

Esercizio 1. Scrivere le espressioni regolari corrispondenti ai linguaggi, con  $\Sigma = \{0,1\}$

- $\{w \mid w \text{ contiene esattamente 1}\} \quad 0^* 1 0^*$
- $\{w \mid w \text{ contiene almeno un 1}\} \quad (0+1)^* 1 (0+1)^* \text{ oppure } 0^* 1 (0+1)^* \text{ oppure } (0+1)^* 1 0^*$
- $\{w \mid w \text{ contiene la stringa 001}\} \quad (0+1)^* 001 (0+1)^*$
- $\{w \mid |w| \text{ è multiplo di 3}\} \quad ((0+1)(0+1)(0+1))^*$

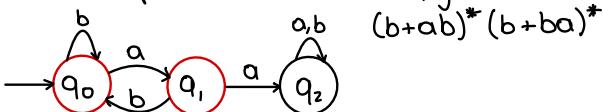
Esercizio 2: Si dica se sono regolari i linguaggi sull'alfabeto  $\Sigma = \{a,b\}$  formati da:

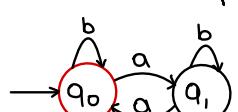
- tutte le parole con non più di tre a: SI  $\leftarrow b^*(a+\epsilon)b^*(a+\epsilon)b^*(a+\epsilon)b^*$  oppure  $b^* + b^*ab^* + b^*ab^*ab^* + b^*ab^*ab^*ab^*$
- tutte le parole con esattamente una occorrenza di aa: SI  $\leftarrow (a+ab)^*aa(b+ba)^*$
- tutte le parole con lo stesso numero di a e di b  $\leftarrow \text{NO} \leftarrow (ab)^*?$  no! perché  $ba \notin (ab)^*$   $aabb \notin (ab)^*$  ...

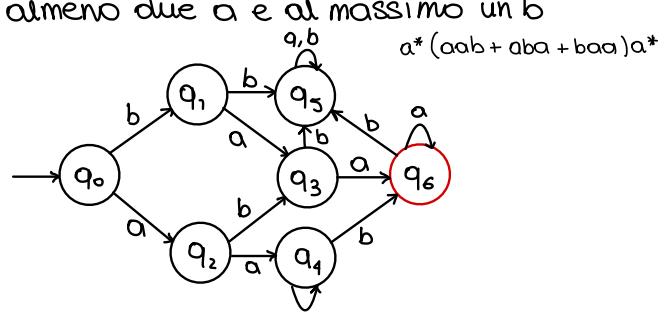
Esercizio 3: quali tra le seguenti affermazioni sono vere:

- $baa \in a^*b^*a^*b^*$  VERO
- $b^*a^* \cap a^*b^* = a^* \cup b^*$  VERO  $a^*b^* \notin 1,3$   $b^*a^* \notin 2,3$   $b^* \in T$   $a^* \in T$
- $(a+b)^* = a^* + b^*$  FALSO

Esercizio 4: Dato  $\Sigma = \{a,b\}$ , costruire automi DFA che accettino il linguaggi formati da:

- tutte e sole le parole che non contengono aa  


$$(b+ab)^*(b+ba)^*$$
- tutte e sole le parole con un numero pari di a  


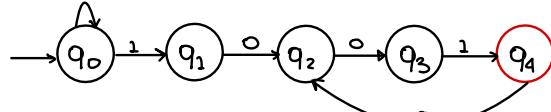
$$(b^*(b^*ab^*ab^*)^*b^*(b^*ab^*ab^*)b^*)^*$$
- tutte e sole le parole che contengono almeno due a e al massimo un b  


$$a^*(aab+aba+baa)a^*$$

Esercizio 5: Si consideri il DFA  $A = (\{q_0, q_1, q_2, q_3, q_4\}, \Sigma = \{0,1\}, \delta, q_0, F = \{q_4\})$ , dove le seguenti tuple costituiscono  $\delta$ :

- $\rightarrow ((q_0, 0), q_0); ((q_0, 1), q_1)$
- $\rightarrow ((q_1, 0), q_2);$
- $\rightarrow ((q_2, 0), q_3);$
- $\rightarrow ((q_3, 1), q_4);$
- $\rightarrow ((q_4, 0), q_2);$

• si costruisca l'automa:



• quali delle seguenti parole sono riconosciute?

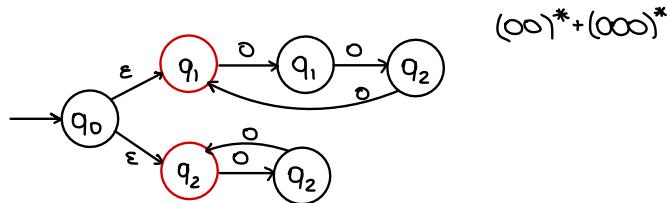
- 1. 0100 NO
- 2.  $w \in 0^*10^*1^*0^*$  NO
- 3.  $w \in 0^*10^*1^*0^+$  NO
- 4.  $w \in 0^*1001$  SI

• si ricavi una espressione regolare che costitui esattamente il linguaggio dell'automa

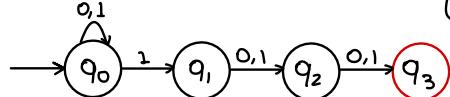
$$L(A) = 0^*1(001)^+$$

Esercizio 6: Costruire automi NFA che accettino i linguaggi formati da:

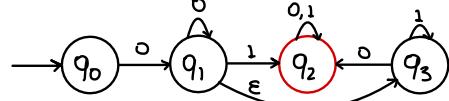
• Tutte e sole le parole dell'alfabeto  $\Sigma = \{0\}$  di lunghezza multiplo di 2 o di 3



• Tutte e sole le parole di  $\Sigma = \{0,1\}$  con un 1 in terzultima posizione  $(0+1)^* 1 (0+1)(0+1)$



Esercizio 7:  $\Sigma = \{0,1\}$   $F = \{q_2\}$   $((q_0, 0), q_1); ((q_1, 0), q_1); ((q_1, 1), q_2); ((q_1, \epsilon), q_3); ((q_2, 0), q_2); ((q_2, 1), q_2); ((q_3, 1), q_3); ((q_3, 0), q_2)$



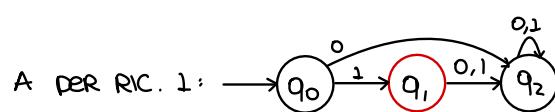
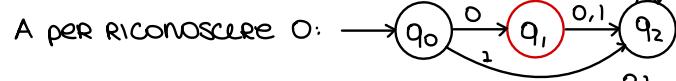
Parole riconosciute:

$0? \text{ NO } w \in 0^*? \text{ NO } \epsilon \in 0^* \text{ ma } \epsilon \notin L(A) \quad w \in 00^+? \text{ SI } \quad w \in 00^*10^*? \text{ SI } \quad w \in 00^*(10)^*? \text{ NO } 0 \notin L(A) \text{ o } w$

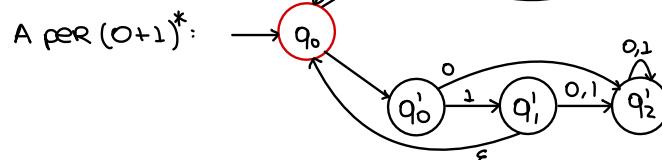
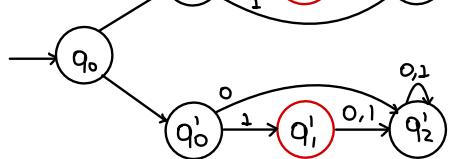
Espressione regolare  $0^+ (1+0) (1+0)^*$   $0^+(1+1^*0)(0+1)^*$

Esercizio 8: Usando il teorema REG  $\rightarrow$  NFA trovare gli automi non deterministici per le espressioni:

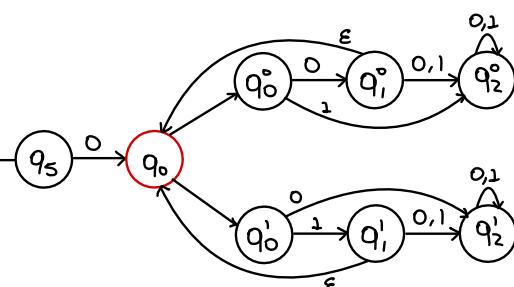
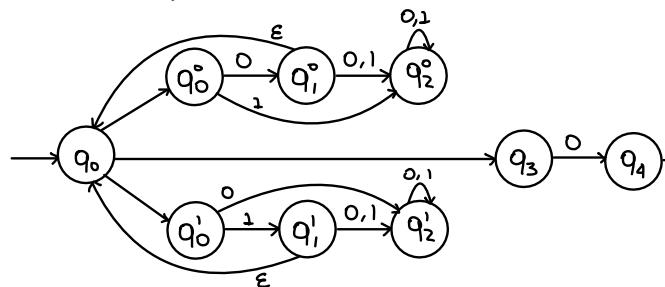
•  $(0+1)^* 0000(0+1)^*$  STRATEGIA BOTTOM-UP



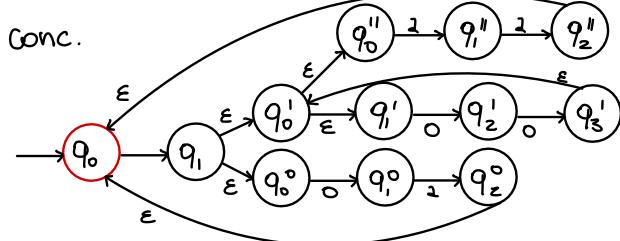
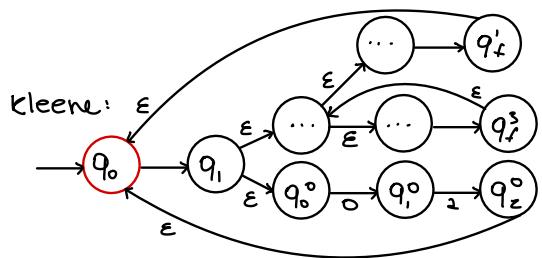
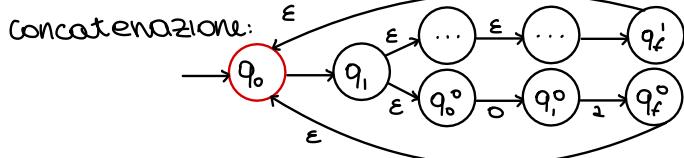
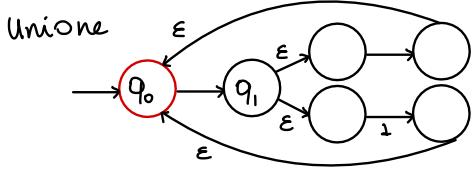
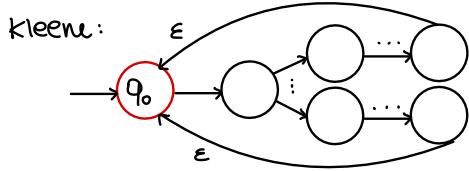
$A$  per  $(0+1)$ :



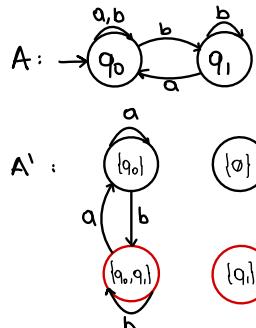
$A$  per  $(0+1)^* 0000(0+1)^*$ :



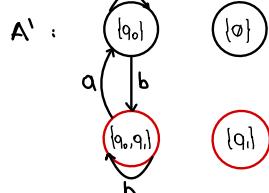
$\cdot (((00)^*(11)) + 01)^* \rightarrow \text{STRATEGIA TOP-DOWN}$



