

Testi consigliati (v. pagina Docenti > Programmi)

Davis et al., Computability, complexity
and languages

Sipser, Introduzione alla teoria
nella computazione

Consigliato: Algebra

Motivi:

— Storia

Anni '30: Church, Gödel, Kleene,
Post, Turing

Motivi:

- Legare con la "pratica"
- Maggiore consapevolezza della potenzialità e dei limiti dell'informatica

Ese.: Calcolabilità (funzioni non calcolabili)

Calcolabilità

Funzioni $f: \mathbb{N}^K \rightarrow \mathbb{N}$ K-arie

$$(a_1, \dots, a_K) \mapsto f(a_1, a_2, \dots, a_K)$$

Non interessa: complessità

Non tutte le f sono calc.

Risultati "positivi": funzione universale

Ling maggio S

Variabili

- di input : X_1, X_2, \dots
- di output : Y
- locali : Z_1, Z_2, \dots

$$X = X_1$$

$$Z = Z_1$$

Etichette

A_1, A_2, \dots

B_1, B_2, \dots

⋮

E_1, E_2, \dots

Istruzioni :

- Incremento: $V \leftarrow V + 1$
- Decremento: $V \leftarrow V - 1$
(solo se $V \neq 0$)
- Salto condizionato:
IF $V \neq 0$ GOTO L

Valore iniziale di $Y, Z_1, \dots, Z_m, \dots$ è 0

[A] $X \leftarrow X - 1$

$Y \leftarrow Y + 1$

IF $X \neq 0$ GOTO A

Condizioni di fermata:

Possima istruzione non trovata

$$f(x) = \begin{cases} 1 & \text{se } x=0 \\ x & \text{se } x \neq 0 \end{cases}$$

[A] IF $X \neq 0$ GOTO B

$Z \leftarrow Z + 1$

IF $Z \neq 0$ GOTO E

} = GOTO E

[B] $X \leftarrow X - 1$

$Y \leftarrow Y + 1$

$Z \leftarrow Z + 1$

IF $Z \neq 0$ GOTO A

} = GOTO A

$f_2(x) = x$

[A] IF $X \neq 0$ GOTO B

GOTO C

[B] $X \leftarrow X - 1$

$Y \leftarrow Y + 1$

$Z \leftarrow Z + 1$

GOTO A

[C] IF $Z \neq 0$ GOTO D

GOTO E

[D] $Z \leftarrow Z - 1$

$X \leftarrow X + 1$

GOTO C

$$f_B(x) = x$$

senza distruggere
il valore di X

$V \leftarrow V'$ Macro:

Ese.

$X_2 \leftarrow Z_3$

Come?

Macro $V \leftarrow V'$

1) Azzerare V : | [A] $V \leftarrow V - 1$
IF $V \neq 0$ GOTO A

2) Riutilizzare il programma che calcola f_3

con opportuni cambi di nome di variabile

(usare V al posto di Y , V' al posto di X
e così via) e di etichetta

$V \leftarrow 0$

[A] IF $V' \neq 0$ GOTO B

GOTO C

[B] $V' \leftarrow V' - 1$

$V \leftarrow V + 1$

$Z \leftarrow Z + 1$

GOTO A

[C] IF $Z \neq 0$ GOTO D

GOTO E ←

[D] $Z \leftarrow Z - 1$

$V' \leftarrow V' + 1$

GOTO C

da sostituire con l'etichetta
dell'istruzione da eseguire dopo $V \leftarrow V'$

$Y \leftarrow X_1$

$Z \leftarrow X_2$

[A] IF $Z \neq 0$ GOTO B
GOTO E

[B] $Z \leftarrow Z - 1$
 $Y \leftarrow Y + 1$
GOTO A

$$f_4(x_1, x_2) = x_1 + x_2$$

Se un programma calcola
la funzione $f(x_1, \dots, x_k)$,
possiamo riutilizzarlo come
sottoprogramma
(macro)

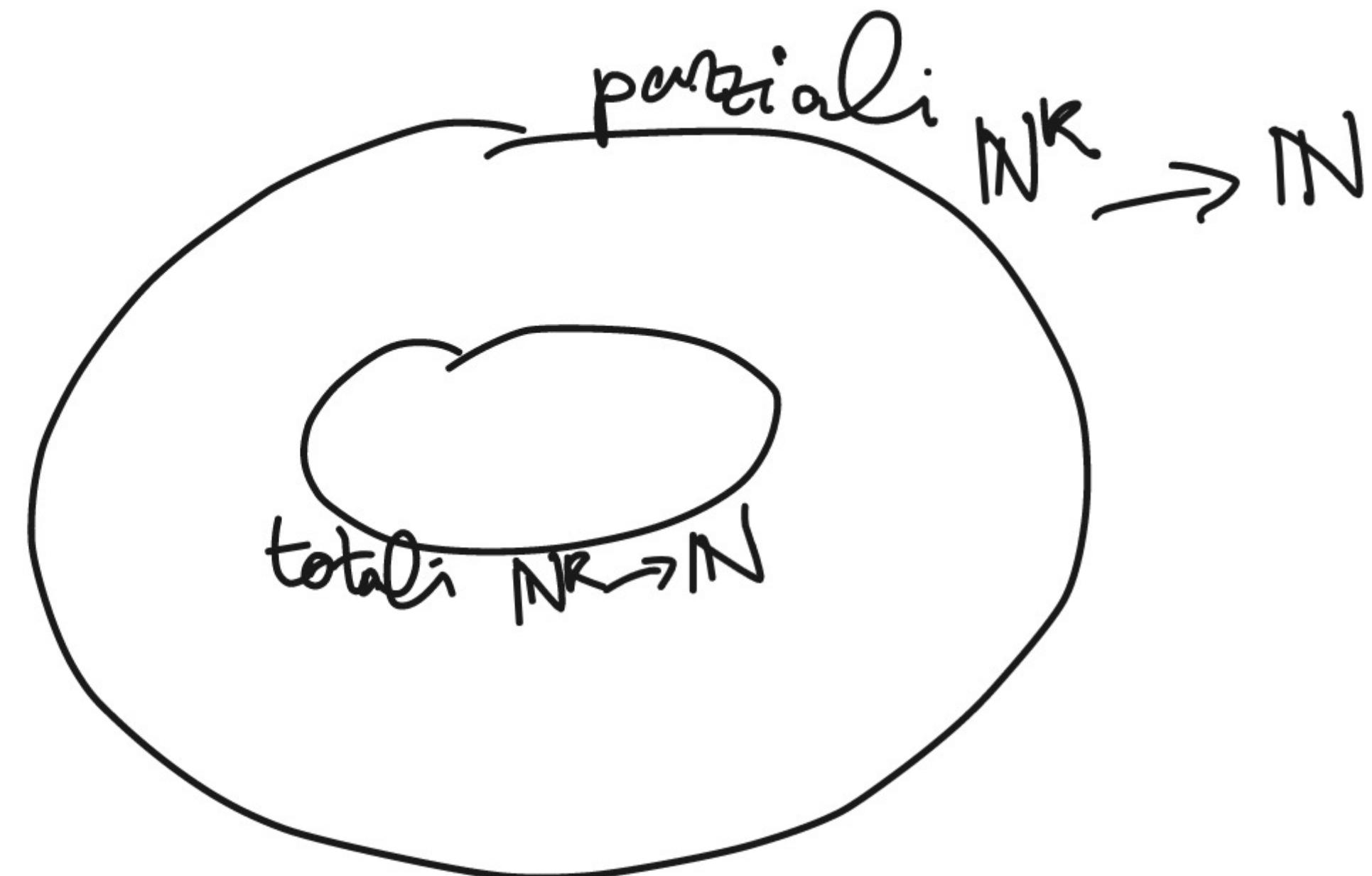
$$V' \leftarrow f(V_1, \dots, V_k)$$

Ad es. potremo scrivere

$Z_1 \leftarrow X_1 + Z_2$

$f: \mathbb{N}^k \rightarrow \mathbb{N}$ funzione totale: $\forall (x_1, \dots, x_k) \in \mathbb{N}^k, f(x_1, \dots, x_k) \in \mathbb{N}$

$g: S \subseteq \mathbb{N}^k \rightarrow \mathbb{N}$ funzione pazziale $\mathbb{N}^k \rightarrow \mathbb{N}$



Esempio: $f(x_1, x_2) = x_1 - x_2$ pazziale, non totale

$Y \leftarrow X_1$

$Z \leftarrow X_2$

[A] IF $Z \neq 0$ GOTO B

GOTO E

[B] IF $Y \neq 0$ GOTO C

GOTO B

[C] $Y \leftarrow Y - 1$

$Z \leftarrow Z - 1$

GOTO A

$$f_5(x_1, x_2) = \begin{cases} x_1 - x_2 & se x_1 \geq x_2 \\ \uparrow & altrimenti \end{cases}$$

Predicati : proposizioni (su una o più variabili)
con valore di verità definito

Esempio : $x_1 = x_2$

Predicati \leftrightarrow Funzioni totali a valori in $\{0, 1\}$

Predicato calcolabile se b è come funzione

Mac²⁰ : dato $P(x_1, \dots, x_k)$ calcolabile, || $Z \leftarrow P(v_1, \dots, v_k)$
IF $P(v_1, \dots, v_k)$ GOTO L { IF $Z \neq 0$ GOTO L

Esercizi (pag. 25 Davis)

Scrivere Sprogrammi per calcolare:

1) $P(x_1, x_2) \Leftrightarrow x_1 = x_2$ (senza macro)

2) $f_6(x) = 3x$

3) $f_7(x_1, x_2) = x_1 x_2$

4) $f_8(x) = \begin{cases} 1 & x \neq 0 \\ \uparrow & \text{altrimenti} \end{cases}$

SINTASSI

Variabili : X_1, X_2, \dots

Y

Z_1, Z_2, \dots

Etichette : A_1, A_2, \dots

\vdots

E_1, E_2, \dots

Statement : $V \leftarrow V+1$

$V \leftarrow V-1$

IF $V \neq 0$ GOTO L

$V \leftarrow V$

Istruzioni :
Statement

[L] Statement

Programma : $P = (I_1, \dots, I_n)$ con I_1, \dots, I_n istruzioni

$n=0$: programma vuoto

n è lunghezza del programma P | Calcola funzione nulla

Stato di un programma P
lista di uguaglianze del tipo $V = m$, con $m \in N$ e
 V variabile, contenente tutte le variabili che occorrono in P ,
una e una sola volta.

Istantanea : coppia (i, σ) con $1 \leq i \leq n+1$
(n lunghezza di P)
e σ stato di P .

Terminale se $i = n+1$

Se (i, σ) non terminale,

la successiva è (j, τ) , dove:

- 1) Se l-i-esima istruzione è $V \leftarrow V + 1$, τ si ottiene da σ incrementando di 1 il valore di V , e $j = i + 1$
- 2) se è $V \leftarrow V - 1$, τ si ottiene da σ decrementando di 1 V se non è già 0, e $j = i + 1$
- 3) se è $V \leftarrow V$, allora $\tau = \sigma$ e $j = i + 1$
- 4) se è IF $V \neq 0$ GOTO L, allora $\tau = \sigma$, e $j = i + 1$ se $V = 0$ in σ , altrimenti j è il minimo tale che la j-esima istruzione sia di dicitura con L, se esiste, altrimenti $j = n + 1$.

IF $x_1 \neq 0$ GOTO B

IF $x_2 \neq 0$ GOTO E

$Z_2 \leftarrow Z_2 + 1$

IF $Z_2 \neq 0$ GOTO C

[A] $x_1 \leftarrow x_1 - 1$

$x_2 \leftarrow x_2 - 1$

IF $x_1 \neq 0$ GOTO B

IF $x_2 \neq 0$ GOTO E

[C] $y \leftarrow y + 1$

$Z_1 \leftarrow Z_1 + 1$

IF $Z_1 \neq 0$ GOTO E

[B] IF $x_2 \neq 0$ GOTO A

		$x_1 = 0$	$x_1 \neq 0$
		$x_2 = 0$	1
		0	?
$x_2 \neq 0$	0		

Se $P(x_1, x_2)$ è calcolabile

IF $P(x_1, x_2)$ GOTO L ←

si può utilizzare come macro

Esempio di stato del programma precedente:

$$\sigma = \{X_1=5, X_2=7, Y=0, Z_1=0, Z_2=0, Z_5=7\}$$

Esempio di istantanea: (6, σ)

Istantanea successiva: (7, τ)

$$\text{dove } \tau = \{X_1=5, X_2=6, Y=0, Z_1=0, Z_2=0, Z_5=7\}$$

Altro esempio di istantanea: (8, σ)

Successiva: (13, σ)

Calcolo (terminante) :

lista (s_1, \dots, s_k) di istanze tali che:

- 1) per $i=2, \dots, k$, s_i è l'istante successivo di s_{i-1}
- 2) s_k è terminale.

Ad es. per il programma precedente, $((8\rho), (13, 0))$ è un calcolo.

Dato programma ρ e $m \geq 1$, definiamo $\psi_\rho^{(m)}$
la funzione m -aria calcolata da ρ

Ad esempio, se ρ è il programma che calcola $g(x_1, x_2) = x_1 + x_2$,
si ha $\psi_\rho^{(4)}(x_1, x_2, x_3, x_4) = x_1 + x_2$ (input ignorati se $m > \#\text{input in } \rho$)

Se P contiene $m' > m$ variabili di input, posso =0 tutte le variabili aggiuntive.

Ad esempio, data P che calcola $g(x_1, x_2) = x_1 + x_2$

$$\psi_P^{(1)}(x) = x$$

(Input non specificati partono da 0)

Dato $f(x_1, \dots, x_m)$ funzione parziale, essa si dà PARZIALMENTE CALCOLABILE se esiste P S-progrma tale da

$$\psi_P^{(m)}(x_1, \dots, x_m) = f(x_1, \dots, x_m) \text{ per ogni } x_1, \dots, x_m \in N$$

Esercizi : pag. 31 n. 4

Per il programma ρ seguente

$$Y \leftarrow X_2$$

[A] IF $X_2 = 0$ GOTO E

$$Y \leftarrow Y + 1$$

$$Y \leftarrow Y + 1$$

$$X_2 \leftarrow X_2 - 1$$

GOTO A

quali sono $\psi_\rho^{(1)}(x)$, $\psi_\rho^{(2)}(x_1, x_2)$, $\psi_\rho^{(3)}(x_1, x_2, x_3)$?

Stato iniziale relativo a P e $m > 0$, (x_1, \dots, x_m)

Se P contiene R var. di input,

lo stato iniziale σ_1 conterrà uguaglianze del tipo

$X_1 = x_1, X_2 = x_2, \dots, X_m = x_m, X_j = 0$ per $m+1 \leq j \leq k$

e $Y = 0, Z_1 = 0, \dots, Z_n = 0$

l'istante iniziale s_1 è $(1, \sigma_1)$

$\psi_{\beta}^{(m)}(x_1, \dots, x_m) = \begin{cases} \text{Valore della var. } Y \text{ nell'istante terminale di} \\ \text{un calcolo } s_1, \dots, s_k \text{ che parte dall'istante iniz. } s_1 \\ \text{relativa a } (x_1, \dots, x_m), \text{ SE ESISTE} \\ \uparrow, \text{ se un tale calcolo NON ESISTE, cioè se} \\ \text{esiste una successione INFINTA di istanze} \\ s_1, s_2, \dots \\ \text{che parte da quella iniziale, in cui} \\ s_{i+1} \text{ è successiva di } s_i \text{ per ogni } i > 0, \\ \text{e tale che nessuna } s_i \text{ è terminale} \end{cases}$

Una funzione m-aria $f(x_1, \dots, x_m)$ è PARZ. CALCOLABILE
(o PARZ. RICORSIVA) se esiste un S-programma ρ tale che

$$f(x_1, \dots, x_m) = \psi_\rho^{(m)}(x_1, \dots, x_m),$$

dove l'uguaglianza è verificata $\forall (x_1, \dots, x_m) \in \mathbb{N}^m$

(e in particolare f non è definita per tutti e soli i valori di (x_1, \dots, x_m) per i quali non è def. $\psi_\rho^{(m)}$).

"Moltiplicazione" ρ

$$Z_2 \leftarrow X_2$$

[A] IF $Z_2 \neq 0$ GOTO B
GOTO E

[B] $Z_2 \leftarrow Z_2 - 1$

$$\left. \begin{array}{l} Z_1 \leftarrow Y + X_1 \\ Y \leftarrow Z_1 \end{array} \right\} Y \leftarrow Y + X_1$$

GOTO A

$$\left| \begin{array}{l} \psi_{\rho}^{(2)}(x_1, x_2) = x_1 x_2 \\ \psi_{\rho}^{(1)}(x) = x \cdot 0 = 0 \\ \psi_{\rho}^{(5)}(x_1, \dots, x_5) = x_1 x_2 \end{array} \right.$$

ma: sostituzione macro del tipo

$$V' \leftarrow f(V_1, \dots, V_k)$$

Sostituzione macro $V' \leftarrow f(V_1, \dots, V_n)$ in un programma P'

Sia f calcolata da P , cioè $f = \psi_f^{(n)}$.

Supponiamo che P usi una sola etichetta di uscita E , più altre etichette A_1, \dots, A_ℓ .

Dunque $P = P(Y, X_1, \dots, X_m, Z_1, \dots, Z_k; E, A_1, \dots, A_\ell)$

Supponiamo che un altro programma P' contenga la macro
 $V' \leftarrow f(V_1, \dots, V_n)$; sia $m > 0$ tale che

$Z_m, Z_{m+1}, \dots, Z_{m+n+k}$, e $\hat{E}_m, A_{m+1}, \dots, A_{m+l}$ NON accorrono altrove in P'

Allora $V' \leftarrow f(V_1, \dots, V_n)$ in P' sta per:

$$Z_m \leftarrow 0$$

$$Z_{m+1} \leftarrow V_1$$

:

:

$$Z_{m+n} \leftarrow V_n$$

$$Z_{m+n+1} \leftarrow 0$$

:

$$Z_{m+n+k} \leftarrow 0$$

Q_m

$$[E_m] \quad V' \leftarrow Z_m$$

dove

$$Q_m = P(Z_m, Z_{m+1}, \dots, Z_{m+n}, Z_{m+n+1}, \dots, Z_{m+n+k}; \\ E_m, A_{m+1}, \dots, A_{m+l})$$

Composizione

Siano $h(x_1, \dots, x_n)$, $g_1(x_1, \dots, x_n)$, \dots , $g_K(x_1, \dots, x_n)$ funzioni (pazziali) n -arie, e $f(z_1, \dots, z_K)$ funzione K -aria, tali che:

$$h(x_1, \dots, x_n) = f(g_1(x_1, \dots, x_n), g_2(x_1, \dots, x_n), \dots, g_K(x_1, \dots, x_n)).$$

Allora h è ottenuta per composizione da f, g_1, \dots, g_K .

Teorema. Se f, g_1, \dots, g_K sono pazz. calcolabili e h è ottenuta da esse per composizione, allora h è pazz. calcolabile.

Dimostrazione. Il programma P

$$z_1 \leftarrow g_1(x_1, \dots, x_m)$$

$$\vdots$$

$$z_k \leftarrow g_k(x_1, \dots, x_m)$$

$$y \leftarrow f(z_1, \dots, z_k)$$

calcola h , cioè $h(x_1, \dots, x_m) = \psi_P^{(n)}(x_1, \dots, x_m)$, CVD

Ad esempio, la funzione $f_8(x) = 2x$ è calcolabile, dato che
 $f_8(x) = f_7(g_1(x), g_2(x))$ con $g_1(x) = 2$, $g_2(x) = x$ e

$$f_7(z_1, z_2) = z_1 z_2$$

Analogamente, la funzione $f_3(x) = 4x^2 - 2x$ è parzialmente alc.
essendo ottenuta per composizione da funzioni costanti, moltip. e sottrazione

Essendo anche totale (dato che $4x^2 \geq 2x$ per ogni x), è
calcolabile.

Ricorsione primitiva, 1^o caso

Siano h funzione unaria, g funzione bivaria, entrambe totali, tali che

$$h(0) = R \quad (\text{con } R \geq 0)$$

$$\text{e } h(t+1) = g(t, h(t)) \quad \text{per ogni } t \geq 0.$$

Allora h si dice ottenuta da g ($\in R$) per ricorsione primitiva

2^o caso.

Sia h funzione $(n+1)$ -aria, f funz. n -aria e g funz. $(n+2)$ -aria, tali tali che

$$h(x_1, \dots, x_n, 0) = f(x_1, \dots, x_n)$$

$$\text{e } h(x_1, \dots, x_n, t+1) = g(t, h(x_1, \dots, x_n, t), x_1, \dots, x_n)$$

Allora h è ottenuta da f e g per ricorsione primitiva.

Esercizi : pag. 32 n. 6 , pag. 36 n. 4

p. 32 n. 4

- a) Dim. che le funzioni (merse) costanti sono calcolabili
- b) Chiamiamo "straightline" un programma che non contiene istruzioni di salto.
Dimostrare (per induz. sulla lunghezza) che se un prog. straightline ha lunghezza K , allora $\psi_P^{(1)}(x) \leq K$ per ogni x
- c) Dim. che se un tale P calcola $\psi_P^{(1)}(x) = K$, allora la lung. di P è almeno K
- d) Dim. che $f(x) = x+1$ non è calcolata da nessun prog. straightline.
Notizie che la classe delle funz. calcolate da programmi del genere è PROPRIAMENTE contenuta in quella delle funz. calcolabili pa??.

p. 36 n. 4

Mostriare che il predicato binario $P(x_1, x_2) = (x_1 \leq x_2)$
è calcolabile.

miro

Base $k=0 \Rightarrow P$ non ha istruzioni

Soluz.
esercizi

$$\Rightarrow \psi_P^{(1)}(x) = 0 \quad \text{per ogni } x.$$

Sia vero per R e dimostriamolo per $K+1$.

Le prime K istruzioni di P formano un prog. straightline di lung. K . Dunque per ipotesi induttiva, dopo le prime K istruz. $y \leq R$. Poiché la $(K+1)$ -esima istruz. non è di salto, il valore di y dopo la sua esecuzione può al massimo incrementare di 1

(se è $y \leftarrow y+1$), altrimenti resta invariato (o diminuisce).

Dunque $\psi_P^{(1)}(x) \leq R+1$

$Z_1 \leftarrow X_1$

$Z_2 \leftarrow X_2$

[A] IF $Z_1 \neq 0$ GOTO B

$Y \leftarrow Y+1$

GOTO E

[B] IF $Z_2 \neq 0$ GOTO D

GOTO E

[D] $Z_1 \leftarrow Z_1 - 1$

$Z_2 \leftarrow Z_2 - 1$

GOTO A

Predicato

$$P(x_1, x_2) = (x_1 \stackrel{?}{\leq} x_2)$$

Teorema

Se $n \geq 0$ e h è una funzione $(n+1)$ -aria ottenuta da g e f calcolabili per ricorsione primitiva, allora h è calcolabile.

Dim.

$$h(x_1, x_2, \dots, x_n, 0) = f(x_1, \dots, x_n)$$

$$h(x_1, \dots, x_n, t+1) = g(t, h(x_1, \dots, x_n, t), x_1, \dots, x_n)$$

Scriviamo S-programma che calcoli h in base a f e g .

$$Y \leftarrow f(X_1, \dots, X_n)$$

[A] IF $X_{n+1} = 0$ GOTO E

$$Y \leftarrow g(Z, Y, X_1, \dots, X_n)$$

$$Z \leftarrow Z + 1$$

$$X_{n+1} \leftarrow X_{n+1} - 1$$

GOTO A

CVD

Classi PRC C

Una classe di funzioni si dice PRC se

1) contiene le funzioni INIZIALI:

- $n(x) = 0$

- $s(x) = x+1$

- $u_i^{(n)}(x_1, \dots, x_n) = x_i \quad \text{per } n > 0 \text{ e } 1 \leq i \leq n$

2) è chiusa rispetto alle operazioni di composizione e ricorsione primitiva.

Cioè: se f e g appartengono a C e h si ottiene da f e g per ric. prim.

allora $h \in C$; se f e g_1, \dots, g_k appartengono a C e h si ottiene da esse per compozit., allora $h \in C$.

Teorema

Le funzioni calcolabili formano una classe PRC.

Dim

Le funzioni iniziali sono calcolabili, e la classe è chiusa rispetto alle operazioni di composiz. e ricors. primitiva in base ai risultati già visti, CVD.

Una funzione (totale) si dice RICORSIVA PRIMITIVA se si può ottenere dalle funzioni iniziali mediante un numero finito di applicazioni delle operazioni di composizione e ricorsione primitiva.

Teorema: Una funzione è ricorsiva primitiva se e solo se appartiene a TUTTE le classi PRC.

ESEMPIO

- $h_1(x, y) = x + y$

$$h_1(x, 0) = x = u_1^{(1)}(x)$$

$$h_1(x, t+1) = x + t + 1 = s(x+t) = s(h_1(x, t)) =$$

$$= s(u_2^{(3)}(t, h_1(x, t), x))$$

- $\overline{h_2(x, y)} = xy$

$$h_2(x, 0) = 0 = n(x)$$

$$h_2(x, t+1) = xt + x = g(t, h_2(x, t), x)$$

dove $g(x_1, x_2, x_3) = u_2^{(3)}(x_1, x_2, x_3) + u_3^{(3)}(x_1, x_2, x_3)$

cioè
 $g(x_1, x_2, x_3)$
 $= h_1(u_2^{(3)}(x_1, x_2, x_3), u_3^{(3)}(x_1, x_2, x_3))$
Quindi g è RP,
e così lo è
 h_2

Corollario Ogni funzione ricorsiva primitiva è calcolabile.

Esercizi : $h_3(x) = x!$ e $h_4(x, y) = x^y$ sono eic. prim.

