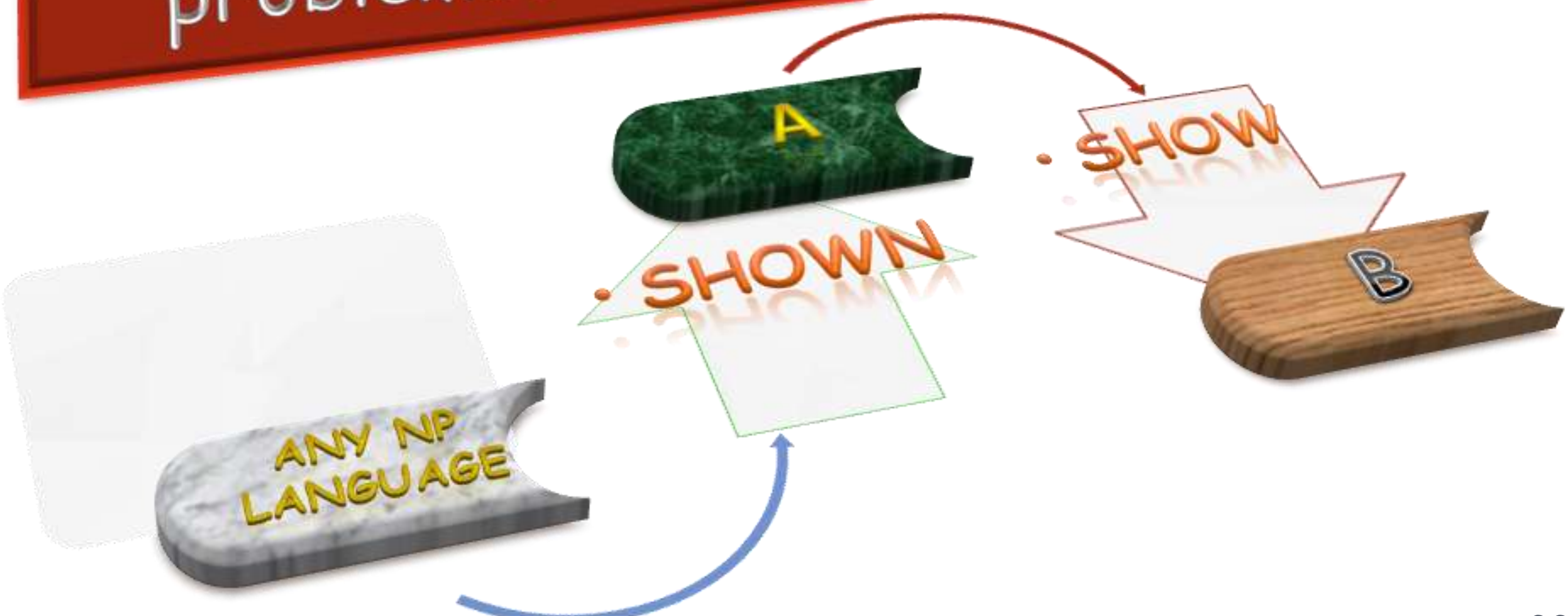


P, NP, co-NP and NPC

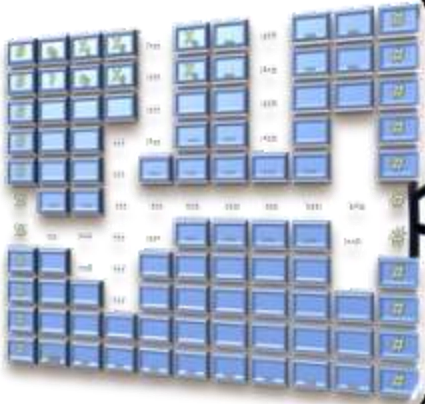


Henceforth, to show a problem **A** is **NP-hard**, it suffices to reduce **SAT** to **A**

Furthermore, once we've shown A is NP-hard, we can reduce from it to show other problems NP-hard



Summary



proved SAT is NP-Complete

Consider SAT the Genesis problem, and explored how to proceed and show other problems are NP-hard

Goal:

- introduce some additional NP-Complete problems.