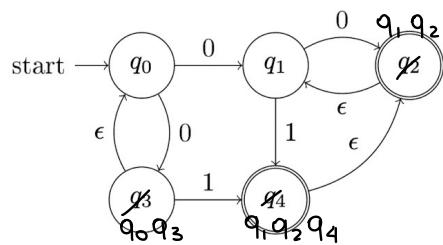
BONUS: si applichi il teorema  $\text{NFA} \rightarrow \text{DFA}$  sull'automa A



$$\begin{aligned} \delta'(\{q_0\}, a) &= \{q_0\} & \delta'(\{q_0\}, b) &= \{q_0, q_1\} & \delta'(\{q_0, q_1\}, a) &= \{q_0\} \\ \delta'(\{q_0, q_1\}, b) &= \{q_1\} \end{aligned}$$



ESERCIZIO 9:  $\text{NFA} \rightarrow \text{DFA}$



Se  $\delta^*((q_0, \epsilon), q_b)$  allora lo stato  $\{q_a\}$  diventa lo stato  $\{q_a q_b\}$  nel DFA

stati:  $2^5 \quad \epsilon(R) = \{\{q_0\}, \{q_1\}, \{q_2\}\}$

$Q' = \{\{\emptyset\}, \{q_0\}, \{q_1\}, \{q_2\}, \{q_3\}, \{q_4\}, \{q_0, q_1\}, \{q_0, q_2\}, \dots, \{q_1, q_2, q_3\}, \{q_0, q_1, q_2\}, \dots, \{q_1, q_2, q_3, q_4\}\}$

$$\begin{aligned} \delta'((\{q_0\}, 0), \{q_0, q_1, q_3\}) & \quad \delta'((\{q_0\}, 1), \{\emptyset\}) \\ \delta'((\{q_0, q_1, q_3\}, 0), \{q_0, q_1, q_2, q_3\}) & \quad \delta'((\{q_0, q_1, q_3\}, 1), \{q_1, q_2, q_4\}) \\ \delta'((\{q_0, q_1, q_2, q_3\}, 0), \{q_0, q_1, q_2, q_3\}) & \quad \delta'((\{q_0, q_1, q_2, q_3\}, 1), \{q_1, q_2, q_4\}) \\ \delta'((\{q_1, q_2\}, 0), \{q_1, q_2\}) & \quad \delta'((\{q_1, q_2\}, 1), \{q_1, q_2, q_4\}) \\ \delta'((\{q_1, q_2\}, 0), \{q_1, q_2\}) & \quad \delta'((\{q_1, q_2\}, 1), \{q_1, q_2, q_4\}) \end{aligned}$$

$0 \xrightarrow{0} 013 \xrightarrow{0} 0123 \supseteq 0$

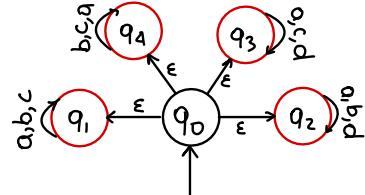
$1 \downarrow \quad 1 \swarrow$   
 $1 \supseteq 124 \xleftarrow[0]{1} 12 \supseteq 0$

Esercizio 10: considerare  $L$  su  $\Sigma = \{a, b, c, d\}$  dato da tutte le parole nelle quali almeno un simbolo di  $\Sigma$  è assente.

- Costruire una espressione regolare per caratterizzare esattamente  $L$ :

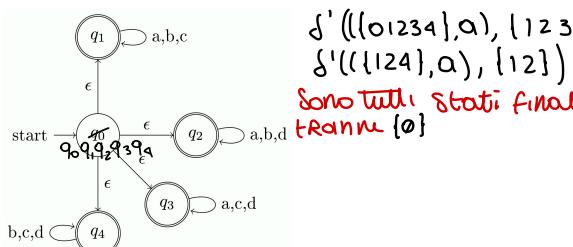
$$(a+b+c)^* + (a+b+d)^* + (a+d+c)^* + (b+c+d)^*$$

- Costruire un NFA per  $L$ :



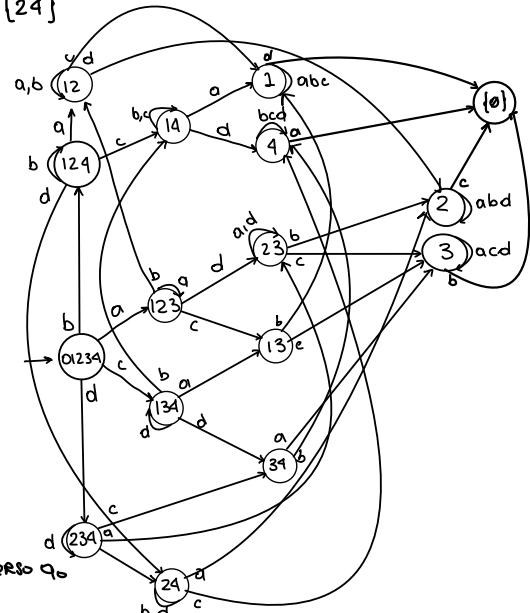
- Applicare  $NFA \rightarrow DFA$ : l'automa ha  $2^5$  stati

$$Q' = P(Q') = \{\emptyset, \{q_0\}, \{q_1\}, \{q_2\}, \{q_3\}, \{q_4\}, \{q_1, q_2\}, \dots, \{q_0, q_1, q_2, q_3, q_4\}\}$$



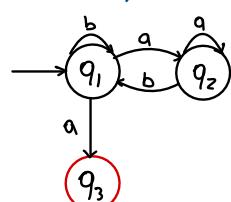
$$\begin{aligned} \delta'(\{\{01234\}, a\}, \{123\}) &= \delta'(\{\{01234\}, b\}, \{124\}) = \delta'(\{\{01234\}, c\}, \{154\}) = \delta'(\{\{01234\}, d\}, \{234\}) \\ \delta'(\{\{124\}, a\}, \{12\}) &= b \rightarrow \{124\} \quad c \rightarrow \{14\} \quad d \rightarrow \{24\} \end{aligned}$$

Sono tutti stati finali tranne  $\emptyset$



Questo è un NFA!

BONUS: DFA  $\rightarrow$  REG



$R^0$	1	2	3
1	b	a	a
2	b	a	$\emptyset$
3	$\emptyset$	$\emptyset$	$\emptyset$

$\kappa = 0 \rightarrow q_0$  è il più grande stato intermedio attraverso cui si passa  $\rightarrow$  si passa solo attraverso  $q_0$

$$R^{k-1} \rightarrow R^k \quad R(i, j, k) = R(i, j, k-1) + R(i, k, k-1) \circ R(k, k, k-1)^* \circ R(k, j, k-1)$$

$R^1$	1	2	3
1	$b^+$	$b^*a$	$b^*a$
2	$b^+$	$b^*a$	$b^*a$
3	$\emptyset$	$\emptyset$	$\emptyset$

$$R(1, 1, 1) = R(1, 1, 0) + R(1, 1, 0) R(1, 1, 0)^* R(1, 1, 0)$$

$$R(1, 2, 1) = R(1, 2, 0) + R(1, 1, 0) \cdot R(1, 1, 0)^* \circ R(1, 2, 0)$$

$\dots$

$R^2$	1	2	3
1	$b^+ (a^* b)^*$	$b^* a^*$	$b^* (a^* b)^*$
2	$b^+$	$b^* a^*$	$\emptyset$
3	$\emptyset$	$\emptyset$	$\emptyset$

$$R(1, 1, 2) = (b^+ b^+) + (a + b^+) (a + b^a)^* (b + b b^+)$$

$$R(1, 2, 2) = (a + b^+) + (a + b^+) (a + b^a)^* (a + b^+)$$

$$R(1, 3, 2) = (a + b^a) + (a + b^+) (a + b^a)^* (b^+ a)$$

$$R(3, 3, 2) = R(3, 3, 1) + R(3, 2, 1) R(2, 2, 1)^* R(2, 3, 1)$$

$\dots = (a + b^a) + b^* a (a + b^a)^* b^+ a$

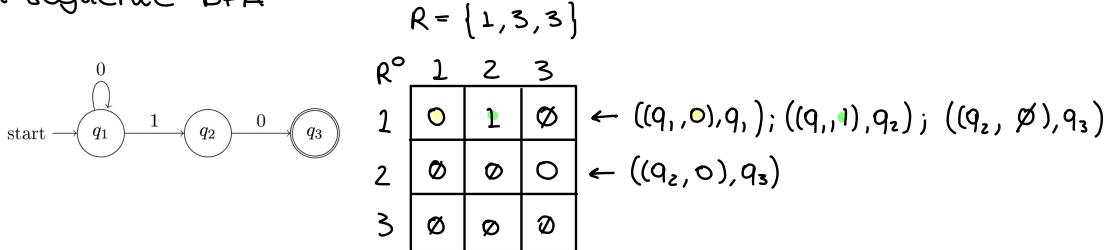
Un sacco di errori nella tabella

$R^3$	1	2	3
1	$b^+ (a^* b)^*$	$b^* a^*$	$b^+ (a^* b)^*$
2	$b^+$	$a^+$	$\emptyset$
3	$\emptyset$	$\emptyset$	$\emptyset$

$$R(1, 3, 3) = R(1, 3, 2) + R(1, 3, 2) R(3, 3, 2)^* R(3, 3, 2) - R(1, 3, 2) + R(1, 2, 2) R(3, 3, 2)^+$$

$$= ((a + b^a) + (a + b^+) (a + b^a)^* (b^+ a)) + ((a + b^a) + (a + b^+) (a + b^a)^* (b^+ a)) ((a + b^a) + b^* a (a + b^a)^* b^+ a)$$

ESERCIZIO 11: Usando il teorema DFA → REG si ricava l'espressione corrispondente al seguente DFA



$$\begin{aligned} R(1, 3, 3) &= R(1, 3, 2) + R(1, 3, 2)R(3, 3, 2)^*R(3, 3, 2) \\ &= (R(1, 3, 1) + R(1, 2, 1)R(2, 2, 1)^*R(2, 3, 1))''(R(3, 3, 1) + R(3, 2, 1)R(2, 2, 1)^*R(2, 3, 1)) \end{aligned}$$

$$R(1, 3, 1) = R(1, 3, 0) + R(1, 1, 0)R(1, 1, 0)^*R(1, 3, 0) = \emptyset + 00^*\emptyset$$

$$R(1, 2, 1) = R(1, 2, 0) + R(1, 1, 0)R(1, 1, 0)^*R(1, 2, 0) = 1 + 00^*1$$

$$R(2, 2, 1) = R(2, 2, 0) + R(2, 1, 0)R(1, 1, 0)^*R(2, 2, 0) = \emptyset + 00^*1$$

$$R(2, 3, 1) = R(2, 3, 0) + R(2, 1, 0)R(1, 1, 0)^*R(2, 3, 0) = 0 + 00^*\emptyset$$

$$\hookrightarrow R(1, 3, 2) = 0^+ + (1 + 0^+)(0^*1)^*0^*$$

$$R(3, 3, 1) = R(3, 3, 0) + R(3, 1, 0)R(1, 1, 0)^*R(3, 3, 0) = \emptyset + 00^*\emptyset$$

$$R(3, 2, 1) = R(3, 2, 0) + R(3, 1, 0)R(1, 1, 0)^*R(3, 2, 0) = \emptyset + 00^*1$$

$$\hookrightarrow = (0^+ + (1 + 0^+)(0^*1)^*0^*) + (0^+ + (1 + 0^+)(0^*1)^*0^*)(0^* + (0^*1)(0^*1)^*0^*)(0^* + (0^*1)(0^*1)^*0^*)$$

