

NP-complete Languages

Polynomial Time Reductions

Polynomial Computable function f :

There is a deterministic Turing machine M
such that for any string w computes $f(w)$
in polynomial time: $O(|w|^k)$

Observation:

the length of $f(w)$ is bounded

$$|f(w)| = O(|w|^k)$$

since, M cannot use
more than $O(|w|^k)$ tape space
in time $O(|w|^k)$

Definition:

Language A

is polynomial time reducible to
language B

if there is a polynomial computable
function f such that:

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

Theorem:

Suppose that A is polynomial reducible to B .
If $B \in P$ then $A \in P$.

Proof:

Let M be the machine that decides B
in polynomial time

Machine M' to decide A in polynomial time:

On input string w :

1. Compute $f(w)$
2. Run M on input $f(w)$
3. If $f(w) \in B$ accept w

Example of a polynomial-time reduction:

We will reduce the

3CNF-satisfiability problem
to the

CLIQUE problem

3CNF formula:

$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge \underbrace{(x_3 \vee \overline{x_5} \vee x_6)}_{\text{clause}} \wedge (x_3 \vee \overline{x_6} \vee x_4) \wedge (x_4 \vee x_5 \vee x_6)$$

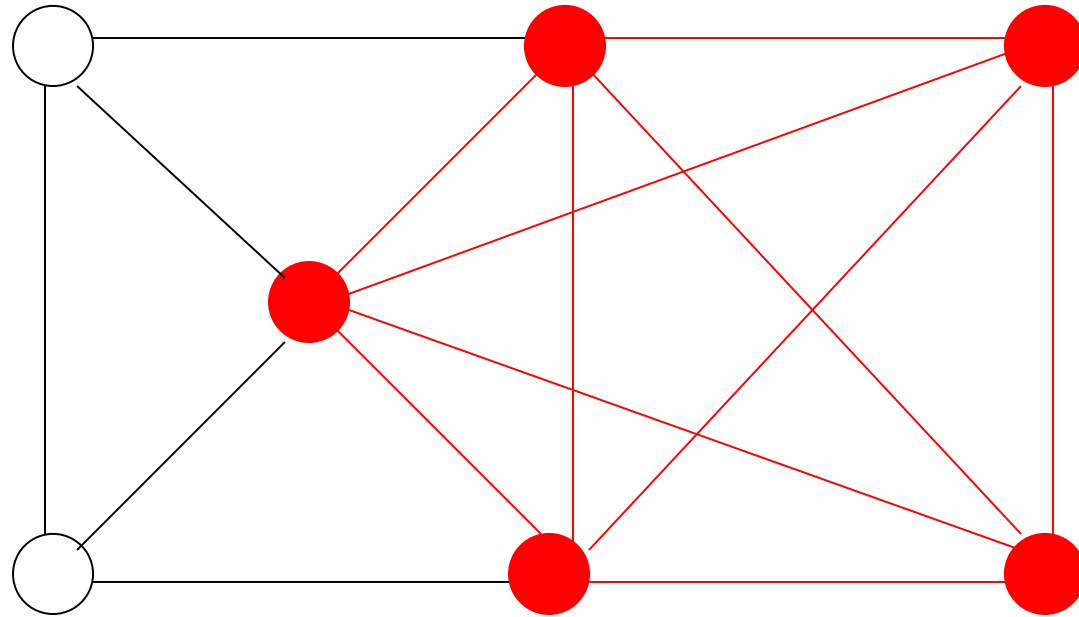
literal variable or its complement

Each clause has three literals

Language:

$3\text{CNF-SAT} = \{ w : w \text{ is a satisfiable } 3\text{CNF formula} \}$

A 5-clique in graph G



Language:

$\text{CLIQUE} = \{ \langle G, k \rangle : \text{graph } G \text{ contains a } k\text{-clique} \}$