Altri esempi

$$p(x) = \begin{cases} x-1 & se \quad x > 0 \\ 0 & altrimenti \end{cases}$$

$$p(0) = 0$$

$$p(t+1) = t = u_1^{(2)}(t, p(t))$$

$$x \div y = \begin{cases} x-y & se \quad x \geq y \\ 0 & altrimenti \end{cases}$$

$$x \div 0 = x = u_1^{(1)}(x)$$

$$x \div (t+1) = p(x \div t) = p(u_2^{(3)}(t, x \div t, x))$$

$$|x - y| = (x \div y) + (y \div x)$$

$$\alpha(x) = (x \div 0) = \begin{cases} 1 & se \quad x=0 \\ 0 & altrimenti \end{cases}$$

$$\alpha(x) = 1 \div x .$$

Predicati ricorsivi primitivi

$$d(x, y) = (x \stackrel{?}{=} y) = \alpha(|x - y|)$$

Esercizio : $x \leq y$

Teorema Se P e Q sono predicati n -ari primitivi ricorsivi, allora sono ricorsivi primitivi anche $\neg P$, $\overset{\wedge}{P \wedge Q}$, $P \vee Q$.

Din.

$$(\neg P)(x_1, \dots, x_n) = \alpha(P(x_1, \dots, x_n))$$

$$(P \wedge Q)(x_1, \dots, x_n) = P(x_1, \dots, x_n) \wedge Q(x_1, \dots, x_n)$$

$$P \vee Q = \neg((\neg P) \wedge (\neg Q)) \quad \text{CVD}$$

Es. pag. 43 n. 3 , 47 n. 5

- Sia $n > 0$, e C una classe di funzioni totali di al più n variabili.
Dimostrare che C non è una classe PRC
- Data una funzione f unaria, sia $f^0(x) = x$ e
 $f^{n+1}(x) = f(f^n(x))$.
Sia $t_f(n, x) = f^n(x)$. Dimostrare che se f è ricorsiva primitiva,
allora lo è anche t_f .

Teorema. Se P e Q sono predici n-arie appartenenti a una classe PRC C , allora anche $\neg P$, $\neg Q$, $P \wedge Q$ e $P \vee Q$ appartengono a C .

Definizione per casi

Siano g, h funzioni n-arie in una classe PRC C , e sia P predicho n-ario in C . Allora se

$$f(x_1, \dots, x_n) = \begin{cases} g(x_1, \dots, x_n) & \text{se } P(x_1, \dots, x_n) \\ h(x_1, \dots, x_n) & \text{altrimenti} \end{cases},$$

si ha $f \in C$.

Dimostrazione Si ha $f(x_1, \dots, x_m) = g(x_1, \dots, x_m)P(x_1, \dots, x_m) +$
 $+ h(x_1, \dots, x_m)(\neg P(x_1, \dots, x_m))$.

Poiché g, h, P , somma e prodotto, $\neg P$ appartengono a C ,
si ottiene $f \in C$ per composizione, CVD.

Corollario. Siano g_1, \dots, g_m e h funzioni in C classe PRC,
e P_1, \dots, P_m predicati in C . Allora la funz. f definita da

$$f(x_1, \dots, x_m) = \begin{cases} g_1(x_1, \dots, x_m) & \text{se } P_1(x_1, \dots, x_m) \\ g_2(x_1, \dots, x_m) & \text{se } P_2(x_1, \dots, x_m) \\ \vdots & \\ g_m(x_1, \dots, x_m) & \text{se } P_m(x_1, \dots, x_m) \\ h(x_1, \dots, x_m) & \text{altrimenti} \end{cases}$$

appartiene a C

Dim. Induzione su m . Caso base $m=1$ già dim.

Sia vero per m e dimostriamo per $m+1$ casi.

Sia dunque $f(x_1, \dots, x_m) = \begin{cases} g_1(x_1, \dots, x_m) & \text{se } P_1(x_1, \dots, x_m) \\ \vdots \\ g_m(x_1, \dots, x_m) & \text{se } P_m(x_1, \dots, x_m) \\ g_{m+1}(x_1, \dots, x_m) & \text{se } P_{m+1}(x_1, \dots, x_m) \\ h(x_1, \dots, x_m) & \text{altrimenti} \end{cases}$

Definiamo $h^1(x_1, \dots, x_m) = \begin{cases} g_{m+1}(x_1, \dots, x_m) & \text{se } P_{m+1}(x_1, \dots, x_m) \\ h(x_1, \dots, x_m) & \text{altrimenti} \end{cases}$

Cosicché $h^1 \in C$

Allora possiamo scrivere $f(x_1, \dots, x_n) = \begin{cases} g_1(x_1, \dots, x_n) & \text{se } P_1(x_1, \dots, x_n) \\ \vdots \\ g_m(x_1, \dots, x_n) & \text{se } P_m(x_1, \dots, x_n) \\ h'(x_1, \dots, x_n) & \text{altrimenti} \end{cases}$

Per ipotesi di induzione, segue $f \in C$, CVD.

Prop. Sia C classe PRC, e $f \in C$ una funz. $(n+1)$ -aria. Allora \exists

$$g(y, x_1, \dots, x_n) = \sum_{t=0}^y f(t, x_1, \dots, x_n) \quad \text{e}$$

$$h(y, x_1, \dots, x_n) = \prod_{t=0}^y f(t, x_1, \dots, x_n) \quad \text{per ogni } y, x_1, \dots, x_n,$$

si ha $g, h \in C$.

Dim. Si ha }
$$\begin{aligned} g(0, x_1, \dots, x_m) &= f(0, x_1, \dots, x_m) \\ g(y+1, x_1, \dots, x_m) &= g(y, x_1, \dots, x_m) + f(y+1, x_1, \dots, x_m) \\ &\quad \vdots \\ r(y, g(y, x_1, \dots, x_m), x_1, \dots, x_m) \end{aligned}$$

Analogamente,

$$\begin{aligned} h(0, x_1, \dots, x_m) &= f(0, x_1, \dots, x_m) \\ h(y+1, x_1, \dots, x_m) &= h(y, x_1, \dots, x_m) + f(y+1, x_1, \dots, x_m) \end{aligned}$$

CJD

Quantificazione limitata

Sia P un predicato $(n+1)$ -ario, appartenente a una classe PRC C. Allora vi appartengono anche i predici

$$Q(y, x_1, \dots, x_n) = (\exists t \leq y : P(t, x_1, \dots, x_n))$$

$$R(y, x_1, \dots, x_n) = (\forall t \leq y, P(t, x_1, \dots, x_n)).$$

Dim. Basta osservare che

$$\left(\forall t \leq y, P(t, x_1, \dots, x_n) \right) = \prod_{t=0}^y P(t, x_1, \dots, x_n) \text{ è}$$

$$\left(\exists t \leq y : P(t, x_1, \dots, x_n) \right) \Leftrightarrow \left(\sum_{t=0}^y P(t, x_1, \dots, x_n) \neq 0 \right)$$

Esempi

$$y | x \Leftrightarrow \left(\exists t \leq x : y \cdot t = x \right) \text{ è predicato ricorsivo prim.}$$

$\text{Prime}(x) \Leftrightarrow x$ è numero primo

$\text{Prime}(x) \Leftrightarrow (x \geq 1) \wedge (\forall t \leq x, (t=1) \vee (t=x) \vee \neg(t|x))$

Anche Prime è predicato ric. prim.

Minimalizzazione limitata

Sia P predicato $(n+1)$ -ario in C classe PRC.

Definiamo $g(y, x_1, \dots, x_n) = \sum_{i=0}^y \prod_{t=0}^i (\neg P(t, x_1, \dots, x_n))$, cosicché $g \in C$.

Supponiamo che $\exists t \leq y : P(t, x_1, \dots, x_m)$, e diamiamo a il MINIMO tale t . Allora

$$\prod_{t=0}^i (\neg P(t, x_1, \dots, x_m)) = (\forall t \leq i, \neg P(t, x_1, \dots, x_m)) = \begin{cases} 1 & \text{se } i < t_0 \\ 0 & \text{altrimenti} \end{cases}$$

e dunque $g(y, x_1, \dots, x_m) = \sum_{i=0}^y \prod_{t=0}^i (\neg P(t, x_1, \dots, x_m)) = t_0$.

Definiamo

$$\min_{t \leq y} P(t, x_1, \dots, x_m) = \begin{cases} g(y, x_1, \dots, x_m) & \text{se } (\exists t \leq y : P(t, x_1, \dots, x_m)) \\ 0 & \text{altrimenti} \end{cases}$$

Essendo questa una def. per casi che usa funzioni e predicati in C,
abbiamo $\left(\min_{t \leq y} P(t, x_1, \dots, x_n) \right) \in C$.

Esempio

$$\lfloor x/y \rfloor = \min_{t \leq x} ((t+1)y > x) \quad \left(\text{assumendo } \lfloor x/0 \rfloor = 0 \right)$$

Esercizio: $x \bmod y$, pag 54 n.3, p. 58 n. 1,2,5

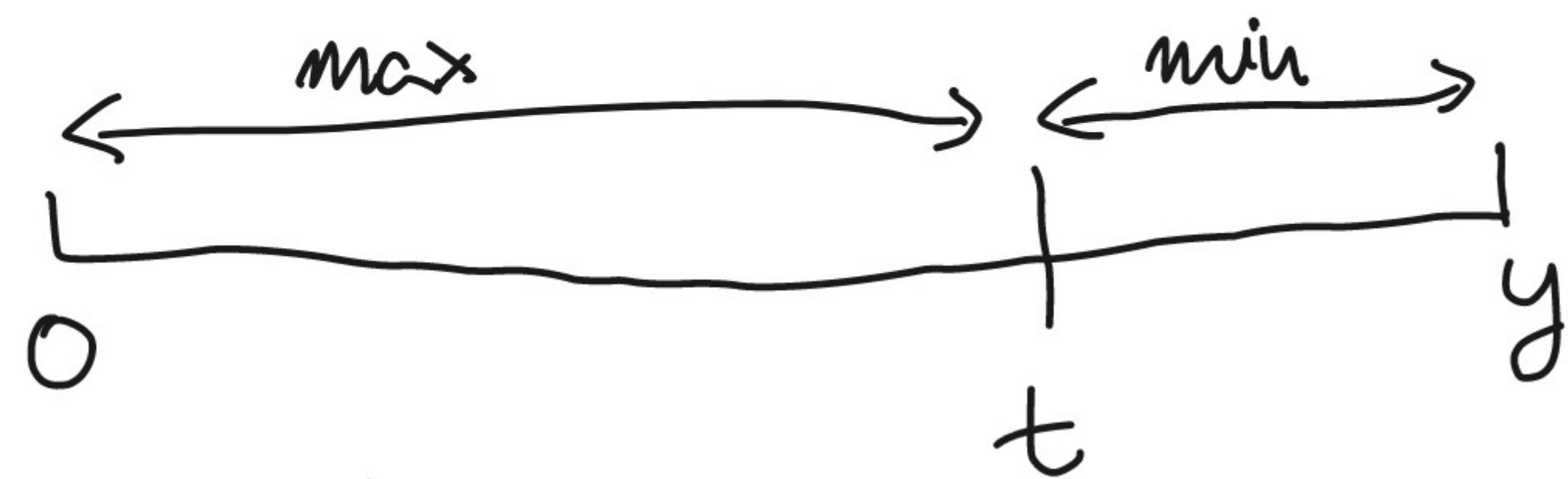
- Sia $\pi(x)$ il numero di primi $\leq x$. Mostrare che π è ric. prim.
- Sia $h(x) = \lfloor \sqrt{2}x \rfloor$, cioè il numero nat. n per cui

$$n \leq \sqrt{2}x < n+1$$
.
 Dim. che h è ric. prim.
- Lo stesso per $h'(x) = \lfloor (1+\sqrt{2})x \rfloor$
- Sia $R(x,t)$ predicato ric. prim. Dim. che

$$\max_{t \leq y} R(x,t) = \begin{cases} \text{massimo } t \leq y \text{ per cui } R(x,t) \\ \text{esiste,} \\ 0 \quad \text{altrimenti} \end{cases}$$
 definisce una funz. ricorsiva primitiva.

Soluzione

$$\max_{t \leq y} R(x, t) = \begin{cases} \text{massimo } t \leq y \text{ tale che } R(x, t) \text{ se esiste} \\ 0 \text{ altrimenti} \end{cases}$$



$$\max_{t \leq y} R(x, t) = \begin{cases} y - \min_{s \leq y} R(x, y-s) & \text{se } \exists t \leq y : R(x, t) \\ 0 & \text{altrimenti} \end{cases}$$

miro

Minimalizzazione non limitata

Sia P predicato $(n+1)$ -ario, e definiamo

$$f(x_1, \dots, x_n) = \min_y P(x_1, \dots, x_n, y) = \begin{cases} \text{il minimo valore di } y \text{ per cui } P(x_1, \dots, x_n, y), \\ \quad \text{se esiste} \\ \uparrow \quad \text{altrimenti} \end{cases}$$

Teorema

Se P è calcolabile, f è parzialmente calcolabile.

Dimostrazione

Il seguente programma calcola f :

[A] IF $P(X_1, \dots, X_n, Y)$ GOTO E

$Y \leftarrow Y + 1$

GOTO A

CVD

Osservazione

Tutte le funz. ricorsive primitive sono calcolabili, ma il viceversa
non vale. Vedremo più avanti.

Sia $p_0=0$, e p_n l'¹_n-esimo numero primo per $n \geq 1$

(ad es. $p_1=2$, $p_2=3$, $p_3=5, \dots$)

Allora $p_{n+1} = \min_{t \leq p_n! + 1} (\text{Prime}(t) \wedge (t > p_n))$

Infatti tra gli interi compresi tra $p_n + 1$ e $p_n! + 1$ dev'essere almeno un numero primo. Se così non fosse, i fattori primi di $p_n! + 1$ sarebbero tutti $\leq p_n$. Questo è assurdo perché $p_n! + 1$ non è multiplo di nessun numero $\leq p_n$ (la divisione dà sempre resto 1).

Poiché ancora non sappiamo che p_n è ric. prim., l'identità (può vera)

$$p_{n+1} = \min_{t \leq p_n! + 1} (\text{Prime}(t) \wedge (t > p_n)) \quad \text{non può essere usata per dimostrarlo.}$$

Poniamo allora $h(y, z) = \min_{t \leq z} (\text{Prime}(t) \wedge (t > y))$ il minimo primo
compresso tra $y+1$ e z ,
se esiste, 0 altrimenti

e $R(x) = h(x, x! + 1)$, cosicché R è ric. prim.

Si ha $\begin{cases} p_{n+1} = R(p_n) \\ p_0 = 0 \end{cases}$, dunque per ricorsione primitiva si ottiene che p_n è ric. prim.

Definiamo, per $x, y \in \mathbb{N}$

$$\langle x, y \rangle = 2^x (2y+1) - 1$$

dato che $2^x (2y+1) \geq 1$

\langle , \rangle è funzione zic. prim. binaria ($\langle , \rangle : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$)

È inoltre biiettiva: infatti se $z = 2^x (2y+1) - 1$, allora $z+1 = 2^x (2y+1)$,

dunque x si ottiene come esponente della massima potenza di 2 che divide

$$z+1, \text{ e } y \text{ si ottiene da } 2y+1 = \frac{z+1}{2^x} \Leftrightarrow y = \left(\frac{z+1}{2^x} - 1 \right) / 2.$$

In aggiunta, le "inverse" l e r definite da

$$\langle l(z), r(z) \rangle = z \quad \text{per ogni } z \in \mathbb{N} \quad \text{sono anch'esse ricorse primitive}$$

NB:
$$\begin{cases} l(z) = u_1^{(2)}(\langle , \rangle^{-1}(z)) \\ r(z) = u_2^{(2)}(\langle , \rangle^{-1}(z)) \end{cases}$$

Infatti, se $\langle x, y \rangle = 2^x (2y+1) - 1 = z$, allora $x \leq z$ e $y \leq z$,
per cui $l(z) = \min_{x \leq z} (\exists y \leq z : \langle x, y \rangle = z)$ e

$$r(z) = \min_{y \leq z} (\exists x \leq z : \langle x, y \rangle = z).$$

La funzione \langle , \rangle ci permetterà di associare a ogni istruzione di un S-programma un numero naturale, e viceversa.

La definizione precisa di \langle , \rangle non è altrettanto importante rispetto alle sue proprietà (biettiva da $\mathbb{N} \times \mathbb{N}$ in \mathbb{N} , ric. primit. con le sue inverse)

Ad esempio, potremmo usare anche $\langle x, y \rangle' = \frac{1}{2} (x+y)(x+y+1) + y$.

Potremmo associare a ogni programma (= lista di istruzioni \rightarrow seq. finita di numeri) un singolo numero naturale, ci serviamo della funzione $[]$ (numero di Gödel)

Data $(a_1, \dots, a_n) \in \mathbb{N}^n$, si definisce $(n \geq 0)$

$$[a_1, \dots, a_n] = \prod_{i=1}^n p_i^{a_i}$$

Esempi

$$\langle 0, 0 \rangle = 2^0 (2 \cdot 0 + 1) - 1 = 0; \quad \langle 1, 2 \rangle = 2^1 (2 \cdot 2 + 1) - 1 = 9$$

$$l(0) = r(0) = 0$$

$$l(1) = ? \quad r(1) = ?$$

$$1 = \langle l(1), r(1) \rangle = 2^{l(1)} (2r(1) + 1) - 1 \Leftrightarrow 2 = 2^{l(1)} (2r(1) + 1) \Leftrightarrow l(1) = 1 \quad r(1) = 0$$

$$\langle 4, 5 \rangle = 2^4 (2 \cdot 5 + 1) - 1 = 175$$

$$[0, 0] = 2^0 \cdot 3^0 = 1 = [0] = [0, 0, 0] = []$$

$$[2, 0, 1] = 2^2 \cdot 5 = 20$$

$[,]$ non è iniettiva né suriettiva come funz. da \mathbb{N}^m in \mathbb{N}

Ma 0 è l'unico valore escluso: se $m > 0$, possiamo ottenere una sequenza di cui m sia il numero di Gödel della scomposiz. di m in primi.

miro

Es. pag. 62 n. 3

- Sia $F(0)=0$, $F(1)=1$, e $F(n+1)=F(n)+F(n-1)$ per $n \geq 1$.

Dimostrare che F è ricorsiva primitiva.

Soluzione

$$g(n) = \langle F(n), F(n+1) \rangle \quad \text{per } n \in \mathbb{N}$$

$$g(0) = \langle F(0), F(1) \rangle = \langle 0, 1 \rangle = 2^0(2 \cdot 1 + 1) - 1 = 2$$

$$\begin{aligned} g(n+1) &= \langle F(n+1), F(n+2) \rangle = \langle F(n+1), F(n) + F(n+1) \rangle \\ &= \langle r(g(n)), l(g(n)) + r(g(n)) \rangle \end{aligned}$$

$$\forall n \in \mathbb{N}, F(n) = l(g(n))$$

$$[,] : \mathbb{N}^* = \bigcup_{n \geq 0} \mathbb{N}^n \rightarrow \mathbb{N}$$

$$(a_1, \dots, a_n) \mapsto \prod_{i=1}^n p_i^{a_i}$$

Non suriettiva: 0 non si può ottenere

Ma per il teorema fondamentale dell'aritmetica, 0 è l'unico valore escluso

Non iniettiva : $[a_1, \dots, a_n] = [a_1, \dots, a_n, 0]$

Ma, sempre per il TFA, $([a_1, \dots, a_n] = [b_1, \dots, b_m], a_n \neq 0, b_m \neq 0) \Rightarrow n=m \text{ e } a_i = b_i \text{ per } 1 \leq i \leq n.$

Per $x, i \in \mathbb{N}$, definiamo

$$(x)_i = \min_{t \leq x} (\neg(p_i^{t+1} | x)), \text{ funzione ric. prim.}$$

Se $x = \prod_{j=1}^n p_j^{a_j}$, supponendo $\forall i \leq n$, allora $(x)_i = a_i$, cioè
l'esponente della massima potenza di p_i che divide x .

$$(x)_0 = 0 \text{ per def. e } (0)_i = 0 = (1)_i$$

$$j > i \Rightarrow (x)_j = 0$$

Definiamo anche

$$\text{Lt}(x) = \min_{i \leq x} ((x)_i \neq 0 \wedge \overbrace{\forall j \leq x, (j \leq i) \vee (x)_j = 0})$$

$$\text{Lt}(0) = 0 = \text{Lt}(1)$$

Esempio: se $x = 100 = 2^2 \cdot 5^2$, allora $(x)_1 = 2$, $(x)_2 = 0$, $(x)_3 = 2$, $(x)_j = 0$ per $j \geq 4$

$$\Rightarrow \text{Lt}(100) = 3$$

In generale, $([a_1, \dots, a_n])_i = \begin{cases} a_i & \text{se } 1 \leq i \leq n \\ 0 & \text{altrimenti} \end{cases}$

e, se $n \geq \text{Lt}(x)$, allora $[(x)_1, (x)_2, \dots, (x)_n] = x$

Codifica di istruzioni

Enumereziamo tutte le etichette: otteniamo $\#(L)$

$A_1, B_1, C_1, D_1, E_1, A_2, B_2, \dots$

$1, 2, 3, \dots$

es. $\#(B_2) = 7$

Facciamo lo stesso per le variabili: otteniamo $\#(V)$

$Y, X_1, Z_1, X_2, Z_2, \dots$

es. $\#(Y) = 1, \#(Z_2) = 5.$

Data un'istruzione I , definiamo $\#(I) = \langle a, \langle b, c \rangle \rangle$, dove:

- a è $\#(L)$ se I è etichettata da L (0 se non ha etichetta)
- c è $\#(V)-1$, dove V è menzionata in I .

$$- b = \begin{cases} 0, & \text{se } I \text{ è } V \leftarrow V \\ 1, & \text{se } I \text{ è } V \leftarrow V+1 \\ 2, & \text{se } I \text{ è } V \leftarrow V-1 \\ \#(L')+2, & \text{se } I \text{ è IF } V \neq 0 \text{ GOTO } L' \end{cases}$$

Esempio: $\#(X_2 \leftarrow X_2 + 1) = \langle 0, \langle 1, 3 \rangle \rangle = \langle 0, 13 \rangle = 26$

$$\#([B] \text{ IF } Z \neq 0 \text{ GOTO } E) = \langle 2, \langle 7, 2 \rangle \rangle = \langle 2, 639 \rangle = 5115$$

Consideriamo I tale che $\#(I) = 2020$:

$$2021 = 2^0 (2 \cdot 1010 + 1) \Rightarrow 2020 = \langle 0, 1010 \rangle$$

$$1010 = \langle 0, 505 \rangle \Rightarrow 2020 = \langle 0, \langle 0, 505 \rangle \rangle$$

$$\overline{I} = \left(\begin{matrix} X_{253} & \leftarrow X_{253} \end{matrix} \right)$$

Se $\beta = (I_1, \dots, I_n)$, definiamo

$$\#(\beta) = [\#(I_1), \dots, \#(I_n)] - 1$$

Aggiungiamo all' def. di S-programma valido la richiesta che l'ultima istruzione NON sia $y \leftarrow y$, poiché

$$\#(y \leftarrow y) = \langle 0, \langle 0, 0 \rangle \rangle = 0.$$

In tal modo, ad ogni programma valido corrisponde un numero naturale e viceversa.

Esempi Sia \varnothing il programma:

$[A] \quad y \leftarrow y$

$y \leftarrow y+1$

Si ha $\#(\varnothing) = [\#(I_1), \#(I_2)] - 1$, con
 $I_1 = ([A] \quad y \leftarrow y)$ e $I_2 = (y \leftarrow y+1)$

$$\#(I_1) = \langle 1, \langle 0, 0 \rangle \rangle = \langle 1, 0 \rangle = 1$$

$$\#(I_2) = \langle 0, \langle 1, 0 \rangle \rangle = \langle 0, 1 \rangle = 2$$

$$\Rightarrow \#(\emptyset) = [1, 2] - 1 = 2^1 \cdot 3^2 - 1 = 17.$$

Se \emptyset è il prog. vuoto, $\#(\emptyset) = 0$

Cerchiamo \emptyset tale che $\#(\emptyset) = 23$

$$\text{Si ha } 24 = [3, 1], \text{ e } 3+1 = 2^2(2 \cdot 0 + 1) \Rightarrow 3 = \langle 2, 0 \rangle = \langle 2, \langle 0, 0 \rangle \rangle \\ 1 = \langle 1, 0 \rangle = \langle 1, \langle 0, 0 \rangle \rangle$$

Dunque \emptyset è:

[B]	$y \leftarrow y$
[A]	$y \leftarrow y$

- Trovare il num. dei seguenti programmi:

a) IF $X \neq 0$ GOTO A

[A] $X \leftarrow X+1$

IF $X \neq 0$ GOTO A

[A] $Y \leftarrow Y+1$

(chi è $\psi_B^{(1)}(x)$?)

- Trovare P con $\#(P) = 575$.

b) [B] IF $X \neq 0$ GOTO A
 $Z \leftarrow Z+1$
IF $Z \neq 0$ GOTO B
[A] $X \leftarrow X$

Soluzioni

a) $2^{46} 3^{21} 5^{46} 7^5 - 1 \quad | \quad \psi_P^{(1)}(x) = \uparrow$

$$\langle 0, \langle 3, 1 \rangle \rangle = \langle 0, 24-1 \rangle = 46$$

b) $\psi_P^{(1)}(x) = \begin{cases} 0 & \text{se } x \neq 0 \\ \uparrow & \text{altrimenti} \end{cases}$

P con $\#(P) = 575$

$$576 = 2^6 3^2 = [6, 2]$$

$$\ell(6) = 0, \quad r(6) = 3 = \langle 2, 0 \rangle$$

$$\ell(2) = 0, \quad r(2) = 1 = \langle 1, 0 \rangle$$

$$y \leftarrow y - 1$$

$$y \leftarrow y + 1$$

Teorema Esistono funzioni che NON sono pazzalmente calcolabili

Dim. Per "argomento diagonale di Cantor".