$R^1$	1	2	3
1	$0^+$	$0^*1$	$\emptyset$
2	$\emptyset$	$\emptyset$	0
3	$\emptyset$	$\emptyset$	$\emptyset$

$R^2$	1	2	3
1	$0^+$	$0^*1$	$0^*10$
2	$\emptyset$	$\emptyset$	0
3	$\emptyset$	$\emptyset$	$\emptyset$

$R^3$	1	2	3
1	$0^+$	$0^*1$	$0^*10$
2	$\emptyset$	$\emptyset$	0
3	$\emptyset$	$\emptyset$	$\emptyset$

ESERCIZIO 12: Mostrare, usando il pumping lemma che i seguenti linguaggi non sono regolari

$$\cdot \{0^n 1^n 2^n \mid n \geq 0\} \quad \Sigma = \{0, 1, 2\} \quad 0001112222$$

Consideriamo una parola dipendente da  $p$  e più lunga di  $p$  →  $w = 0^p 1^p 2^p \quad p < |w|$

$$\text{Poiché } |xy| \leq p \text{ e } |y| > 0 \text{ allora } n = 0^j \quad y = 0^{k>0} \quad z = 0^r 1^p 2^p \quad j+k+r = p$$

Ma  $\exists i \mid ny^i z \notin L$  perché se ripeto la  $y$ , allora il numero di 0 sarà superiore al numero di 1 e di 2 ←  $j+ik+r > p \forall i$

$$\cdot \{www \mid w \in (0+1)^*\}, \Sigma = \{0, 1\} \quad w: \underbrace{(01^p)}_{|y|} \underbrace{(01^p)}_{j+k=r=p} (01^p)$$

$$|y| < p \quad n = 0^j 1^s \quad y = 0^k 1^r \quad z = 0^s 0^p 0^r 1^p \quad j+k+r = p \quad s+r+1 = p \quad k, r > 0$$

$\exists i \mid ny^i z \notin L$  in quanto  $s+ir+u \geq p$

$$\cdot \{n=y+z \mid n, y, z \in (0+1)^*, B(n, y, z)\} \quad \Sigma = \{0, 1, +, =\} \quad B(n, y, z): n \text{ è la somma binaria di } y \text{ e } z$$

$$w: \underbrace{1^p}_{ny} = \underbrace{0^p + 1^p}_z \quad n = 1^j \quad y = 1^{k>0} \quad z = (1^r = 0^p + 1^p) \quad j+k+r = p \rightarrow ny^i z \notin L \quad y+ik+r \geq p$$

$$\cdot \{a^i \mid n \geq 0\} \quad \Sigma = \{a\} \quad w: a^{2^p} \rightarrow |w| = 2^p$$

$$nyz = a^j a^k a^s, k > 0 \quad j+k < p \quad j+k+s = 2^p$$

$ny^i z \in L \forall i? \quad j+ik+s = 2^q \leftarrow$  perché in questo caso non ho bisogno che il numero di 'a' sia uguale al numero di un altro letterale se  $i=0 \quad j+s = 2^m$

$$\begin{cases} j+ik+s = 2^q \\ j+s = 2^m \end{cases} \quad \begin{cases} ik = 2^q - (j+s) \\ ik = 2^q - 2^m \end{cases} \xrightarrow{\text{pari}} ik = 2^q - 2^m \quad \rightarrow J+2R+S = 2^q$$

$$J+i2^k+s = 2^t \rightarrow J+i2R+S = 2^t \rightarrow i2R = 2^t - 2^m \quad (j+s)2R = 2^t - 2^m \rightarrow 2^m 2R = 2^t - 2^m \rightarrow 2R = \frac{2^t - 2^m}{2^m} = 2^{t-m} - 1 \quad p = d? \text{ IMPOSSIBILE}$$

Esercizio 13: Si mostri che il seguente linguaggio non è regolare:

$$\{a^i b^j c^k \mid i, j, k \geq 0, i-1 \rightarrow j = k\} \quad w: \overbrace{a^i b^p c^p}^{ny} \overbrace{z}^{z}$$

$$ny = ab^* \quad z = b^+ c^+ \\ = ab^n \quad = b^m c^p \quad n = a^s b^q \quad y = a^t b^r \quad z = b^m c^p \quad s+t=1 \quad q+r+m=p$$

se  $s=0 \rightarrow t=1 \rightarrow s+t \geq 1$  NO!

$ny^i z \in L \forall i$ ? NO perché: se  $s=1 \rightarrow r > 0 \rightarrow q+r+m > p$  NO!

OPPURE:  $ab^n c^n \in L \xrightarrow{m} ab^n c^n \in \text{REG}$  REG è chiuso per l'INTERSEZIONE:

$$\rightarrow ab^n c^n \cap a^* b^* = b^n c^n \in \text{REG} \leftarrow \text{NO!}$$

COMPLEMENTO: L REG  $L(A) = L \quad A = (Q, \sigma, \delta, q_0, F)$   
 $\bar{A} = (Q, \sigma, \delta, q_0, Q \setminus F)$   
 Accetta se A rifiuta

Esercizio 14:  $\{0^n 1^m\}$

$$\sigma^P, P$$

Non è vero che non posso dividere in  $xyz$ :

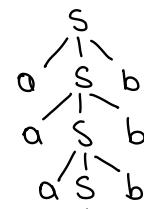
$$ny = 0^j 1^k \quad z = 1^s \quad nyz = 0^j 1^k 1^s \quad j=p \quad k+s=p \\ n = 0^j \quad q = 1^k \quad ny^i z = 0^j (1^k)^i 1^s \in L$$

INTERSEZIONE:  $L \cap L_2 = (\overline{L_1} \cup \overline{L_2})$  →  $(\overline{L}) = \overline{L} \rightarrow \overline{L}$  regolare

DIFF. INSIEMISTICA:  $L_1 \setminus L_2 = L_1 \cup \overline{L_2}$

Esercizio 15:

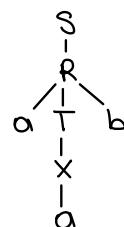
Si consideri la grammatica libera  $S ::= \epsilon | aSb$   
e si disegni l'albero di derivazione di aaabbb



Esercizio 16:

Si consideri la grammatica libera  
 $S ::= XSX | R$     $R ::= aTb | bTa$     $T ::= XT\bar{X} | X | \epsilon$     $X ::= a | b$

e si disegni l'albero di derivazione della parola aab



Esercizio 17:

Si trovi la grammatica libera dal contesto corrispondente ai seguenti linguaggi.

• {w | w contiene più uno che 0}

è regolare? No  $\rightarrow$  Pumping lemma

Considero una lunghezza p ed una parola di L dipendente da p

$$0^p 1^{p+1} = \underbrace{0 \dots 0}_p \underbrace{1 \dots 1}_{p+1} = xy \in L \quad |xy| \leq p \rightarrow xy \in 0^p \dots 0$$

$\xrightarrow{\text{sicuramente } ny^i z \notin L \forall i \geq 1}$

$p=2 \quad ny^2z = 00 \quad \text{almeno per}$

$$n=0^j \quad y=0^j \quad z=0^{p-2j}, p+1$$

$$\text{or: } p=5 \quad j=1 \quad i=2 \quad ny^2z = \overbrace{000}^n \overbrace{000}^y \overbrace{11111}^z \notin L$$

Scrivo l'espressione CFG

Costruisco una stringa dove ci sono  $|0|=|1|$ , un 1, e una stringa  $|0|=111$

$$G = (\{S, T\}, \{0, 1\}, R, S)$$

$$\begin{cases} S ::= T1T \\ T ::= 1TOT | T1TO | \epsilon | T1T \end{cases}$$

• {w | w contiene più 1 che 0}

$$\begin{cases} S ::= T1T \\ T ::= T1T | 1TOT | TOT1 | \epsilon \end{cases}$$

• {w#v | w, v  $\in (0+1)^*$ ,  $w^{-1} \sqsubseteq v$ } che vuol dire?

sol. scheda:  $S ::= STC$

$$\begin{cases} T ::= 011 \\ C ::= 0C0 | C1 | \# \end{cases}$$

•  $\{a^i b^j c^k | i=j \wedge j=k\} \rightarrow$  stesso numero di a e b & stesso numero di b e c in ordine abc

$$\begin{cases} S ::= RT | MN \\ R ::= aRb | \epsilon \\ T ::= CT | \epsilon \\ M ::= aM | \epsilon \\ N ::= bNc | \epsilon \end{cases}$$

Esercizio 18: Si converta la seguente grammatica in una equivalente forma di Chomsky

$$\begin{array}{lcl} S & ::= & XSX \mid R \\ R & ::= & aTb \mid bTa \\ T & ::= & XTX \mid X \mid \epsilon \\ X & ::= & a \mid b \end{array}$$

$$\begin{array}{ll} 1. \text{ Esplicito le disgiunzioni:} \\ S ::= XSX \quad S ::= R \\ R ::= aTb \quad R ::= bTa \\ T ::= XTX \quad T ::= X \quad T ::= \epsilon \\ X ::= a \quad X ::= b \end{array}$$

2. T ha una  $\epsilon$  produzione senza essere il simbolo iniziale  
 $\rightarrow$  la si rimuove introducendo nuove regole:

$$\begin{array}{l} R ::= ab \\ R ::= ba \\ T ::= XX \end{array}$$

3. Si applica la funzione Ch

$$\begin{array}{l} R_1 \quad Ch(S ::= XSX) = \{S ::= XB_1, B_1 ::= SX\} \\ R_2 \quad Ch(S ::= R) = \{S ::= R\} \leftarrow \text{era qui in forma normale} \\ R_3 \quad \begin{aligned} Ch(R ::= aTb) &= \{R ::= A_1B_2, A_1 ::= a\} \cup Ch(B_2 ::= Tb) \\ &= \{R ::= A_1B_2, A_1 ::= a, B_2 ::= TB_3, B_3 ::= b\} \\ Ch(R ::= bTa) &= \{R ::= B_3A_2, A_2 ::= TA_1\} \\ Ch(R ::= ab) &= \{R ::= A_1B_3\} \\ Ch(R ::= ba) &= \{R ::= B_3A_1\} \\ Ch(T ::= XTX) &= \{T ::= XB_4, B_4 ::= TX\} \\ Ch(T ::= X) &= \{T ::= X\} \\ Ch(T ::= XX) &= \{T ::= XX\} \end{aligned} \end{array}$$

$$\rightarrow G = \begin{cases} S ::= XB_1 | A_1B_2 | A_1B_3 | B_3A_1 | B_3A_2 \\ T ::= XB_4 | XX | a | b \\ X ::= a | b \\ A_1 ::= a \\ A_2 ::= TA_1 \\ B_1 ::= SX \\ B_2 ::= TB_3 \\ B_3 ::= b \\ B_4 ::= TX \end{cases}$$

Esercizio 19: Se  $G$  è una grammatica libera in forma di Chomsky e  $w \in L(G)$ , allora esiste una derivazione di  $w$  in  $G$  di al più  $2|w|-1$  passi

Una parola generata da una grammatica libera può essere rappresentata come la frontiera dell'albero di derivazione, e tutte le grammatiche di Chomsky danno luogo ad un albero che è binario fino al penultimo livello, con l'ultimo livello unario, in quanto in Ch un simbolo non terminale è da solo all'interno di una regola  $\rightarrow$  non genera branch ma un'unica foglia.