Theorem: 3CNF-SAT is polynomial time reducible to CLIQUE

Proof: give a polynomial time reduction of one problem to the other

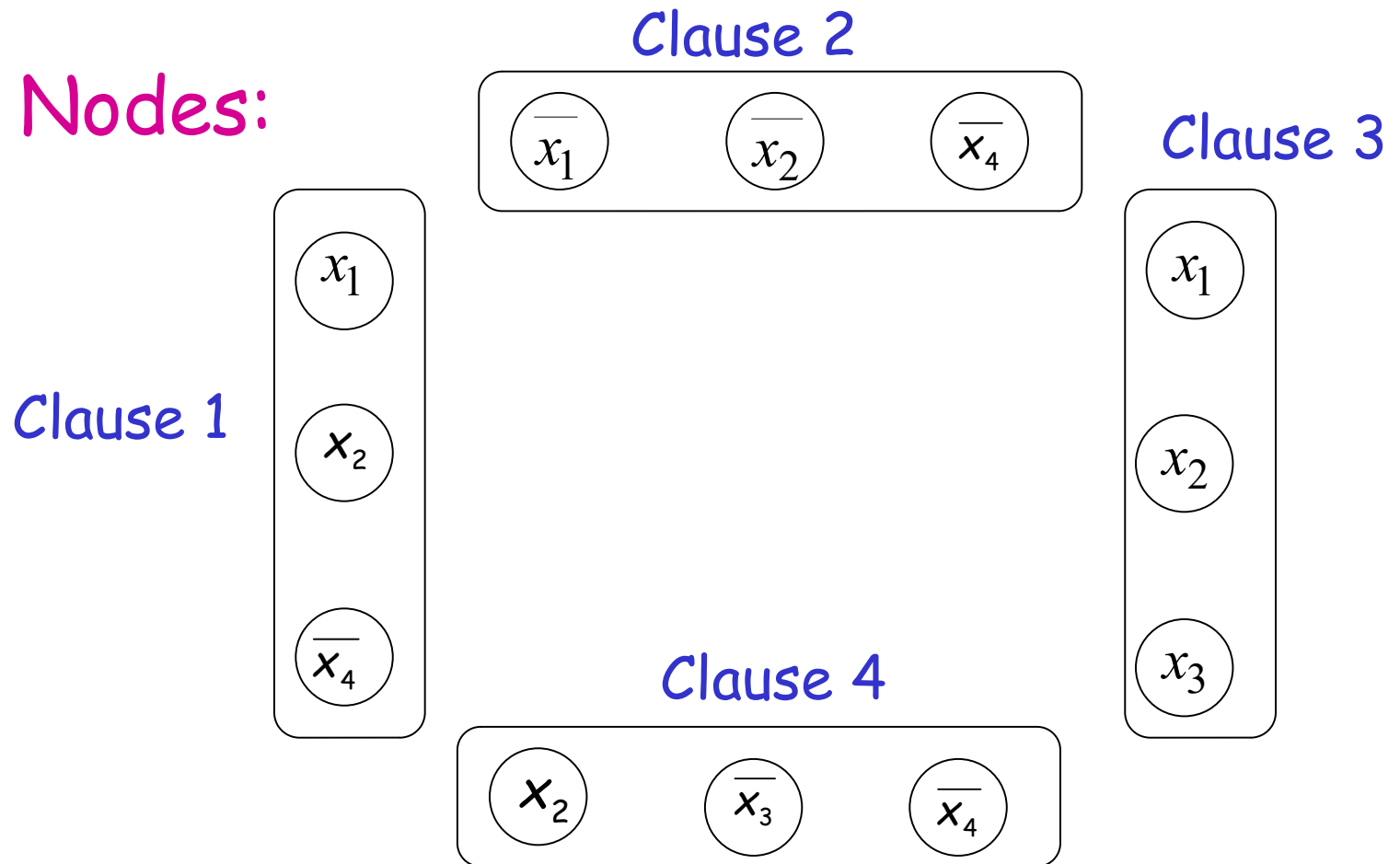
Transform formula to graph

Transform formula to graph.

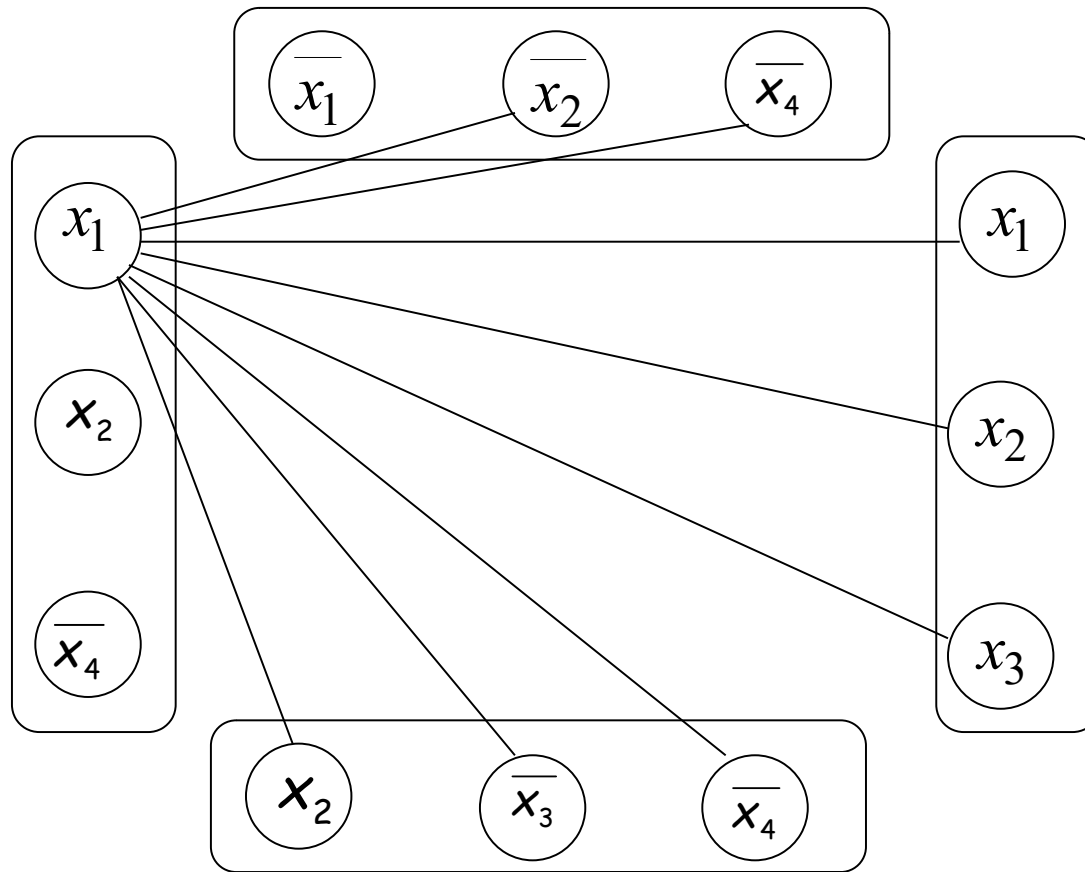
Example:

$$(x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$$

Create Nodes:

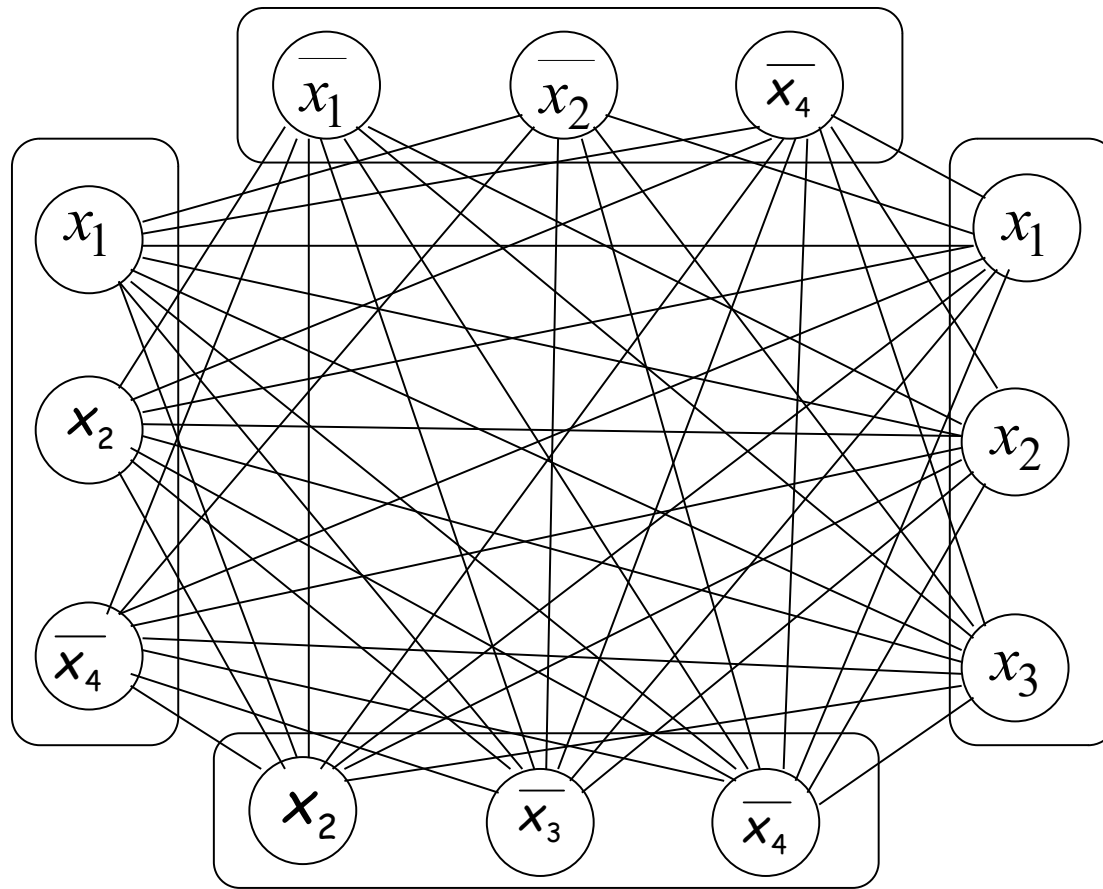


$$(x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$$



Add link from a literal ξ to a literal in every other clause, except the complement $\overline{\xi}$

$$(x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$$



Resulting Graph

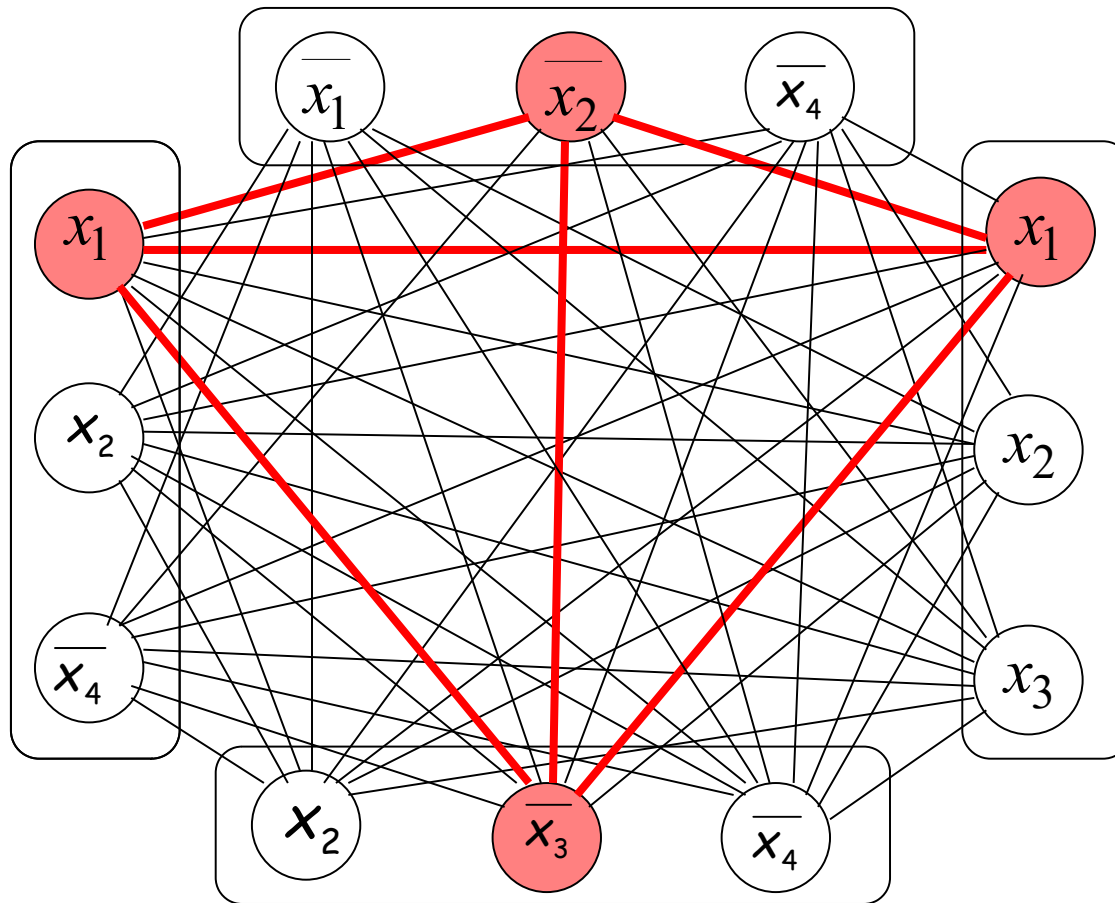
$$(x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4}) = 1$$