È possibile ordinare le funzioni pazz. calc.^{unarie} in una successione (infinita):

Sia P_n il programma tale che $\#(P_n) = n$

Allora P_0, P_1, \dots è la successione di TUTTI gli S-programmi

$\psi_{P_0}^{(1)}(x), \psi_{P_1}^{(1)}(x), \dots$ (prendendo solo le funz. distinte da tutte le precedenti)

è successione di TUTTE le funz. unarie pazz. calc.

Tuttavia, NON è possibile ordinare tutte le funzioniузie in una successione! Neanche considerando solo quelle totali.

Per assurdo, sia $f_0, f_1, \dots, f_n, \dots$ una successione di TUTTE le funz. totaliузie. Definiamo $g(x) = f_x(x) + 1$. Allora $g \neq f_i$ per ogni $i \in \mathbb{N}$, dato che $g(i) = f_i(i) + 1 \neq f_i(i)$, quindi g non compare nella successione, assurdo.

Dunque, non è possibile ordinare tutte le funzioniузie in una successione e pertanto non è possibile che esse coincidano con tutte le funzioni пazz. calc., CVD.

Argomento diagonale

$f_0(0)$ $f_0(1)$ $f_0(2)$ - - -

$f_1(0)$ $f_1(1)$ $f_1(2)$ - . -

$f_2(0)$ $f_2(1)$ $f_2(2)$ - . -

- - -

$$g(n) = f_n(n) + 1 \neq f_n(n)$$

Definiamo una funzione

$$\text{HALT}(x, y) = \begin{cases} 1 & \text{se } \psi_{P_y}^{(1)}(x) \downarrow \\ 0 & \text{altrimenti, cioè se } \psi_{P_y}^{(1)}(x) \uparrow \end{cases}$$

Teorema HALT non è calcolabile

Dim. Per assurdo, sia HALT calcolabile.

Scriviamo allora il programma P

[A] IF HALT(X,X) GOTO A

e sia $y_0 = \#(P)$. Allora $\text{HALT}(x, y_0) \Leftrightarrow \neg \text{HALT}(x, x)$ per ogni x .

In particolare, si avrebbe $\text{HALT}(y_0, y_0) \Leftrightarrow \neg \text{HALT}(y_0, y_0)$, assurdo, \square .

Il risultato precedente stabilisce che

non è possibile trovare un algoritmo che ci dice se un dato programma si ferma oppure no su un input assegnato. ("Problema della fermata")

Questo perché assumiamo che

ogni algoritmo che lavora su numeri naturali si può realizzare tramite
un S-programma
(Tesi di Church-Turing)

Ciò non significa che non esistono programmi dei quali sappiamo dire con certezza su quali input si fermino!

Il programma vuoto (o uno qualsiasi fra i tanti esempi di programmi visti finora) è un esempio di programma per cui ciò è possibile.

Quello che non è possibile avere è un algoritmo GENERALE
che fornisca la risposta per OGNI programma e input assegnati.

Ciò non deve sorprendere: è facile costruire programmi di cui nessuno sa dire se si fermino.

Ad esempio, consideriamo la congettura di Goldbach:

"Ogni numero pari ≥ 4 è la somma di due numeri primi", problema aperto da secoli.

È facile scrivere un programma per trovare (se esiste) il più piccolo controesempio alla congettura: si noti che basta testare il predicato zic_prim .

$$G(n) = \neg \left(\exists x \leq n : \left(\exists y \leq n : \text{Prime}(x) \wedge \text{Prime}(y) \wedge x+y=n \right) \right)$$

Dato che la risposta alla congettura non è nota, nessuno sa se un programma che testi il predicato $G(n)$ su tutti gli $n \geq 4$ pari si fermerebbe mai.

La difficoltà nel problema delle fermate è rappresentata dal caso in cui il programma assegnato NON si ferma su un dato input.

Viceversa, se il programma si ferma e quindi restituisce un output, è possibile determinarla algoritmicamente (con un S-programma).

unico

Definiamo cioè

$$\Phi^{(n)}(x_1, \dots, x_n, y) = \psi_{P_y}^{(n)}(x_1, \dots, x_n), \text{ funzione UNIVERSALE } (n+1)\text{-aria}$$

Dimostreremo che $\forall n$, $\Phi^{(n)}$ è sempre parzialmente calcolabile -

Esercizi pag. 69 n. 2-4

2. Sia $\widehat{\text{HALT}}(x, y)$ il predicato definito da

$\widehat{\text{HALT}}(x, y) \Leftrightarrow$ il programma P_y non si ferma su x

Dim. che $\widehat{\text{HALT}}$ non è calcolabile

3. Sia $\text{HALT}^1(x) = \text{HALT}(l(x), R(x))$. Dim. che HALT^1 non è calc.

4. Dimostrare o confutare: "Se f è una funzione totale n-aria limitata, cioè tale da esiste R costante per cui $|f(x_1, \dots, x_n)| \leq R$ per ogni x_1, \dots, x_n , allora f è calcolabile".

Soluzioni

4. Falso: HALT è funzione binaria totale e limitata (≤ 1)
ma non calcolabile
3. Per assurdo, sia $\text{HALT}^1(x) = \text{HALT}(l(x), r(x))$ calcolabile.
Ma $\forall x, y \in \mathbb{N}$, $\text{HALT}(x, y) = \text{HALT}^1(\langle x, y \rangle)$, osicché
HALT si ottiene per composizione di HALT^1 e \langle , \rangle e sarebbe
dunque calcolabile, assurdo.

Ricordiamo che $\Phi^{(n)}(x_1, \dots, x_n, y) = \psi_{P_y}^{(n)}(x_1, \dots, x_n)$

Teorema (Universalità)

$\forall n \geq 1$, $\Phi^{(n)}$ è parzialmente calcolabile

Dim.

Scriviamo un programma U_n che calcola $\Phi^{(n)}$.

Memorizziamo gli stati di un programma attraverso un numero di Gödel: ad es.

$(y=y, X_1=x_1, Z_1=z_1, X_2=x_2, X_3=x_3) \mapsto [y, x_1, z_1, x_2, 0, x_3]$

Per comodità di lettura del codice, usiamo nomi di variabili ed etichette non previsti dal linguaggio S .

$$Z \leftarrow X_{m+1} + 1 \quad (\text{numero di Gödel della lista di istruzioni})$$

$$K \leftarrow 1 \quad (\text{istruzione da eseguire})$$

$$S \leftarrow \prod_{i=1}^m p_{2i}^{x_i} \quad (\text{stato, memorizzato come } [0, X_1, 0, X_2, 0, \dots, X_m])$$

$$[C] \text{ IF } (K = \lfloor t(Z) + 1 \vee K = 0) \text{ GOTO } F \quad (K=0 \text{ rappresenta salto verso})$$

etichetta mancante

$$U \leftarrow r((Z)_K) \quad ((Z)_K \text{ è il numero della } K\text{-sima istruzione,}\\ \text{di forma } \langle a, \langle b, c \rangle \rangle \text{ in cui } a,\\ \text{eventuale etichetta, non interessa ora})$$

IF $\ell(U) = 0$ GOTO N

$(U = \langle b, c \rangle, \text{ quindi } \ell(U) = b;)$
 $b=0 \Leftrightarrow \text{istruz. pigra}$

$P \leftarrow P_{\ell(U)+1}$

$(S = [Y, X_1, Z_1, X_2, \dots])$

IF $\ell(U) = 1$ GOTO A

(incremento)

IF $\neg(P | S)$ GOTO N

($\Leftrightarrow V = 0$)

IF $\ell(U) = 2$ GOTO M

$(\ell(U) = \ell((Z)_i) + 2)$

$K \leftarrow \min_{i \leq \ell(Z)}$

$\begin{cases} \text{caso solto: IF } V \neq 0 \text{ GOTO L} \\ \#(L) + 2 = c \\ K \leftarrow 0 \text{ se L non è presente} \\ \text{nel programma} \end{cases}$

GOTO C

[A] $S \leftarrow S \cdot P$

GOTO N

[M] $S \leftarrow [S/P]$

(fa in modo che in $S=[Y, X_1, Z_1, X_2, \dots]$,
l'esponente corrispondente a V aumenti di 1)

[N] $K \leftarrow K+1$

GOTO C

[F] $Y \leftarrow (S)_1$

(decremento)

CVD -

Nota: Scriviamo semplicemente Φ per $\Phi^{(1)}$

Definiamo il predicato

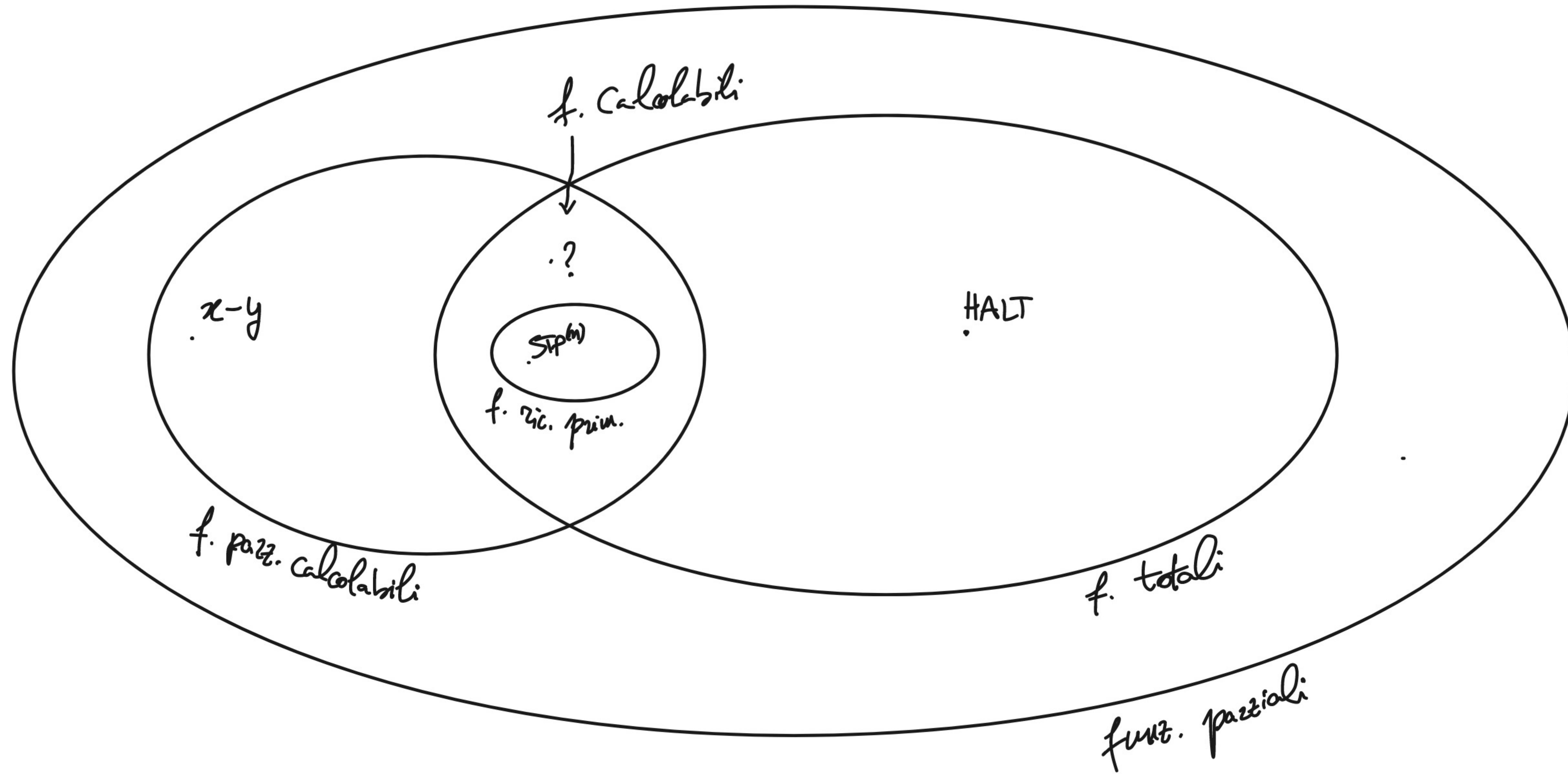
$\text{STP}^{(n)}(x_1, \dots, x_n, y, t) \Leftrightarrow$ Il programma P_y si ferma su (x_1, \dots, x_n) dopo al più t istruzioni.

A differenza di HALT , $\text{STP}^{(n)}$ è calcolabile : basta aggiungere a Σ_n un contatore per il numero di istruzioni eseguite.

In realtà, $\text{STP}^{(n)}$ è addizitiva ricorsivo primitivo.

Teorema Una funzione f è parzialmente calcolabile se e solo se si può ottenere dalle funzioni iniziali tramite l'applicazione di un numero finito di composizioni, ricorsioni primitive, e minimizzazioni (non limitate).

Funzioni a valori in \mathbb{N}



Intuitivamente, le funzioni ricorsive primitive sono quelle per le quali si possa trarre un algoritmo in cui ogni ciclo ha una durata massima predeterminata (= FOR).

Esempio di funzione calcolabile ma non ricorsiva primitiva:

funzione di Ackermann-Péter

$$A(x, y) = \begin{cases} y+1 & \text{se } x=0 \\ A(x-1, 1) & \text{se } y=0 \\ A(x-1, A(x, y-1)) & \text{altrimenti} \end{cases}$$

NB: $A(4, 2)$ ha oltre 19000 cifre decimali!

$$\text{Es. } A(0, 0) = 1$$

$$A(1, 0) = A(0, 1) = 2$$

$$A(2, 1) = A(1, A(2, 0)) = A(1, A(1, 1)) =$$

$$= A(1, A(0, A(1, 0))) = A(1, A(1, 0) + 1) = A(1, 3)$$

$$= A(0, A(1, 2)) = A(1, 2) + 1 = A(0, A(1, 1)) + 1 =$$

$$= A(0, 3) + 1 = 4 + 1 = 5$$

Esercizi pag. 77 n. 2 (albero a), 3

2a). Sia $H_1(x) = \begin{cases} 1 & \text{se } \Phi(x, x) \downarrow \\ \uparrow & \text{altrimenti} \end{cases}$

Mostrare che H_1 è pazz. calc.

b) Sia $A = \{a_1, \dots, a_n\} \subseteq \mathbb{N}$ un insieme finito tale che $\Phi(a_i, a_i) \uparrow$ per $1 \leq i \leq n$,

e sia $H_2(x) = \begin{cases} 1 & \text{se } \Phi(x, x) \downarrow \\ 0 & \text{se } x \in A \\ \uparrow & \text{altrimenti} \end{cases}$

Mostrare che H_2 è pazz. calc.

c) Trovare un insieme INFINITO $B \subseteq \mathbb{N}$ tale che $\Phi(b,b) \uparrow$ per ogni $b \in B$ e

$$H_3(x) = \begin{cases} 1 & \text{se } \Phi(x,x) \downarrow \\ 0 & \text{se } x \in B \\ \uparrow & \text{altrimenti} \end{cases}$$

sia part. calc.

d) Trovare un ins. INFINITO $C \subseteq \mathbb{N}$ tale che $\Phi(c,c) \uparrow$ per ogni $c \in C$ e

$$H_4(x) = \begin{cases} 1 & \text{se } \Phi(x,x) \downarrow \\ 0 & \text{se } x \in C \\ \uparrow & \text{altrimenti} \end{cases}$$

NON sia part. calc.

3. Dare un esempio di programma P tale che il predicato

$$H_P(x_1, x_2) \Leftrightarrow \text{Il programma } P \text{ si ferma su } (x_1, x_2)$$

NON sia calcolabile.

Soluzioni

2a) $H_1(x) = \begin{cases} 1 & \text{se } \Phi(x, x) \downarrow \\ \uparrow & \text{altrimenti} \end{cases}$

Scriviamo un programma che calcola H_1 :

$$Z \leftarrow \Phi(X, X)$$

$$Y \leftarrow Y + 1$$

Trovare C infinito per cui

2d) $H_4(x) = \begin{cases} 1 & \text{se } \text{HALT}(x, x) \\ 0 & \text{se } x \in C \\ \uparrow & \text{altrimenti} \end{cases}$

NON pazz. calc.

Il testo chiede che C sia
sottoinsieme di

$$\{x \in \mathbb{N} \mid \Phi(x, x) \uparrow\} =: \mathcal{H}$$

$$= \{x \in \mathbb{N} \mid \neg \text{HALT}(x, x)\}$$

Se scegliamo $C = \mathcal{H}$,

otteniamo $H_4(x) = \text{HALT}(x, x)$.

3) $H_P(x_1, x_2) \Leftrightarrow P$ si ferma su (x_1, x_2)

Cerchiamo P tale che $H_P(x_1, x_2) \Leftrightarrow \text{HALT}(x_1, x_2) \Leftrightarrow P_{x_2}$ si ferma su x_1 .

La soluzione è proprio $P = U_1$.

Infatti U_1 calcola $\Phi = \Phi^{(1)}$, definita da $\Phi(x, y) = \chi_{P_y}(x)$.

Dato $B \subseteq \mathbb{N}^n$, la sua funzione caratteristica di B come

$$f_B(x_1, \dots, x_n) = \begin{cases} 1 & \text{se } (x_1, \dots, x_n) \in B \\ 0 & \text{altrimenti} \end{cases}$$

(f_B è il predicato di appartenenza a B).

Diciamo che B è $\begin{cases} \text{ricorsivo primitivo} & \text{se e solo se lo è } f_B \\ \text{ricorsivo (cioè calcolabile)} & \end{cases}$

Più in generale, diciamo che B "appartiene" a una classe PRC C se e solo se $f_B \in C$

Ad esempio, l'insieme $P^{\subseteq \mathbb{N}}$ di tutti i numeri primi è un insieme ricorsivo primitivo. Infatti $f_p(x) = \text{Primes}(x)$ per ogni x .

Invece l'insieme $H' = \{(x, y) \in \mathbb{N}^2 \mid \text{HALT}(x, y)\}$ non è ricorsivo:

infatti $f_{H'}(x, y) = \text{HALT}(x, y)$.

Teorema Sia $B \subseteq \mathbb{N}^m$ e sia $B' \subseteq \mathbb{N}$ definito da $B' = \{[x_1, \dots, x_m] \mid (x_1, \dots, x_m) \in B\}$

Allora B appartiene a una data classe PRC se e solo se vi appartiene B' .

Dim.: Per ogni $x \in \mathbb{N}$,

$$x \in B' \Leftrightarrow ((x)_1, (x)_2, \dots, (x)_n) \in B \wedge H(x) \leq n \wedge x > 0$$

Viceversa, per ogni $x_1, \dots, x_n \in \mathbb{N}$

$$(x_1, \dots, x_n) \in B \Leftrightarrow [x_1, \dots, x_n] \in B'$$

Dunque f_B e $f_{B'}$ si ottengono l'una dall'altra per composizione con funzioni ricorsive primitive; pertanto se C è classe PRC, allora $f_B \in C \Leftrightarrow f_{B'} \in C$, c.d.

Teorema. Se $A, B \subseteq \mathbb{N}$ appartengono a una classe PRC, vi appartengono anche $A \cup B$, $A \cap B$, $\bar{A} := \mathbb{N} \setminus A$ ($\neg \bar{B}$).

Dim. Basta osservare che, per $x \in \mathbb{N}$,

$$f_{A \cup B}(x) = f_A(x) \vee f_B(x) \quad \left(x \in A \cup B \iff (x \in A) \vee (x \in B) \right)$$

Analogamente $x \in A \cap B \iff (x \in A) \wedge (x \in B)$

e $x \in \bar{A} \iff x \notin A \iff \neg(x \in A)$.

La tesi si ottiene dalla chiusura delle classi PRC rispetto alle operazioni booleane, CVD.

Un insieme $B \subseteq \mathbb{N}$ è ricorsivamente enumerabile (r.e.) se

esiste f unaria e parzialmente calcolabile tale che $B = \{x \in \mathbb{N} \mid f(x) \downarrow\}$

Laddove un insieme B è ricorsivo se esiste una procedura di decisione per l'appartenenza a B (un algoritmo sempre terminante che permetta di distinguere se l'input appartiene o no a B),

B è invece ricorsivamente enumerabile se esiste una proc. di semi-decisione (un algoritmo che termini se e solo se l'input appartiene a B).

Teorema B ricorsivo \Rightarrow B r.e.

Dcm. Troviamo g tale che $B = \{x \in \mathbb{N} \mid g(x) \downarrow\}$ scrivendo un programma che lo calcola:

[A] IF $f_B(X) = 0$ GOTO A

CVD.

Theorema

$B \in \mathbb{N}$ è ricorsivo $\Leftrightarrow B, \bar{B}$ sono R.e.

Dire "=>"

Se B è ricorsivo, lo è anche \bar{B} . Dal risultato precedente, B e \bar{B} sono allora r.l.

“ ”

" \Leftarrow " Siano f e g p.zz. calc. tali che $B = \{x \in N \mid f(x) \downarrow\}$ e $\bar{B} = \{x \in N \mid g(x) \downarrow\}$

Supponiamo due f sia calcolata dal programma di numero p
e g " " "

Allora il programma seguente calcola la funz. caratteristica di B:

[A] IF $STP^{(1)}(X, p, Z)$ GOTO C

IF $STP(X, q, Z)$ GOTO E

$Z \leftarrow Z + 1$

GOTO A

[C] $Y \leftarrow Y + 1$

CVD.

Es. pag. 84 n. 1

Diciamo che $B \subseteq \mathbb{N}^n$ è z.e. se $\exists f$. pass. calc. tale che $B = \{(x_1, \dots, x_n) \in \mathbb{N}^n \mid f(x_1, \dots, x_n) \downarrow\}$.

Dimostrare che B è z.e. se e solo se lo è $B' := \{[x_1, \dots, x_n] \mid (x_1, \dots, x_n) \in B\}$.

Soluzione es. p. 77 n. 2b

Sia $B = \{b_1, \dots, b_m\}^{\mathbb{C}^N}$ un insieme finito tale che $\Phi(b_i, b_i) \uparrow$. Dim.

che la funzione f definita da

$$f(x) = \begin{cases} 1 & \text{se } \Phi(x, x) \downarrow \\ 0 & \text{se } x \in B \\ \uparrow & \text{altrimenti} \end{cases}$$

è pass. calc.

calcola f .

IF $(X=b_1) \vee (X=b_2) \vee \dots \vee (X=b_m)$ GOTO E

$Z \leftarrow \Phi(X, X)$

$Y \leftarrow Y+1$

Teorema Se $A, B \subseteq \mathbb{N}$ sono r.e., allora anche $A \cap B$ è r.e.

Dim Siano f e g pazz. calc. tali che $A = \{x \in \mathbb{N} \mid f(x) \downarrow\}$ e $B = \{x \in \mathbb{N} \mid g(x) \downarrow\}$.

Il seguente programma termina se e solo se $X \in A \cap B$:

$$Z \leftarrow f(X)$$

$$Z \leftarrow g(X)$$

Cosicché il dominio della funzione da esso calcolata è $A \cap B$, CVD.

Teorema Se $A, B \subseteq \mathbb{N}$ sono z.e., lo è anche $A \cup B$.

Dim. Sia $A = \{x \mid f(x) \downarrow\}$, $B = \{x \mid g(x) \downarrow\}$, con f, g calcolate rispettivamente dai programmi di numero p e q .

Allora il seguente programma termina se e solo se $X \in A \cup B$:

[c] IF STP (X, p, Z) GOTO E
 IF STP (X, q, Z) GOTO E
 $Z \leftarrow Z + 1$
 GOTO C

CND .

Definiamo, per $n \in \mathbb{N}$, $W_n = \{x \in \mathbb{N} \mid \Phi(x, n) \downarrow\} = \{x \mid \psi_{P_n}^{(1)}(x) \downarrow\}$.

Evidentemente, $B \subseteq \mathbb{N}$ è r.e. $\Leftrightarrow \exists n \in \mathbb{N} : B = W_n$.

Infatti: B r.e. $\Leftrightarrow \exists f$ pazz. calc. tale che $B = \{x \mid f(x) \downarrow\}$
 $\Leftrightarrow \exists$ programma P tale che $B = \{x \mid \psi_P^{(1)}(x) \downarrow\}$
 $\Leftrightarrow B = W_n$ dove $n = \#(P)$

Insieme diagonale

$$K = \{n \in \mathbb{N} \mid \Phi(n, n) \downarrow\}.$$

Teorema. K è r.e., ma non ricorsivo

Distr.

K è evidentemente r.e. dato che $K = \{x \mid h(x) \downarrow\}$ dove $h(x) = \Phi(x, x)$.

Per dimostrare che K non è ricorsivo basta far vedere che $\overline{K} = N \setminus K = \{x \mid \Phi(x, x)\uparrow\}$ (insieme autidiagonale) non è ricorsivamente enumerabile.

Per assurdo, sia \overline{K} r.e., dunque esista $i \in \mathbb{N}$ per cui $\overline{K} = W_i = \{x \mid \Phi(x, i)\downarrow\}$.

Allora $i \in K \Leftrightarrow \Phi(i, i) \downarrow \Leftrightarrow i \in W_i \Leftrightarrow i \in \overline{K}$, assurdo, WD.

Prop. Sia $B \subseteq \mathbb{N}$ r.e. Allora esiste R predicato ricorsivo primitivo binario tale che
 $B = \{x \in \mathbb{N} \mid \exists t : R(x, t)\}$

Dim. Sia $B = W_n$. Allora $B = \{x \in N \mid \exists t : \text{STP}(x, n, t)\}$, c.d.