Per una parola di lunghezza  $|w|$ , l'albero di derivazione avrà  $|w|$  foglie e altezza massima  $\lceil \log_2 |w| \rceil + 1$ .

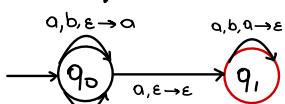
I passi di derivazione sono pari al numero di nodi in un albero di altezza  $\lceil \log_2 |w| \rceil$ , ovvero  $2^{\lceil \log_2 |w| \rceil - 1} \leq 2^{\lceil \log_2 |w| \rceil + 1} = 2|w|-1$

## Esercizio 20:

Si consideri il PDA  $A = (\{q_0, q_1\}, \Sigma = \{a, b\}, \Gamma = \{a\}, \delta, F = \{q_1\})$  dove le seguenti tuple costituiscono  $\delta$ :

- $((q_0, a, \epsilon), (q_0, a))$ ;
- $((q_0, b, \epsilon), (q_0, a))$ ;
- $((q_0, a, \epsilon), (q_1, \epsilon))$ ;
- $((q_1, a, a), (q_1, \epsilon))$ ;
- $((q_1, b, a), (q_1, \epsilon))$ .

- si disegni l'automa:



- quali delle seguenti parole sono riconosciute?

- aba: NO! la pila non è vuota a fine lettura
- aa: NO
- baa: SI
- bab: SI

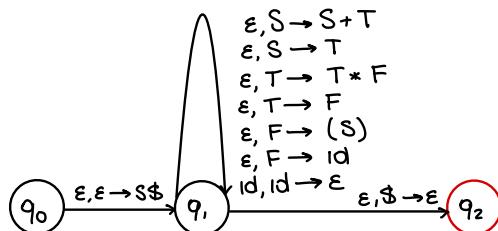
- abb NO

- si spieghi in linguaggio naturale la struttura delle parole accettate: le parole di  $L(A)$  devono avere lo stesso numero di caratteri a destra e a sinistra, con una 'a' al centro →  $|w|$ : dispari  
prot.: l'automa accetta le parole in  $(a+b)^*$  di lunghezza dispari con a centrale

## BONUS: Grammatica per espressioni

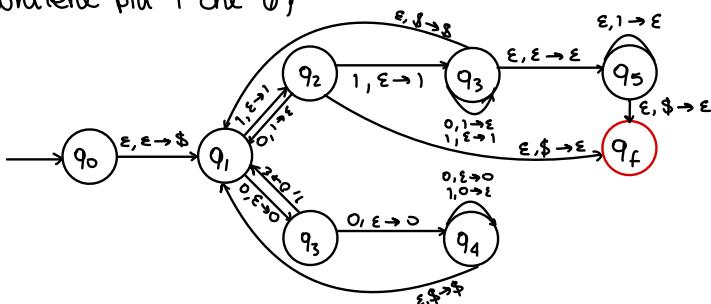
SCRIVIAMO UNA GRAMMATICA  $G$  CHE GENERA TUTTE E SOLO LE ESPRESSIONI ARITMETICHE CON SOMMA E MOLTIPLICAZIONE CORRETTEMENTE PARENTESIZZATE.

$G = (V = \{S, E, T, F\}, \Sigma = \{\text{id}, *, +, (, )\}, R, S)$ , dove  $R$  e $\cdot$   $S ::= S + T \mid T$   $T ::= T * F \mid F$   $F ::= (S) \mid \text{id}$

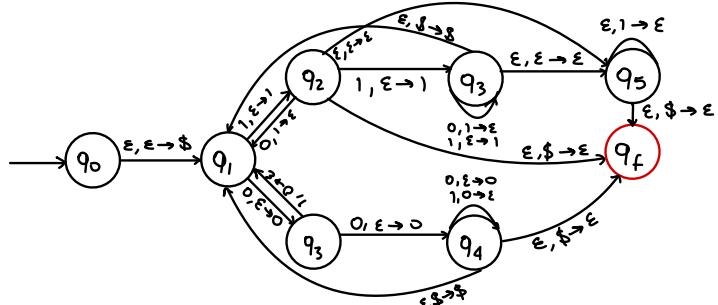


## Esercizio 21: Si produca un PDA per ognuno dei seguenti linguaggi

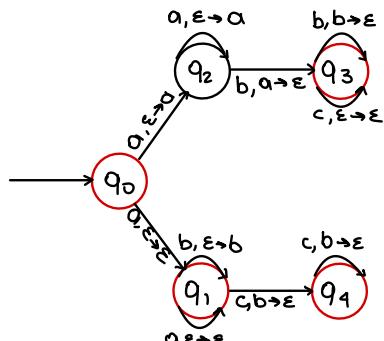
- $\{w | w \text{ contiene più 1 che } 0\}$



- $\{w | w \text{ contiene almeno tanti } 1 \text{ quanti } 0\}$

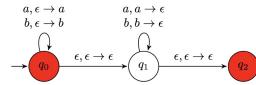


- $\{a^i b^j c^k \mid i = j \text{ o } j = k\}$

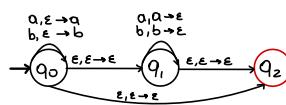


### ESERCIZIO 22:

Usando il teorema PDA  $\rightarrow$  CFG, si trovi una grammatica per il seguente automa a pila, e si caratterizzi il linguaggio riconosciuto:



Per applicare il teorema dobbiamo prima trasformare l'automa in modo che abbia un solo stato finale  
→ si ottiene legando gli stati finali a un unico stato finale

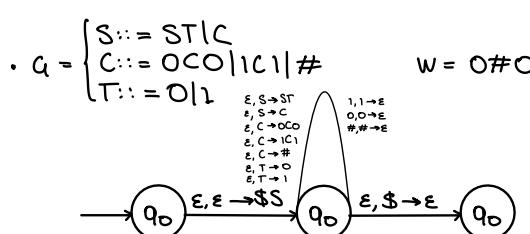


$S ::= A_{02}$   
 $A_{01} ::= aA_{01} \cup bA_{01}b$  in linguag  
 $A_{01} ::= \epsilon$   
 $A_{12} ::= \epsilon$   
 $A_{02} ::= \epsilon$   
 $A_{02} ::= A_{01}A_{12} \cup A_{00}A_{02} \cup A_{02}A_{22}$   
 $A_{00} ::= \epsilon$   
 $A_{11} ::= \epsilon$   
 $A_{22} ::= \epsilon$

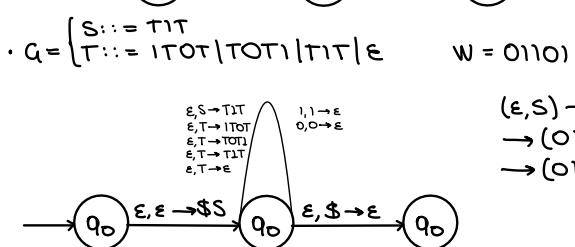
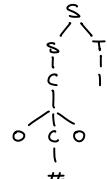
Il linguaggio riconosciuto è  $ww^T$ ,  $w \in \{a,b\}^*$

## ESERCIZIO 23:

Usando il teorema CFG  $\rightarrow$  PDA per ciascuna delle seguenti grammatiche, si produca un PDA che riconosca i seguenti linguaggi, e si mostrino i passi computazionali per riconoscere la stringa w specificata.



$$(\varepsilon, S) \rightarrow (\varepsilon, ST) \rightarrow (\varepsilon, C\Gamma) \rightarrow (\varepsilon, OCOT) \rightarrow (O, COT) \rightarrow (O, \#OT) \\ \rightarrow (O\#, OT) \rightarrow (O\#O, T) \rightarrow (O\#O, I) \rightarrow (O\#OI, U)$$



$\vdash \tau_1 \tau \rightarrow (\varepsilon, \tau_0 \tau_1 \tau) \rightarrow (\varepsilon, \varepsilon \sigma \tau_1 \tau) \rightarrow (\sigma, \tau_1 \tau) \rightarrow (\sigma, \varepsilon \tau_1 \tau)$   
 $\vdash \tau \rightarrow (\sigma \tau, \tau) \rightarrow (\sigma \tau, \tau_0 \tau_1) \rightarrow (\sigma \tau, \varepsilon \sigma \tau_1) \rightarrow (\sigma \tau \sigma, \tau_1) \rightarrow (\sigma \tau \sigma, \varepsilon_1)$   
 $\vdash \vdash$

Esercizio 24: Mostrare che i seguenti linguaggi, con  $\Sigma = \{a, b, c\}$  non sono liberi

- $\cdot \{ww \mid w \in \Sigma^*\}$  - confrontare con  $\{ww' \mid w \in \Sigma^*\}$  che  $\in \text{CFG}$  ma  $\notin \text{REG}$

Considero  $w = a^p b^p c^p a^p b^p c^p$ , le uniche divisioni in  $w = xyz$  che permettono di sommare a  $w = xyz$   
 $x = y = z = w \wedge x \cap y = \emptyset$   
sono  $x = y = z = w$  e  $x = y = \emptyset$  e non rispettano né  $|xy| > 0$  né (in quanto  $|w|=3p$ )  $|xyz| \leq p$

- $\{www \mid w \in \Sigma^*\}$  l'unica opzione sarebbe dividere come sopra (e qua' qui...) ma la 3<sup>a</sup> parola non verrebbe comunque mai pompata.

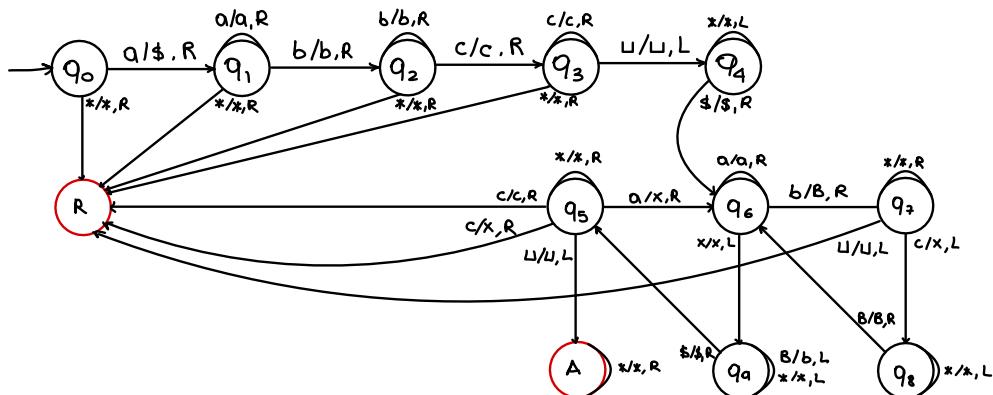
ESERCIZIO 25: Siano  $L_{CFG}$  e  $L_{REG}$ , si dimostri, tramite la costruzione di un automa apposito, che  $L_{CFG} \cap L_{REG} \in CFG$

Costruiamo un automa a pila  $A_{CFG}$  e un automa deterministico a stati finiti  $A_{REG}$ .  $A_{CFG} \in PDA$  e  $A_{REG} \in DFA$ . Costruiamo il PDA prodotto  $A$  tale che  $Q = Q_{CFG} \times Q_{REG}$ ,  $F = F_{CFG} \times F_{REG}$  e  $((q_{c1}, q_{r1}), r_1, r_2), (q_{c2}, q_{r2}), r_3 \rangle \in \delta$  per ogni coppia  $\in Q$  con  $((q_{c1}, r_1, r_2), (q_{c2}, r_3)) \in \delta_{CFG}$  e  $((q_{r1}, r_1), q_{r2}) \in \delta_{REG}$ .