$$x_1 = 1$$

$$x_2 = 0$$

$$x_3 = 0$$

$$x_4 = 1$$

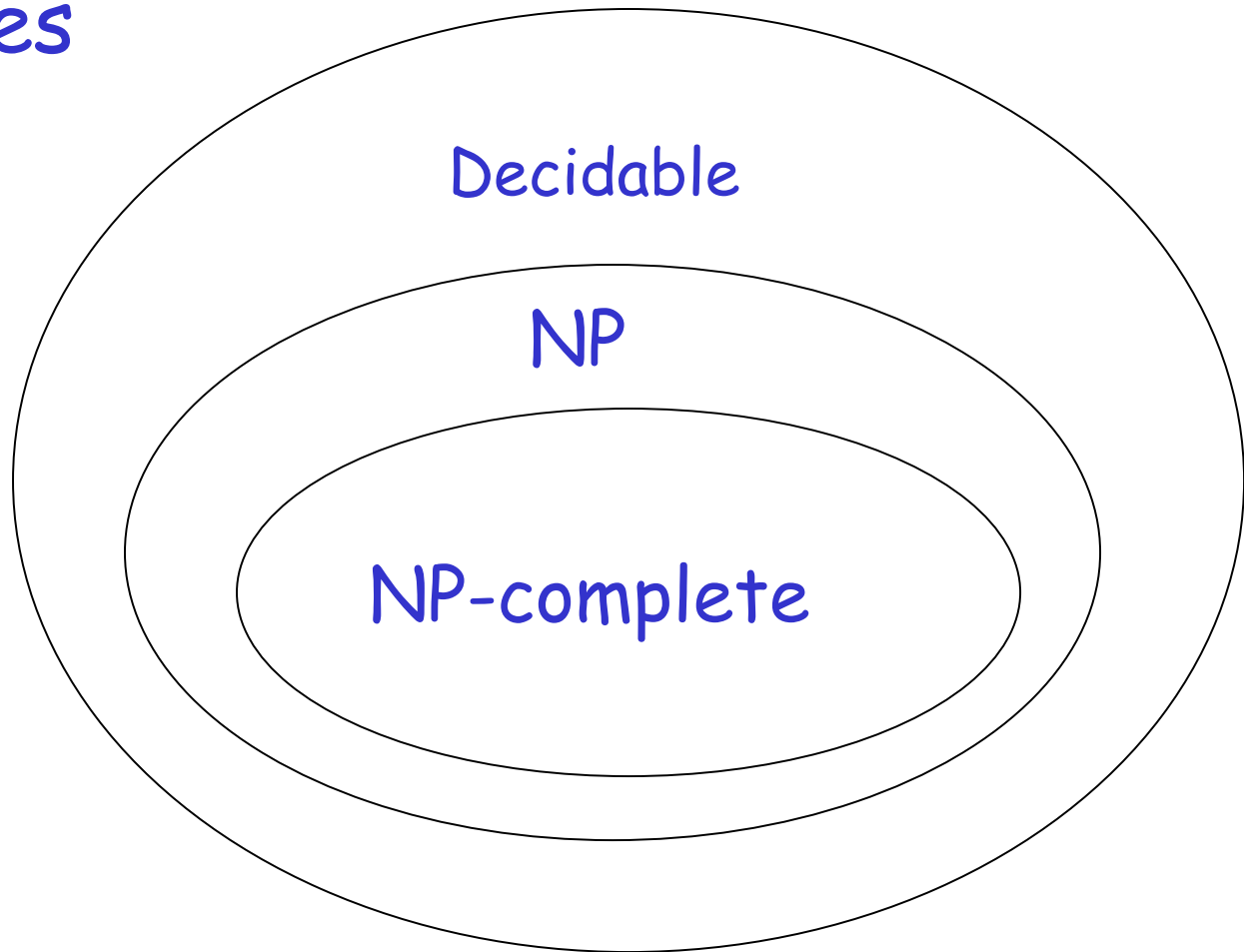


The formula is satisfied if and only if
the Graph has a 4-clique

End of Proof

NP-complete Languages

We define the class of NP-complete languages



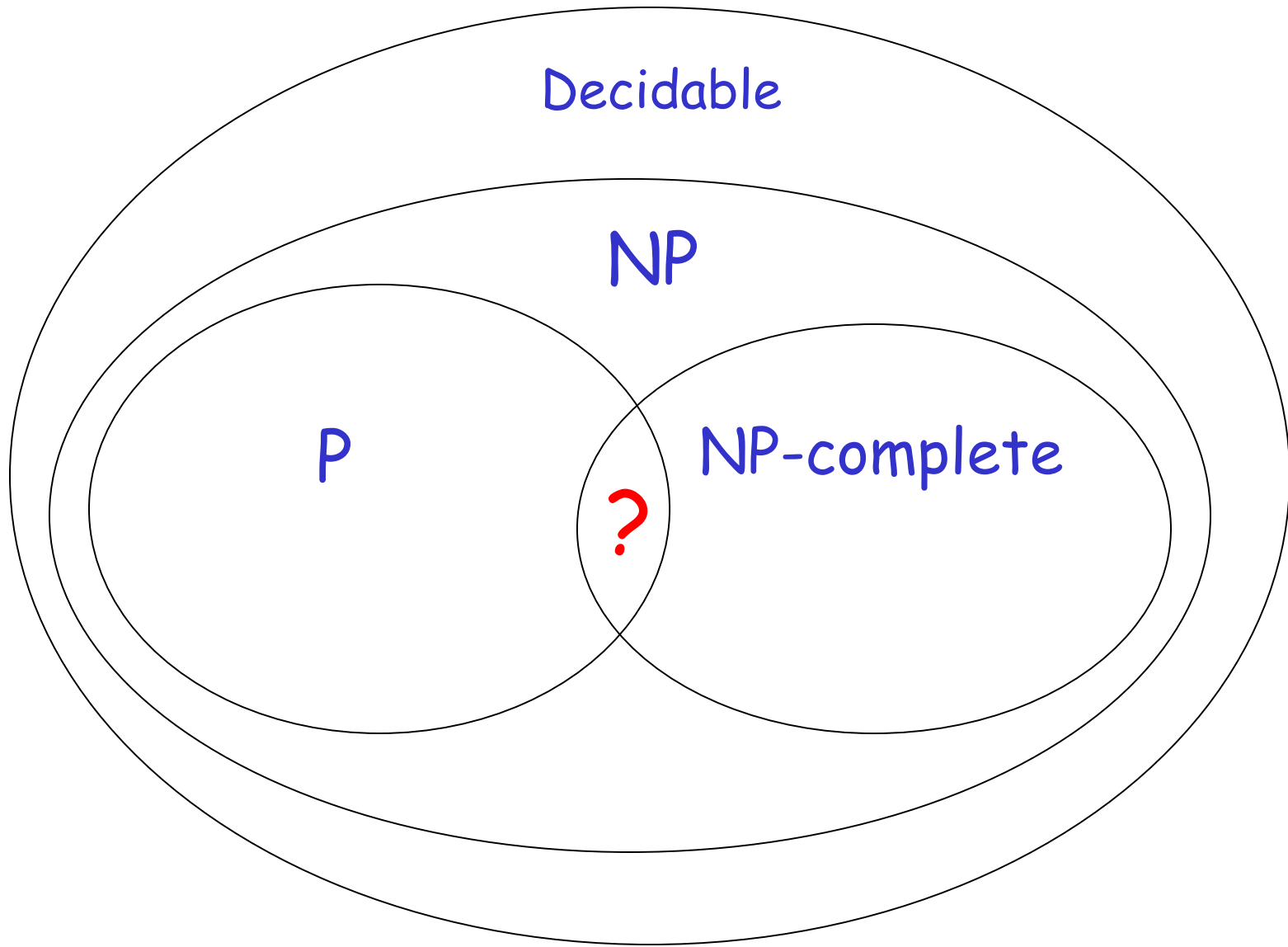
A language L is NP-complete if:

- L is in NP, and
- Every language in NP is reduced to L in polynomial time

Observation:

If a NP-complete language
is proven to be in P then:

$$P = NP$$



An NP-complete Language

Cook-Levin Theorem:

Language SAT (satisfiability problem)
is NP-complete

Proof:

Part1: SAT is in NP

(we have proven this in previous class)

Part2: reduce all NP languages
to the SAT problem
in polynomial time

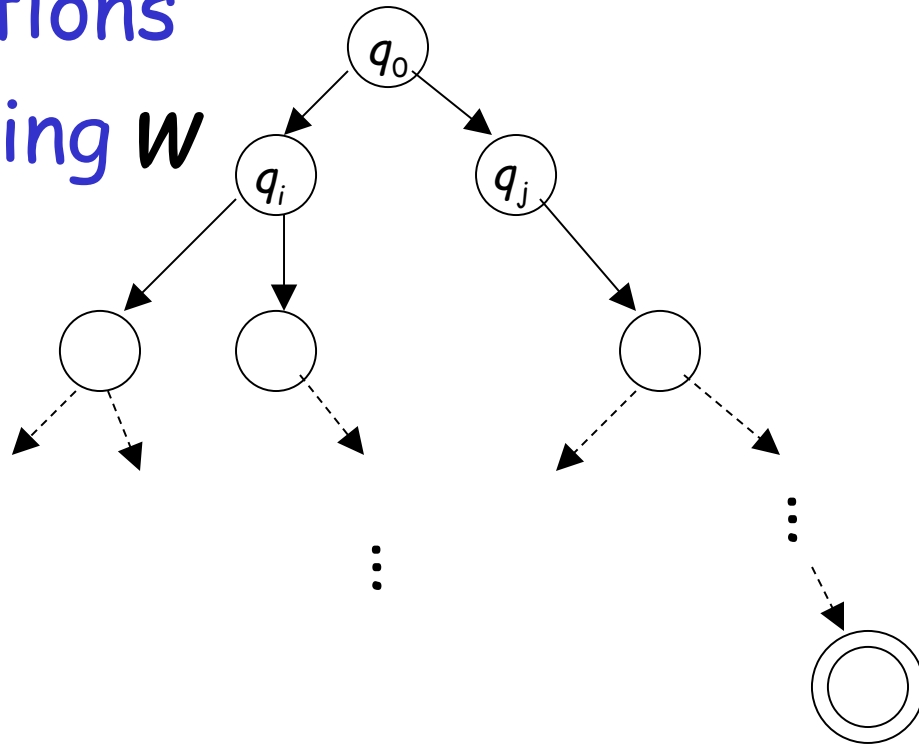
Take an arbitrary language $L \in NP$

We will give a polynomial reduction of L to SAT

Let M be the **NonDeterministic**
Turing Machine that decides L in polyn. time

For any string w we will construct
in polynomial time a **Boolean expression** $\varphi(M, w)$
such that: $w \in L \iff \varphi(M, w)$ is satisfiable

All computations
of M on string w

$$|w| = n$$


depth

$$n^k$$

accept

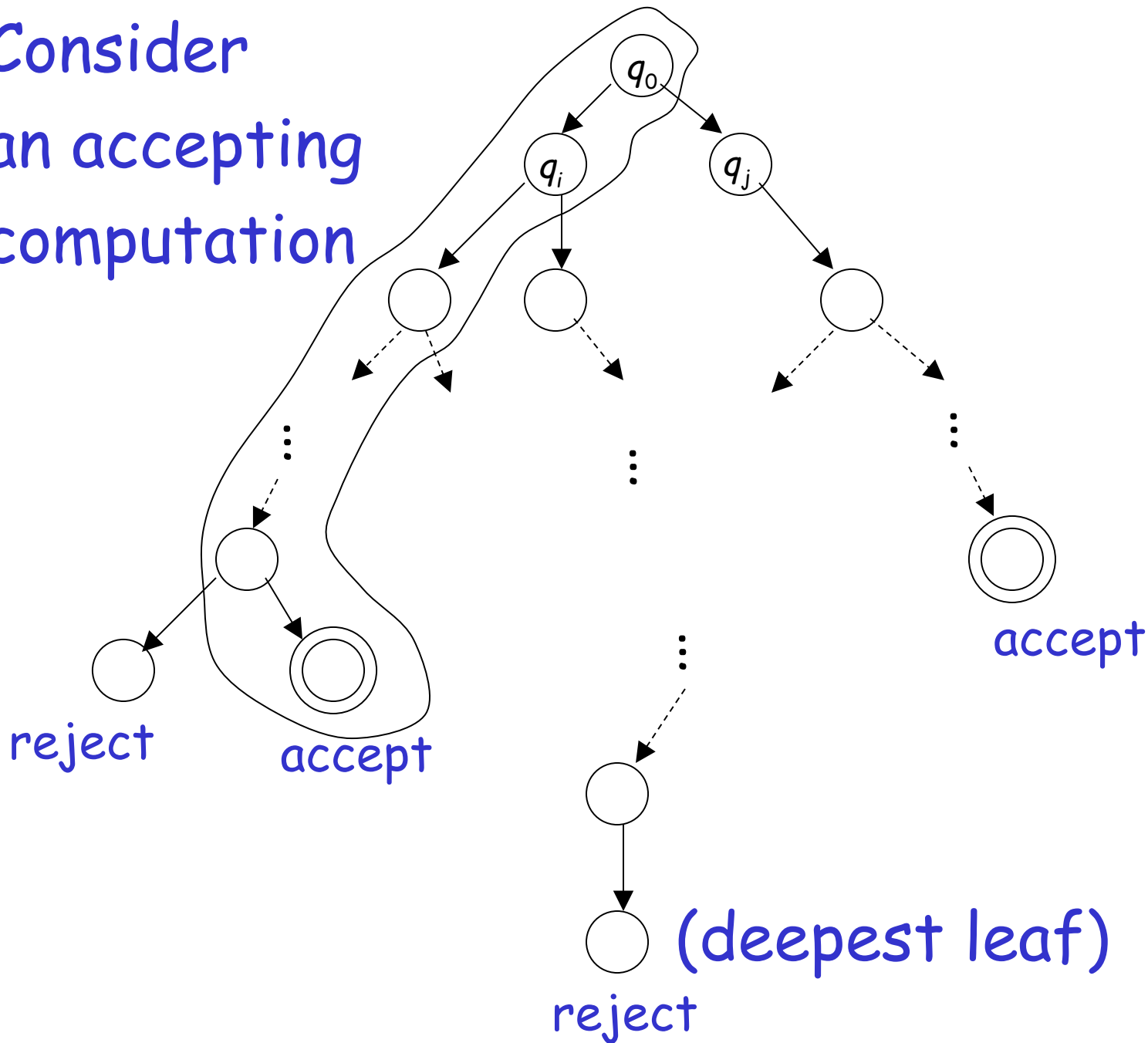
reject

accept

(deepest leaf)