Corollario

Sia $B \neq \emptyset$ r.e. Allora esiste f ricorsiva primitiva tale che $B = f(N) = \{f(x) \mid x \in N\}$.

Dim.

Dal risultato prec. esiste R ric. prim. per cui $B = \{x \mid \exists t : R(x, t)\}$.

Sia $x_0 \in B$ e definiamo $f(x) = \begin{cases} l(x) & se R(l(x), z(x)) \\ x_0 & altrimenti \end{cases}$

f è ricorsiva primitiva. Dobbiamo dim. che $B = f(N)$

Se $n \in N$, allora $f(n) = l(n) \in R(l(n), z(n))$, cioè in particolare se $\exists t = z(n) : R(l(n), t)$, cioè $f(n) = l(n) \in B$; altrimenti $f(n) = x_0 \in B$. Viceversa, se $b \in B$, esiste $t : R(b, t)$.

Ma allora $f(\langle b, t \rangle) = \ell(\langle b, t \rangle) = b$, CVD.

Teorema Sia f part. alc. e $B = \{f(x) \mid f(x) \downarrow\}$. Allora B è z.e.

Dim. Sia f calcolata dal programma di numero p .

Allora il seguente programma termina se e solo se $X \in B$:

[A] IF $\neg \text{STP}(Z, p, T)$ GOTO C
 $V \leftarrow f(Z)$
IF $X = V$ GOTO E

(uso T anziché Z_2 per leggibilità)
(uso V anziché Z_3 -- --)

[C] $Z \leftarrow Z+1$
IF $Z \leq T$ GOTO A
 $T \leftarrow T+1$
 $Z \leftarrow 0$
GOTO A

Infatti, se il programma termina, necessariamente $X = f(Z)$ per qualche Z , e allora $X \in B$. Viceversa, se $X = f(Z)$ per qualche Z , il programma di numero p si ferma su Z in un certo numero di passi Q . Allora se $T \geq Q$ e $T \geq Z$, $\text{STP}(Z, p, T)$ è verificato e dunque la condizione di uscita viene raggiunta. CVD

Teorema Sia $B \subseteq \mathbb{N}$. Le seguenti affermazioni sono equivalenti:

- 1) B è r.e.
 - 2) B è l'insieme dei valori di una f ric. prim. (unaria)
 - 3) $B \sim$ " " " " " calcolabile
 - 4) $B \sim$ " " " " " parz.-calcolabile.

Din. Abbiamo visto $1 \Rightarrow 2$ e $4 \Rightarrow 1$; ma $2 \Rightarrow 3 \Rightarrow 4$ è banale, CVD.

Abbiamo visto come ogni algoritmo da lavorare su numeri naturali possa scomporsi in operazioni di base quali incrementi, decrementi e salti condizionati (da controlli "se $\neq 0$ "). Tali operazioni non sono davvero "di base" lavorando in base 10 (o 2).

Lo sono solamente in "base 1", rappresentando cioè ogni $n \in \mathbb{N}$

come $111 \cdots 1$.


Tale rappresentazione non è conveniente (n ha lunghezza n anziché $\log_{10} n$ o $\log_2 n$).

Si può allora pensare di passare a un linguaggio S_b , che lavora su stringhe su un alfabeto $\{0, 1, \dots, b-1\}$ (= numeri naturali, in base b). Le istruzioni ammesse in S_b saranno:

$V \leftarrow \sigma V$ (per ogni $\sigma \in A_b = \{0, \dots, b-1\}$)

$V \leftarrow V^-$ (cancello l'ultimo carattere se presente)

If V ENDS σ GOTO L (o ultimo carattere).

Il concetto di calcolabilità rispetto ai linguaggi $S (=S_1)$ e S_b è equivalente:
ogni funzione che sia purz. calca può essere calcolata da S_b -programmi e viceversa.

Esercizi pag. 84 n. 2, 7, 8

2) Dim. che $\{\langle x, y \rangle \mid x \in W_y\}$ è r.e.

7) Siano $A, B \subseteq \mathbb{N}$. Dimostrare o confutare: a) Se $A \cup B$ è r.e., allora A, B entrambi r.e.
b) Se $A \subseteq B$ e B è r.e., allora A è r.e.

8) Dim. che non esiste f calcolabile tale che $f(x) = \Phi(x, x) + 1$ se $\Phi(x, x) \downarrow$.

Soluzioni

Es. pag 84 n.7: Dati $A, B \subseteq N$

Dimostrare o confutare: a) B r.e. e $A \subseteq B \Rightarrow A$ r.e.
b) $A \cup B$ r.e. $\Rightarrow A, B$ r.e.

Per confutare a) basta trovare B r.e. e $A \subseteq B$ tale che A non è r.e.

$A = \bar{K}$, $B = N$ verifica questo requisito

b) $A = \bar{K}$, $B = N$ confuta anche b)
 $B = K$

Esercizio pag. 84 n. 8

Dimostrare che non esiste f calcolabile tale che $f(x) = \Phi(x, x) + 1$ se $\Phi(x, x) \downarrow$

Notiamo che $\forall x \in \mathbb{N}, \Phi(x, x) \downarrow \iff \text{HALT}(x, x)$

Per assurdo, supponiamo che esista f calcolabile, con $f(x) = \begin{cases} \Phi(x, x) + 1 & \text{se } \Phi(x, x) \downarrow \\ ? & \text{altrimenti} \end{cases}$

continua (\star)

Esercizio pag. 77 n. 2c

Trarre $B \subseteq \mathbb{N}$ infinito tale che $\Phi(b, b) \uparrow$ per ogni $b \in B$ e

$H_3(x) = \begin{cases} 1 & \text{x se } \Phi(x, x) \downarrow \\ 0 & \text{se } x \in B \\ \uparrow & \text{altrimenti} \end{cases}$ sia pazz. calcolabile.

Scelgiamo B in modo che i suoi elementi siano numeri di programmi che non terminano su nessun input. Ad esempio, tutti i programmi che iniziano con:

[A] $y \leftarrow y+1$
IF $y \neq 0$ GOTO A

Si ha $\#([A] \ y \leftarrow y+1) = \langle 1, \langle 1, 0 \rangle \rangle = \langle 1, 1 \rangle = 5$

e $\#(\text{IF } y \neq 0 \text{ GOTO A}) = \langle 0, \langle 3, 0 \rangle \rangle = \langle 0, 7 \rangle = 14$

Se \emptyset inizia con tali istruzioni, allora $2^5 3^{14} \mid (\#(\emptyset)+1)$ e
 $\neg (2^6 \mid (\#(\emptyset)+1))$, $\neg (3^{15} \mid (\#(\emptyset)+1))$

miro

In altre parole, detto $b = \#(P)$, $(b+1)_1 = 5$ e $(b+1)_2 = 14$.

Il seguente programma calcola H_3 :

IF $(X+1)_1 = 5 \wedge (X+1)_2 = 14$ GOTO E

$Z \leftarrow \Phi(X, X)$

$Y \leftarrow Y + 1$

(\star) Sia p il numero di un programma che calcola f , cosìché $f(x) = \Phi(x, p)$ per ogni x . In particolare dovrebbe aversi

$$f(p) = \Phi(p, p) = \Phi(p, p) + 1, \text{ assurdo.}$$

p. 84 n. 1

Sia $B \subseteq \mathbb{N}^m$ e diciamo che è r.e. se esiste f m-aria e pazz. calc. tale che

$$B = \{(x_1, \dots, x_m) \mid f(x_1, \dots, x_m) \downarrow\}$$

Dim. che B r.e. $\Leftrightarrow B' = \{[x_1, \dots, x_m] \mid (x_1, \dots, x_m) \in B\}$ è r.e.

" \Rightarrow " Basta trovare un programma da termini se e solo se $X \in B'$, ad es:

[A] IF $X=0 \vee \text{Lt}(X) > n$ GOTO A

$$Z \leftarrow f((X)_1, \dots, (X)_m)$$

" \Leftarrow " Supponiamo che $B' = \{x \mid g(x) \downarrow\}$ con g pazz. calc.

$$\text{Allora } B = \{(x_1, \dots, x_m) \mid g([x_1, \dots, x_m]) \downarrow\}$$

Macchine di Turing

Al posto di "programmi" (sequenze di istruzioni), abbiamo un dispositivo con un insieme finito di stati che legge e scrive su un "nastro" (potenzialmente infinito), scorrendo sul nastro di un carattere alla volta.

L'(eventuale) risultato del calcolo è rappresentato dal contenuto del nastro al quando la macchina si ferma. Formalmente:

Una macchina di Turing M è una quadrupla (Q, A, δ, q_1) in cui:

- Q è insieme finito di stati
- A è alfabeto finito | δ è funzione parziale da $Q \times (A \cup \{\sqcup\})$ a
- $q_1 \in Q$ (stato iniziale) | $Q \times (A \cup \{\sqcup, L, R\})$, con $L, R \notin A$

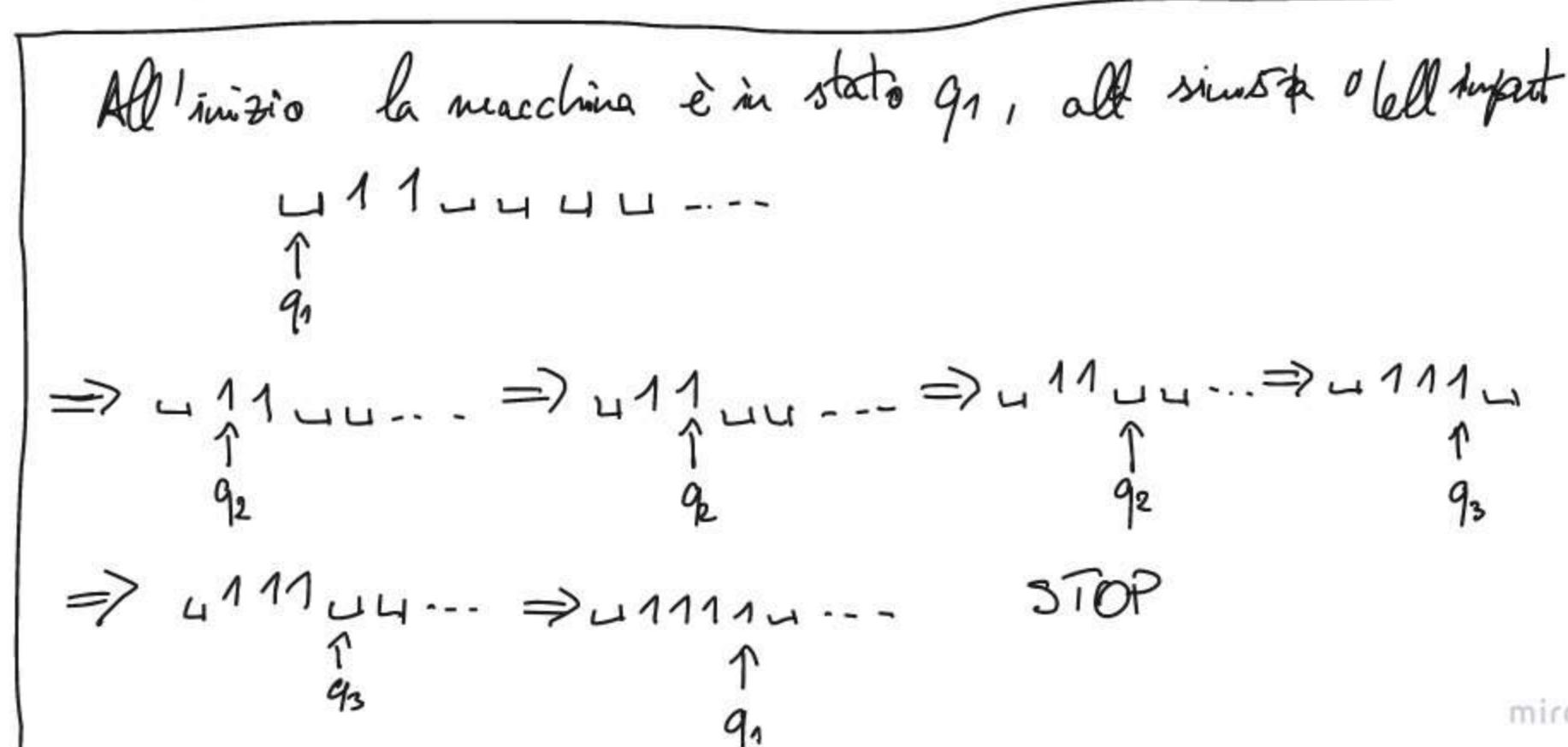
Una macchina M si può anche vedere come insieme di quadrupli

(q, a, q', a') dove $\delta(q, a) = (q', a')$, e $q, q' \in Q$, $a \in A \cup \{\leftarrow\}$,
 $a' \in A \cup \{\leftarrow, L, R\}$

Esempio

Sia $A = \{1\}$, $Q = \{q_1, q_2, q_3\}$, e M data da

$$\{(q_1, \leftarrow, R, q_2), \\ (q_2, 1, R, q_2), \\ (q_2, \leftarrow, 1, q_3), \\ (q_3, 1, R, q_3) \\ (q_3, \leftarrow, 1, q_1)\}$$



Il calcolo termina se si arriva a una coppia $(q, a) \in Q \times (A \cup \{\}\}$ non appartenente al dominio di δ , ossia tale da nessuna quadrupla di istruzioni inizia con (q, a) .

La macchina dell'esempio calcola la funzione $f(x) = x + 2$.

La macchina di Turing di esempio calcola la funzione $f(x)=x+2$

Tesi di Church-Turing: ogni algoritmo che lavora su (n-piè di) numeri naturali può essere realizzato mediante un'apposita macchina di Turing.

Quindi macchine di Turing e S-programmi (e S_b -programmi) sono funzionalmente equivalenti.

Nella definizione di MdT abbiamo usato un alfabeto qualsiasi (finito) A .

Dunque possiamo considerare come insieme dei possibili input non più N ($\circ N^n$),

ma $A^* = \{a_1 a_2 \dots a_n \mid n \in \mathbb{N}, a_1, \dots, a_n \in A\}$. NB: A^* contiene la

stringa vuota ϵ , di lunghezza $n=0$ (scriviamo $|\epsilon|=0$)

La memoria di una MdT è rappresentata dal nastro e dall'insieme finito di stati.

Chiamiamo linguaggio un sottoinsieme di A^* .

Diciamo che $L \subseteq A^*$ è accettato da una macchina M (scriviamo $L = L(M)$) se L è l'insieme di tutte le stringhe per le quali il calcolo di M ha termine (gli input su cui M si ferma). Ricordiamo che una MdT si ferma quando non ha istruzioni per la propria condizione attuale, rappresentata da una coppia stato - carattere.

Diciamo che L è ricorsivamente enumerabile se esiste una macchina M tale che $L = L(M)$. Un insieme di numeri è d.e. se e solo se lo è la sua rappresentazione in ogni base b .

Un automa finito deterministico (DFA) è come un MdT che possa scorrere una stringa solo in una direzione, senza sovrascrivere il nastro.

Formalmente, M è un DFA se:

$$M = (Q, A, \delta, q_1, F) \text{ dove}$$

- Q è un insieme finito di stati
- A un alfabeto finito
- $q_1 \in Q$ è lo stato iniziale
- $F \subseteq Q$ è l'insieme degli stati terminali o di accettazione.
- $\delta: Q \times A \rightarrow Q$ è una qualsiasi funzione totale, detta di transizione.

Dato un DFA, definiamo una funzione δ^* per ricorrenza, come segue:

$$\forall q \in Q, \quad \delta^*(q, \varepsilon) = q$$

$$\forall q \in Q, u \in A^*, a \in A, \quad \delta^*(q, ua) = \delta(\delta^*(q, u), a).$$

Il linguaggio accettato $L(M) \subseteq A^*$ è: $\{w \in A^* \mid \delta^*(q_1, w) \in F\}$.

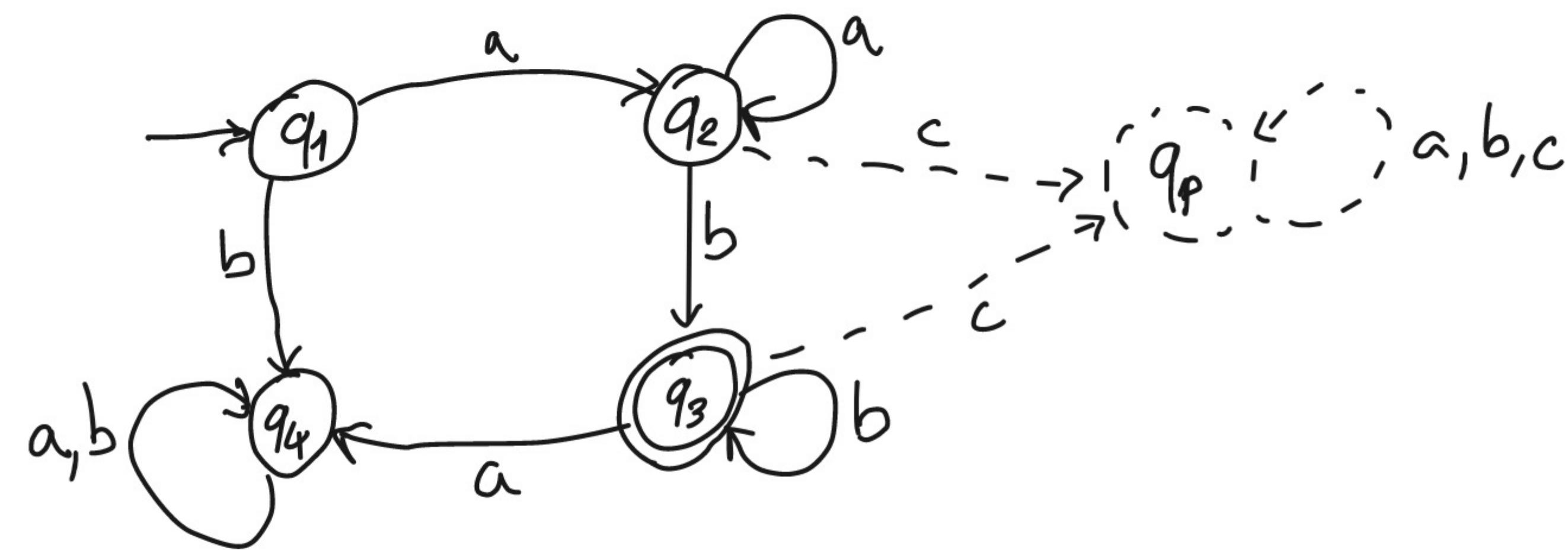
Esempio

Sia $M = (Q, A, \delta, q_1, F)$, con $Q = \{q_1, q_2, q_3, q_4\}$, $A = \{a, b\}$, $F = \{q_3\}$, e δ data da:

δ	a	b
q_1	q_2	q_4
q_2	q_2	q_3
q_3	q_4	q_3
q_4	q_4	q_4

Si ha $L(M) = \{a^n b^m \mid n, m \geq 0\}$, dove in generale w^i indica la concatenazione $\underbrace{w \dots w}_{i \text{ volte}}$ (Formalmente, $w^0 = \varepsilon$ e $w^{i+1} = w^i w$).

Diagramma di transizione (di stato) del DFA



Un linguaggio $L \subseteq A^*$ è regolare se esiste un DFA M tale che $L = L(M)$.

La nozione di regolarità è indipendente dall'alfabeto, cioè:

se $L \subseteq A^*$ e $A \subseteq B$, allora esiste DFA su A che accetta L se e solo se ne esiste uno su B . Infatti, se $M = (Q, A, \delta, q_1, F)$ accetta L , allora

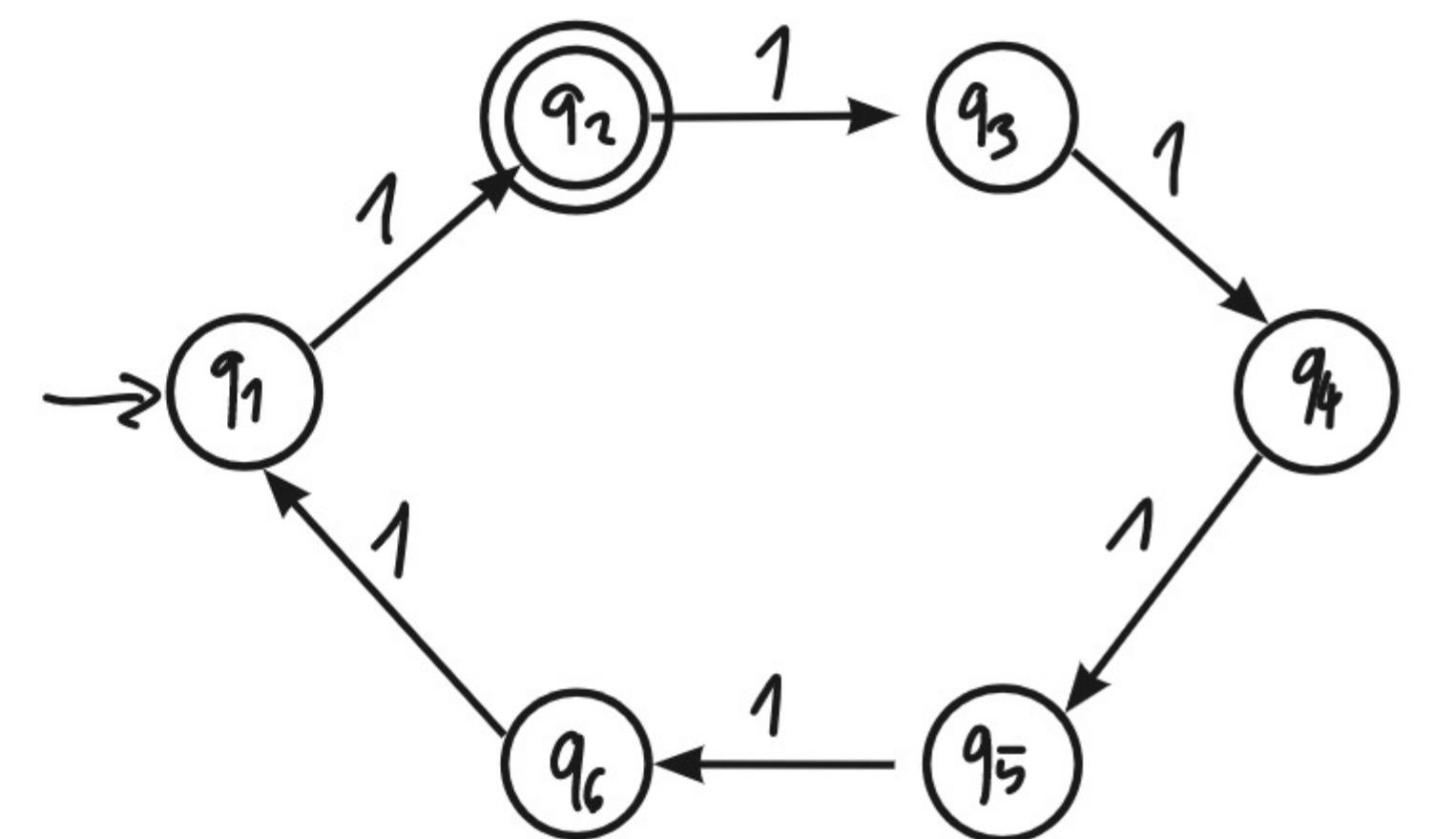
$M' = (Q \cup \{q_p\}, B, \delta', q_1, F)$ con $\delta'(q, a) = \delta(q, a)$ per $a \in A$ e $q \in Q$; $\delta'(q_p, x) = q_p$ per $x \in A \cup B$, e $\delta'(q, b) = q_p$ per $b \in B \setminus A$, accetterà L .

Viceversa, se $\mathcal{M}_B = (Q_B, B, \delta_B, q_1, F_B)$ è un DFA che accetta $L \subseteq A^*$, con $A \subseteq B$, allora anche $\mathcal{M}_A = (Q_B, A, \delta_A, q_1, F_B)$ accetta L , dove $\delta_A = \delta_B|_{Q \times A}$.

Esercizi di sintesi (pag. 240 n. 1a)

Sia $A = \{1\}$, $L = \{1^{6k+1} \mid k \geq 0\}$ (linguaggio delle rappresentaz. in base 1 dei numeri di forma $6k+1$). Dim. che è regolare. L'automa il cui diagramma è mostrato di seguito

accetta L .



Esercizi pag. 241 n. 1b, g, i, 3c, 4.

1b $A = \{a, b\}$. L : stringhe che finiscono con $bbab$

1g $A = \{a, b\}$. $L = \{a_1 \dots a_{m-2} a_{m-1} a_m^{ \in A^* } \mid a_{m-2} = b\}$.

1i // L : stringhe che non contengono aaa .

