

La classe NP
Non-Deterministic Polynomial time

Per ogni k

$$NP = \cup TIME(n^k)$$

La classe P
Deterministic Polynomial time

Per ogni k

$$P = \cup TIME(n^k)$$

Linguaggi NP-completi

Una funzione $f:A \rightarrow B$

è calcolabile in tempo polinomiale se esiste una macchina M che calcola la funzione f in tempo polinomiale

$$w \longrightarrow wOf(w)$$

Un linguaggio A è riducibile in tempo

polinomiale a un linguaggio B , $A \leq B$, se esiste una funzione polinomiale f tale che per ogni x : $x \in A$ implica $f(x) \in B$

Nota che se $A \leq B$ e B è polinomiale
allora anche A è polinomiale.

Sia M che decide B ,

Sia f che riduce A a B

Sia x elemento di A

1. Calcola $f(x)$
2. Calcola $M(f(x))$

$A \leq B$

$\text{Pari} \leq \text{Dispari}$

$\text{Dispari} \leq \text{Pari}$

X è elemento di Pari

(div 2 (resto=0))

Dispari

X è elemento di dispari

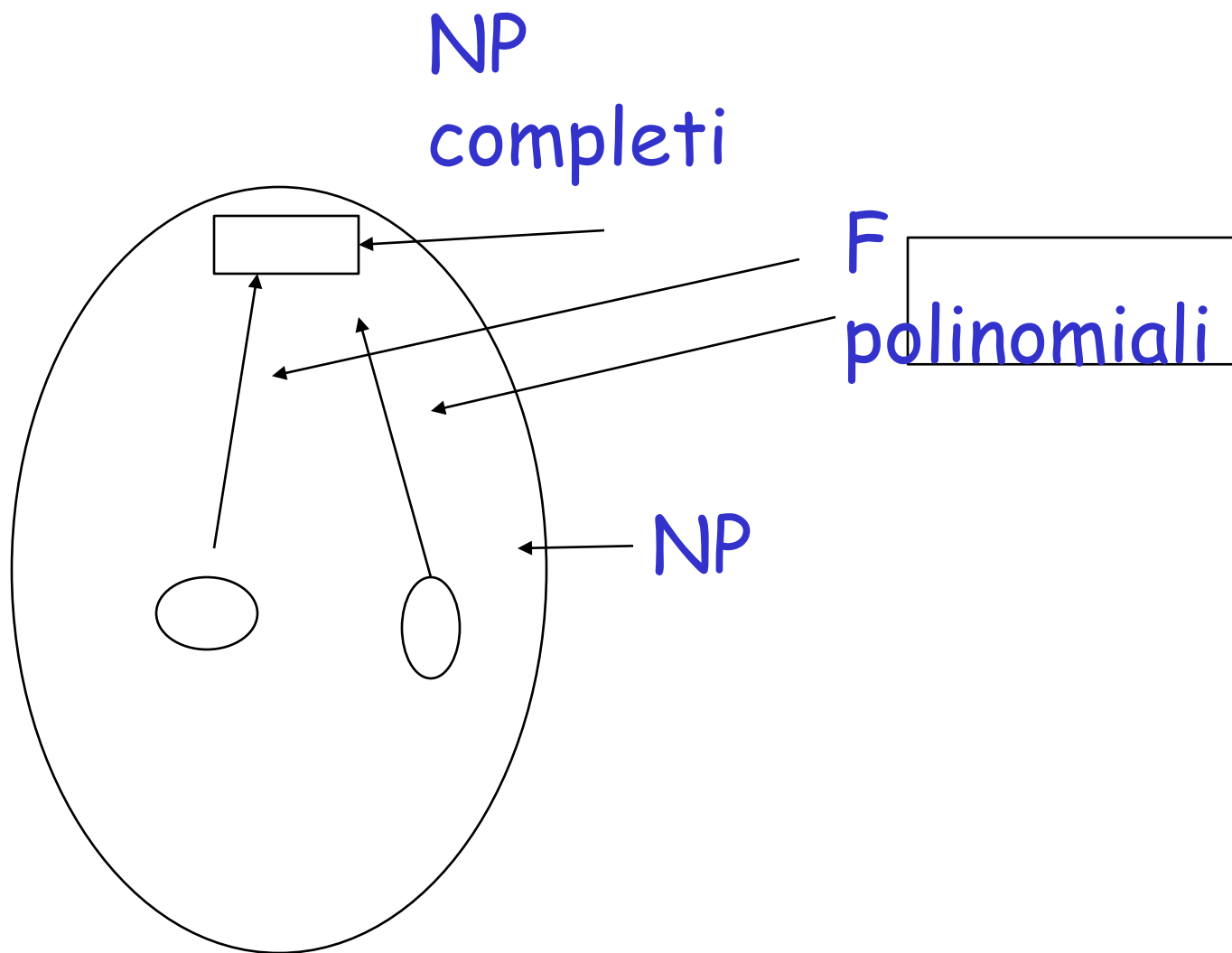
Non (div 2)

NP-Completezza

Un problema A è NP-completo se:

- A è in NP
- Ogni problema NP è riducibile ad A

(in tempo polinomiale)



Osservazione:

Se possiamo risolvere un problema
NP-complete
in tempo Deterministico Polinomiale
allora :