A esegue quindi i due automi contemporaneamente e di conseguenza riconosce  $L_{CFG} \cap L_{REG}$ .

ESERCIZIO 26: Si consideri  $L = \{a^i b^j c^k \mid i \neq j \neq k, i, j, k \geq 0\}$

1. Controlla che l'input sia di tipo  $a^+ b^+ c^+$
2. Sostituisci la prima  $a$  con  $x$
3. Sostituisci la prima  $b$  con  $B$  e la prima  $c$  con  $X$ , finché ci sono ancora  $b$  ( $B$ ). Se non ci sono  $c$  per una  $b$  ( $B$ ) rifiuta
4. Se non ci sono più  $b$  ( $B$ ), ma ci sono ancora  $a$  e  $c$ , ritorna al punto 2



Si diano due configurazioni  $g, g'$  qualsiasi tali che  $g \rightsquigarrow g': \$aBbq_7ccc \rightsquigarrow \$aBbXq_8cc$

ESERCIZIO 27: Considerare nuovamente il Teorema  $K-TM \rightarrow TM$  del blocco che introduce le macchine di Turing. Riformulare la dimostrazione

Una  $K-TM$  differisce da una TM per il numero di nastri  $\leq$  perché la funzione di transizione da ogni carattere sotto ogni testina e può causare la scrittura e lo spostamento di ciascuna in un singolo passo computazionale.

Per simulare una  $K-TM$  useremo una  $stay-TM$  che, se  $K-TM$  ha alfabeto  $\Sigma$ , ha alfabeto  $\{\Sigma, \$, \alpha, \underline{\Sigma}\}$ , dove  $\underline{\Sigma}$  simboleggia tutti i caratteri di  $\Sigma$ , ma sottolineati, che useremo per tenere traccia delle posizioni delle testine;  $\$$  servirà come separatore del contenuto dei nastri;  $\alpha$  serve per indicare la fine del contenuto del nastro  $\leftarrow$  appare anche alla fine dei contenuti di  $K-TM$ .

Se la stringa di input è  $w=w_1 \dots w_n$ , a inizio computazione il nostro nastro dovrà avere forma:

$w_1 \dots w_n \$ \underline{\$} \underline{\$} \dots \underline{\$} \alpha$  Per ottenere questo stato dobbiamo avere una transizione di questo genere:  $\textcircled{w_1 \dots w_n} \xrightarrow{\forall i \in \Sigma, \forall j \in \Sigma} \textcircled{\alpha}$  che sostituisca  $w_i$  con  $\underline{w}_i$ , dopodiché scorriamo tutto il nastro  $\textcircled{\alpha} \xrightarrow{\forall i \in \Sigma, \forall j \in \Sigma} \textcircled{\alpha}$  e usiamo  $2K$  stati per scrivere  $K-1$  volte  $\underline{\$}$  sul nastro  $\textcircled{\alpha} \xrightarrow{\underline{\$}/\$, R} \textcircled{\alpha} \xrightarrow{\underline{\$}/\$, R} \dots$  e facciamo tornare la testina allo stato iniziale.

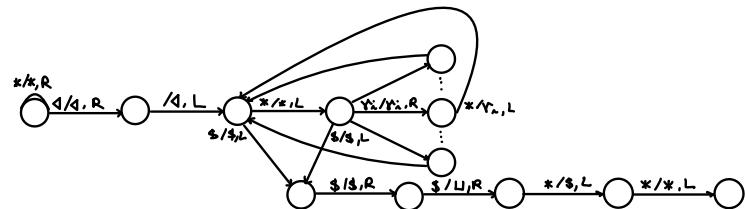
Simuliamo ora ogni passo computazionale della  $k$ -TM, che ha  $f: (q, a_1, a_2, \dots, a_k) \rightarrow (q', b_1, \dots, b_k, M_1, \dots, M_k)$

Avremo quindi bisogno di  $2k$  stati aggiuntivi per simulare questo tipo di transizione.

$k$  per leggere ciascun carattere  $a_i$ , e altri  $k$  per scrivere  $b_i$  e fare  $M_i$ .

Per spostare la testina da un  $a_i$  a un altro e sostituire con  $b_i$

Gadget: se uno aggiungere una lettera al "nastro  $i$ ", traslo a destra finendo su  $\sqcup$  e lo sostituisco così dopo che:



ESERCIZIO 28: Considerare nuovamente il teorema left-TM  $\rightarrow$  TM

Una left-TM può essere simulata con una 2-TM, oppure con una TM usando le caselle pari per l'estremo destro e quelle dispari per l'estremo sinistro, con le transizioni che vanno di due in due - tutto questo assumendo un meccanismo per il riconoscimento dell'estremo sinistro del nastro.

ESERCIZIO 29: Considerare il verso RAM  $\rightarrow$  TM del teorema RAM = TM

Una TM può essere vista come una RAM con: 2 aree di memoria infinite (istruzioni e input), due registri contenenti data pointer e program counter, e una istruzione con 4 argomenti: due valori ( $\Sigma$ ), un booleano (R, L) e un indirizzo della memoria di programma.

Per simulare una RAM, invece, possiamo usare una  $k$ -TM con almeno 2 nastri, uno per  $T[]$  e uno per ogni registro  $R[]$ . Nel nastro che simula la memoria utilizzo  $\$$  per separare i dati.

Ogni istruzione corrisponde a uno TM.

ESERCIZIO 30: PDA C DEC

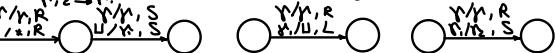
Esiste almeno un linguaggio DEC che non è PDA ( $\leftarrow a^n b^n c^n$  - il che dimostra che in ogni caso l'inclusione non può essere stretta)

Per dimostrare PDA C DEC dovo dimostrare che per ogni PDA ne esiste sempre un'altro che riconosce lo stesso linguaggio e termina sempre; e che per ogni PDA esiste una macchina di Turing che lo simula.

Un PDA può non terminare se esiste nel suo grafo un loop di  $\epsilon$ -transizioni tra stati non finali. Mostriamo che per ogni automa a pila ne esiste uno equivalente senza  $\epsilon$ -cicli: per ogni situazione di looping di un PDA, se questa è accettante aggiungiamo una  $\epsilon$ -transizione ad un nuovo stato accettante, altrimenti la aggiungiamo ad un nuovo stato cieco.

Per ogni PDA A possiamo costruire una 2-TM non deterministica M che riconosca gli stessi linguaggi.

Sulla pila si possono eseguire 3 tipi di azioni: aggiungere un simbolo, toglierlo, oppure entrambe.



Esercizio 33: Proprietà di chiusura di DEC, sapendo che è chiusa per unione e complemento

DEC è chiusa per complemento perché se  $L \in \text{DEC}$  e la TM  $M$  la decide, se inverti Accetta e Rifiuta (e viceversa) ottengo una TM  $M'$  che decide  $\overline{L}$

Unione: se  $L, L' \in \text{DEC}$  costruisco una TM che si comporta come  $M_L$  e se  $M_{L'}$  rifiuta, si comporta come  $M_{L'}$ , accettandone la risposta.

Intersezione:  $L_1 \cap L_2 = \overline{(L_1 \cup L_2)}$

Differenza insiemistica:  $L_1 \setminus L_2 = L_1 \cup \overline{L_2}$

Composizione: Dati  $L, L_2$  e  $M_1, M_2$ , costruiamo una 2-TM  $M$  che opera sul primo nastro come  $M_1$ , fino all'accettazione, e poi opera sul secondo come  $M_2$

Kleene: Dato  $L$  e  $M_L$ , per riconoscere  $L^*$  costruiamo una 2-TM che divide (non deterministicamente) l'input in  $n$  parti e usa il secondo nastro per simularne ciascuna in  $M$ , accettando se accettano tutte le simulazioni

Esercizio 34: considerare la gerarchia di Chomsky e dare un esempio completo di grammatica per ogni tipo.

Tipo 3 - Regolare: Ammette regole di produzione del tipo  $A ::= AB$  o  $A ::= a$ ,  $a^*bc^*$   $G = \left\{ \begin{array}{l} S ::= aSbT \\ T ::= cTc \end{array} \right.$

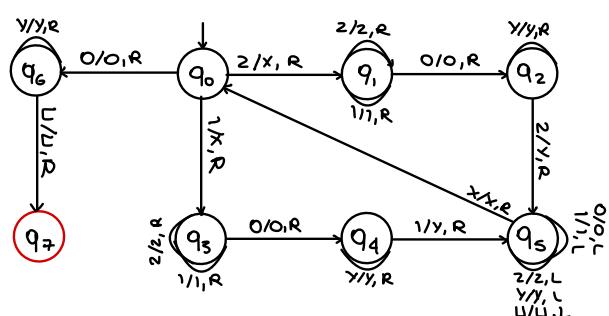
Tipo 2 - libera dal contesto: Ammette  $A ::= \alpha$ , può essere limitata alla forma di Chomsky, ossia  $A ::= BC$ ,  $A ::= a^n b^n$   $G = \{S ::= OS1 | \epsilon\}$

Tipo 1 - dipendente dal contesto: Ammette  $\beta A \gamma ::= \beta \alpha \gamma$ ,  $0^n 1^n 2^n$   $G = \{ \dots \}$

Tipo 0 - non ristretta:  $\alpha ::= \beta \quad \alpha^2$

$$\left\{ \begin{array}{l} S ::= OBC | OSBC \\ CB ::= BC \\ OB ::= O1 \\ 1B ::= 11 \\ 1C ::= 12 \\ 2C ::= 22 \end{array} \right.$$

Esercizio 35:



. Si è deterministica

. 102 · NO · 202 · SI · 2012 · NO · 12012 · SI

. Accetta le parole di tipo WOW,  $w \in \{1,2\}^*$

### ESERCIZIO 36: $\text{EQ}_{\text{DFA}} \in \text{DEC}$

$$\text{EQ}_{\text{DFA}} = \{(A_1, A_2) \mid A_1, A_2 \in \text{DFA} \quad L(A_1) = L(A_2)\}$$

Poiché non posso utilizzare la simulazione per dimostrarlo (in quanto elencare tutte le possibili stringhe di input mi porterebbe alla non terminazione), per la dimostrazione mi avallero del fatto che REG è chiuso per la differenza insiemistica  
 → Costruisco una TM M basata sulla differenza insiemistica che decide  $\text{EQ}_{\text{DFA}}$

1. Data una string w, controlla se è in formato  $\langle A_1, A_2 \rangle$ , altrimenti rifiuta
2. Costruisce  $B = A_1 \setminus A_2$  e  $C = A_2 \setminus A_1 \leftarrow \text{REG chiuso quindi DFA chiuso in quanto:}$   
 $A_i \rightarrow L(A_i) \quad A_j \rightarrow L(A_j) \quad L(A_k) = L(A_i) \setminus L(A_j) \quad L(A_k) \rightarrow A_k$
3. Se  $\langle B \rangle \in \text{E}_{\text{DFA}}$  e  $\langle C \rangle \in \text{E}_{\text{DFA}}$ , allora accetta  
 $\hookrightarrow L(C) = \emptyset$

### • $\text{E}_{\text{CFG}} \in \text{DEC}$ :

1. Controlla se w è nel formato corretto di una grammatica
2. Esegui una visita dell'albero di produzione della grammatica, ignorando le regole già utilizzate, ~~rifiutando~~ se si raggiunge una foglia con simboli terminali

### ESERCIZIO 37: Dimostrare le proprietà di chiusura di R.E.

- R.E. chiuso sotto unione: dati  $l_1, l_2 \in \text{R.E.}$  esistono  $M_1, M_2$  che li semidicono. Con una 2TM che simula  $M_1$  in un nastro e  $M_2$  in un altro, semidicono  $L_1 \cup L_2$
- Intersezione: come sopra, ma accetta solo se entrambe accettano
- Composizione: una TM non deterministica sceglie una divisione della parola in due parti, simula  $M_1$  sulla prima parte e  $M_2$  sulla seconda. Accetta se entrambe accettano
- Star di Kleene: come prima, ma divisione in n parti, e simulazione sempre su  $M$
- Complemento:  $L \in \text{R.E.} \Rightarrow \bar{L} \in \text{R.E.}$ , se anche  $\bar{L} \in \text{R.E.}$ , allora  $L \in \text{DEC}$ . Se R.E. fosse chiuso per complemento, allora  $\text{R.E.} = \text{DEC} \subseteq \text{R.E.}$
- Differenza insiemistica:  $T = \Sigma^*/L \rightarrow$  se R.E. fosse chiuso per d.i. lo sarebbe anche per complemento

ESERCIZIO 38: Dimostrare che se  $l_1, l_2$  sono linguaggi tali che  $l_1 \leq_m l_2$  e che  $l_2 \in \text{R.E.}$ , allora  $l_1 \in \text{R.E.}$

Se  $l_2 \in \text{R.E.}$  allora esiste una TM  $M_2$  che lo semidice. Inoltre, siccome  $l_1 \leq_m l_2$ . Possiamo avere una macchina che semidice  $l_1$  che, con input w, calcola  $f(w)$  tramite  $M_f$ , ed esegue  $M_2(f(w))$ , accettando se accetta e rifiutando altrimenti.