3c.

	a	b	c
q_1	q_2	q_2	q_1
q_2	q_3	q_2	q_1
q_3	q_1	q_3	q_2

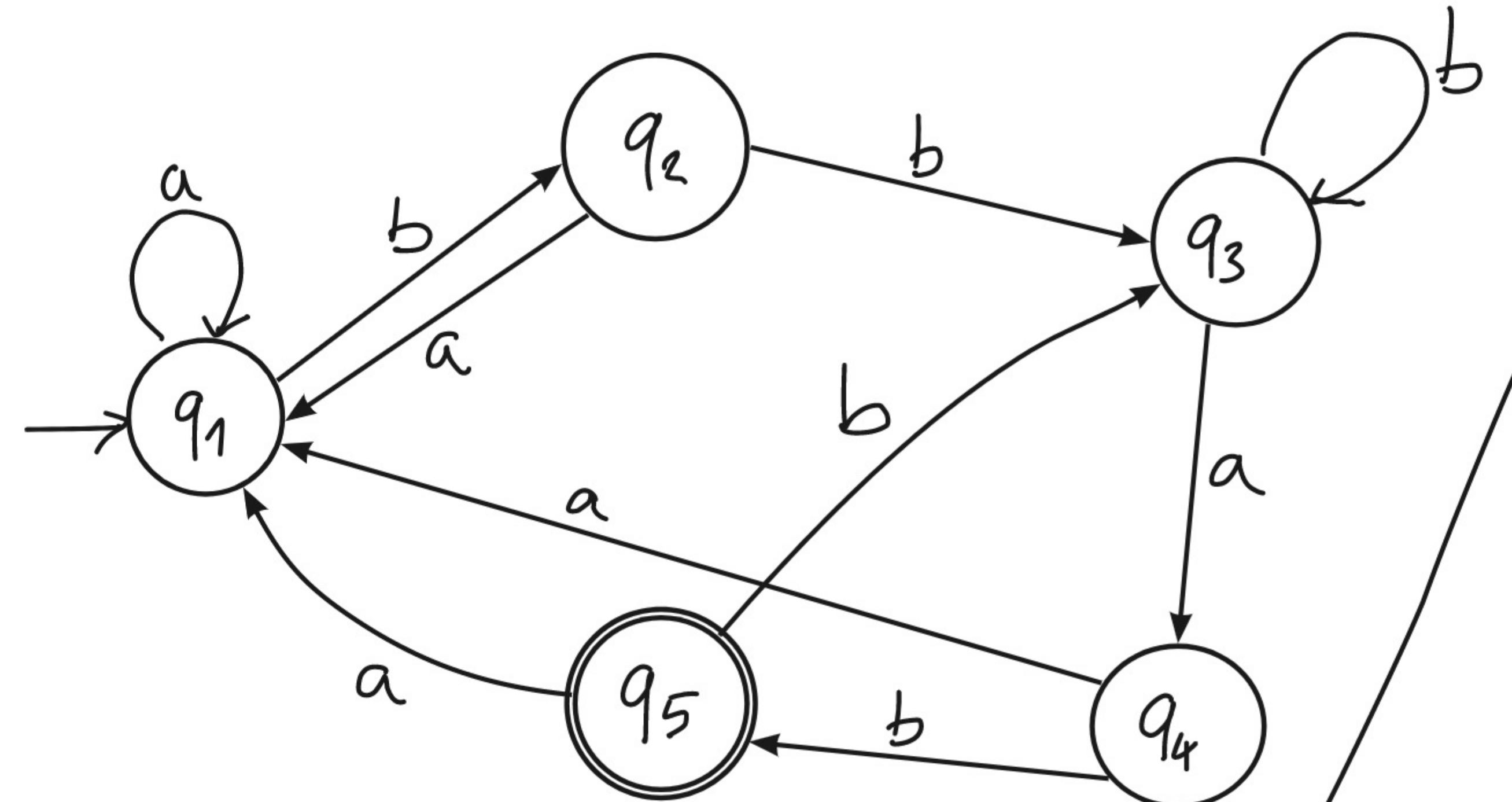
, st. init. q_1 , $F = \{q_2\}$

$L(m) = ?$

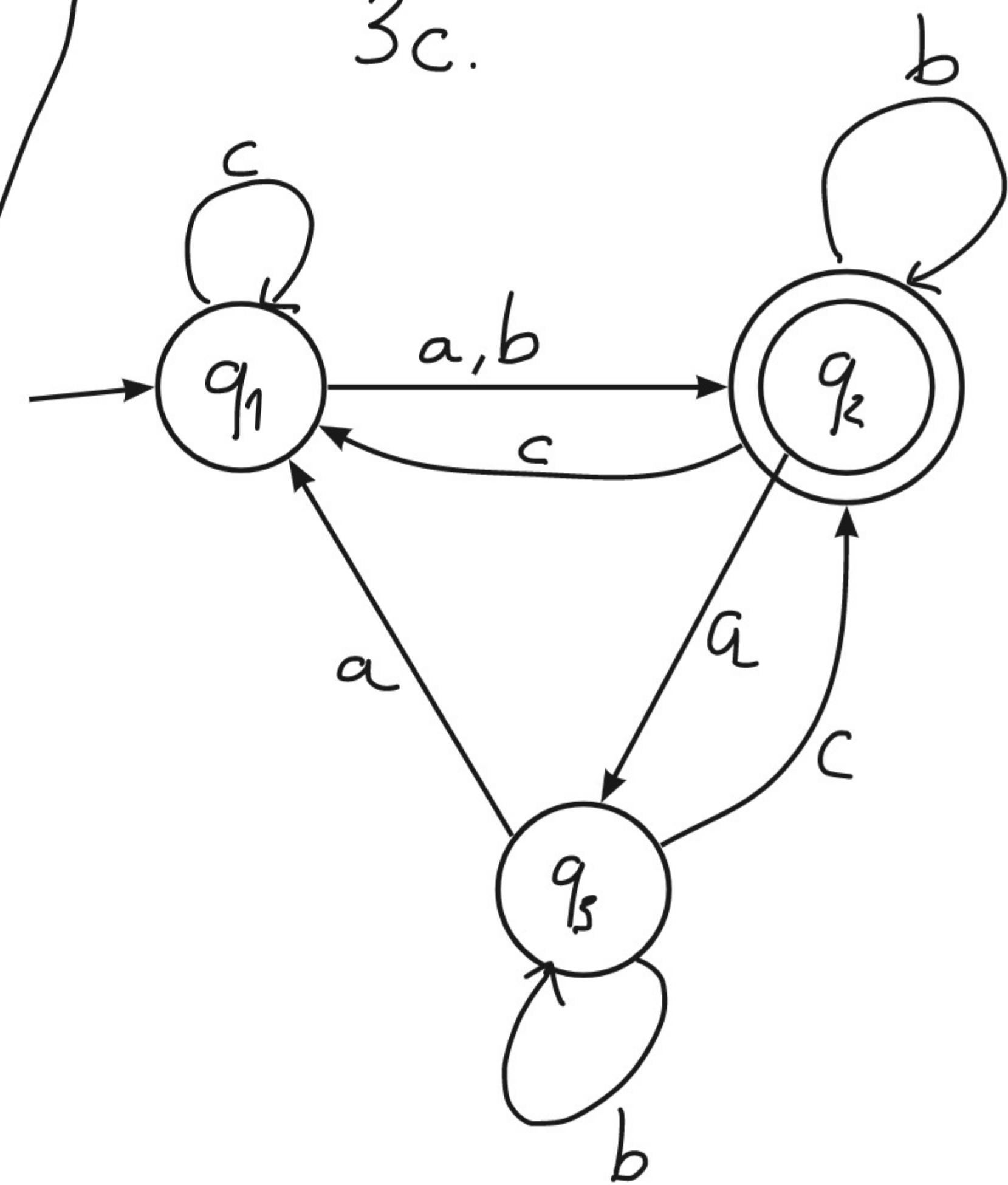
4. Quanti DFA esistono con m stati, su un alfabeto di cardinalità n ?
miro

Soluzioni

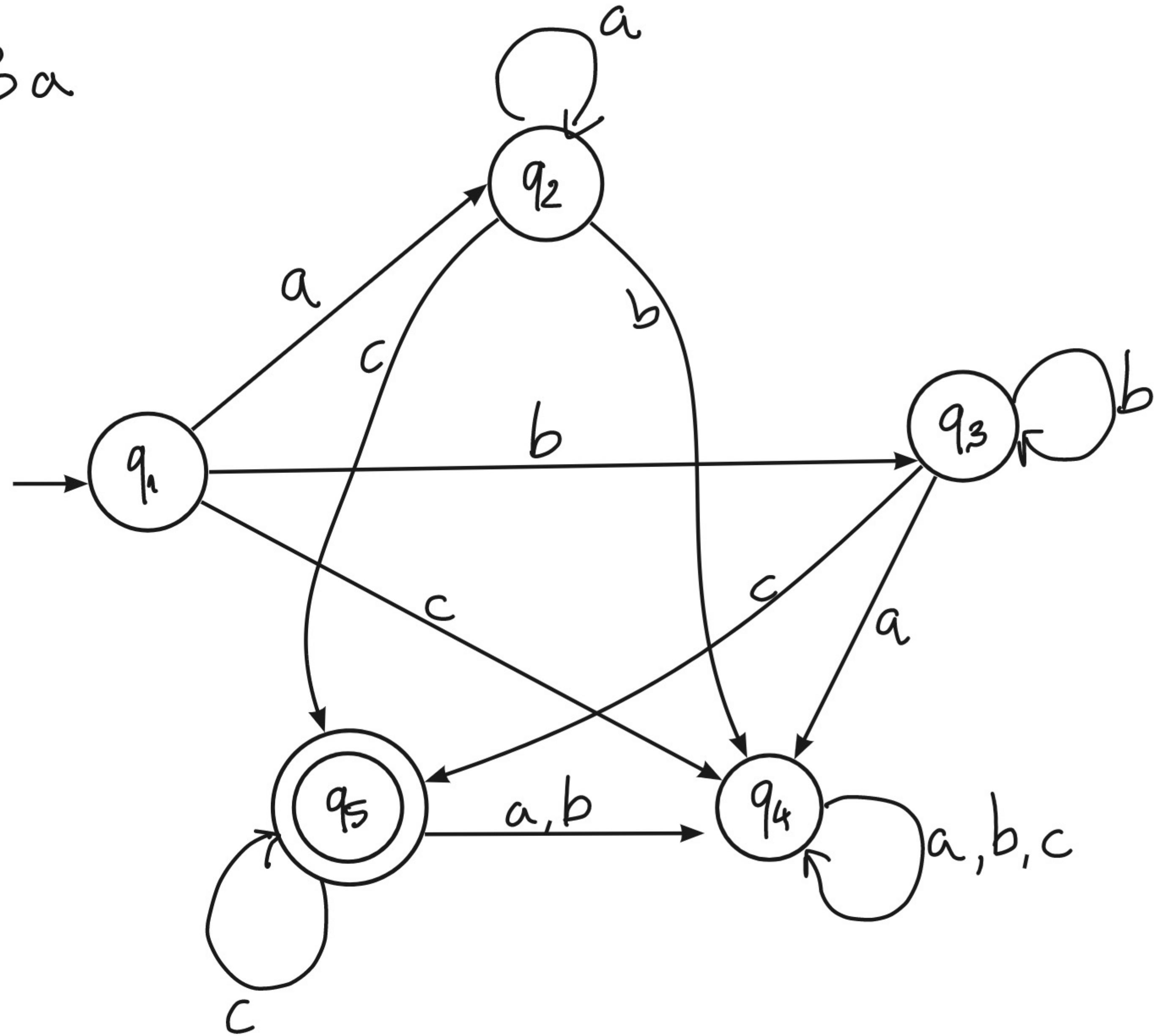
1b.



3c.

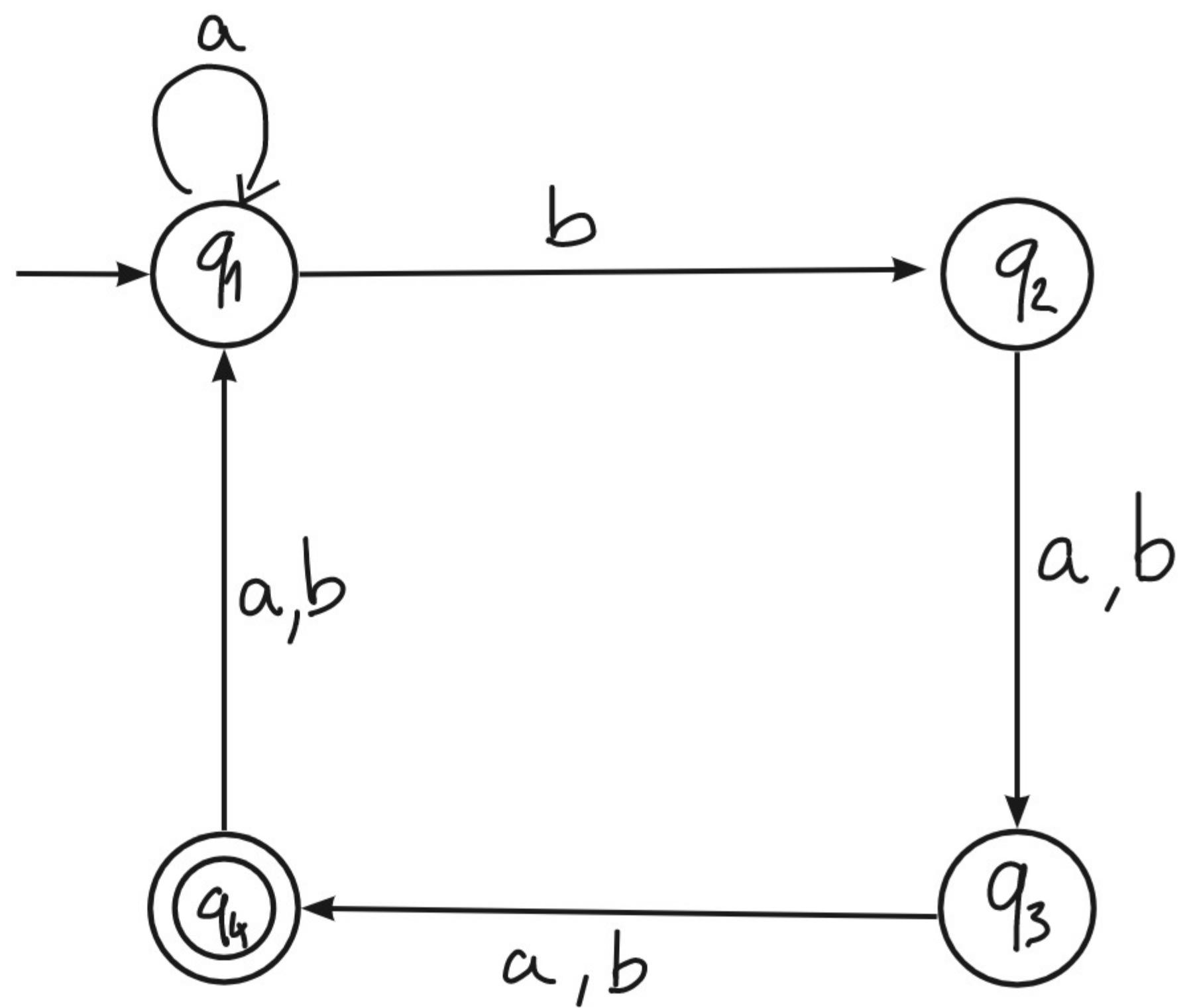


Ese. 1. 3a



$$L = \{a^n c^m \mid n, m \geq 0\} \cup \\ \cup \{b^n c^m \mid n, m \geq 0\}$$

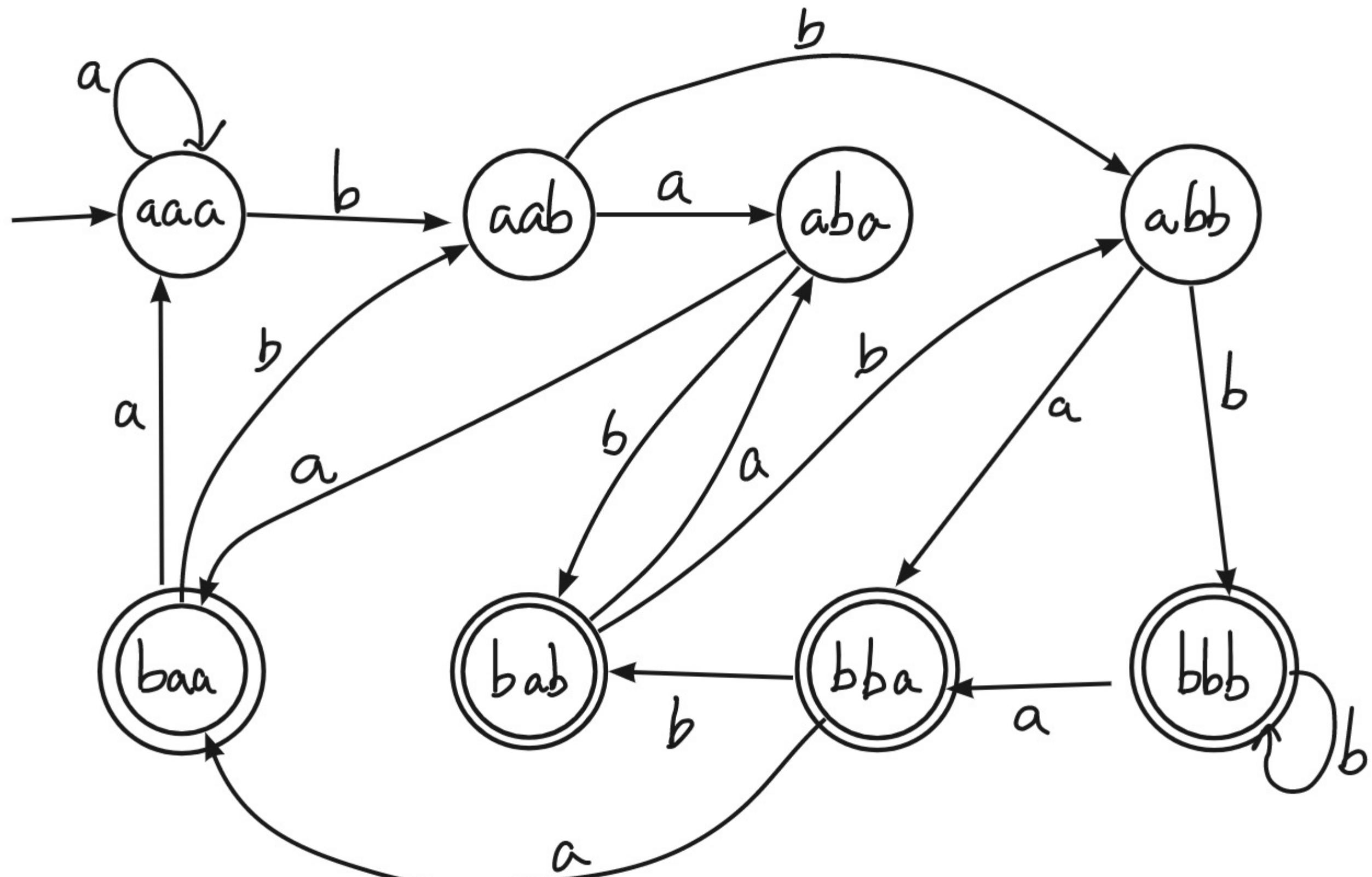
E.D. 1g.



$$L(m) \subseteq \{a_1 \dots b a_{m-1} a_m \mid a_i \in \{a, b\} \text{ per } i=1, \dots, m\}$$

ma inclusione opposta non vale: ad es.

$w = bbbb$ non accettata
↑



Soluzione corretta

Abbiamo introdotto informalmente i DFA come particolari MdT.

Più precisamente, possiamo dim. che

Teorema Ogni linguaggio regolare è ricorsivamente enumerabile.

Dim. Sia $L \subseteq A^*$ regolare e sia $M = (Q, A, \delta, q_1, F)$ un DFA che lo accetta.

Allora la macchina di Turing $\hat{M} = (Q \cup \{q_0\}, A, \hat{\delta}, q_0)$, dove $\hat{\delta}$ è descritta dalla quadruple (q_0, \sqcup, R, q_1) , $(q, a, R, \delta(q, a))$ per ogni $q \in Q$ e $a \in A$, e (q, \sqcup, R, q) per ogni $q \in Q \setminus F$, accetta L , CVD.

Teorema $L \subseteq A^*$ regolari $\Rightarrow \bar{L} := A^* \setminus L$ regolari.

Dim. Se (Q, A, δ, q_1, F) accetta L , $(Q, A, \delta, q_1, Q \setminus F)$ accetta \bar{L} , CVD.

Corollario L regolare $\Rightarrow L$ ricorsivo (predicato di appartenenza a L è calcolabile).

Dim. Segue dal fatto che L ricorsivo $\Leftrightarrow L, \bar{L}$ r.e., CVD.

Soluzione es. 4

Fissato Q e A , un DFA è individuato da:

- $\delta: Q \times A \rightarrow Q$ (m^{mn} scelte)

- q_1 (m scelte)

- F (2^m scelte)

Risposta $m^{mn+1} 2^m$.

Automi finiti non deterministici

Non determinismo = possibilità di scelta tra più stati (anche nessuno) ad ogni passo.

Un automa finito non det. (NDFA o NFA) è una quintupla

$M = (Q, A, \delta, q_1, F)$, con Q insieme finito di stati ecc.

ma $\delta: Q \times A \rightarrow P(Q)$

Linguaggio accettato: parole che etichettano cammini "di accettazione", cioè che iniziano nello stato iniziale e terminano in uno stato terminale.

Formalmente, definiamo $\delta^*: Q \times A^* \rightarrow P(Q)$ in questo modo:

- $\forall q \in Q, \quad \delta^*(q, \varepsilon) = \{q\}$
- $\forall q \in Q, \forall u \in A^*, \forall a \in A, \quad \delta^*(q, ua) = \bigcup_{q' \in \delta^*(q, u)} \delta(q', a)$

Il linguaggio accettato $L(\mathcal{M})$ è allora l'insieme

$$\{w \in A^* \mid \delta^*(q_0, w) \cap F \neq \emptyset\}$$

NB: Un DFA $\mathcal{M} = (Q, A, \delta, q_0, F)$ può essere interpretato come l'NDFA

$\mathcal{M}' = (Q, A, \delta', q_0, F)$, in cui $\delta'(q, a) = \{\delta(q, a)\}$ per ogni $q \in Q$ e $a \in A$.

Teorema $L \subseteq A^*$ regolare $\Leftrightarrow L$ accettato da un NDFA su A .

Dim. " \Rightarrow " Come sopra (e il linguaggio accettato coincide)

" \leftarrow "

Sia $L = L(M)$ con $M = (Q, A, \delta, q_1, F)$ NDFA. Allora sia \hat{M} il DFA

$(\hat{Q}, A, \hat{\delta}, \{q_1\}, \hat{F})$, dove $\hat{Q} = \mathcal{P}(Q)$,

$$\hat{F} = \left\{ Q' \subseteq Q \mid Q' \cap F \neq \emptyset \right\}$$

$\hat{\delta}(Q', a) = \bigcup_{q \in Q'} \delta(q, a)$ per ogni $Q' \subseteq Q$ e $a \in A$.

Ora $u \in L(\hat{M}) \Leftrightarrow \hat{\delta}^*(\{q_1\}, u) \in \hat{F} \Leftrightarrow \hat{\delta}^*(\{q_1\}, u) \cap F \neq \emptyset$

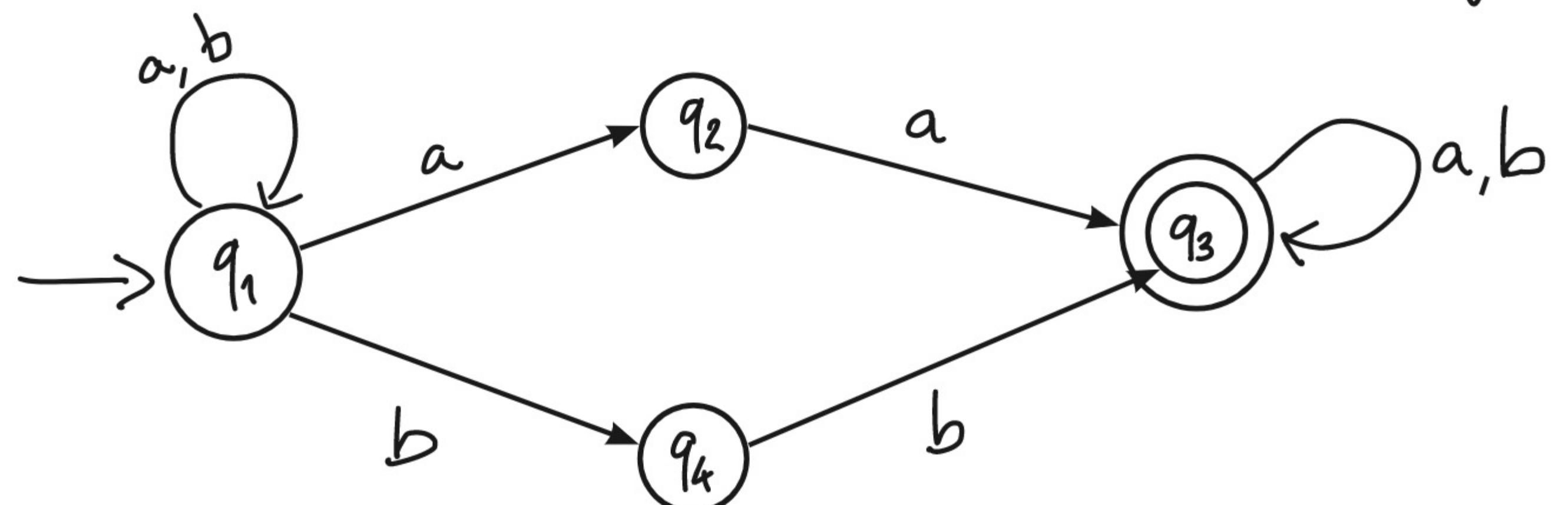
$\Leftrightarrow \bigcup_{x \in \{q_1\}} \delta^*(x, u) \cap F \neq \emptyset \Leftrightarrow \delta^*(q_1, u) \cap F \neq \emptyset$

$\Leftrightarrow u \in L(M)$, CVD.

Esempio (sintesi di NDFA)

Sia $L \subseteq A^*$, con $A = \{a, b\}$, il linguaggio delle parole che contengono $aa \circ bb$.