$$P = NP$$

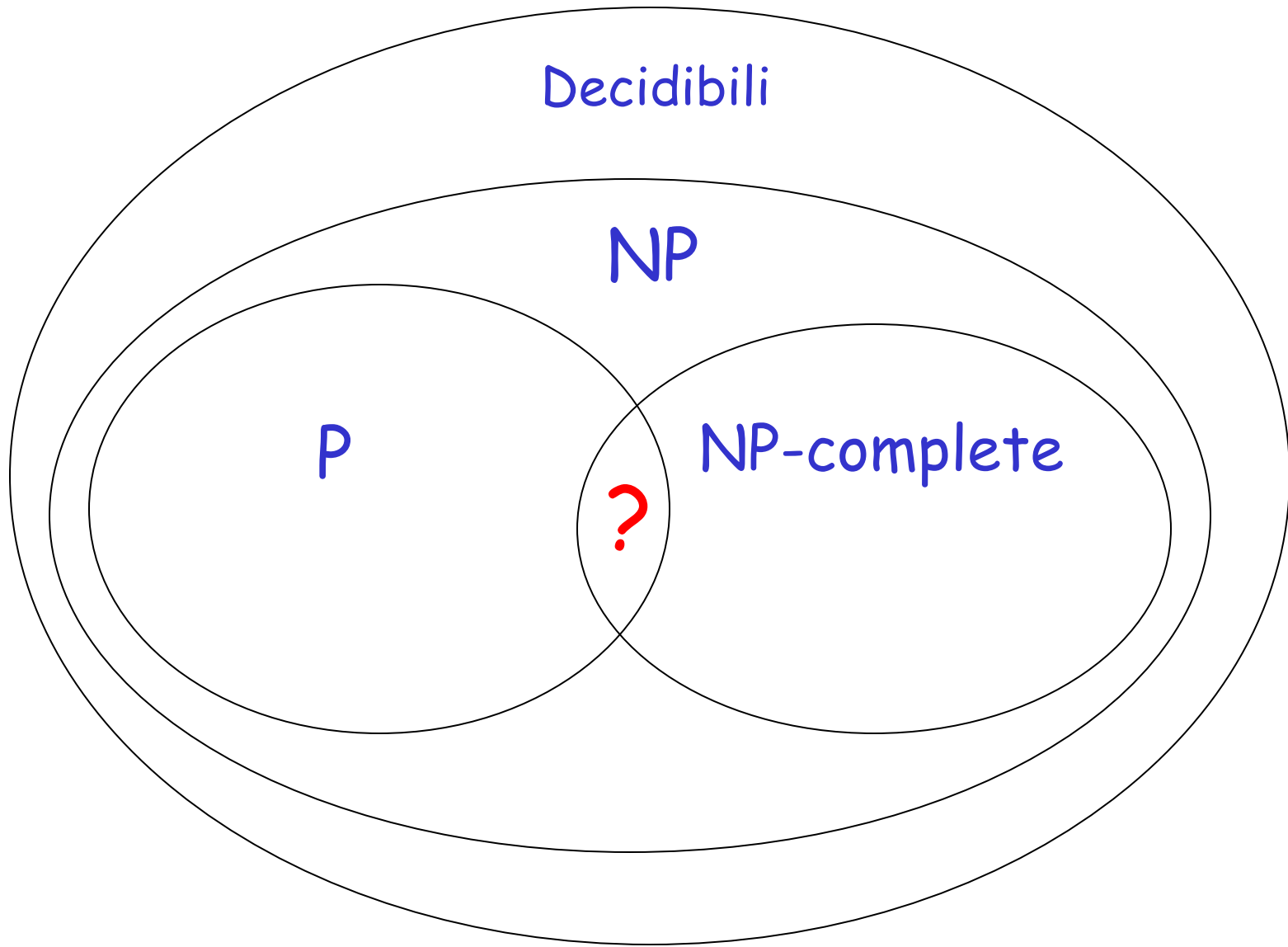
Osservazione :

Se proviamo che
non possiamo risolvere un problema
NP-complete
in tempo Deterministico Polinomiale
Allora :

$$P \neq NP$$

Un Linguaggio L è NP-completo se:

- L è in NP, e
- Ogni Linguaggio in NP può essere ridotto a L in Tempo Polinomiale



Formule SAT:

literal
↙

$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6} \vee x_4) \wedge (x_4 \vee x_5 \vee x_6)$$

L: letterale o letterale negato

O: Gruppi di L collegati da \vee

Sat: Gruppi di O collegati da \wedge

Un Linguaggio NP-completo

Teorema di Cook-Levin :

Linguaggio SAT (satisfiability problem)
è NP-complete

Dim:

Part1: SAT è in NP
(già provato)