ESERCIZIO 39: Si verifichi se valgono o no le seguenti proprietà per  $\leq_m$

Riflessività: ogni problema si può ridurre a se stesso usando come riduzione la funzione identità.

Simmetria: No.  $A_{\text{TM}} \leq_m \text{EQ}_{\text{TM}}$  e sappiamo anche che  $A_{\text{TM}} \in \text{Co.R.E.}$ , se a fosse simmetrica  $\text{EQ}_{\text{TM}} \leq_m A_{\text{TM}} \rightarrow \text{EQ}_{\text{TM}} \in \text{Co.R.E.}$  ma questo è falso in quanto  $A_{\text{TM}} \leq_m \overline{\text{EQ}_{\text{TM}}} \in \overline{\text{A}_{\text{TM}}} \in \text{Co.R.E.}$   
 $\overline{\text{A}_{\text{TM}}} \in \text{Co.R.E.C.}$

Trasitività: Sì. Se  $X \leq_m Y \leq_m Z \quad f: X \rightarrow Y \quad g: Y \rightarrow Z$  e di conseguenza  $g \circ f: X \rightarrow Z$

Antisimmetria: No. Due problemi che si possono ridurre reciprocamente sarebbero lo stesso problema.  
 Facciamo l'esempio di  $A_{\text{TM}}$  e  $H_{\text{TM}}$ :  
 $A_{\text{TM}} \leq_m H_{\text{TM}} \quad f: \langle M, w \rangle \rightarrow \langle M', w' \rangle$  dove  $M'$  esegue  $M$ , se  $M$  accetta  $\rightarrow M'$  accetta,  
 altrimenti  $M'$  entra in un ciclo infinito  
 $H_{\text{TM}} \leq_m A_{\text{TM}}$ : Esegue e accetta.  $A_{\text{TM}} \in \text{R.E.C.} \quad H_{\text{TM}} \in \text{R.E.C.}$

ESEMPIO 40: Dimostrare i seguenti fatti riguardo  $E_{TM}$ :

•  $E_{TM} \notin \text{DEC}$ : Supponiamo che  $E_{TM}$  sia decidibile. Dovrei poter costruire una macchina  $A(H_{TM}) \in \text{DEC}$  ma  $A(H_{TM})$  è  $A_{TM} \notin \text{DEC}$

•  $A_{TM} \leq_m E_{TM}$ : Se  $A_{TM} \leq_m E_{TM} \rightarrow \bar{A}_{TM} \leq_m \bar{E}_{TM}$  ma  $\bar{E}_{TM} \in \text{R.E.} \rightarrow \bar{A}_{TM} \in \text{R.E.}$  ma sappiamo  $\bar{A}_{TM} \in \text{Co.R.E.}$ .  
ma se  $\bar{A}_{TM} \in \text{R.E.}$  e  $\bar{A}_{TM} \in \text{Co.R.E.} \rightarrow \bar{A}_{TM} \in \text{DEC} \rightarrow A_{TM} \in \text{DEC}$  che è falso

BONUS: Mostriamo che  $(p \rightarrow q) \wedge \neg q$

ossia mostriamo che  $I \models p \rightarrow q$  e  $I \models \neg q$

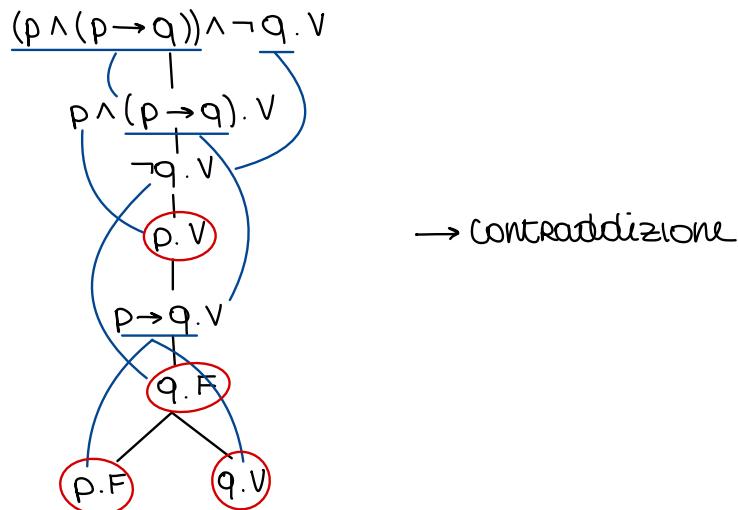
$I \models \neg q$ :  $q$  deve essere falsa

$I \models p \rightarrow q$ :  $I \not\models p$  o  $I \models q$

} vero se  $I \not\models p$  (ossia  $I \models \neg p$ )

→ la formula si soddisfa quando  $p$  e  $q$  sono entrambe false

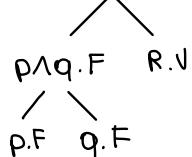
BONUS 2: Espressione tableau della formula  $(p \wedge (p \rightarrow q)) \wedge \neg q$



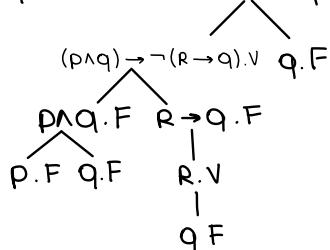
ESERCIZIO 43: classificare le seguenti formule tra Tautologie, contraddizioni e contingenze. Tra quest'ultime, dire quali sono soddisfatte da  $I_1 = \{p=1, q=0, R=0\}$  e quali da  $I_2 = \{p=1, q=1, R=0\}$

•  $p \vee \neg p$  tautologia

•  $(p \wedge q) \rightarrow R$  contingenza. Soddisfatta da  $I_1$ , non da  $I_2$



•  $((p \wedge q) \rightarrow \neg(R \rightarrow q)) \vee \neg q$  contingenza. Soddisfatta da  $I_1$ , non da  $I_2$



•  $((\neg p) \rightarrow q) \vee (p \rightarrow R) \equiv p \vee q \vee \neg p \vee R$  tautologia

•  $((p \wedge (p \rightarrow q)) \rightarrow q) \quad \neg(p \wedge (\neg p \vee q)) \vee q \quad \underline{\neg(p \wedge (\neg p)) \vee \neg(p \wedge q) \vee q}$   
tautologia!

$$\begin{aligned} & \cdot ((p \wedge (p \rightarrow q)) \wedge \neg q) \\ & (p \wedge (\neg p \vee q)) \wedge \neg q \\ & (p \wedge \neg p \vee \underline{p \wedge q}) \wedge \neg q \\ & \text{contrad.} \quad \text{contraddizione} \end{aligned}$$

- $(p \vee q) \wedge (\neg q \wedge \neg p)$  Contraddizione
- $(p \vee q) \wedge (p \rightarrow R \wedge q) \wedge (q \rightarrow \neg R \wedge p)$
- $(p \vee q) \wedge ((\neg p \vee R) \wedge q) \wedge ((\neg q \vee \neg R) \wedge p)$
- $((p \wedge R \wedge q) \vee (q \wedge \neg p) \vee (q \wedge R)) \wedge ((\neg q \wedge p) \vee (\neg R \wedge p))$
- Contraddizione

ESEMPIO 44: Dati gli operatori booleani  $\neg$ ,  $\perp$ ,  $\top$ ,  $\vee$ ,  $\wedge$ , e  $\rightarrow$ , un sottoinsieme di operatori si dice completo se, usando solo operatori del sottoinsieme, è possibile esprimere tutti gli operatori rimanenti.

$$\begin{aligned} \cdot \{\perp, \rightarrow\} \quad & \neg p \equiv (p \rightarrow \perp) \quad \top \equiv (\perp \rightarrow \perp) \quad (p \vee q) \equiv (\neg p \rightarrow q) \equiv ((p \rightarrow \perp) \rightarrow q) \\ & (p \wedge q) \equiv \neg \neg (p \wedge q) \equiv \neg (\neg p \vee \neg q) \equiv (((p \rightarrow \perp) \vee (q \rightarrow \perp)) \rightarrow \perp) \\ & \qquad \qquad \qquad \equiv ((p \rightarrow \perp) \rightarrow \perp) \rightarrow ((q \rightarrow \perp) \rightarrow \perp) \\ & \qquad \qquad \qquad \equiv (p \rightarrow \perp) \rightarrow (q \rightarrow \perp) \rightarrow \perp \end{aligned}$$

$$\cdot \{ \vee, \neg \} \quad T \equiv p \vee \neg p \quad \perp \equiv \neg(\neg p \vee p) \quad (p \wedge q) \equiv \neg \neg(p \wedge q) \equiv \neg(\neg p \vee \neg q)$$

$$p \rightarrow q \equiv \neg p \vee q$$

## ESEMPIO 45:

$\neg((\neg p \vee q) \wedge (\neg q \vee r) \wedge (p \vee \neg r) \wedge (p \vee q \vee r)) \rightarrow (p \wedge q \wedge r)$	
$(\neg p \vee q) \wedge (\neg q \vee r) \wedge (p \vee \neg r) \wedge (p \vee q \vee r) . F$	$p \wedge q \wedge r . V$
$\neg p \vee q . F$	$p . V$
$\neg q \vee r . F$	$q . V$
$p \vee \neg r . F$	$r . V$
$p \vee q \vee r . F$	
$p . F$	
$q . F$	
$r . F$	

$\neg(\neg q) \wedge (p \rightarrow q) \wedge (\neg p \rightarrow \neg R) \rightarrow \neg R$	
	R.F
$\neg q \wedge (p \rightarrow q) \wedge (\neg p \rightarrow \neg R)$	.F
$q.V$	
$p \rightarrow q.F$	
$\neg p \rightarrow \neg R.F$	
$p.V$	
$p.F$	
$q.F$	
$R.V$	