reject

Consider
an accepting
computation



depth

n^k

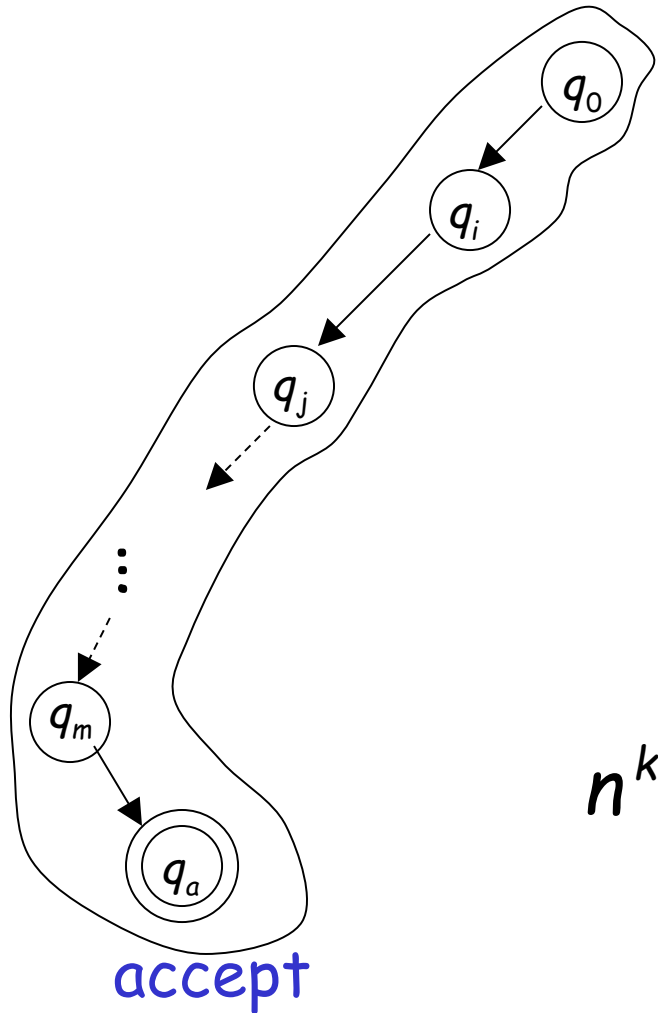
accept

accept

reject

(deepest leaf)
reject

Computation path



Sequence of Configurations

initial state

$$1: \quad \textcircled{q_0} \sigma_1 \sigma_2 \cdots \sigma_n$$

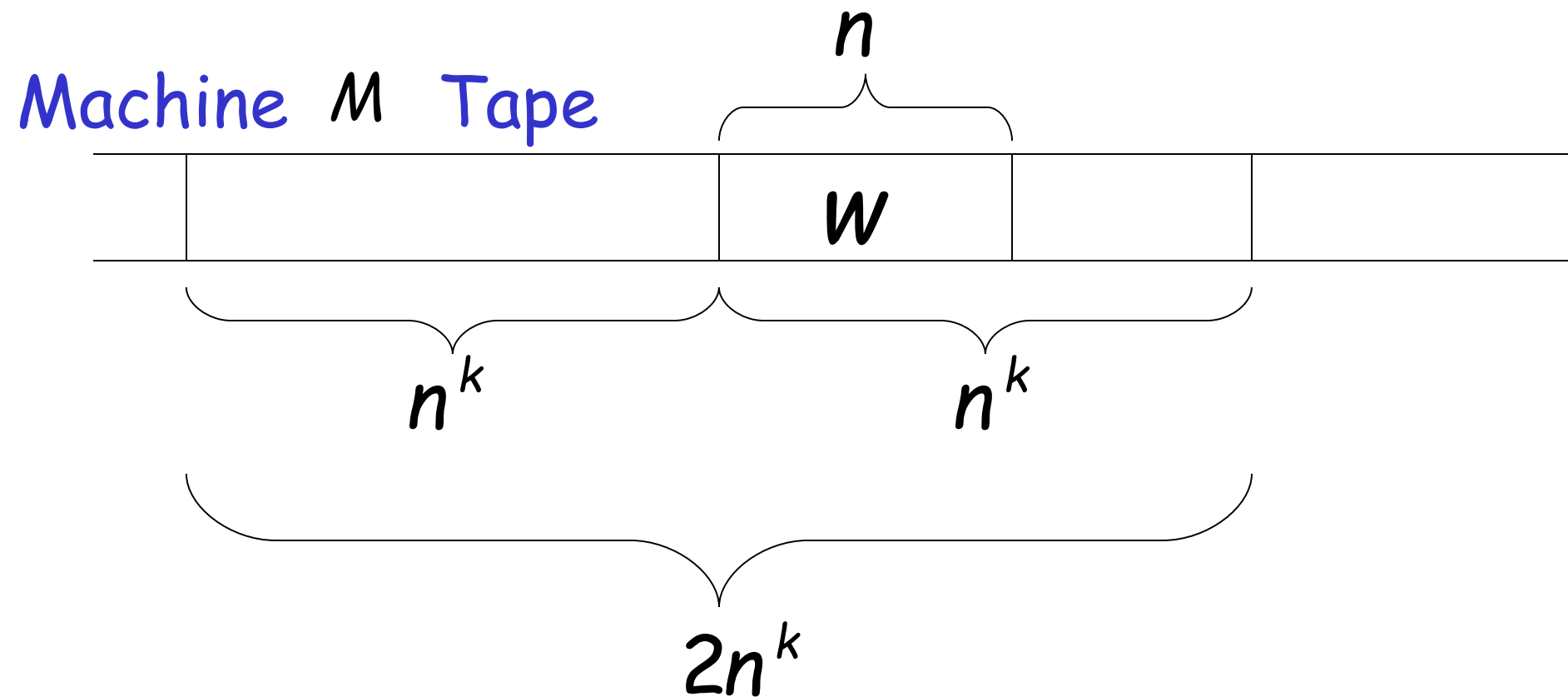
$$2: \quad \succ \sigma'_1 \textcircled{q_i} \sigma_2 \cdots \sigma_n$$