Part2: ridurre tutti i Linguaggi NP
al problema SAT
in Tempo Polinomiale

Sia $L \in NP$, un arbitrario linguaggio

Definiamo una riduzione Polinomiale of L to SAT

Sia M Macchina di Turing Nondeterministica che decide L in Tempo polinomiale

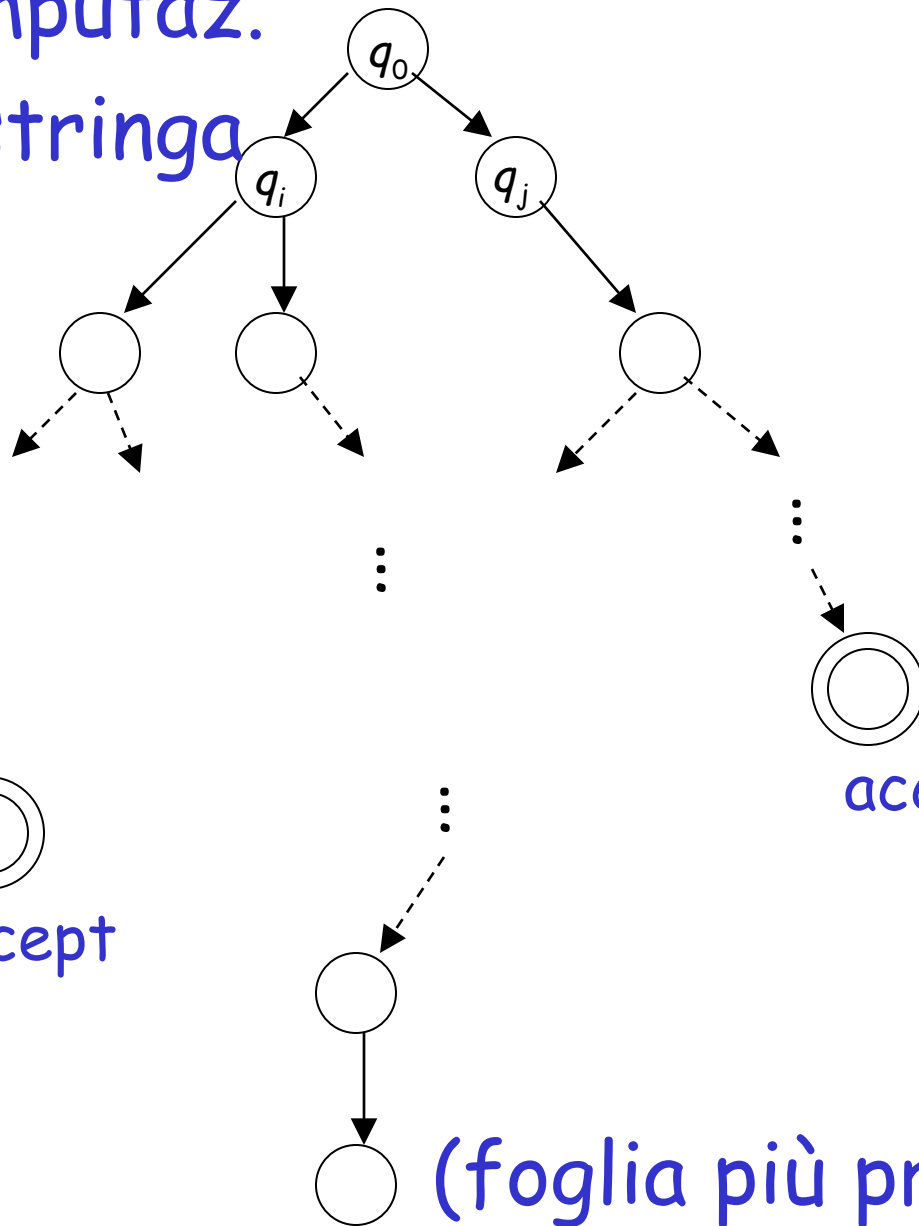
Per ogni stringa w costruiamo in Tempo

Polinomiale una **espressione Booleana** $\varphi(M, w)$

tale che: $w \in L \iff \varphi(M, w)$ è soddisfacibile

Tutte le computaz.
of M sulla stringa
 w

$$|w| = n$$



Prof
ondità
 n^k

reject

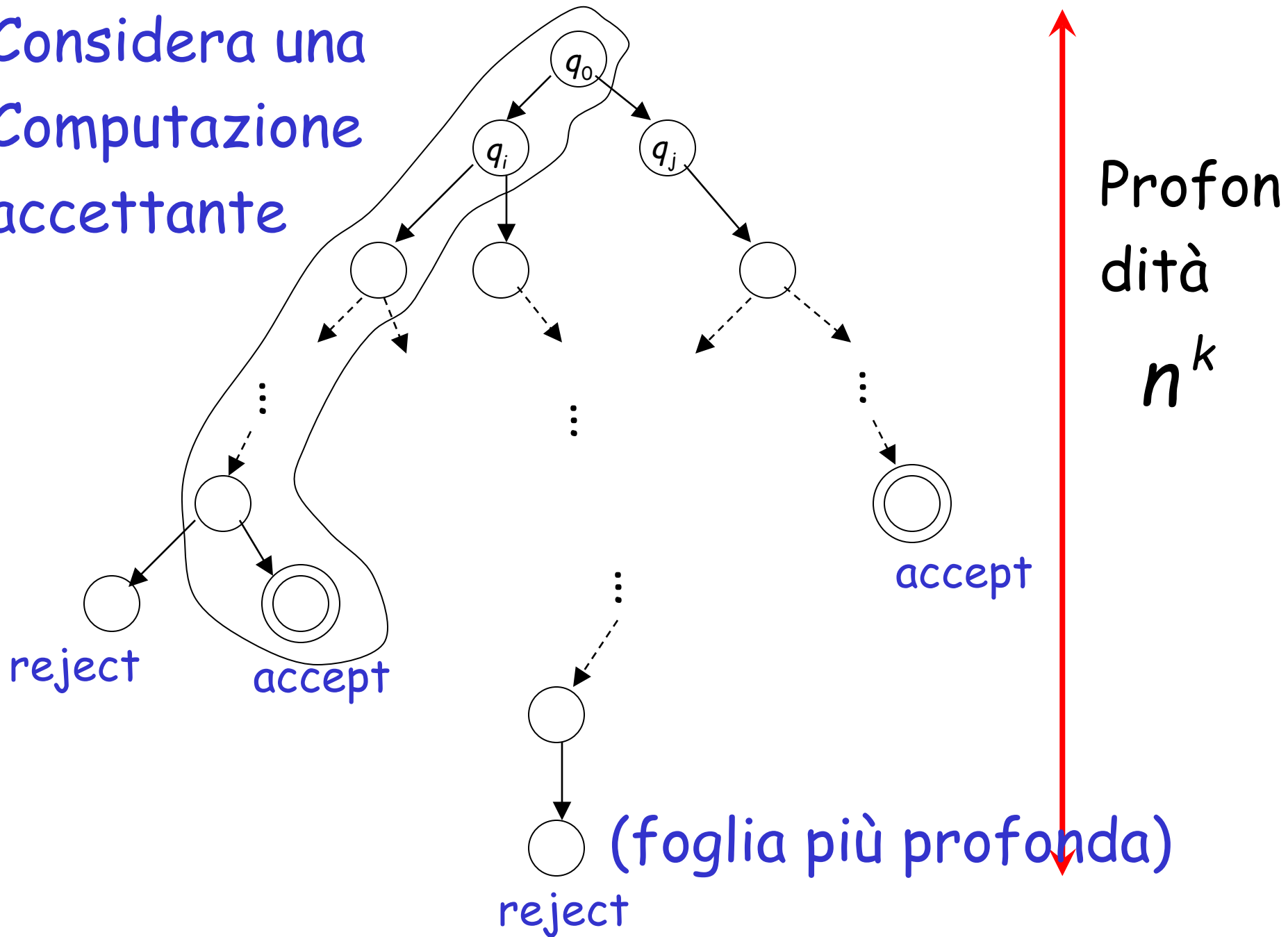
accept

accept

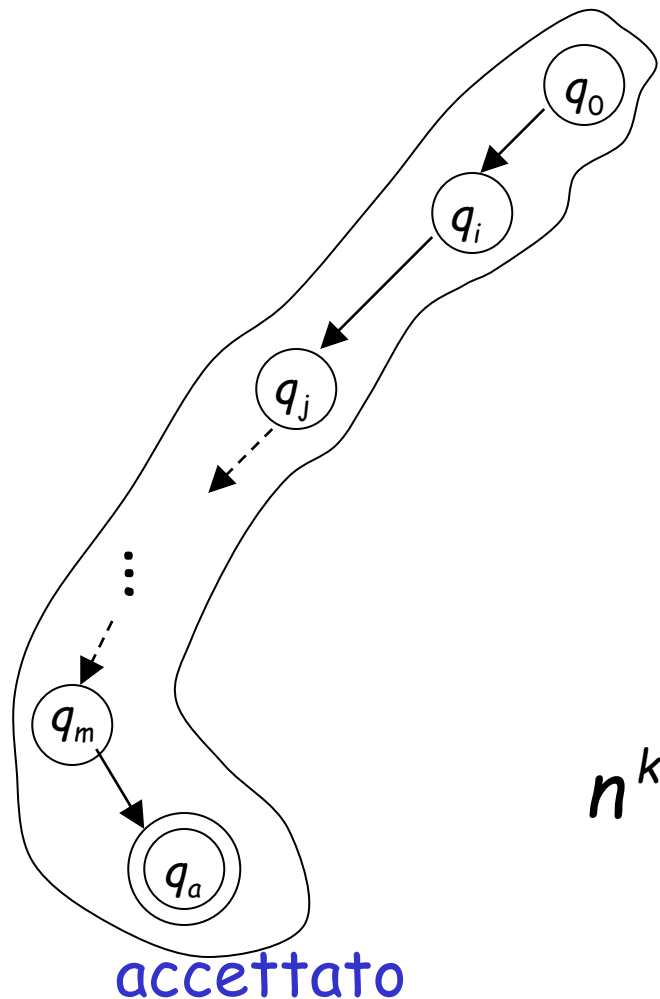
reject

(foglia più profonda)

Considera una
Computazione
accettante



Cammino di computazione



Sequenze di
configurazioni

Stato iniziale

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

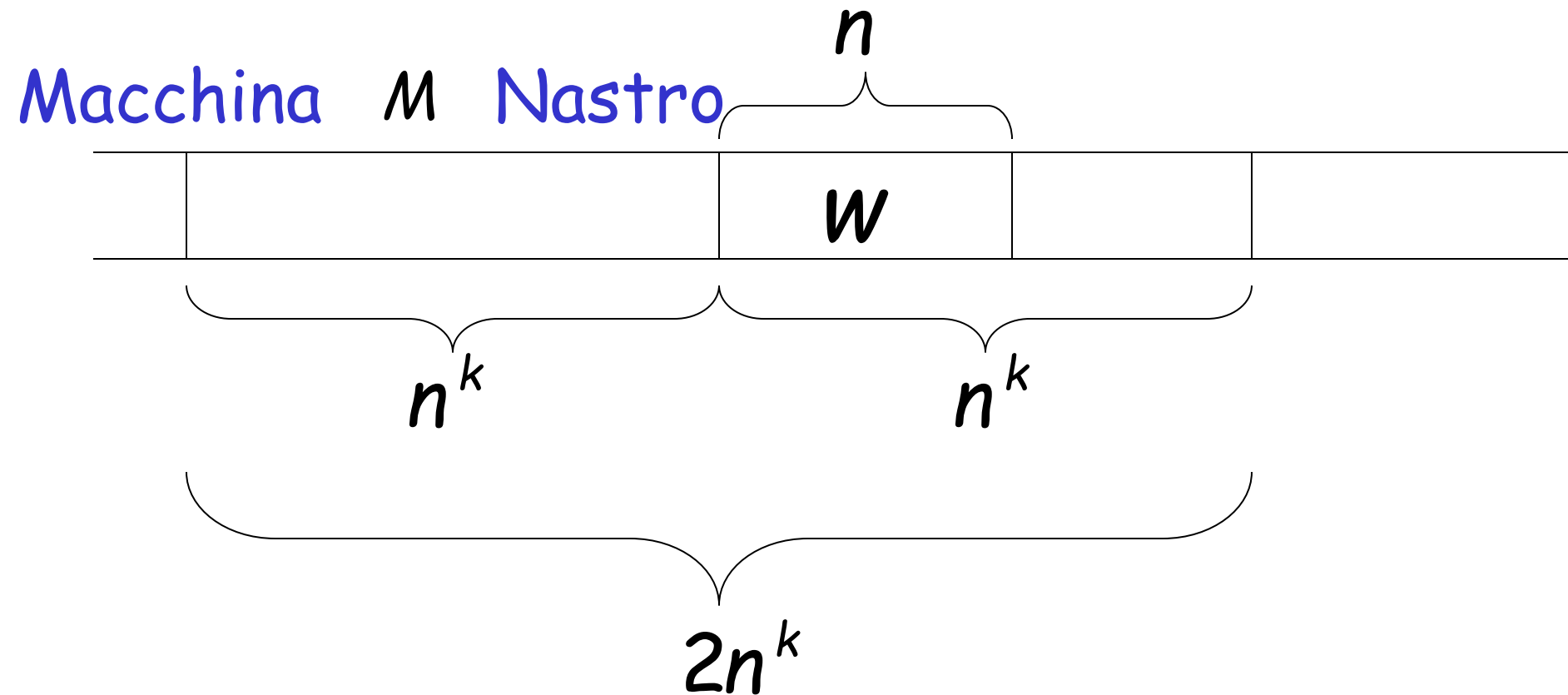
$$2: \quad \succ \sigma'_1 \textcircled{q_i} \sigma_2 \cdots \sigma_n$$

$$\vdots$$

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

Stato accettazione

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



Massima area di calcolo sul nastro

durante n^k steps (passi) temporali

Tableau delle configurazioni

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

Configurazione accettante

Righe identiche

$\longleftarrow n^k \longrightarrow$ $\longleftarrow n^k \longrightarrow$

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

Alfabeto del Tableau

$$\begin{aligned} C &= \{\#\} \cup \{\text{alfabeto nastro}\} \cup \{\text{insieme degli stati}\} \\ &= \{\#\} \cup \{\alpha_1, \dots, \alpha_r\} \cup \{q_1, \dots, q_t\} \end{aligned}$$

Dimensione finita (costante)

Per ogni cella con posizione i, j
E per ogni simbolo nell'alfabeto
del tableau $s \in \mathcal{C}$

Definiamo la variabile $x_{i,j,s}$

tale che se la cella i, j contiene il simbolo s
allora $x_{i,j,s} = 1$ altrimenti $x_{i,j,s} = 0$

Esempio:

[illegible]