**Esercizio 46:** si trasformino le seguenti formule in forma clausale

$$\cdot p \vee \neg(p \wedge \neg R) \equiv p \vee (\neg p \vee R) \equiv p \vee \neg p \vee R$$

$$\begin{aligned} & \neg(((q \rightarrow p) \rightarrow \neg R) \wedge ((q \rightarrow p) \rightarrow q)) \equiv \neg(((\neg q \vee p) \rightarrow \neg R) \wedge ((\neg q \vee p) \rightarrow q)) \equiv \neg((\neg(\neg q \vee p) \vee \neg R) \wedge (\neg(q \vee p) \vee q)) \\ & \equiv \neg((q \wedge \neg p) \vee \neg R) \wedge (\neg q \wedge \neg p) \vee q) \equiv \neg(q \vee \neg R) \wedge (\neg p \vee \neg R) \wedge (\neg q \vee q) \wedge (\neg p \vee q) \end{aligned}$$

$$\cdot (\rho \wedge (\rho \rightarrow \sigma)) \wedge \neg q \equiv (\rho \wedge (\neg \rho \vee q)) \wedge \neg q$$

$$\cdot (\rho \vee \neg(\rho \wedge q)) \vee \neg q \equiv \neg \rho \wedge (\rho \wedge q) \wedge q$$

ESERCIZIO 47: Si dica se le seguenti formule si possono scrivere come formule di Horn, e se possibile si faccia

- $p \rightarrow (\neg q \vee r) \equiv \neg p \vee (\neg q \vee r) \equiv \neg p \vee \neg q \vee r = (p \wedge q) \rightarrow r$
- $(\neg p \vee q) \rightarrow r \equiv \neg(\neg p \vee q) \vee r \text{ NO}$
- $(p \wedge q) \rightarrow r \text{ NO e' gia'}$
- $(p \wedge (p \rightarrow q)) \rightarrow q = (p \wedge (\neg p \vee q)) \rightarrow q = (p \wedge q) \rightarrow q$
- $(p \wedge (p \rightarrow q)) \wedge \neg q = (p \wedge (\neg p \vee q)) \wedge \neg q \equiv (p \wedge q) \wedge \neg q = p \wedge q \wedge \neg q = \perp$

ESERCIZIO 48: Dimostrare i seguenti fatti usando l'algoritmo del forward checking

- da  $\neg q, p \rightarrow q, \neg p \rightarrow \neg r$  si deduce  $\neg r$   
 $(q \rightarrow \perp) \wedge (\neg p \rightarrow q) \wedge (r \rightarrow p)$

Per dimostrare  $\neg r$  si ipotizza  $r$  per assurdo, derivando poi  $\perp$ . Dunque inizialmente  $A = \{r\}$  poi uso  $(r \rightarrow p)$  e ottengo  $A = \{r, p\}$ , poi  $(p \rightarrow q)$  e  $A = \{r, p, q\}$ , poi  $(q \rightarrow \perp)$  e  $A = \{r, p, q, \perp\}$  poiché  $\perp \in A$ ,  $\neg r$  è soddisfatto (in quanto assurdo)

- da  $(\neg p \vee \neg q \vee r) \wedge (\neg r \vee s) \wedge (\neg s \vee \neg t \vee u) \wedge p \wedge q$  si deduce  $u$   
 $((p \wedge q) \rightarrow r) \wedge (r \rightarrow s) \wedge (s \rightarrow t) \wedge ((s \rightarrow t) \rightarrow u) \wedge p \wedge q$

$A = \{p, q\} \Rightarrow A = \{p, q, r\} \Rightarrow A = \{p, q, r, s\} \Rightarrow A = \{p, q, r, s, t\} \Rightarrow A = \{p, q, r, s, t, u\}$   
 $\perp \notin A$  e  $u \in A$  soddisfatto

ESERCIZIO 49: Si dica se le seguenti formule proposizionali sono vere oppure no:

- $\forall p \exists q (p \rightarrow q)$  SI :  $\{p=0, \forall\} \quad \{p=1, q=1\}$
- $\forall p \forall q (p \rightarrow q)$  NO
- $\exists p \forall q (p \rightarrow q)$  SI se  $p$  è falsa, qualsiasi valore di verità di  $q$  rende vero  $p \rightarrow q$
- $\forall p \forall q \exists r ((p \wedge q) \rightarrow r)$  SI  $(\neg(p \wedge q) \vee r)$  basta scegliere  $r$  vera
- $\forall p \exists R \forall q ((p \wedge q) \rightarrow R)$  SI basta scegliere  $R$  falsa

ESERCIZIO 50: Si dica se le formule proposizionali quantificate sono vere, usando il truth-checking

- $\forall p \exists q (p \rightarrow q)$

$$TC(\forall p \exists q (p \rightarrow q), \emptyset) = TC(\exists q (p \rightarrow q), \{p=1\}) \wedge TC(\exists q (p \rightarrow q), \{p=0\})$$

$$= (TC((p \rightarrow q), \{p=1, q=1\}) \vee TC((p \rightarrow q), \{p=1, q=0\})) \wedge (TC((p \rightarrow q), \{p=0, q=1\}) \vee TC((p \rightarrow q), \{p=0, q=0\})) \\ = ('si' \vee 'no') \wedge ('si' \vee 'si') = 'si' \wedge 'si' = 'si'$$

- $\forall p \forall q (p \rightarrow q)$   $TC(\forall p (p \rightarrow q), \{p=0\}) \wedge TC(\forall q (p \rightarrow q), \{p=1\})$

$$= (TC((p \rightarrow q), \{p=0, q=0\}) \wedge TC((p \rightarrow q), \{p=0, q=1\})) \wedge (TC((p \rightarrow q), \{p=1, q=0\}) \wedge TC((p \rightarrow q), \{p=1, q=1\})) \\ = (1 \wedge 1) \wedge (0 \wedge 1) = 1 \wedge 0 = 0 \text{ NO}$$

- $\forall p \exists R \forall q ((p \wedge q) \rightarrow R)$   $TC(\exists R \forall q ((p \wedge q) \rightarrow R), \{p=0\}) \wedge TC(\exists R \forall q ((p \wedge q) \rightarrow R), \{p=1\})$

$$= (TC(\forall q ((p \wedge q) \rightarrow R), \{p=0, R=0\}) \vee TC(\forall q ((p \wedge q) \rightarrow R), \{p=0, R=1\})) \wedge (TC(\forall q ((p \wedge q) \rightarrow R), \{p=1, R=0\}) \vee$$

$$= ((TC((p \wedge q) \rightarrow R, \{p=0, R=0, q=0\}) \wedge TC((p \wedge q) \rightarrow R, \{p=0, R=0, q=1\})) \vee (TC((p \wedge q) \rightarrow R, \{p=0, R=1, q=0\}) \wedge TC((p \wedge q) \rightarrow R, \{p=0, R=1, q=1\})))$$

$$\wedge ((TC((p \wedge q) \rightarrow R, \{p=1, R=0, q=0\}) \wedge TC((p \wedge q) \rightarrow R, \{p=1, R=0, q=1\})) \vee (TC((p \wedge q) \rightarrow R, \{p=1, R=1, q=0\}) \wedge TC((p \wedge q) \rightarrow R, \{p=1, R=1, q=1\})))$$

$$= ('si' \wedge 'si') \vee ('si' \wedge 'no') \vee ('si' \wedge 'si') = 'si'$$

$$= ('si' \wedge 'si') \vee ('no' \vee 'si') = 'si' \wedge 'si' = SI$$

basta invertire  $q_0$  e  $q_r$

- Esercizio 51:
  - Complemento: Se  $M$  è una TM che termina in  $O(n^k)$  e decide  $L$ , allora esiste una TM che decide  $\bar{L}$  in  $O(n^k)$
  - Unione: Siano  $L_1, L_2$  due linguaggi decisi da due macchine di Turing  $M_1, M_2$  rispettivamente in tempo  $O(n_1^{k_1})$  e  $O(n_2^{k_2})$ . Una TM per  $L_1 \cup L_2$  termina in tempo  $O(n_1^{k_1+k_2})$   $\leftarrow$  Nel caso peggiore, prima tutta  $M_1$ , e poi tutta  $M_2$
  - Composizione: Supponiamo che  $L_1$  e  $L_2$  siano linguaggi in  $P$  decisi da  $M_1$  e  $M_2$ . Vogliamo mostrare che  $L_1 \circ L_2 \in P$ . Costruiremo una macchina  $M$  che:
    - $\forall i \mid w_1 \dots w_i \circ w_{i+1} \dots w_n = w$  esegua  $M_1(w_1 \dots w_i) \circ M_2(w_{i+1} \dots w_n)$  e accetti se entrambe accettano
    - Se  $w$  ilo termina, rifiuti
 Termina in  $O((n_1+1)n^{\max(k_1, k_2)}) \rightarrow$  polinomica  $\vee$
- Star di Kleene: Se  $L \in P$ ,  $L^* \in P$ ? Poiché  $L \in P \Rightarrow \exists M_L$  che in tempo polinomiale termina e decide se  $w \in L$ 
  - Quando una stringa  $w \in L^*$ ?
    - $w = \epsilon \leftarrow$  chiunque sia  $L$ ,  $L^*$  contiene tra le sue stringhe anche la stringa vuota
    - $w \in L \leftarrow$  la chiusura di Kleene comprende il ling. stesso
    - $w = uv, u, v \in L^* \leftarrow$  una stringa  $z \in L^*$  se è possibile dividere  $z$  in segmenti appartenenti al linguaggio  $L$

- Intersezione:  $O(n^{k_1+k_2}) \leftarrow$  Abbiamo  $L_1$  e  $L_2$  decisi rispettivamente da  $M_1$ , in  $O(n_1^k)$  e  $M_2$ , in  $O(n_2^k)$   
Una TM esegue prima  $M_1$ , e poi  $M_2$ , accettando se accettano entrambe

- Differenza insiemistica:  $L_2 / L_1 = L_2 \cap \bar{L}_1$

### Esercizio 52.

Mostrare che il seguente linguaggio è  $P$ :  $RP = \{(n, m) \mid n, m \in \mathbb{N} \text{ e } n, m \text{ sono relativi primi}\}$   
 $L$  (due numeri sono relativamente primi se non hanno comuni divisori diversi da uno).