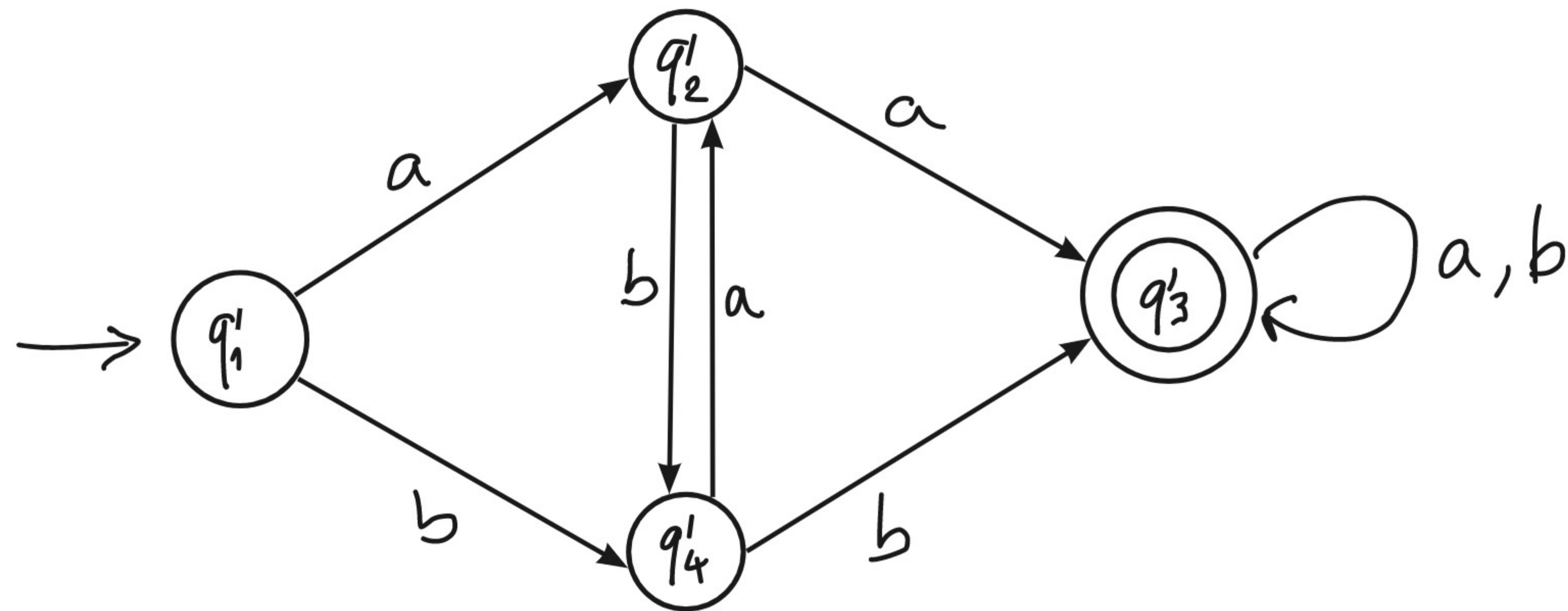
Allora un NDFA che accetta L è il seguente:



$M = (Q, A, \delta, q_1, \{q_3\})$, dove $Q = \{q_1, q_2, q_3, q_4\}$ e δ definita come in tabella (a destra)

	a	b
q_1	$\{q_1, q_2\}$	$\{q_1, q_4\}$
q_2	$\{q_3\}$	\emptyset
q_3	$\{q_3\}$	$\{q_3\}$
q_4	\emptyset	$\{q_3\}$

Osserviamo che il teorema assicura l'esistenza di un DFA che accetta L , con al più $2^4 = 16$ stati.

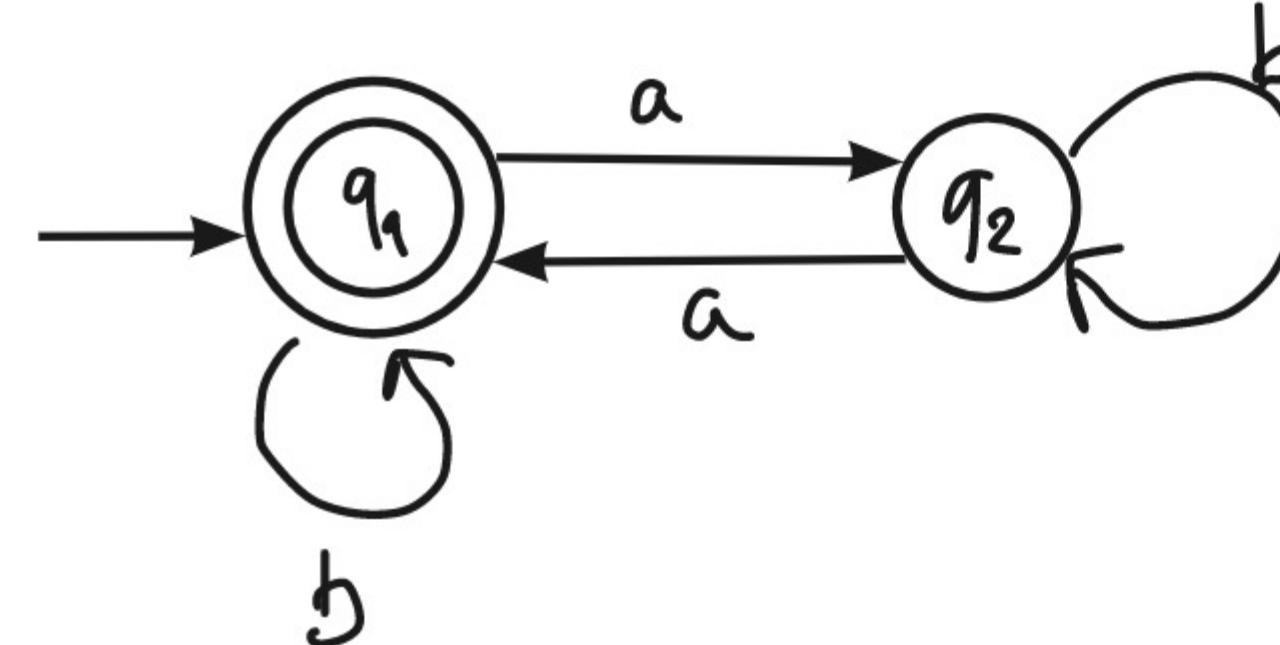


Un DFA $\overset{m}{\sim}$ si dice nonrestating se $M = (Q, A, \delta, q_1, F)$ e $q_1 \notin \delta(Q \times A)$, cioè non è possibile tornare nello stato iniziale.

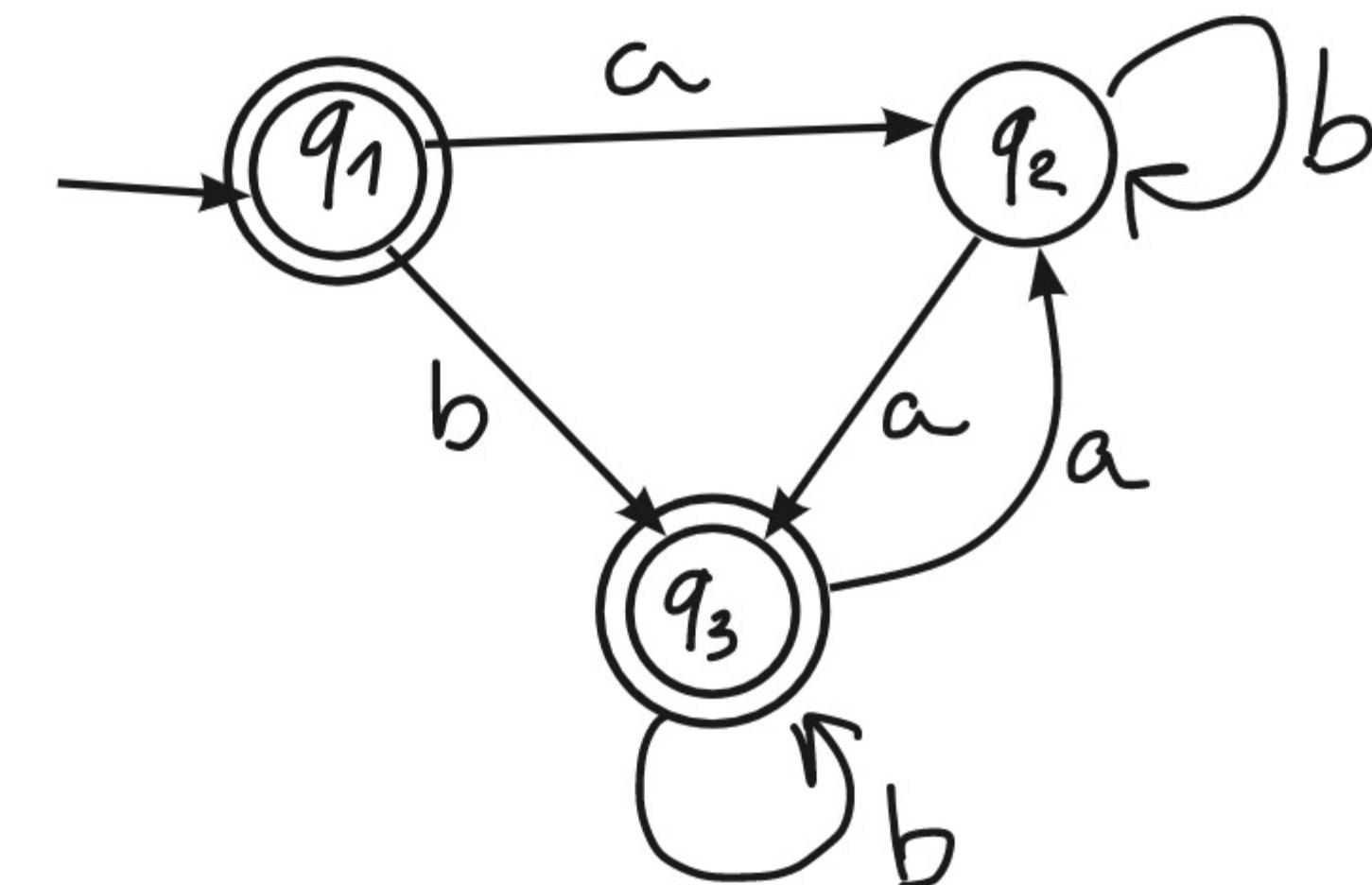
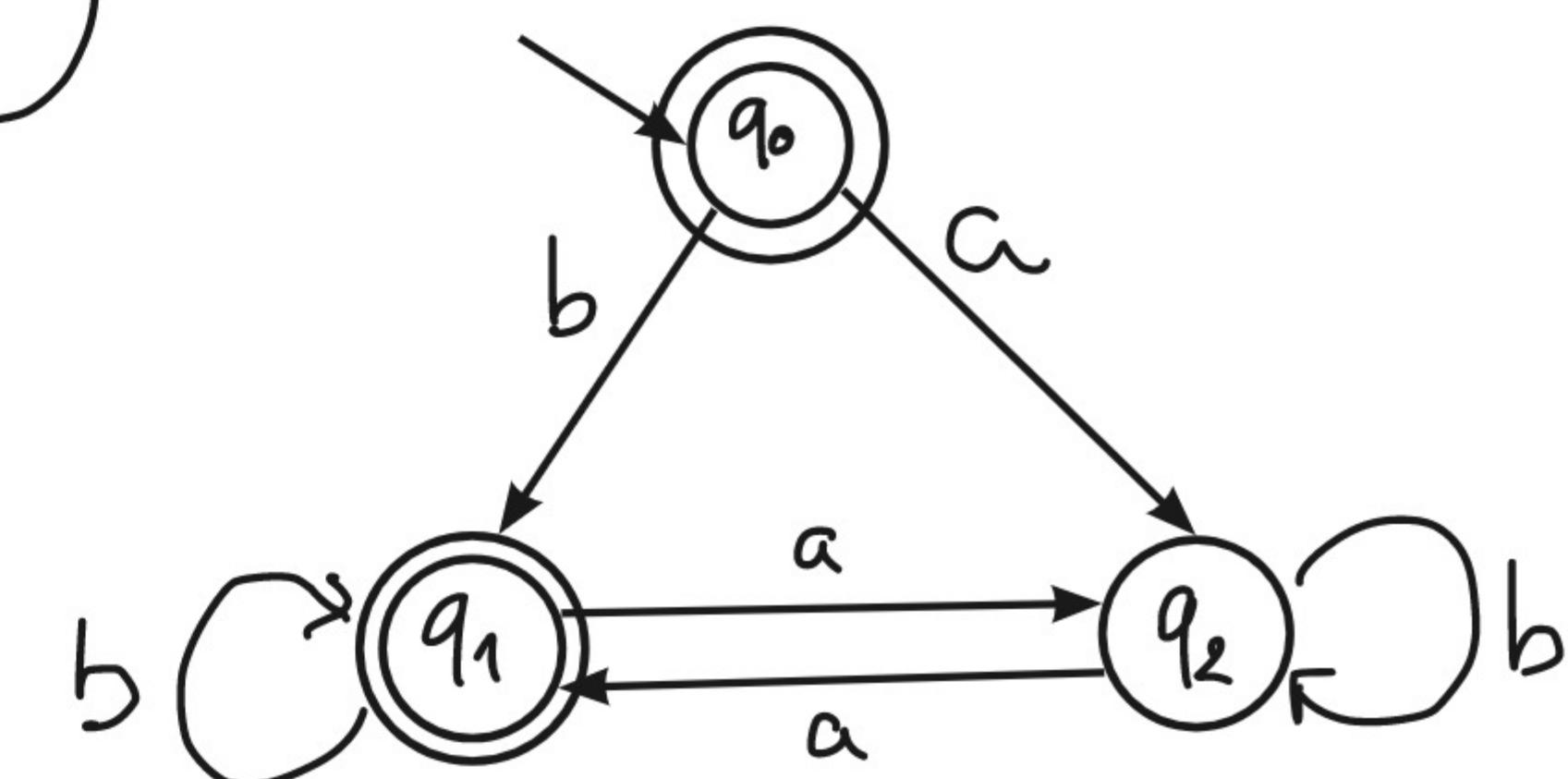
Dato qualsiasi DFA $M = (Q, A, \delta, q_1, F)$, ne esiste uno non restarting equivalente, cioè che accetta lo stesso linguaggio.

Ad esempio, $(Q \cup \{q_0\}, A, \delta', q_0, F')$ dove $\delta'(q, a) = \delta(q, a)$ per $q \in Q$ e $a \in A$ e $\delta'(q_0, a) = \delta'(q_1, a)$ per ogni $a \in A$, e $F' = \begin{cases} F & \text{se } q_1 \notin F \\ F \cup \{q_0\} & \text{altrimenti} \end{cases}$.

Ad esempio:



(accetta le parole con un numero pari di occorrenze di a)



miro

Teorema L, L' regolari ($\subseteq A^*$) $\Rightarrow L \cup L'$ regolare

Dim. Siamo $M = (Q, A, \delta, q_1, F)$ e $M' = (Q', A, \delta', q'_1, F')$ dei DFA nonrestazting che accettano rispettivamente L e L' , con $Q \cap Q' = \emptyset$.

Definiamo un NDFA $\hat{M} = (Q \cup Q' \cup \{q_0\} \setminus \{q_1, q'_1\}, A, \hat{\delta}, q_0, \hat{F})$, dove

$$\hat{F} = \begin{cases} F \cup F', & \text{se } q_1 \notin F \text{ e } q'_1 \notin F' \\ F \cup F' \cup \{q_0\} \setminus \{q_1, q'_1\}, & \text{altrimenti} \end{cases}$$

$$\text{e } \hat{\delta}(q, a) = \begin{cases} \{\delta(q, a)\} & \text{se } q \in Q \setminus \{q_1\} \text{ e } \delta(q, a) \neq q_1 \\ \{\delta'(q, a)\} & \text{se } q \in Q' \setminus \{q'_1\} \text{ e } \delta(q, a) \neq q'_1 \\ \{q_0\} & \text{se } q \in Q \cup Q' \setminus \{q_1, q'_1\} \text{ e } \delta(q, a) \in \{q_1, q'_1\} \text{ oppure se } q = q_0, \delta(q, a) = q_1 \text{ e } \delta'(q, a) = q'_1 \\ \{\delta(q_1, a), \delta'(q'_1, a)\} & \text{se } q = q_0 \text{ e } \delta(q_1, a) \neq q_1, \delta'(q'_1, a) \neq q'_1 \\ \{q_0, \delta'(q'_1, a)\} & \text{se } q = q_0 \text{ e } \delta(q_1, a) = q_1 \text{ e } \delta'(q'_1, a) \neq q'_1 \\ \{\delta(q_1, a), q_0\} & \text{se } q = q_0 \text{ e } \delta'(q'_1, a) = q'_1 \text{ e } \delta(q_1, a) \neq q_1 \end{cases}$$

\hat{M} accetta $L \cup L'$, CVD.

E_Δ - pag. 246 n. 1

a.

f_1	a	b	c
q_1	$\{q_1, q_2, q_3\}$	\emptyset	\emptyset
q_2	\emptyset	$\{q_4\}$	\emptyset
q_3	\emptyset	\emptyset	$\{q_4\}$
q_4	\emptyset	\emptyset	\emptyset

$F_1 = \{q_4\}$

b. $S_2 = S_1$, $F_2 = \{q_1, q_2, q_3\}$

c.

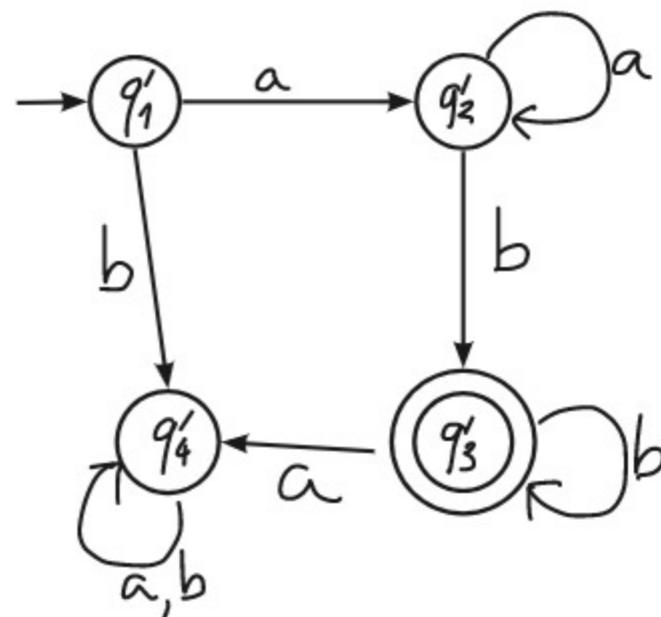
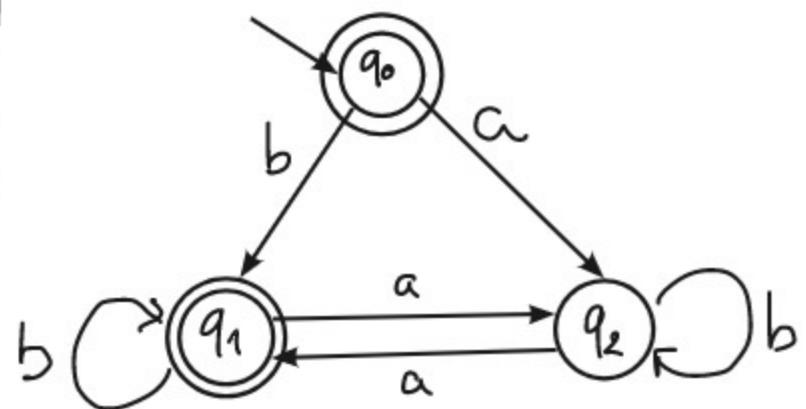
	a	b
q_1	$\{q_2\}$	\emptyset
q_2	\emptyset	$\{q_1, q_3\}$
q_3	$\{q_1, q_3\}$	\emptyset

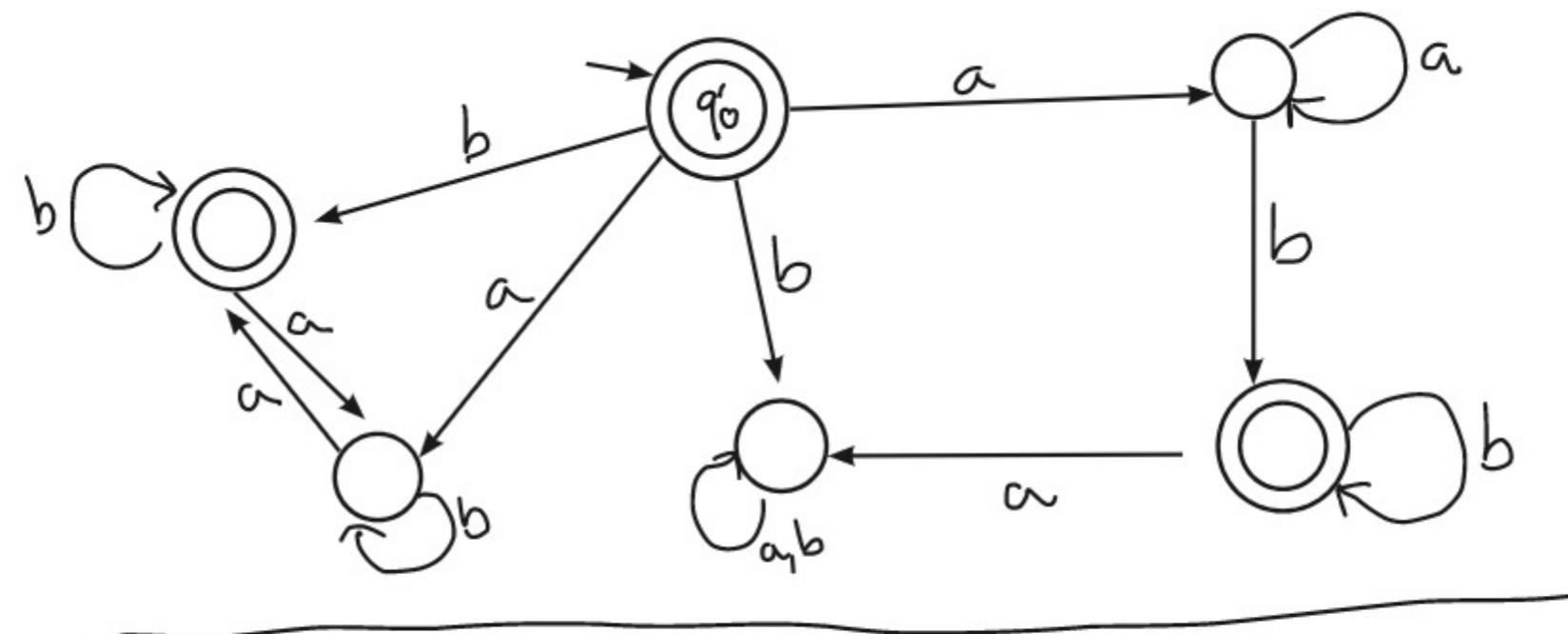
$F_3 = \{q_2\}$.

Correzione: funzione di transizione di \hat{M} che accetta $L \cup L'$

$$\hat{\delta}(q, a) = \begin{cases} \{\delta(q, a)\} & \text{se } q \in Q \setminus \{q_1\} \\ \{\delta'(q, a)\} & \text{se } q \in Q' \setminus \{q'_1\} \\ \{\delta(q_1, a), \delta'(q'_1, a)\} & \text{se } q = q_0 \end{cases}$$

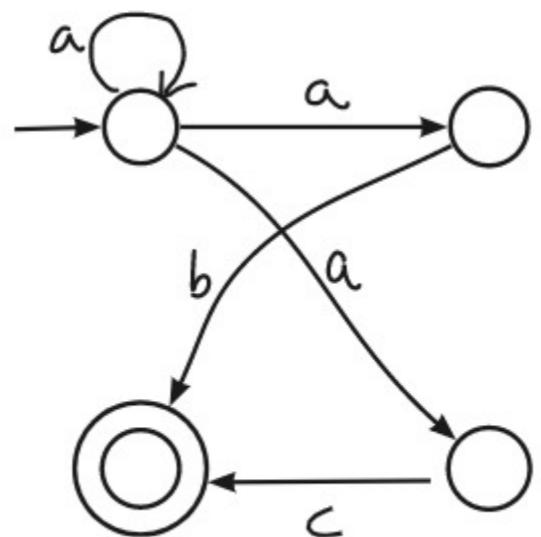
Esempio: $L = \{w \in \{a, b\}^* \mid w \text{ contiene un numero pari di } a\}$
 $L' = \{a^n b^m \mid n, m > 0\}$





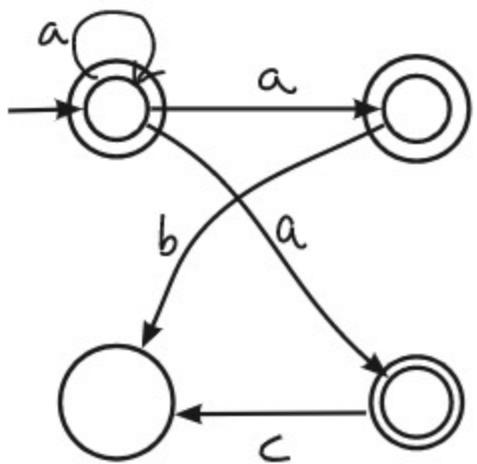
NDFA che accetta $L \cup L'$.