Plan:

- 3SAT
- CLIQUE & INDEPENDENT-SET

SAT and NPC

Recall: L is NPC if

- L In NP
- L NP-hard – via Karp-reduction

So far we only showed one such problem: SAT

- which, however, is not up for the tasks ahead

Next we show a special case of SAT is NPC:

- 3SAT

3SAT Instance:

- 3CNF formula

Conjunctive Normal Form – 3 literals in each clause

Decision Problem:

- Is it satisfiable?

3CNF:

$(x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z)$

$(x \vee x \vee x) \wedge (\neg x \vee \neg x \vee \neg x)$

3SAT is NPC

Claim:
- 3SAT ∈ NP ♦

3SAT is a special case of SAT.

Claim:
- 3SAT ∈ NP-hard

Proof:
- amend our SAT formula, so it becomes 3CNF
- First make it a CNF: use DNF→CNF on 3rd line

Does this suffice?

Are all others OK?

$$\varphi_{M,w} = \bigwedge_{1\le i,j\le cn^e} \left[\left(\bigvee_{s\in(\Gamma\cup Q\cup\{\#\})} x_{i,j,s} \right) \wedge \left(\bigwedge_{s\ne t\in(\Gamma\cup Q\cup\{\#\})} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right]$$

$$\wedge X_{0,0,\#} \wedge X_{0,1,q_0} \wedge X_{0,2,w_1} \wedge \cdots \wedge X_{0,n+2,w_n} \wedge X_{0,n+3,\_} \wedge \cdots \wedge X_{0,cn^e,\_} \wedge X_{0,cn^e+1,\#}$$

$$\bigwedge_{0\le i,j\le cn^e} \left[\bigvee_{\langle s_{0,0},s_{0,1},s_{1,0},s_{1,1},s_{2,0},s_{2,1}\rangle\in\Delta_M} \left[\bigwedge_{i'=0,1,2,\, j'=0,1} \left[X_{i+i',j+j',s_{i',j'}} \right] \right] \right]$$

$$\bigwedge \bigvee_{0\le i,j\le cn^e} X_{i,j,q_{acc}}$$

What is the size of new formula?

CNF→3CNF

$$(x \vee y) \wedge (x_1 \vee x_2 \vee \dots \vee x_t) \wedge \dots$$

clauses with 1 or 2 literals

clauses with more than 3 literals

replication

split

$$(x \vee y \vee x)$$

$$(x_1 \vee x_2 \vee c_{11}) \wedge (\neg c_{11} \vee x_3 \vee c_{12}) \wedge \dots \wedge (\neg c_{1t-3} \vee x_{t-1} \vee x_t)$$

QED

3SAT is NP-Complete

CLIQUE is NPC

CLIQUE instance:

- A graph $G=(V,E)$ and a threshold k

Decision problem:

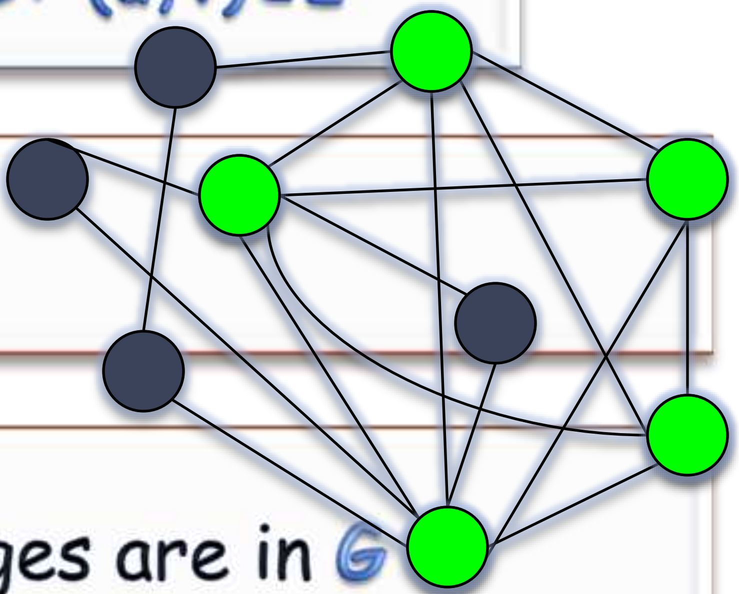
- Is there a set of nodes $C=\{v_1,\ldots,v_k\}\subseteq V$, s.t. $\forall u,v\in C$: $(u,v)\in E$