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

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

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

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

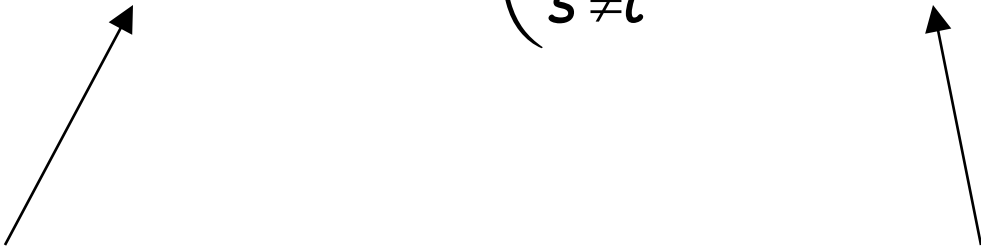
$\varphi(M, w)$ è costruito con le variabili $x_{i,j,s}$

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

Quando la formula è soddisfatta allora descrive
una computazione di accettazione nel
tableau della macchina M su input w

φ_{cell}

Ci rende sicuri che ogni cella
nel tableau contiene esattamente
un simbolo

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


Ogni cella contiene
almeno un simbolo

Ogni cella contiene
al massimo un simbolo

Dimensione di φ_{cell} :

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

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

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

φ_{start} il tableau parte con la configurazione iniziale

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

Descrive la configurazione iniziale
Nella riga 1 del tableau

Dimensione di φ_{start} :

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

\downarrow

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

φ_{accept} Ci da la sicurezza che La
computazione raggiunge stato di
Accettazione

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

Stati di accettazione

Uno stato di accettazione deve apparire
Da qualche parte nel tableau

Size of φ_{accept} :

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



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

φ_{move} Ci rende sicuri che il tableau
ci da una sequenza valida di
configurazioni

φ_{move} è espresso in termini di
windows legali

Tableau

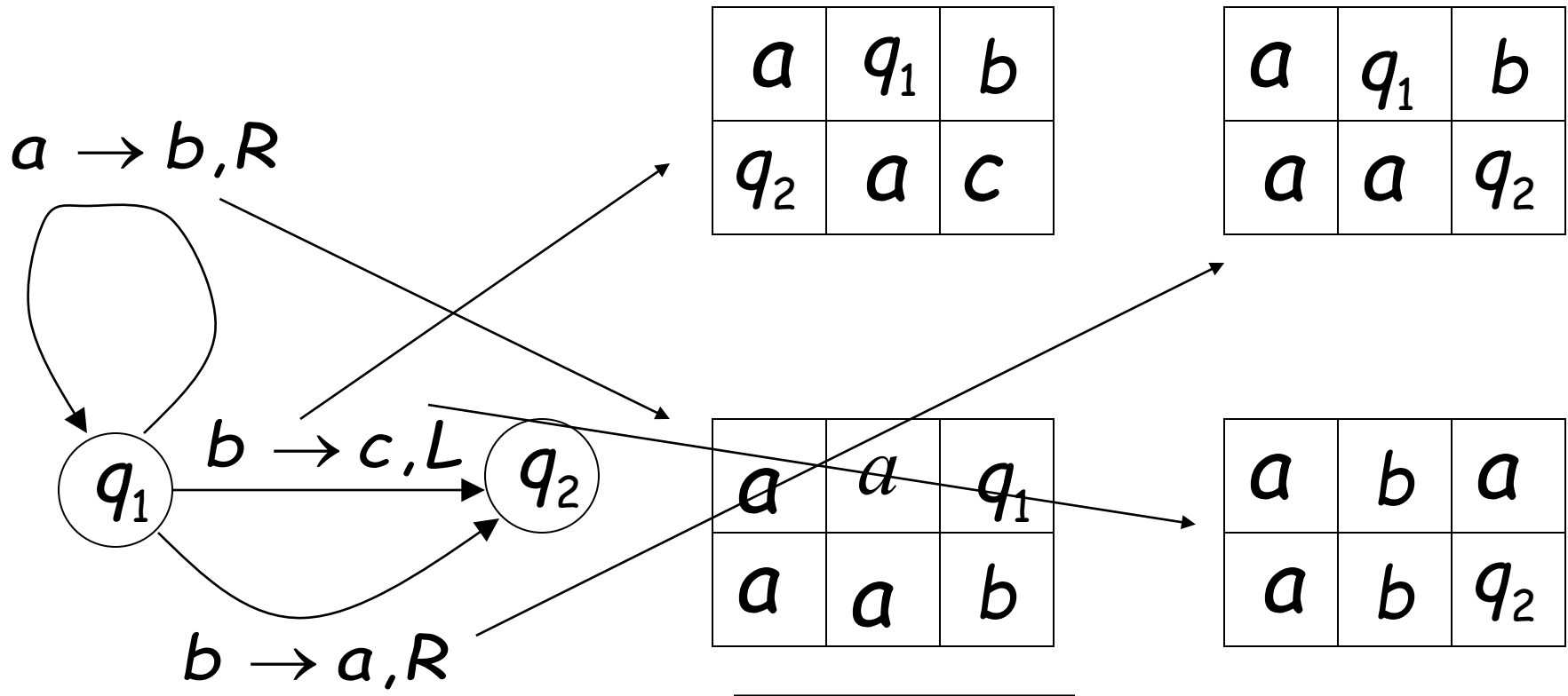
Per ogni
 i, j ,
elemento
centrale
superiore

Window

a	q_1	b
q_2	a	c

2x6 area di celle

Possibili windows Legali



window Legali obbediscono alle transizioni

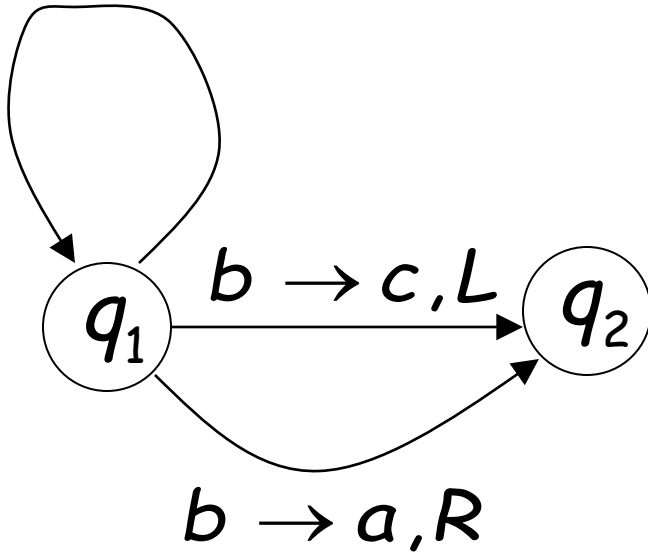
Altre finestre legali

#	b	a
#	b	a

b	b	b
c	b	b

Possibili window illegali

$a \rightarrow b, R$



a	b	a
a	a	a

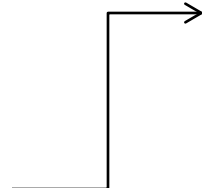


La b in a ?

a	q_1	b
q_1	a	a



b	q_1	b
q_2	b	q_2



Due stati

Se $\delta(q_1, b) = (q_1, c, R)$

	j		
i	a	q_1	b
	q_2	a	c

(è legale)

Formula:

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

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

window (i,j) è legale:

	j	
i	a	q_1
	q_2	c

	j	
i	a	q_1
	a	q_2

	j	
i	a	a
	a	q_1

((is legal) e

(is legal) e

(is legal)

...