Soluzione esercizi, n. 1a pag. 246



$$L(M_1) = \{a^n b \mid n \geq 0\} \cup \{a^n c \mid n \geq 0\}$$

Esercizio 1b

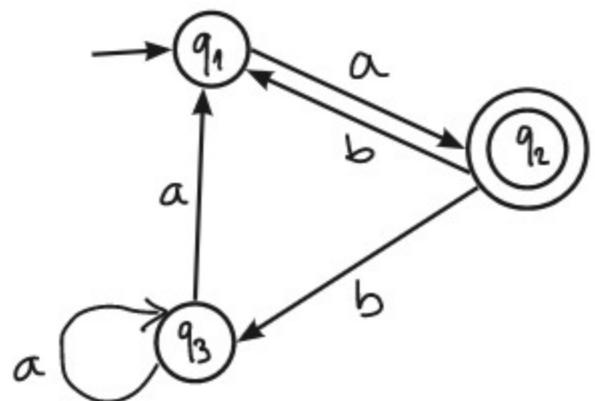


$$L(M_2) = \{a^n \mid n \geq 0\}$$

NB: $L(M_2) \neq \overline{L(M_1)}$!!

In generale, funziona solo per i DFA.

Esercizio 1c



$$\begin{aligned} L(M_3) &= \{a(ba^{n_1})(ba^{n_2}) \dots (ba^{n_k}) \mid k \geq 0, n_1, \dots, n_k \geq 0\} \\ &= \{w \in \{a, b\}^* \mid w \text{ inizia e finisce per } a, \text{ non contiene } bb \text{ e} \\ &\quad \text{non inizia per } aa\} \end{aligned}$$

Teorema $L, L' \subseteq A^*$ regolari $\Rightarrow L \cap L'$ regolare

Dim. Basta osservare che $L \cap L' = \overline{(L \cup L')} =$

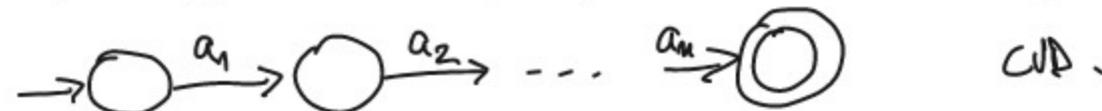
$$= A^* \setminus ((A^* \setminus L) \cup (A^* \setminus L')) , \text{ CVD.}$$

Prop. \emptyset e $\{\epsilon\}$ sono regolari

Dim Basta prendere rispettivamente un DFA con $F = \emptyset$, o un DFA nonstarting con $F = \{q_1\}$, CVD.

Prop. $\forall u \in A^*$, $\{u\}$ regolare

Dim Se $u = a_1 \dots a_m$ con $a_1, \dots, a_m \in A$, l'N DFA seguente accetta $\{u\}$:



Corollario. Tutti i linguaggi finiti sono regolari.

Dim. Se $L = \{u_1, \dots, u_n\}$, allora $\{u_1\}, \dots, \{u_n\}$ sono regolari e dunque lo è anche la loro unione, CVD.

Dati due linguaggi $L, L' \subseteq A^*$, definiamo

$$L \cdot L' = \{ww' \in A^* \mid w \in L, w' \in L'\}$$

Teorema Se L e L' sono regolari, lo è anche $L \cdot L'$.

Dim Siano $M = (Q, A, \delta, q_1, F)$ e $M' = (Q', A, \delta', q_1, F')$ DFA che accettano risp. L e L' , con $Q \cap Q' = \emptyset$

Definiamo un NDFA $\hat{M} = (\hat{Q}, A, \hat{\delta}, q_1, \hat{F})$ dove $\hat{Q} = Q \cup Q'$,

$$\hat{\delta}(q, a) = \begin{cases} \{\delta(q, a)\} & \text{se } q \in Q \setminus F \\ \{\delta(q, a), \delta'(q', a)\} & \text{se } q \in F \\ \{\delta'(q, a)\} & \text{se } q \in Q' \end{cases}$$

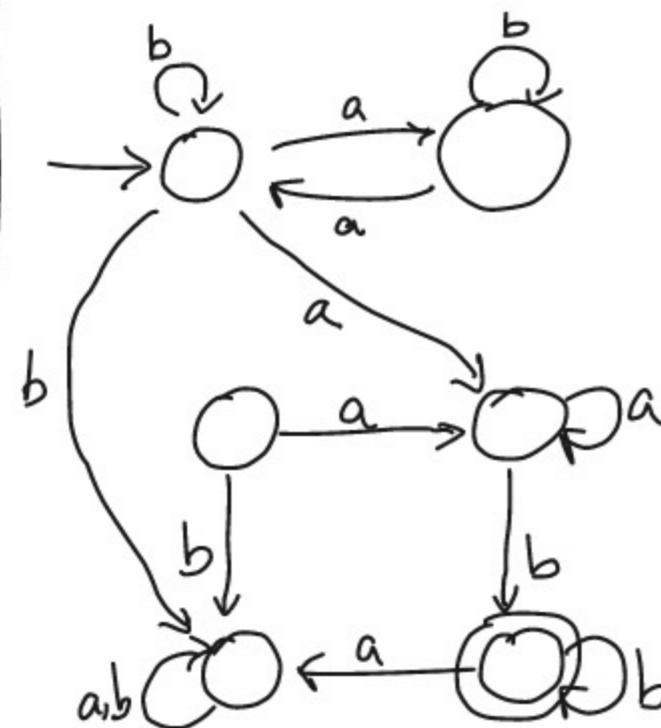
e $\hat{F} = \begin{cases} F' & \text{se } \epsilon \notin L', \text{ cioè se } q_0 \notin F' \\ F \cup F' & \text{altrimenti} \end{cases}$

Tale NDFA accetta $L \cdot L'$, cioè.

Se $L \subseteq A^*$, definiamo

$$L^* = \{w_1 \dots w_n \mid n \geq 0, w_1, \dots, w_n \in L\}.$$

Esempio $L = \{w \in A^* \mid w \text{ contiene min. 2 azi di } a\}$
 $A = \{a, b\}$ $L' = \{a^n b^m \mid n, m \geq 0\}$



Definendo $L^0 = \{\epsilon\}$, $L^{n+1} = L^n \cdot L$ per $n \geq 0$

si ha $L^* = \bigcup_{n \geq 0} L^n$

Esercizi pag. 252 n. 1, 3

1. Sia $A = \{a, b\}$, $L_1 = \{w \in A^* \mid w \text{ contiene almeno 2 occ. di } a\}$, $L_2 = \{w \in A^* \mid w \text{ contiene almeno 2 occ. di } b\}$
Trovare degli NDFA che accettino i seguenti linguaggi:

- a) $L_1 \cup L_2$
- b) $A^* - L_1$
- c) $A^* - L_2$
- d) $L_1 \cap L_2$

3. a) Siano L, L' regolari. Dim. che $L - L'$ è regolare

b) Siano L e L' tali che L è regolare, $L \cup L'$ è regolare, e $L \cap L' = \emptyset$. Dim. che L' è regolare.

Teorema $L \subseteq A^*$ regolare $\Rightarrow L^*$ regolare

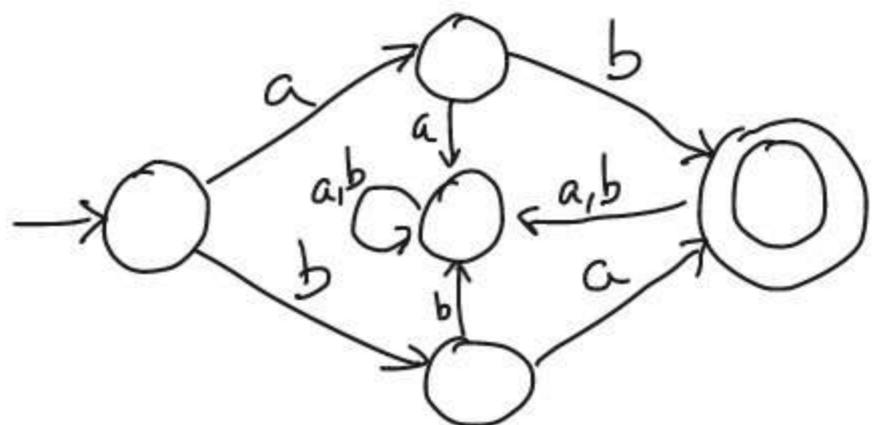
Dim. Sia $M = (Q, A, \delta, q_1, F)$ un DFA nonstarting che accetta L .

Definiamo un NDFA ricopiando su q_1 tutte le transizioni entranti in stati di F , e impostando come unico stato terminale q_1 . Formalmente,

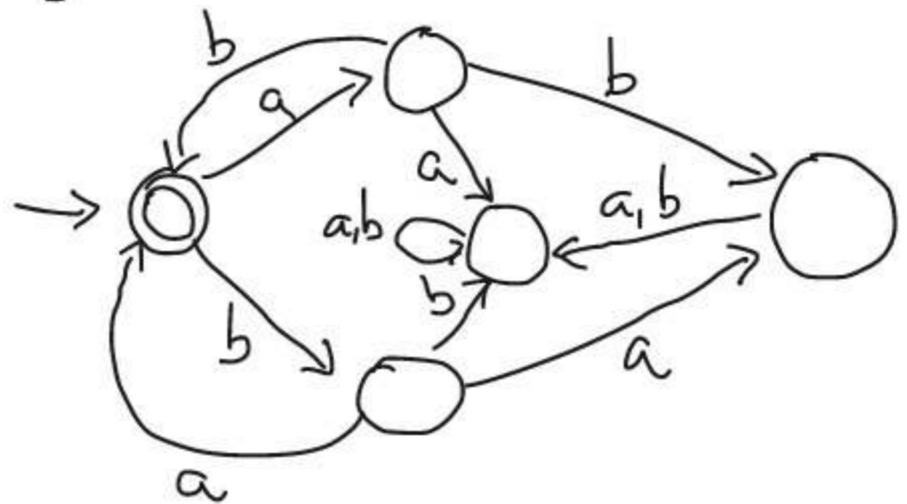
$$M' = (Q, A, \delta', q_1, \{q_1\}) \text{ con } \delta'(q, a) = \begin{cases} \{\delta(q, a)\} & \text{se } \delta(q, a) \notin F \\ \{\delta(q, a), q_1\} & \text{altrimenti} \end{cases}$$

L' NDFA M' accetta L^* , CVD.

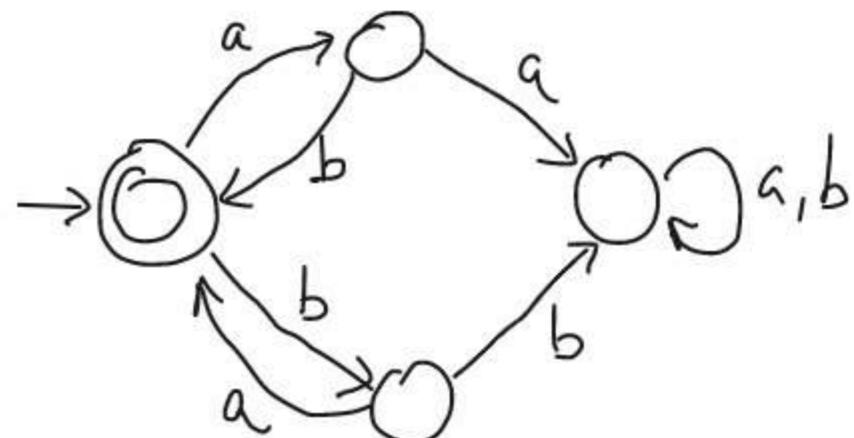
Esempio: $L = \{ab, ba\}$ accettato dal DFA con diagramma seguente



$L^* = \{ab, ba\}^*$ accettato da:



equivalente
al DFA



Teorema (Kleene). Sia A alfabeto finito.

$L \subseteq A^*$ regolare $\Leftrightarrow L$ si ottiene da linguaggi finiti applicando U , \cdot e * un numero finito di volte.

Dim. " \Leftarrow " segue dalla chiusura rispetto a U , \cdot e * .

" \Rightarrow "

Sia $M = (Q, A, \delta, q_1, F)$ un DFA che accetta L , con $Q = \{q_1, \dots, q_n\}$.

Per $1 \leq i, j \leq n$ e $0 \leq k \leq n$, definiamo

$R_{i,j}^{(k)} = \{w \in A^* \mid \delta^*(q_i, w) = q_j \text{ e } \delta^*(q_i, w^l) \in \{q_1, \dots, q_k\} \text{ per ogni prefisso proprio e non vuoto di } w\}$.

Osserviamo che $R_{i,i}^{(0)} = \{\varepsilon\} \cup \{a \in A \mid \delta(q_i, a) = q_i\}$ per $1 \leq i \leq n$.

e $R_{i,j}^{(0)} = \{a \in A \mid \delta(q_i, a) = q_j\}$ per i, j diversi e $\leq n$.

In entrambi i casi, $R_{i,j}^{(0)}$ è finito (e dunque regolare).

Per $k \geq 0$, si ha:

$$R_{i,j}^{(k+1)} = R_{i,j}^{(k)} \cup \left(R_{i,k+1}^{(k)} \cdot \left(R_{k+1,k+1}^{(k)} \right)^* \cdot R_{k+1,j}^{(k)} \right)$$

(in altre parole: un cammino da q_i a q_j che passi solo per i primi $k+1$ stati, se effettivamente passa almeno una volta da q_{k+1} , lo si può scomporre come la concatenazione di un cammino da q_i a q_{k+1} passante solo per i primi k stati, più un numero arbitrario di cammini da q_{k+1} a sé stesso passando per i primi k , seguito infine da un cammino da q_{k+1} a q_j , sempre passando per i primi k).

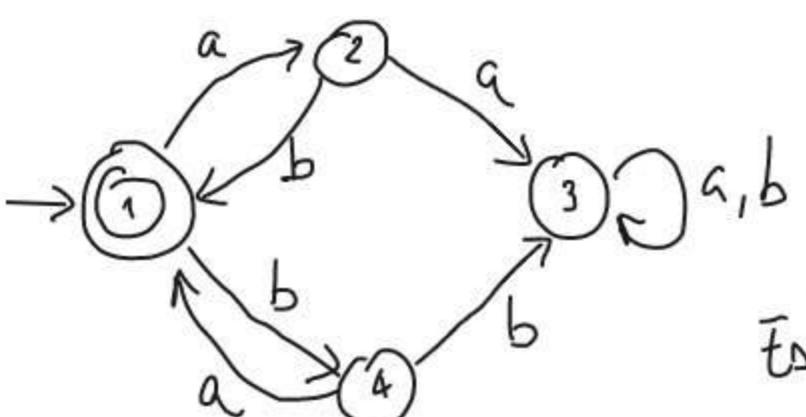
Dunque abbiamo ottenuto per induzione che gli $R_{i,j}^{(k)}$ si ottengono a partire da linguaggi finiti (il caso $k=0$) applicando un numero finito di volte \cup , · e *.

Ma inoltre $L = \bigcup_{q_j \in F} R_{1,j}^{(n)}$, CVD.

Esempio. $L = \{ab, ba\}^*$

Si ha $L = R_{1,1}^4 = R_{1,1}^{(3)} \cup \left(R_{1,4}^{(3)} \cdot (R_{4,4}^{(3)} \cdot R_{4,1}^{(3)}) \right) =$
 $= R_{1,1}^{(2)} \cup \left(R_{1,3}^{(2)} \cdot R_{3,3}^{(2)} \cdot R_{3,1}^{(2)} \right) \cup \left(R_{1,4}^{(2)} \cup \left(R_{1,3}^{(2)} \cdot R_{3,3}^{(2)*} \cdot R_{3,4}^{(2)} \right) \cdot \left(R_{4,4}^{(2)} \cup \left(R_{4,3}^{(2)} \cdot R_{3,3}^{(2)*} \cdot R_{3,1}^{(2)} \right)^* \right) \right) = \dots$

Ed. $R_{4,3}^{(1)} = R_{4,3}^{(0)} \cup \left(R_{4,1}^{(0)} \cdot R_{1,1}^{(0)*} \cdot R_{1,3}^{(0)} \right) = \{b\} \cup \left(\{a\} \cdot \{\varepsilon\}^* \cdot \emptyset \right) = \{b\}$



Dato A alfabeto finito, definiamo $\underline{A} = \{a \mid a \in A\}$ e

$$\hat{A} = \underline{A} \cup \{\emptyset, \varepsilon, \cup, \cdot, ^*, (,)\}$$

Un' espressione regolare su A è una particolare parola di \hat{A}^* , secondo le regole:

- 1) $\emptyset, \varepsilon, a$ (per ogni $a \in A$) sono espressioni regolari
- 2) Se α, β sono esp. regolari, anche $(\alpha \cup \beta)$ lo è.
- 3) Se α, β sono esp. regolari, anche $(\alpha \cdot \beta)$ lo è.
- 4) Se α è esp. regolare, anche α^* lo è.
- 5) Tutte le esp. regolari sono ottenute applicando le regole 1-4.

Ad esempio, $((\underline{a} \cdot \underline{b}) \cup (\underline{b} \cdot \underline{a}))^*$ è un' esp. reg. su $\{a, b\}$.

Semantica delle esp. regolari.

Data un'esp. reg. α , definiamo $\langle \alpha \rangle \subseteq A^*$ come segue:

$$\langle a \rangle = \{a\} \quad \text{per } a \in A$$

$$\langle \emptyset \rangle = \emptyset$$

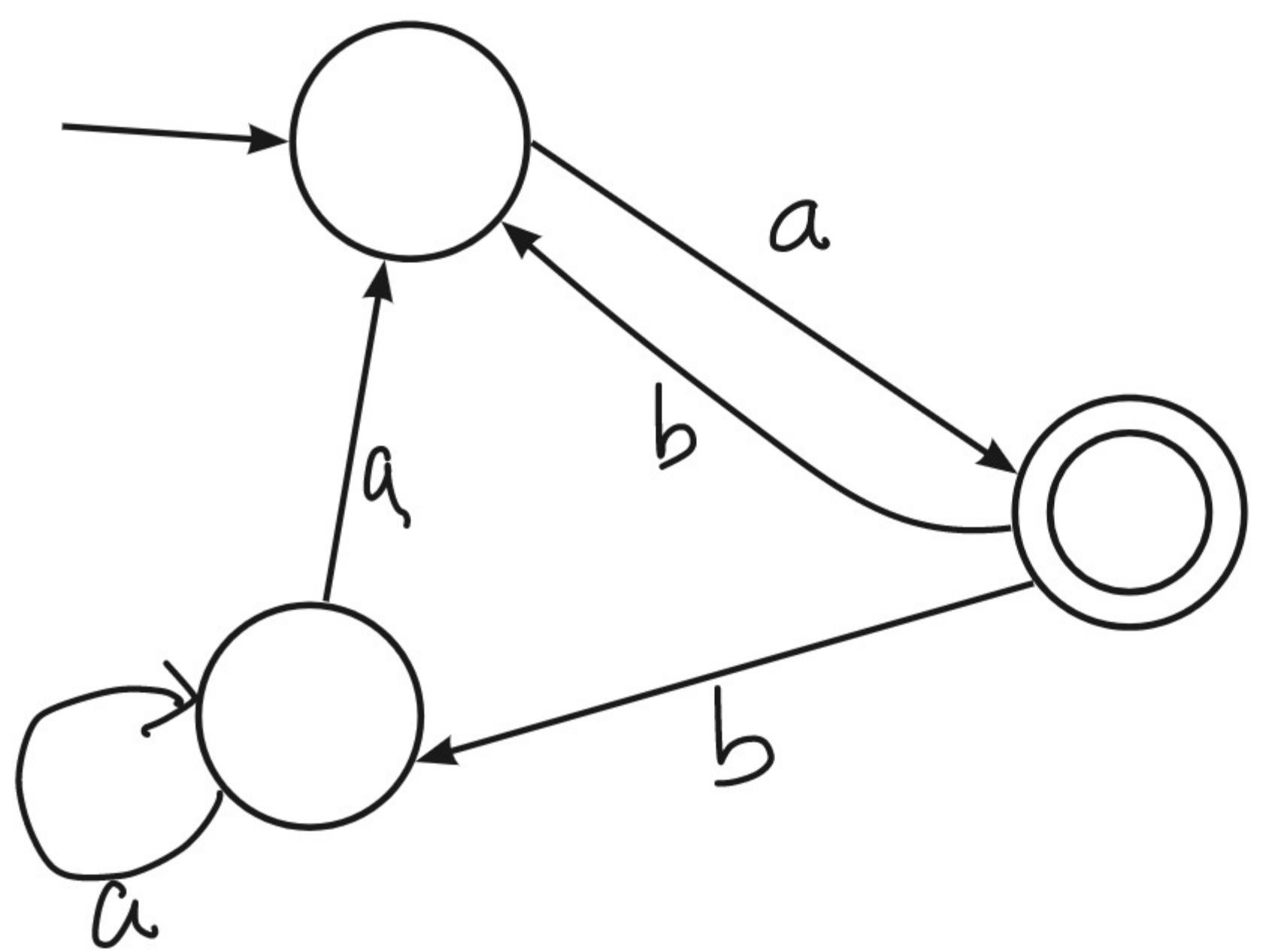
$$\langle \varepsilon \rangle = \{\varepsilon\}$$

$$\langle (\alpha \cup \beta) \rangle = \langle \alpha \rangle \cup \langle \beta \rangle$$

$$\langle (\alpha \cdot \beta) \rangle = \langle \alpha \rangle \cdot \langle \beta \rangle$$

$$\langle \alpha^* \rangle = \langle \alpha \rangle^*$$

Ad es., $\langle ((\underline{a} \cdot b) \cup (\underline{b} \cdot \underline{a}))^* \rangle = \{ab, ba\}^*$.



$$\begin{aligned}
 L(M) &= \left\{ a(ba^{n_1}) \cdots (ba^{n_k}) \mid k \geq 0, n_1, \dots, n_k \geq 0 \right\} \\
 &= \left\langle \underline{a} \cdot ((\underline{b} \cdot \underline{a}) \cup (\underline{b} \cdot (\underline{a}^* \cdot (\underline{a} \cdot \underline{a}))))^* \right\rangle \\
 &= \left\langle \underline{a} \cdot (\underline{b} \cdot (\underline{a}^* \cdot \underline{a}))^* \right\rangle
 \end{aligned}$$

Data l'associatività di \cup e \cdot . (es.: $L_1(L_2 \cdot L_3) = (L_1 \cdot L_2)L_3$) , le espr. regolari si possono semplificare omettendo le parentesi non necessarie. L'espressione risultante non rispetta le regole che definiscono le espr. regolari, ma è facile da interpretare (e trasformata in una espr. reg. propriamente detta). Ad es. un'espressione semplificata per il linguaggio precedente è

$$a(ba^*a)^*$$

Teorema

(Pumping lemma per linguaggi regolari)

Sia $M = \langle Q, A, \delta, q_0, F \rangle$ un DFA con n stati, e $x \in L(M)$ è tale che $|x| \geq n$.

Allora $\exists u, v, w \in A^*$ tali che :

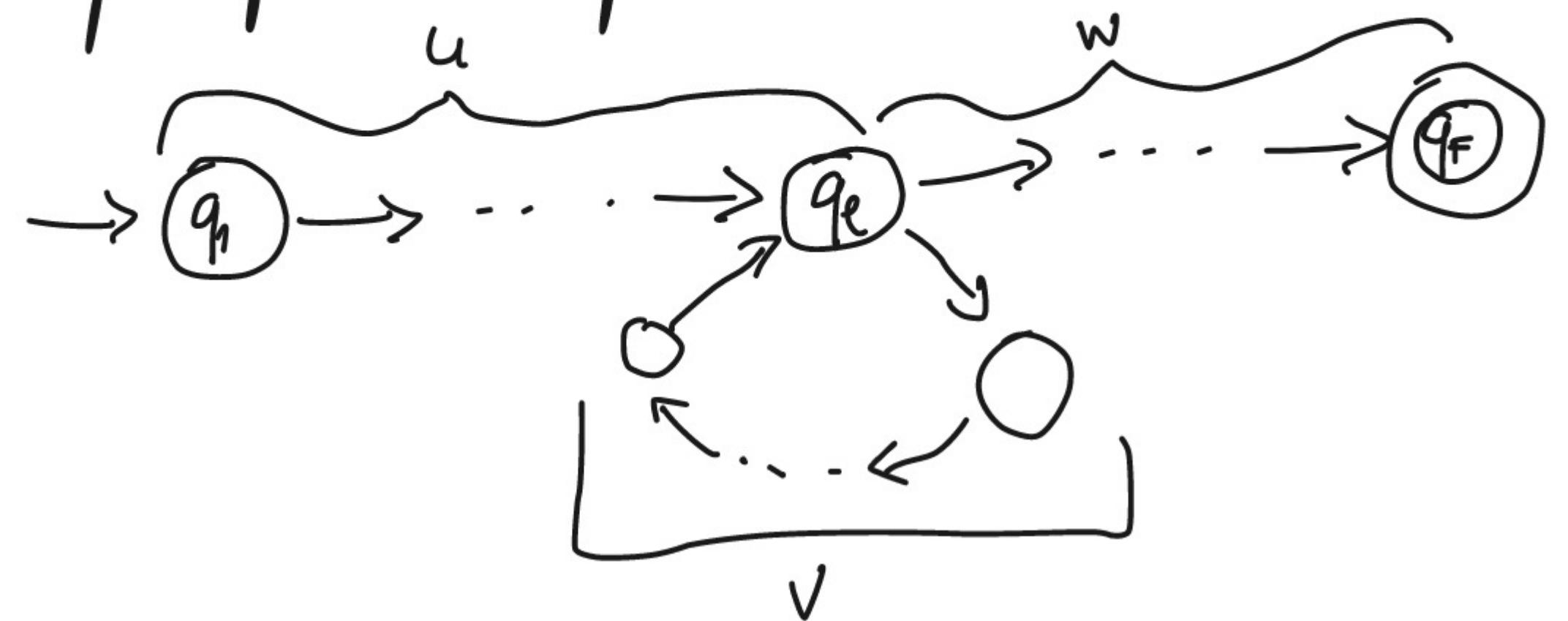
- 1) $v \neq \epsilon$
- 2) $x = uvw$
- 3) $\forall i \geq 0, uv^i w \in L(M)$.