\vdots

$$n^k \geq x: \quad \succ \sigma'_1 \cdots \sigma'_l \textcircled{q_a} \sigma'_{l+1} \cdots \sigma'_{n^k}$$

accept state

$$W = \sigma_1 \sigma_2 \cdots \sigma_n$$



Maximum working space area on tape
during n^k time steps

Tableau of Configurations

1:	#	◇	...	◇	q_0	σ_1	σ_2	...	σ_n	◇	...	◇	#
2:	#	◇	...	◇	σ'_1	q_i	σ_2	...	σ_n	◇	...	◇	#
⋮													
x:	#	◇	...	σ''	σ'_1	σ'_2	σ'_3	...	q_a	σ'_{l+1}	...	σ_{n^k}	#
n^k :	#	◇	...	σ''	σ'_1	σ'_2	σ'_3	...	q_a	σ'_{l+1}	...	σ_{n^k}	#

Accept configuration

identical rows

$\longleftrightarrow n^k \longrightarrow$

$\longleftrightarrow n^k \longrightarrow$

$\longleftrightarrow 2n^k + 3 \longrightarrow$

Tableau Alphabet

$$\begin{aligned}\mathcal{C} &= \{\#\} \cup \{\text{tape alphabet}\} \cup \{\text{set of states}\} \\ &= \{\#\} \cup \{\alpha_1, \dots, \alpha_r\} \cup \{q_1, \dots, q_t\}\end{aligned}$$

Finite size (constant)

$$|\mathcal{C}| = O(1)$$

For every cell position i, j
and
for every symbol in tableau alphabet $s \in \mathcal{C}$

Define variable $x_{i,j,s}$

Such that if cell i, j contains symbol s

Then $x_{i,j,s} = 1$

Else $x_{i,j,s} = 0$

Examples:

Diagram illustrating a sequence of length $n^k + 3$. The sequence is shown in two rows:

- Row 1: $\#$, \diamond , \dots , \diamond , q_0 , σ_1 , σ_2 , \dots , σ_n , \diamond , \dots , \diamond , $\#$
- Row 2: $\#$, \diamond , \dots , \diamond , σ'_1 , q_i , σ_2 , \dots , σ_n , \diamond , \dots , \diamond , $\#$

Red circles highlight the first $\#$ and q_i in their respective rows, with lines pointing to the bottom of the diagram.

$$x_{1,1,\#} = 1$$

$$x_{1,1,\diamond} = 0$$

$$x_{2,n^k+3,q_i} = 1$$

$$x_{2,n^k+3,\#} = 0$$


$\varphi(M, w)$ is built from variables $x_{i,j,s}$

$$\varphi(M, w) = \varphi_{\text{cell}} \wedge \varphi_{\text{start}} \wedge \varphi_{\text{accept}} \wedge \varphi_{\text{move}}$$

When the formula is satisfied,
it describes an accepting computation
in the tableau
of machine M on input w

φ_{cell}

makes sure that every
cell in the tableau contains
exactly one symbol

$$\varphi_{\text{cell}} = \bigwedge_{\text{all } i,j} \left[\left(\bigvee_{s \in C} \mathbf{x}_{i,j,s} \right) \wedge \left(\bigwedge_{\substack{s,t \in C \\ s \neq t}} \left(\overline{\mathbf{x}_{i,j,s}} \vee \overline{\mathbf{x}_{i,j,t}} \right) \right) \right]$$


Every cell contains
at least one symbol

Every cell contains
at most one symbol

Size of φ_{cell} :

$$\varphi_{\text{cell}} = \bigwedge_{\text{all } i,j} \left[\left(\bigvee_{s \in \mathcal{C}} x_{i,j,s} \right) \wedge \left(\bigwedge_{\substack{s,t \in \mathcal{C} \\ s \neq t}} \left(\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}} \right) \right) \right]$$

$(2n^k + 3)n^k \times (|\mathcal{C}| + |\mathcal{C}|^2)$


$= O(n^{2k})$

φ_{start} makes sure that the tableau starts with the initial configuration

$$\begin{aligned} \varphi_{\text{start}} = & \mathbf{x}_{1,1,\#} \wedge \mathbf{x}_{1,2,\diamond} \wedge \cdots \wedge \mathbf{x}_{1,n^k+1,\diamond} \\ & \wedge \mathbf{x}_{1,n^k+2,q_0} \wedge \mathbf{x}_{1,n^k+3,\sigma_1} \wedge \cdots \wedge \mathbf{x}_{1,n^k+n+2,\sigma_n} \\ & \wedge \mathbf{x}_{1,n^k+n+3,\diamond} \wedge \mathbf{x}_{1,2n^k+2,\diamond} \wedge \cdots \wedge \mathbf{x}_{1,2n^k+2,\#} \end{aligned}$$

Describes the initial configuration
in row 1 of tableau

Size of φ_{start} :

$$\begin{aligned}\varphi_{\text{start}} &= \mathbf{x}_{1,1,\#} \wedge \mathbf{x}_{1,2,\diamond} \wedge \cdots \\ &\quad \wedge \mathbf{x}_{1,n^k+1,\diamond} \wedge \mathbf{x}_{1,n^k+2,q_0} \wedge \mathbf{x}_{1,n^k+3,\sigma_1} \wedge \cdots \\ &\quad \wedge \mathbf{x}_{1,2n^k+2,\diamond} \wedge \mathbf{x}_{1,2n^k+3,\#}\end{aligned}$$

$$2n^k + 3 = O(n^k)$$

φ_{accept} makes sure that the computation leads to acceptance

$$\varphi_{\text{accept}} = \bigvee_{\substack{\text{all } i,j \\ \text{all } q \in F}} X_{i,j,q}$$