Tutte le possibili celle legali
nella posizione (i,j)

Dimensione di φ_{move} :

Dimensione di una formula per una window

Legale in una cella i,j : **6**

X Numero di possibili legal window
in una cella i,j : al max $|C|^6$

X Numero di possibili celle: $(2n^k + 3)n^k$

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

Dimensione di $\varphi(M, w)$:

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

Costruita in Tempo $O(n^{2k})$

Quindi Polinomiale in n

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

Abbiamo che:

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

poichè,

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

e

$\varphi(M, w)$ è costruita
in Tempo Polinomiale



L è riducibile in tempo Polinomiale a SAT

END OF Dim

Osservazione 1:

La $\varphi(M, w)$ formula può essere convertita
in CNF (conjunctive normal form) formula
in Tempo Polinomiale

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

già CNF



NOT CNF

Ma può essere convertita
in CNF (distribuitività)

leggi: Distributività

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

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

Osservazione 2:

La $\varphi(M, w)$ formula può essere
Convertita in una formula 3CNF
in Tempo Polinomiale

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

Conv
erti

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

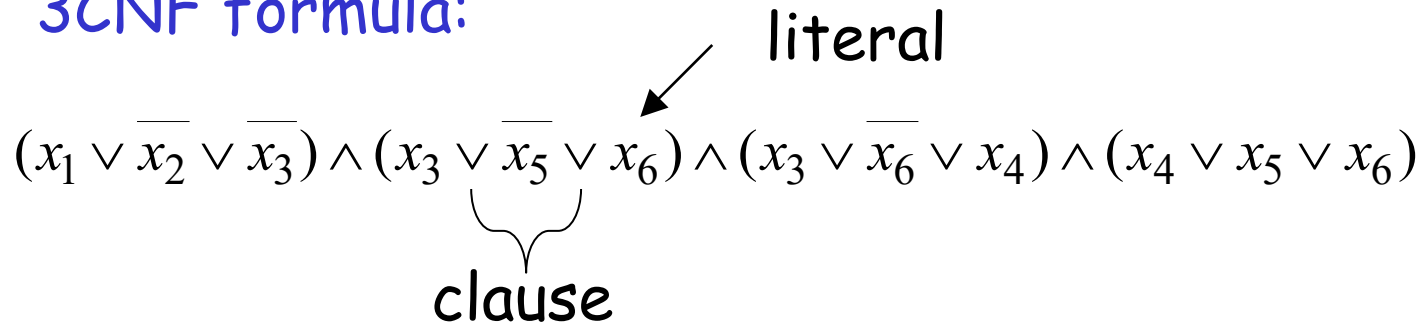
Nuove z

3CNF formula:

$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6} \vee x_4) \wedge (x_4 \vee x_5 \vee x_6)$$

literal

clause



Ogni clausola ha tre letterali

Linguaggio:

3CNF-SAT = { w : w è una formula
3CNF soddisfacibile }

CNF-SAT e
3CNF-SAT sono
Linguaggi NP-completi

(sappiamo che sono NP Linguaggi)

Esempio di riduzione Tempo-Polinomiale:

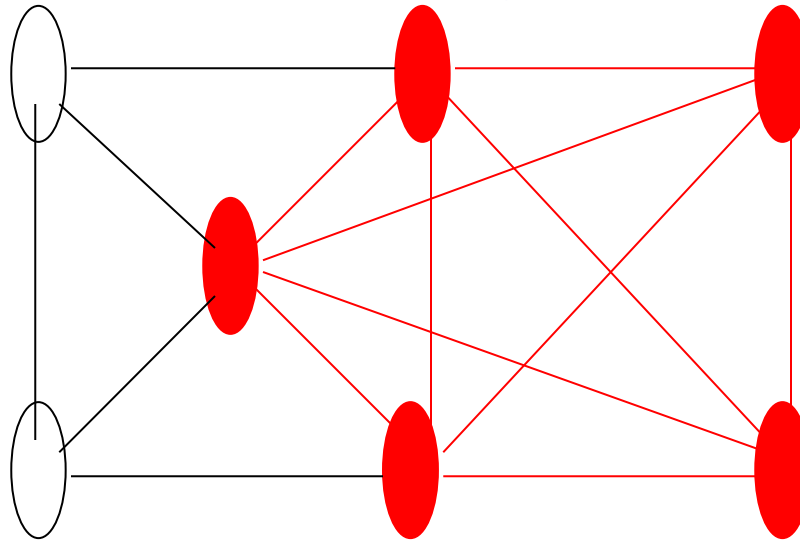
Ridurremo il problema

3CNF-satisfiability

al

problema CLIQUE

Una 5-cricca (Clique) nel grafo G



Linguaggio:

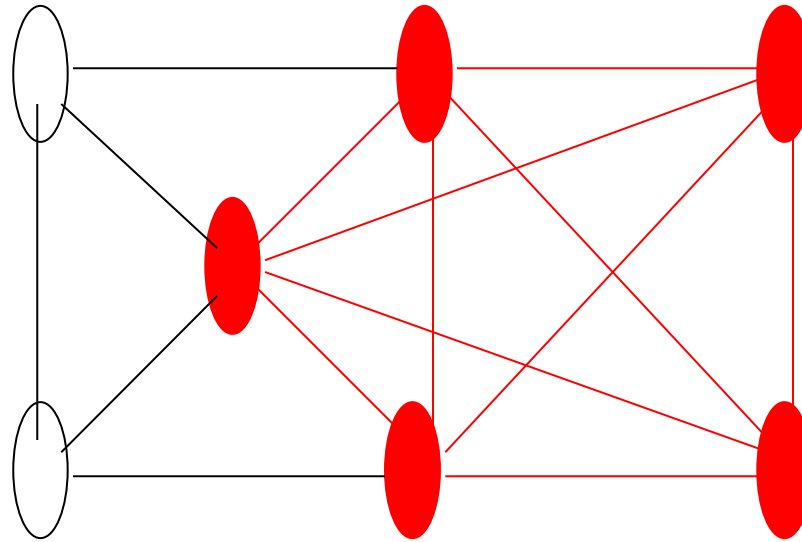
$CLIQUE = \{ \langle G, k \rangle : \text{grafo } G \text{ contiene un } k\text{-clique} \}$