Creiamo una TM  $M(w)$  che verifica  $RP$ :

1. Verifica che  $w$  sia di tipo  $\langle n, m \rangle$ , altrimenti rifiuta  $\leftarrow$  Come con COMP ci affidiamo al non determinismo per indovinare il risultato e poi verifichiamoci
  2. Scegli non deterministicamente un numero  $x \neq 1$
  3. Se  $n/x, m/x \in \mathbb{N}$ , rifiuto
  4. Accetta
- $\downarrow$   
immaginiamo che il non determinismo esplori in  $O(1)$  tutto lo spazio delle soluzioni

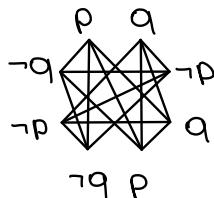
### Esercizio 53: Proprietà di chiusura di NP

- Unione: si -> TM non deterministica  $\leftarrow$  Intersezione

- Composizione: si -> TM non deterministica

- Kleene: si -> TM non deterministica  $\leftarrow$  si sceglie in modo non deterministico come dividere l'input, si copia a turno ciascuna parte sul secondo nastro in cui si scrive

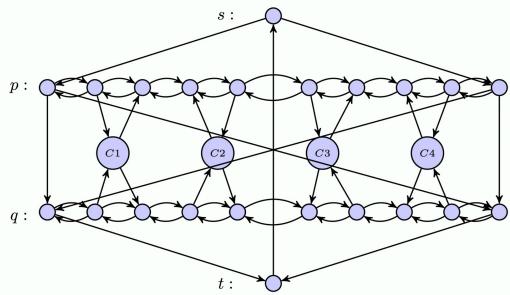
### Esercizio 54: $(p \vee q) \wedge (\neg p \vee q) \wedge (p \vee \neg q) \wedge (\neg p \vee \neg q)$ Trova il grafo facendo riferimento a 3SAT $\leq$ CUQUE



Per ogni formula con  $k$  clausole,  $q$  è soddisfacibile solo se  $G_q$  ha una  $k$ -clique  
Esistono varie 3-clique, ma non una 4-clique  $\rightarrow$  non soddisfacibile

Esercizio 55: Teorema  $\text{3SAT} \leq_p \text{HAM}$

$$\Psi: (p \vee q) \wedge (\neg p \vee q) \wedge (p \vee \neg q) \wedge (\neg p \vee \neg q)$$



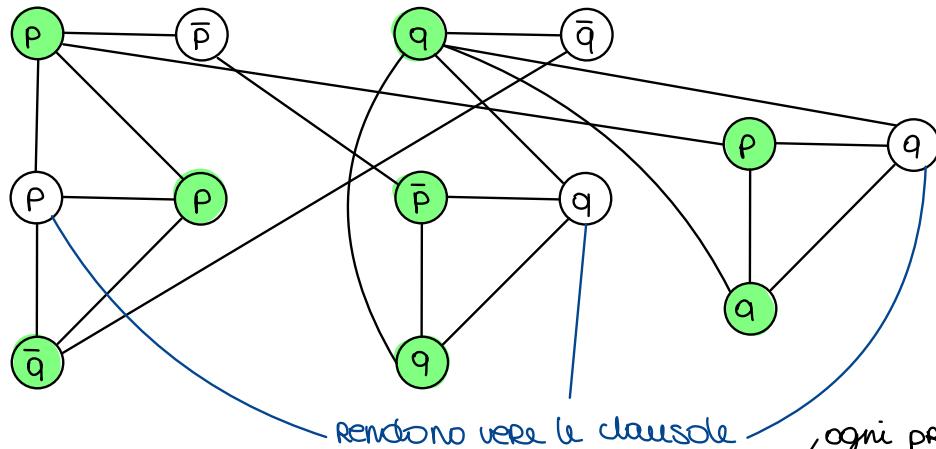
Nel grafo non esiste nessun ciclo Hamiltoniano, in quanto non è possibile percorrere le due tracce corrispondenti a  $p$  e  $q$  in nessun verso, tale che il percorso possa coprire contemporaneamente tutti e 4 i nodi  $C_1, C_2, C_3$  e  $C_4$ .

Infatti, poiché la formula non è soddisfacibile, non esiste un ciclo Hamiltoniano nel grafo.

Esercizio 56: Usando il teorema  $\text{HAM} \leq_p \text{UHAM}$  si dimostri che se il grafo di partenza non ha un ciclo Hamiltoniano, non lo ha neppure quello di arrivo.

Poiché il teorema  $\text{HAM} \leq_p \text{UHAM}$  dimostra che per ogni grafo diretto non pesato esiste un grafo indiretto non pesato tale che il primo ha un ciclo Hamiltoniano se e solo se lo ha il secondo, è ovvio anche il contrario.

Esercizio 57: Si consideri la formula 3CNF:  $(p \vee p \vee \neg q) \wedge (\neg p \vee q \vee q) \wedge (p \vee q \vee q)$



$$R = t + 2S = 2 + 2 \cdot 3 = 8$$

Un arco è coperto quando almeno uno dei suoi vertici è nell'insieme di copertura

Punto al minimo possibile di vertici per coprire ogni clausola → sono obbligato a sceglierne due ← con un solo vertice non riesco a coprire tutti e tre gli archi

Esercizio 58: Si identifichi e spieghi l'errore nella seguente dimostrazione.

Sia  $L = \{0^n 1^n | n \geq 0\}$ , mostriamo  $L \leq_p \text{COVER}$  tramite:  $f(\langle G, k \rangle) = \begin{cases} 0 & \text{se } G \text{ ha } k \text{-copertura} \\ 1 & \text{altrimenti} \end{cases}$

Dato che COVER è NP,  $L \leq_p \text{COVER} \in L \in P$ , otteriamo  $P = NP$

$f \notin P$

Esercizio 59: Proprietà di chiusura di NPC  $\leftarrow$  NP completo

$\Sigma = \{0, 1\}$   $L_0 = \emptyset \notin \text{NPC}$ ,  $\Sigma^* \in \text{NPC}$ ,  $L_1 = \{0w | w \in L\}$   $L_2 = \{1w | w \in L\}$   $L \in \text{NPC}$

↑ no star di Kleene

· Intersezione:  $L_0, L_1 \in \text{NPC}$  ma  $L_0 \cap L_1 = L_0 \notin \text{NPC}$

· Unione:  $L_0' = L_0 \cup L_1 \in \text{NPC}$ , ma  $L_0' \cup L_1 = \Sigma^* \notin \text{NPC} \leftarrow$  composizione

· Differenza insiemistica:  $L \setminus L = L_0 \notin \text{NPC}$

### Esercizio 60: chiusura di PSPACE

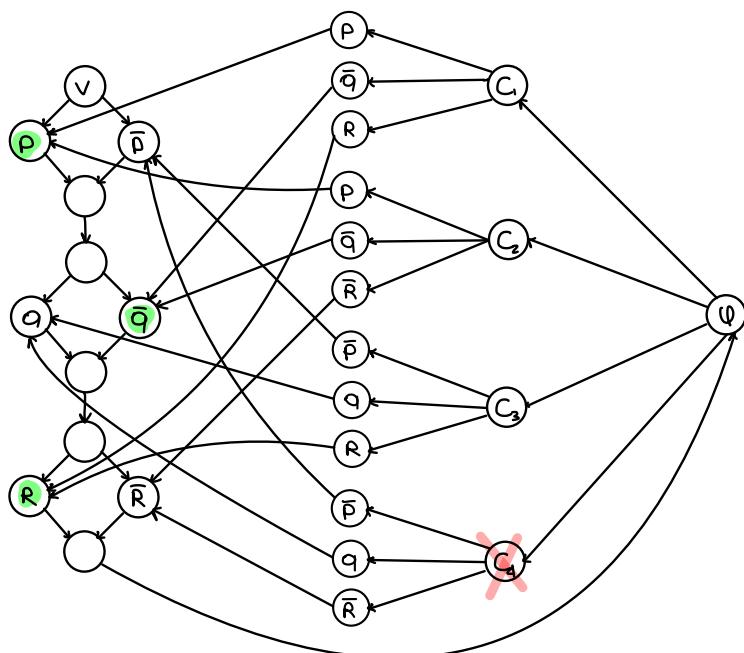
- Unione: date  $L_1, L_2 \in P$  decise da  $M_1$  e  $M_2$ , uno NTM  $M$  che esegue prima  $M_1$  e poi  $M_2$  riconosce l'unione di  $L_1$  e  $L_2$  in spazio polinomiale.  
Lo spazio necessario è pari a quello massimo necessario per  $M_1$  e  $M_2$ .
- Composizione: non determinismo per dividere l'input → eseguo  $M_1$  e  $M_2 \rightarrow$  spazio polinomiale
- Complemento: si  $\leftarrow$  invertire il risultato
- Star di Kleene: si, si divide la parola in n sotto parola e si esegue su ognuna  $\leftarrow$  in sequenza
- Intersezione:  $L_1 \cap L_2 = \overline{(L_1 \cup L_2)}$
- Differenza:  $L_1 \setminus L_2 = L_1 \cup \overline{L_2}$

Esercizio 61: Se una TM deterministica e terminante ha  $C_S = O(f(n))$  possiamo dare un limite superiore alla sua complessità in tempo?

Sì, in quanto il numero di posizioni per la testina è limitato anch'esso a  $C_S$   
 $\rightarrow$  il numero di configurazioni è limitato da  $O(qn^3sf(n)) = O(z^{f(n)})$ . La macchina non può mai entrare due volte nella stessa configurazione  
 $\rightarrow C_T = (z^{f(n)})$

Esercizio 63: Si usi QSAT  $\leq_p$  GG per dimostrare la verità della seguente formula proposizionale

$$\exists p \forall q \exists r ((p \vee \neg q \vee r) \wedge (p \vee q \vee \neg r) \wedge (\neg p \vee q \vee r) \wedge (\neg p \vee q \vee \neg r))$$



Esercizio 64: si dimostri che  $PSPACE\text{-hard} \subseteq NP\text{-hard}$

Poiché sappiamo che  $NP \subseteq PSPACE$ , in quanto una macchina che termina in tempo  $O(f(n))$  scrive alla peggio su  $O(f(n))$  celle di spazio, e ogni linguaggio  $PSPACE\text{-hard}$  non conosciamo la relazione esatta.  
 E' tale che ogni linguaggio si puo' ridurre a esso → puo' ridurre tutti i problemi in  $NP \rightarrow$  e' anche  $NP\text{-hard}$ .

Esercizio 65: Dimostra le proprietà di chiusura di LOG:

intersezione

Unione: Sia  $L_1$  decisa in spazio  $O(k_1 \lg(n))$  e  $L_2$  in  $O(k_2 \lg(n))$ . Una TM esegue prima  $M_1$ , e poi, in caso  $M_1$ , rifiuti,  $M_2$ . E' una TM per  $L_1 \cup L_2$  che nel caso peggiore termina in spazio  $O((k_1 + k_2) \lg(n))$  che è ancora logaritmico.

Composizione: Divido  $w$  in tutte le sue possibili scomposizioni in due parti, cercando le due sottostringhe riconosciute da  $M_1$  e  $M_2$ .

$M$  non scrive tutte le possibili ss ma mantiene un contatore che indica la fine della prima  
↳  $M_1$  sulla prima parte e  $M_2$  sull'altra → come in unione

Complemento: Se  $M_1$  termina in  $O(k \lg(n))$ , allora lo fa anche  $\bar{M}$ .

Differenza:  $L_1 \setminus L_2 = L_1 \cap \bar{L}_2$

Esercizio 66:  $A_{DFA} \in LOG$

E' possibile simulare un DFA usando due contatori, che indicano lo stato del DFA e la posizione della testina di lettura

↳ Spazio logaritmico

Esercizio 67: BIN:  $\{(a,b,c) | a, b, c \text{ sono interi binari e } a \cdot b = c\} \in LOG$

Il controllo dell'input costa tempo, ma non spazio.

Il controllo del prodotto si può fare controllando i risultati delle singole cifre usando un indice per la cifra  $c$  da controllare, due indici per le cifre di  $a$  e  $b$ , un accumulatore con il risultato corretto della cifra in considerazione e un accumulatore per il riporto → LOG

Esercizio 68: Chiusura di NLG, sapendo che è chiuso per complemento.

Unione:  $L_1, L_2 \in NLG$ . M esegue un passo non deterministico per decidere se eseguire  $M_1$  o  $M_2$ , riconoscendo così  $L = L_1 \cup L_2$  in spazio logaritmico

Complemento: come sopra. Usa un indice per indicare la divisione della parola

Kleene: come sopra, con due indici, analizzo due sottoparole alla volta.

Intersezione:  $L_1 \cap L_2 = (\overline{L_1} \cup \overline{L_2})^c$

Differenza:  $L_1 \setminus L_2 = L_1 \cap \overline{L_2}$

Esercizio 71: come si rapporta REG rispetto a PSPACE, NLG e LG?

La caratteristica degli automi regolari è che non hanno memoria

→ REG ⊂ NSPACE(1) → REG ⊂ PSPACE, REG ⊂ NLG e REG ⊂ LG.