Observation:

- CLIQUE $\in$ NP

Proof:

- Given C, verify all inner edges are in G



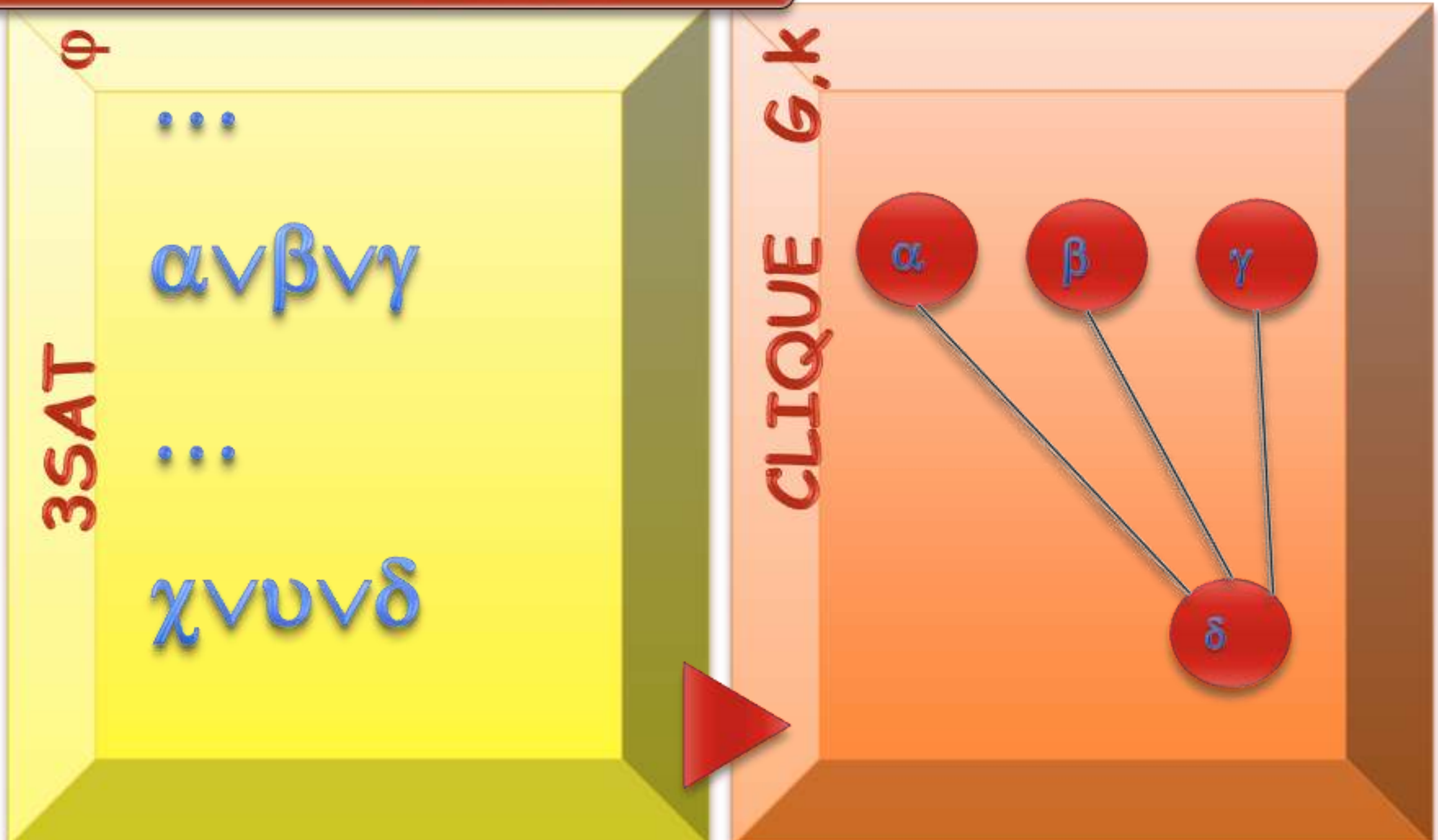
SAT $\leq_p$ CLIQUE

1 vertex for 1 occurrence

inconsistency ⇔ non-edge

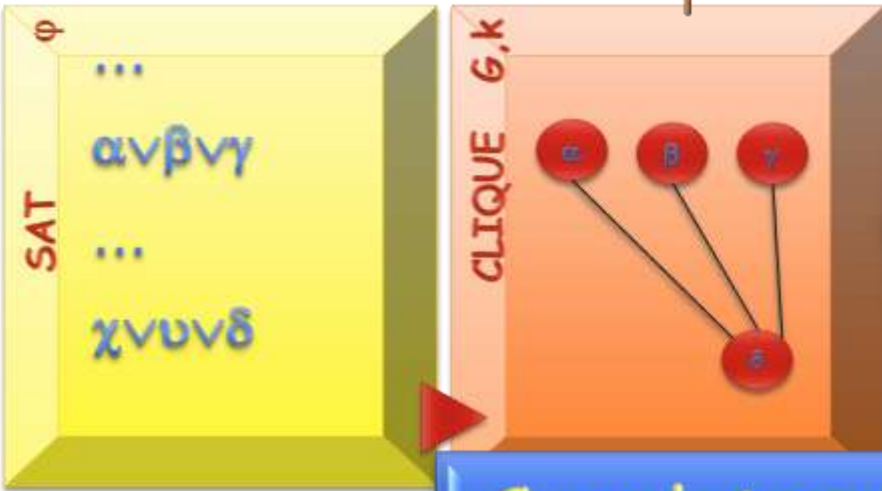
• within triplets α=¬δ

K = number of clauses



SIP 251-253

SAT $\leq_p$ CLIQUE: proof



each triplet disconnected

k-clique has 1 vertex in each

Completeness:

- Let A be a satisfying assignment to φ, $C(A)$ contains 1 v_α s.t. $A(v_\alpha)$ for every clause

Soundness:

- In a clique C in G of size k, each variable has ≤1 of its literals-vertex in C
- extend to a satisfying assignment to φ

INDEPENDENT-SET is NPC

IS instance:

- A graph $G=(V,E)$ and a threshold k

Decision problem:

- Is there a set of nodes $I=\{v_1,\ldots,v_k\}\subseteq V$, s.t. $\forall u,v\in I: (u,v)\notin E$

Observation:

- $IS \in NP$

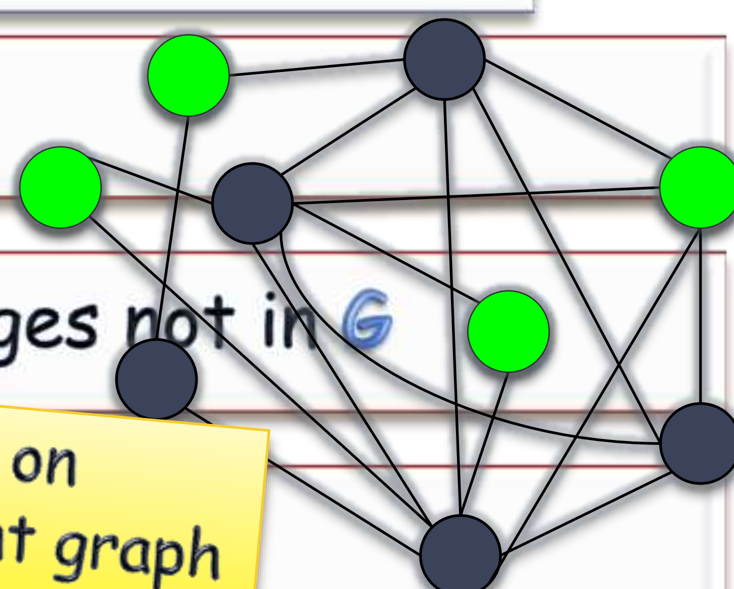
Proof:

- Given I, verify all inner edges not in G

Observation:

- IS is NP-hard

Clique≡IS on complement graph



WWindex

Reductions

Polynomial Time Reductions

Completeness

Hamiltonian Path

Log Space Reductions

Completeness

Complexity Classes

NP

co-NP

P

L

NL

EXPTIME

PSPACE

Hamilton, William Rowan

Karp, Richard

Cook, Stephen Arthur

Levin, Leonid

WWindex

SAT

Cook-Levin Theorem

3SAT

Cook-Levin Theorem

Clique

Independent Set

Subset Sum

CNF

NPC

NP Hard