Teorema: 3CNF-SAT è riducibile
In Tempo Polinomiale
a CLIQUE

Dim: Una riduzione in Tempo Polinomiale
da un problema all'altro

Transformare una formula in un grafo

Una 5-cricca (Clique) nel grafo G



Linguaggio:

$CLIQUE = \{ \langle G, k \rangle : \text{grafo } G \text{ contiene un } k\text{-clique} \}$

Teorema: 3CNF-SAT è riducibile
In Tempo Polinomiale
a CLIQUE

Dim: Una riduzione in Tempo Polinomiale
da un problema all'altro

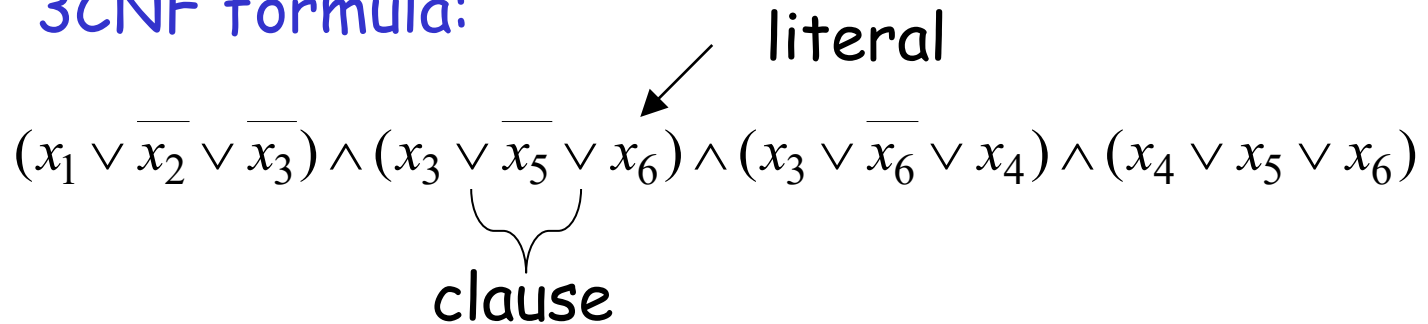
Transformare una formula in un grafo

3CNF formula:

$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6} \vee x_4) \wedge (x_4 \vee x_5 \vee x_6)$$

literal

clause



Ogni clausola ha tre letterali

Linguaggio:

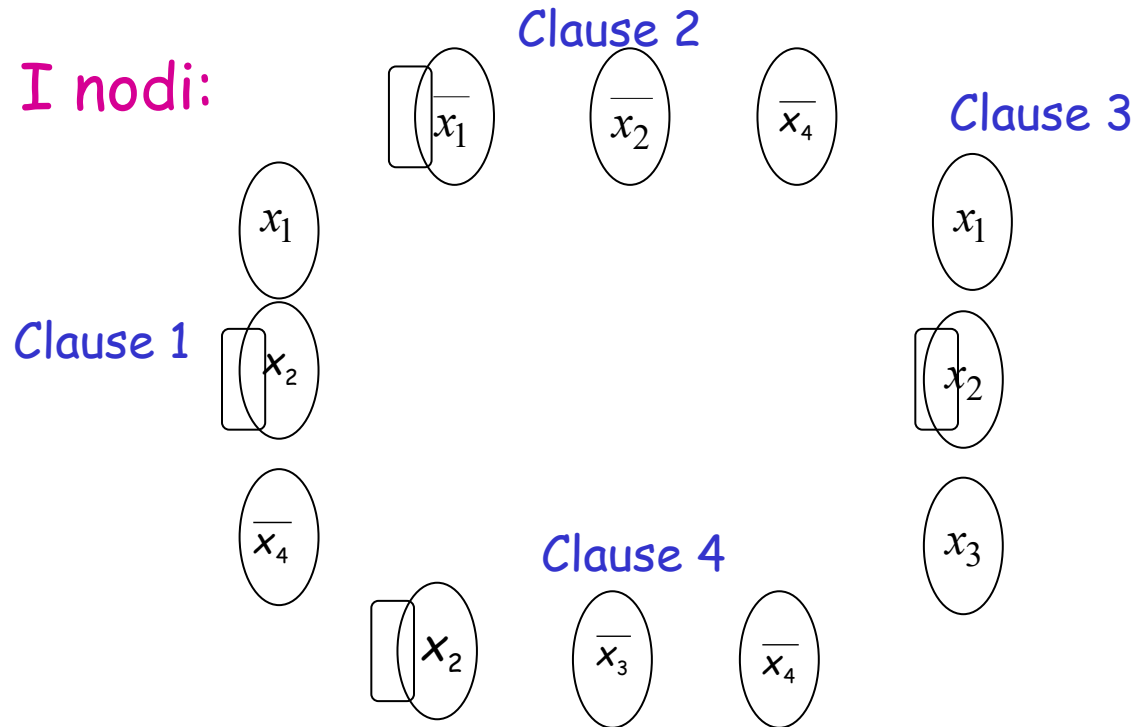
3CNF-SAT = { w : w è una formula
3CNF soddisfacibile }