Accepting states

An accept state should appear somewhere in the tableau

Size of φ_{accept} :

$$\varphi_{\text{accept}} = \bigvee_{\substack{\text{all } i,j \\ \text{all } q \in F}} X_{i,j,q}$$



$$(2n^k + 3)n^k = O(n^{2k})$$

φ_{move} makes sure that the tableau
gives a valid sequence
of configurations

φ_{move} is expressed in terms of
legal windows

Tableau

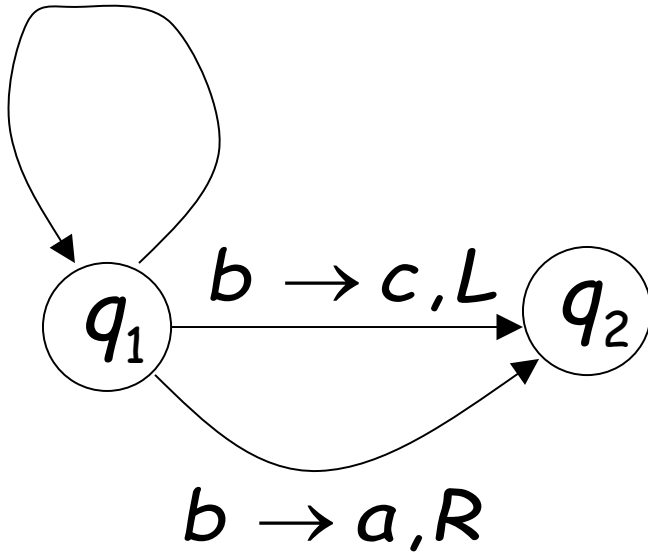
Window

a	q_1	b
q_2	a	c

2x6 area of cells

Possible Legal windows

$a \rightarrow b, R$



a	q_1	b
q_2	a	c

a	q_1	b
a	a	q_2

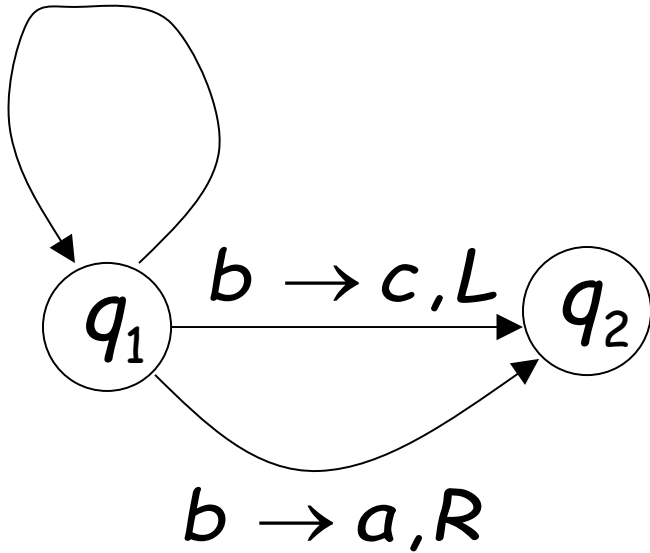
a	a	q_1
a	a	b

a	b	a
a	b	q_2

Legal windows obey the transitions

Possible illegal windows

$a \rightarrow b, R$



a	b	a
a	a	a

a	q_1	b
q_1	a	a

b	q_1	b
q_2	b	q_2

$$\varphi_{\text{move}} = \bigwedge_{\text{all } i,j} (\text{window } (i,j) \text{ is legal})$$

window (i,j) is legal:

	j		j		j		
i	a	q_1	b	i	a	q_1	b
	q_2	a	c		a	a	q_2
	((is legal) \vee (is legal) \vee (is legal))						

all possible legal windows
in position (i,j)

	j		
i	a	q_1	b
	q_2	a	c

(is legal)

Formula:

$$\begin{aligned}
 & x_{i,j,a} \wedge x_{i,j+1,q_1} \wedge x_{i,j+2,b} \\
 & \wedge x_{i+1,j,q_2} \wedge x_{i+1,j+1,a} \wedge x_{i+1,j+2,c}
 \end{aligned}$$

Size of φ_{move} :

Size of formula for a legal window
in a cell i,j : 6

Number of possible legal windows
in a cell i,j : at most $|C|^6$

Number of possible cells: $(2n^k + 3)n^k$

$$\leq |C|^6 \cdot (2n^k + 3)n^k = O(n^{2k})$$

Size of $\varphi(M, w)$:

$$\begin{aligned}\varphi(M, w) &= \varphi_{\text{cell}} \wedge \varphi_{\text{start}} \wedge \varphi_{\text{accept}} \wedge \varphi_{\text{move}} \\ &\quad \swarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ &O(n^{2k}) + O(n^k) + O(n^{2k}) + O(n^{2k}) \\ &= O(n^{2k})\end{aligned}$$

it can also be constructed in time $O(n^{2k})$
polynomial in n

$$\varphi(M, w) = \varphi_{\text{cell}} \wedge \varphi_{\text{start}} \wedge \varphi_{\text{accept}} \wedge \varphi_{\text{move}}$$

we have that:

$$w \in L \iff \varphi(M, w) \text{ is satisfiable}$$

Since,

$w \in L \iff \varphi(M, w)$ is satisfiable

and

$\varphi(M, w)$ is constructed
in polynomial time



L is polynomial-time reducible to SAT

END OF PROOF

Observation 1:

The $\varphi(M, w)$ formula can be converted to CNF (conjunctive normal form) formula in polynomial time

$$\varphi(M, w) = \varphi_{\text{cell}} \wedge \varphi_{\text{start}} \wedge \varphi_{\text{accept}} \wedge \varphi_{\text{move}}$$

Already CNF



NOT CNF



But can be converted to CNF using distributive laws

Distributive Laws:

$$P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$$

$$P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$$

Observation 2:

The $\varphi(M, w)$ formula can also be converted to a 3CNF formula in polynomial time

$$(a_1 \vee a_2 \vee \cdots \vee a_l)$$

convert

$$(a_1 \vee a_2 \vee z_1) \wedge (\overline{z_1} \vee a_3 \vee z_2) \wedge (\overline{z_2} \vee a_4 \vee z_3) \wedge \cdots \wedge (\overline{z_{l-3}} \vee a_{l-1} \vee z_l)$$

From Observations 1 and 2:

CNF-SAT and
3CNF-SAT are
NP-complete languages

(they are known NP languages)

Theorem:

- If:
- a. Language A is NP-complete
 - b. Language B is in NP
 - c. A is polynomial time reducible to B

Then: B is NP-complete

Proof:

Any language L in NP

is polynomial time reducible to A .

Thus, L is polynomial time reducible to B

(sum of two polynomial reductions,
gives a polynomial reduction)

Corollary: CLIQUE is NP-complete

Proof:

- a. 3CNF-SAT is NP-complete
- b. CLIQUE is in NP (shown in last class)
- c. 3CNF-SAT is polynomial reducible to CLIQUE (shown earlier)

Apply previous theorem with

A=3CNF-SAT and B=CLIQUE