Transformare una formula in un grafo

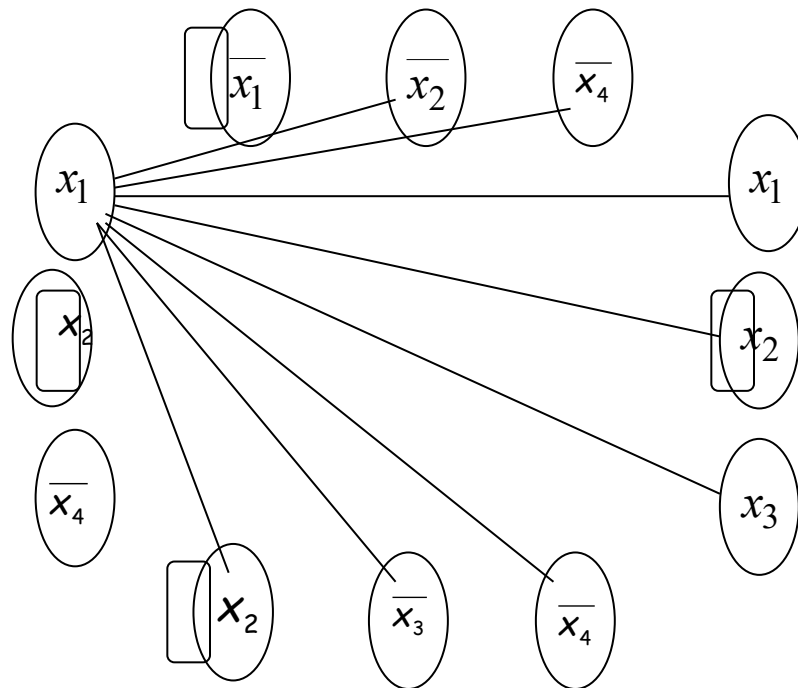
Esempio:

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

Creare i nodi:

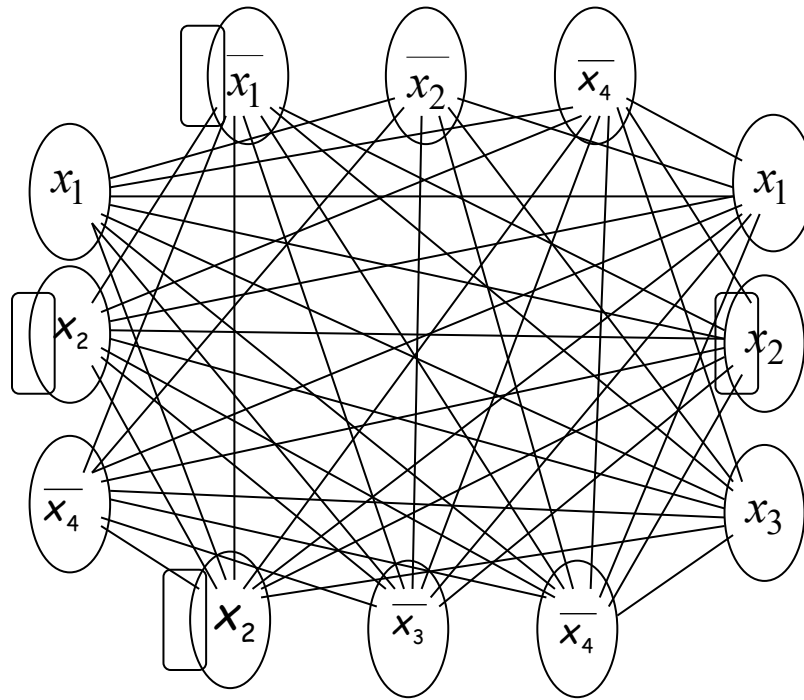


$$(x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$$



Addizionare un arco da un letterale ξ
 A ogni altro letterale in ogni clausola salvo a ξ

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



Grafo risultante

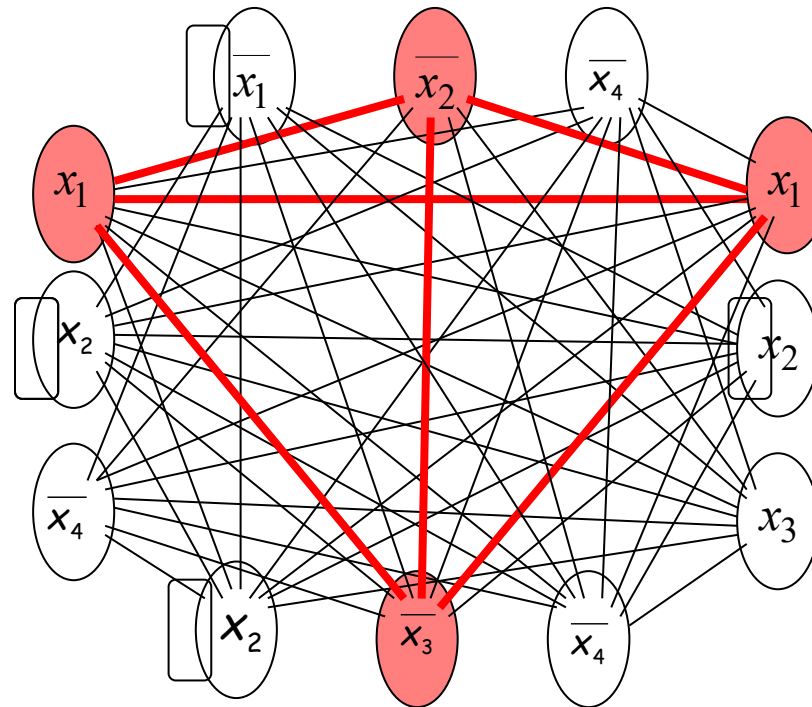
$$(x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4}) = 1$$

$$x_1 = 1$$

$$x_2 = 0$$

$$x_3 = 0$$

$$x_4 = 1$$



La formula è soddisfacibile se e solo se il grafo ha
un 4-clique

End of Dim

Teorema:

If: a. Linguaggio A is NP-complete

b. Linguaggio B is in NP

c. A è riducibile Tempo Polinomiale a B

Then: B is NP-complete

Dim:

Ogni linguaggio L in NP

È riducibile in Tempo Polinomiale a A

allora, L è riducibile in Tempo Polinomiale

a B

(somma di due riducibilità Polinomiali dà una riduzione Polinomiale)

Corollario: CLIQUE è NP-complete

Dim:

- a. 3CNF-SAT è NP-complete
- b. CLIQUE è in NP (già visto)
- c. 3CNF-SAT è riducibile Polinomiale a CLIQUE (già visto)

Applichiamo il teorema precedente

$A=3\text{CNF-SAT}$ e $B=\text{CLIQUE}$

Fine chap

Osservazione:

SE si dimostra che un Linguaggio
NP-complete
è in P allora:

$$P = NP$$