La dimostrazione usa il cosiddetto princípio della piccioneia : Se $n+1$ oggetti sono distribuiti tra m insiemi, allora almeno uno di tali insiemi contiene almeno 2 oggetti.

Dim (del pumping lemma)

Poiché $x \in L(M)$ e $|x| \geq n$, esiste un cammino che parte dallo stato iniziale q_1 , arriva in uno stato terminale q_F , etichettato con x , che attraversa almeno $n+1$ stati.

Per il principio della piccioranza, esiste almeno uno stato q_e che è attraversato almeno 2 volte.



$$\text{Quindi : } \delta^*(q_1, u) = q_e, \quad \delta^*(q_e, v) = q_e, \quad \delta^*(q_e, w) = q_F$$

da cui è evidente che $\delta^*(q_1, uv^i w) = q_F$ per ogni $i \geq 0$, e $v \neq \epsilon$ perché q_e è visitato almeno 2 volte, c.d.

Corollario

Se M è DFA con n stati e $L(M) \neq \emptyset$, allora esiste una parola di $L(M)$ avente lunghezza $< n$.

Dimo.

Se per assurdo la parola più corta in $L(M)$ avesse lunghezza $\geq n$, per il pumping lemma potremmo scomporla in uvw , con $v \neq \epsilon$ e $uvw \in L(M)$. Ma $|uw| < |uvw|$, assurdo CVD.

Possiamo usare ciò per determinare, dati due DFA M_1 e M_2 , se $L(M_1) \subseteq L(M_2)$.
Basta costruire un DFA che accetti $L(M_1) \setminus L(M_2) = L(M_1) \cap (A^* \setminus L(M_2))$; tale automa accetta \emptyset ($\Leftrightarrow L(M_1) \subseteq L(M_2)$) se e solo se non accetta nessuna parola di lunghezza inferiore al numero dei suoi stati. Applicando questo test nelle due direzioni otteniamo un test per $L(M_1) = L(M_2)$.

Teorema Dato M DFA con n stati,

$L(M)$ è infinito $\Leftrightarrow L(M)$ contiene parole w con $n \leq |w| < 2n$.

Dim

" \Leftarrow " viene direttamente dal pumping lemma.

" \Rightarrow "

Se $L(M)$ è infinito, dato che A è finito, $L(M)$ contiene parole arbitrariamente lunghe.

Sia allora x la più corta parola di lunghezza almeno $2n$ accettata da M . Possiamo scrivere $x = yz$, con $|y|=n$ e $|z| \geq n$. Dal principio della piccioreria, come nella dim. del pumping lemma, si ottiene $y=uvw$ con $v \neq \epsilon$ e $uv^iwz \in L(M)$ per ogni $i \geq 0$.

(Formalmente: $\delta^*(q_1, y) = q_y$, ma tale cammino visita almeno due volte uno stato q_e , e allora defino u, v, w da $\delta^*(q_1, u) = q_k$, $\delta^*(q_k, v) = q_e$, $\delta^*(q_e, w) = q_y$. Poiché $\delta^*(q_y, z) \in F$, segue che $uv^iwz \in L(M)$ per ogni $i \geq 0$).

In particolare $uwz \in L(M)$, e $|uwz| \geq n$ (dato che $|z| \geq n$), $|uwz| < 2n$ dato che è più corta di x , CVD.

Il linguaggio $L = \{a^n b^n \mid n \geq 0\}$ non è regolare.

Infatti, supponiamo per assurdo che L sia accettato da un DFA con p stati. Allora la parola $x = a^p b^p$ dovrebbe scomporsi come $x = uvw$, con $v \neq \epsilon$ e $uv^i w \in L$ per ogni $i \geq 0$.

Ci sono 3 casi per v : o contiene solo occorrenze di a , o solo di b , o di entrambe.

1) Se $a^p b^p = uvw$ con $v = a^k$, $k \geq 1$, allora $uw = a^{p-k} b^p \notin L$.

2) Come 1)

3) Se $x = uvw$ con $v = a^h b^k$, $h, k \geq 1$, allora $uv^2 w = a^{p-h} (a^h b^k)^2 b^{p-k} = a^p b^{2k} a^h b^k \notin L$.

Dall'assurdo segue che L non è regolare.

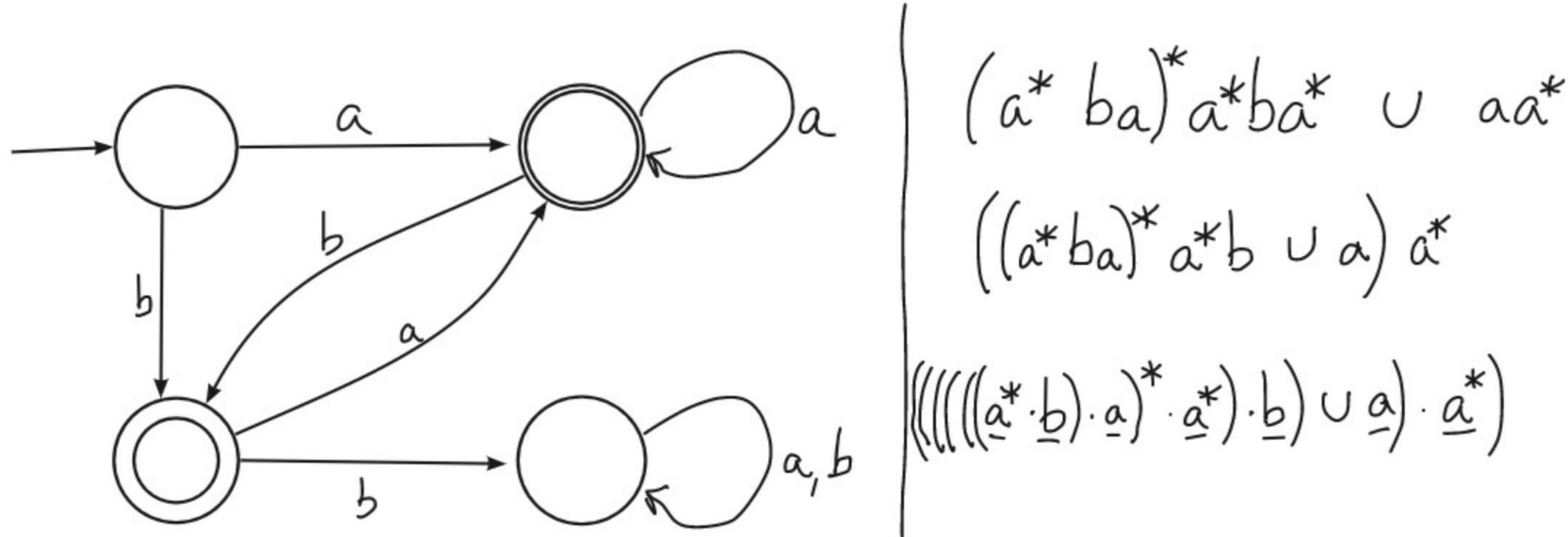
Questo ragionamento può essere applicato per dim. che vari linguaggi non sono regolari.

Ad es. $\{a^m b^n \mid m \geq n \geq 0\}$, esercizio.

Esercizi pag. 259 n. 1, 5

- 1) Per ogni linguaggio regolare menzionato in esercizi precedenti, determinare un'espressione regolare che lo rappresenti.
- 5) Sia $L = \{x \in \{a,b\}^* \mid x \neq \epsilon \text{ e } bb \text{ non è sottostringo di } x\}$
Mostrare che L è regolare costruendo un DFA che lo accetti, e trovare un'espressione regolare α tale che $L = \langle \alpha \rangle$.

Soluzione es n. 5



$L = \{a^m b^n \mid m \geq n \geq 0\}$ non è regolare.

Se p.a. fosse accettato da DFA con p stati. Allora $a^p b^p$ dovrebbe scomporre come uvw , $v \neq \epsilon$ e $uv^i w \in L$ per $i \geq 0$. I casi $v=a^i$ e $v=a^i b^j$ si trattano come per $\{a^i b^n \mid n \geq 0\}$; se invece $v=b^j$ con $j > 0$, $uv^2 w = a^p b^{p+j} \notin L$. Dunque L non è regolare.

Grammatiche context-free

Abbiamo visto vari modelli di calcolo: linguaggio S, equivalente a MdT, delle quali un caso particolare sono gli automi finiti.

Le grammatiche CF permettono invece di generare linguaggi (detti CF) descrivendone la sintassi. Nate per ling. naturali, poi usate ad es. per linguaggi di programmazione.

Una grammatica CF è una quadrupla (V, Σ, R, S) , dove:

- V e Σ sono alf. finiti (rispettivamente delle variabili e dei terminali)
- $S \in V$ è detto variabile iniziale o assiomma

- $R \subseteq V \times (V \cup \Sigma)^*$ è un insieme di regole o produzioni della forma:

$$X \rightarrow w \quad \text{con } X \in V \text{ e } w \in (V \cup \Sigma)^*.$$

Data una grammatica CF $G = (V, \Sigma, R, S)$ e due parole $u, v \in (V \cup \Sigma)^*$, diciamo che v si ottiene da u per derivazione unitaria ($u \xrightarrow{G} v$) se:

$u = u_1 X u_2$, $v = u_1 w u_2$, e R contiene la regola $X \rightarrow w$.

Diciamo che da u deriva v ($u \xrightarrow{G}^* v$) se esistono $K \geq 1$ parole $u_1, \dots, u_K \in (V \cup \Sigma)^*$ tali che $u = u_1$, $v = u_K$ e $u_1 \xrightarrow{G} u_2 \xrightarrow{G} \dots \xrightarrow{G} u_K$.

NB: $K=1$ corrisponde a $u=v$.

Il linguaggio generato $L(G)$ è : $L(G) = \{w \in \Sigma^* \mid S \xrightarrow{G}^* w\}$.

Esempio.

$L = \{a^n b^n \mid n \geq 0\}$ è context-free. Infatti $G = (V, \Sigma, R, S)$ genera L , con:

$$V = \{S\}, \quad \Sigma = \{a, b\}, \quad R = \{S \rightarrow aSb, \quad S \rightarrow \epsilon\}.$$

Una derivazione per $aabb$ è:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb.$$

Per specificare una grammatica CF, basta elencarne le regole, assumendo come var. iniziale la prima menzionata, e supponendo che variabili e terminali siano facilmente distinguibili (es. maiuscole e minuscole). Quando più regole corrispondono a una var. X , le si separa con un " | ". Ad es., la grammatica precedente è individuata dalle regole

$$S \rightarrow aSb \mid \epsilon$$

Alt_{2o} es. di distinzione tra var. e terminali: la grammatica deve specificare la sintassi di un linguaggio di programmaz. potrebbe avere regole del tipo

$\langle \text{for_cycle} \rangle \rightarrow \text{for } \langle \text{expr} \rangle \text{ in } \dots$

$\langle \text{expr} \rangle \rightarrow \dots$

Consideriamo la grammatica avente come regole

$E \rightarrow E+T \mid T$

$T \rightarrow T \cdot F \mid F$

$F \rightarrow (E) \mid a$

e come terminali $\Sigma = \{a, +, \cdot, (,)\}$.

Ad es. La stringa $a+a \cdot a$ appartiene al linguaggio generato:

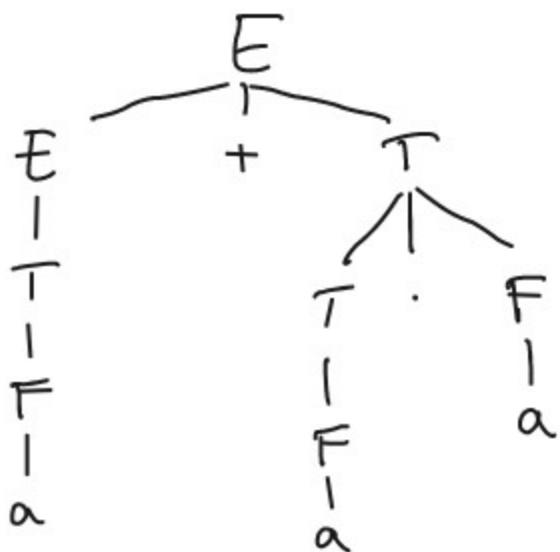
$E \Rightarrow E+T \Rightarrow T+T \Rightarrow F+F \Rightarrow a+T \Rightarrow a+T \cdot F$

$\Rightarrow a+F \cdot F \Rightarrow a+a \cdot F \Rightarrow a+a \cdot a$.

Albero di derivazione

Una derivazione si può rappresentare mediante un albero: la radice è la var. iniziale, e l'applicazione di una regola $X \rightarrow a_1 \dots a_K$ si indica aggiungendo K figli (etichettati a_1, \dots, a_K) al modo che rappresenta l'accorrenza di X da sostituire. (NB: per $X \rightarrow \epsilon$ aggiungiamo 1 figlio, con etichetta ϵ)

E., $E \Rightarrow E+T \Rightarrow T+T \Rightarrow F+T \Rightarrow a+T \Rightarrow a+T \cdot F \Rightarrow a+F \cdot F \Rightarrow a+a \cdot F \Rightarrow a+a \cdot a$.



NB: è albero anche per
 $E \Rightarrow E+T \Rightarrow E+T \cdot F \Rightarrow E+T \cdot a \Rightarrow T+T \cdot a$
 $\Rightarrow T+F \cdot a \Rightarrow F+F \cdot a \Rightarrow F+a \cdot a \Rightarrow a+a \cdot a$.

Nota: La grammatica individuata dalle regole

$$E \rightarrow E+E \mid E \cdot E \mid (E) \mid a$$

genera lo stesso linguaggio (ad es. $(a+a) \cdot a$ ne è un elemento)

ma in maniera ambigua: ad es. per $a+a \cdot a$ ci sono più derivazioni essenzialmente distinte

$$E \Rightarrow E \cdot E \Rightarrow E+E \cdot E \Rightarrow \dots \Rightarrow a+a \cdot a$$

$$E \Rightarrow E+E \Rightarrow E+E \cdot E \Rightarrow \dots \Rightarrow a+a \cdot a$$

(gli alberi corrispondenti sono distinti!)

Esempio pag 162 Sipser n. 2.1 e 2.4

n.1: Dare alberi di derivazione (rispetto alla grammatica precedente, con E, T, F) per le seguenti stringhe:

a , $a+a$, $a+a+a$, $((a))$

n. 4 : Trovare grammatiche CF da generare i seguenti linguaggi su $\{0, 1\}$:

a) $\{w \mid w \text{ contiene almeno tre } 1\}$

b) $\{w \mid w \text{ inizia e finisce con lo stesso simbolo}\}$

c) $\{w \mid |w| \text{ è dispari}\}$

d) $\{w \mid |w| \text{ è dispari e il simbolo centrale è } 0\}$

e) $\{w \mid w \text{ è palindroma}\}$

f) \emptyset .

$$E \rightarrow E+E \mid E \cdot E \mid (E) \mid a$$

$$a+a \cdot a \in L(G)$$

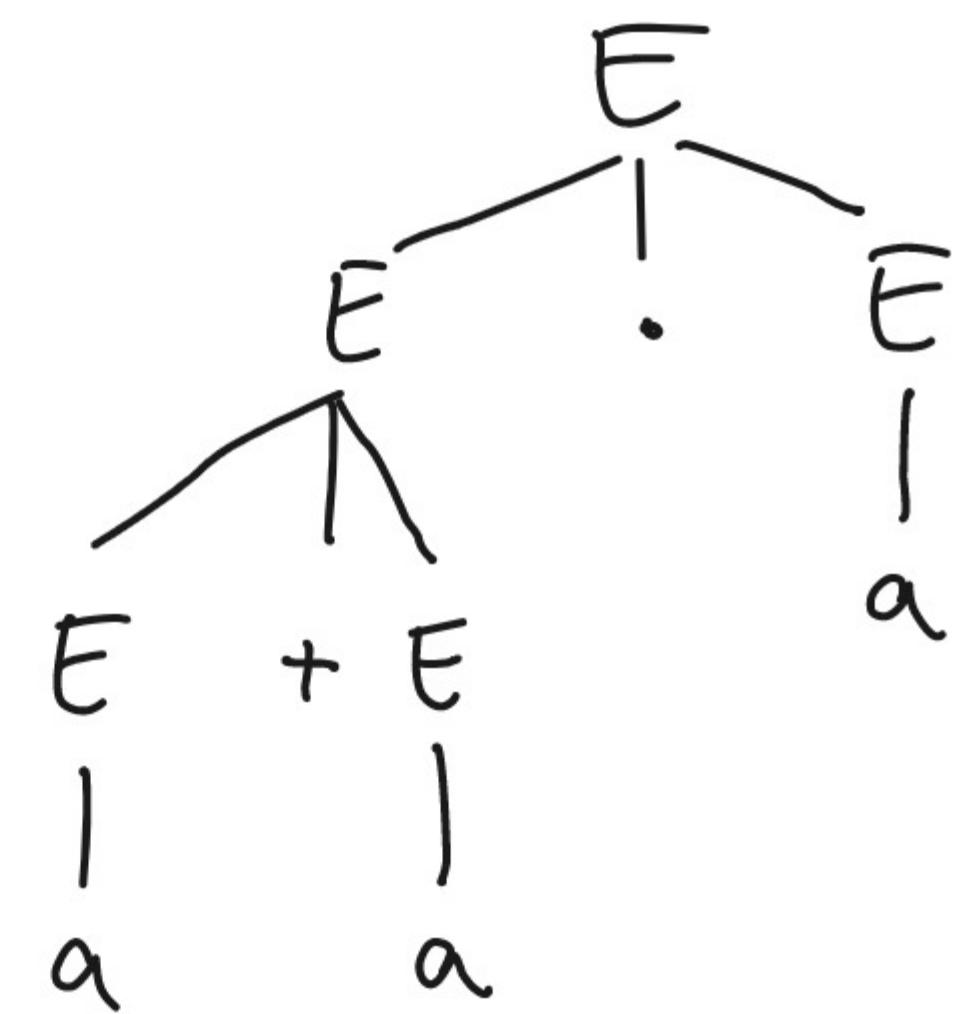
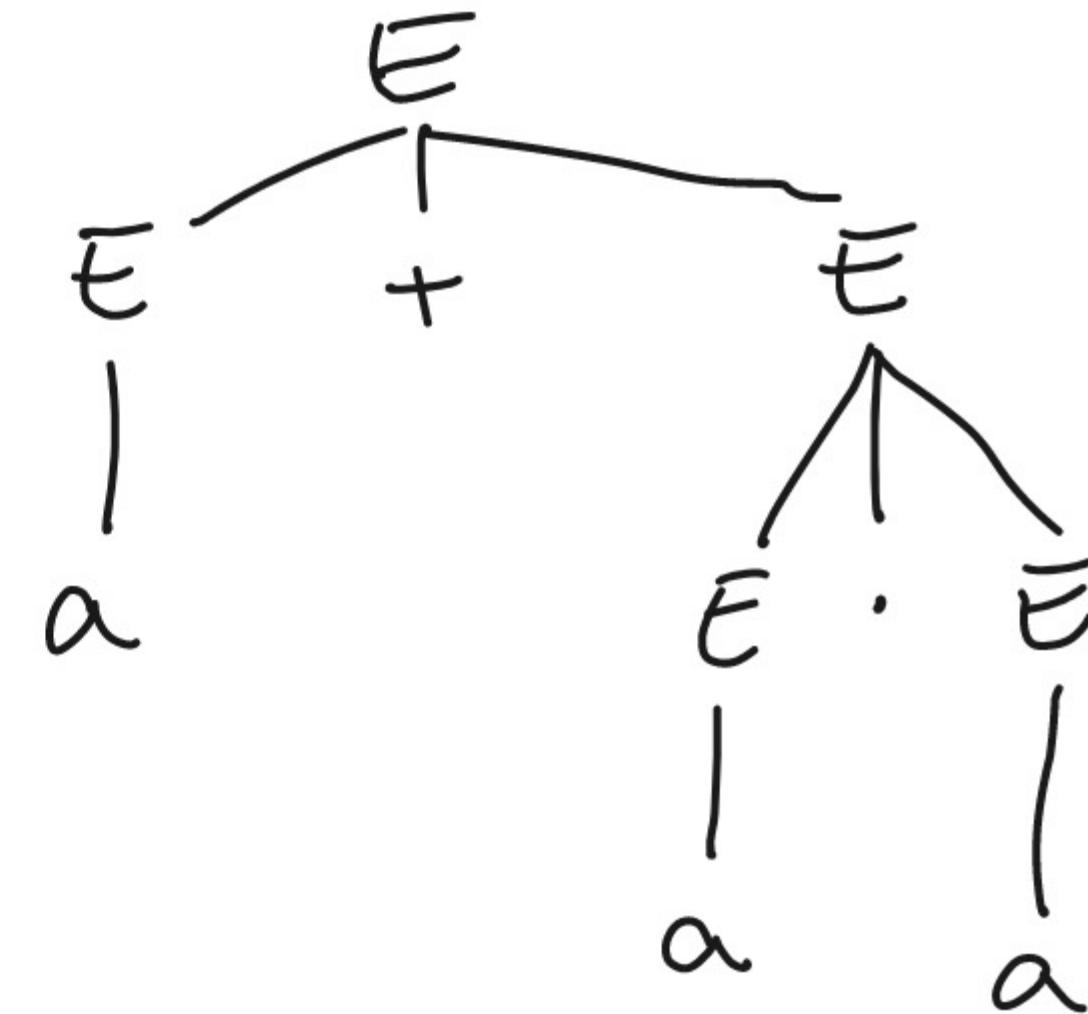
Derivazione 1

$$\underline{E \Rightarrow E+E \Rightarrow a+E \Rightarrow a+E \cdot E \Rightarrow a+a \cdot E \Rightarrow a+a \cdot a}$$

Derivazione 2

$$\underline{E \Rightarrow E \cdot E \Rightarrow E+E \cdot E \Rightarrow a+E \cdot E \Rightarrow a+a \cdot E \Rightarrow a+a \cdot a}$$

Una derivazione è detta estremo sinistra se in ogni derivazione elementare che la compone, la variabile sostituita è sempre la prima.



Una grammatica CF G è non ambigua se ogni stringa $w \in L(G)$ ammette un'unica derivazione estrema sinistra, o equivalentemente un unico albero di derivazione.

Proprietà di chiusura

Teorema Se L_1 e L_2 sono linguaggi CF, anche $L_1 \cup L_2$ lo è.

Dim Sia L_1 generato da $G_1 = (V_1, \Sigma, R_1, S_1)$ e $L_2 = L(G_2)$ con $G_2 = (V_2, \Sigma, R_2, S_2)$. Allora definiamo $G = (V, \Sigma, R, S)$ con:

$V = V_1 \cup V_2 \cup \{S\}$, e R è dato da $R_1 \cup R_2$ più: $S \rightarrow S_1 \mid S_2$.

Evidentemente $L(G) = L_1 \cup L_2$, CVD.

Teorema Se $L_1, L_2 \subseteq \Sigma^*$ sono CF, anche $L_1 \cdot L_2$ lo è.

Dim Sia $L_1 = L(G_1)$ e $L_2 = L(G_2)$, come prima.

Definiamo $G' = (V', \Sigma, R', S')$ con:

$V' = V_1 \cup V_2 \cup \{S'\}$ e R' dato da $R_1 \cup R_2$ più: $S' \rightarrow S_1 S_2$.

Evidentemente $L(G) = L_1 \cdot L_2$, CVD.

Teorema Se $L_1 \subseteq \Sigma^*$ è CF, anche L_1^* lo è.

Dim Sia $L_1 = L(G_1)$ e definiamo $G_0 = (V_0, \Sigma, R_0, S_0)$ con:

$V_0 = V_1 \cup \{S_0\}$ e R_0 dato da R_1 più $S_0 \rightarrow S_0 S_1 \mid \varepsilon$.

Si vede facilmente che $L(G_0) = L_1^*$, CVD.

Soluzione es. 2.1

Derivazione per $a+a+a$:

$$\begin{aligned} E &\Rightarrow E+T \Rightarrow E+T+T \Rightarrow T+T+T \Rightarrow F+T+T \Rightarrow a+T+T \Rightarrow a+F+T \\ &\Rightarrow a+a+T \Rightarrow a+a+F \Rightarrow a+a+a \end{aligned}$$

Derivazione per $((a))$:

$$E \Rightarrow T \Rightarrow F \Rightarrow (E) \Rightarrow (T) \Rightarrow (F) \Rightarrow ((E)) \Rightarrow ((T)) \Rightarrow ((F)) \Rightarrow ((a)).$$

Es. n. 2.4

f) Possiamo considerare $G = (\{S\}, \{0,1\}, R, S)$ con R contenente l'unica regola $S \rightarrow S$ (o qualunque altra regola che produca sempre almeno una variabile).

b) Il linguaggio desiderato è generato dalla grammatica sottivista dalle seguenti regole:

$$S \rightarrow 0|1|0X0|1X1$$

$$X \rightarrow 0X|1X|\epsilon$$

e) Regole:

$$S \rightarrow OS0|1S1|\epsilon|0|1$$

Teorema I linguaggi finiti sono CF.

Dim. Sia $L = \{w_1, \dots, w_k\}$. Allora la grammatica delle regole

$$S \rightarrow w_1 \mid w_2 \mid \dots \mid w_k$$

genera L , CVD.

Corollario Tutti i linguaggi regolari sono CF.

Dim. Immediato dal teorema di Kleene e dalle proprietà di chiusura viste, CVD.

NB: Il viceversa non vale: ad es. $\{a^n b^n \mid n \geq 0\}$ è CF ma non regolare.

Quindi la famiglia dei linguaggi regolari è propriamente contenuta in quella dei linguaggi CF.

Grammatiche regolari

Sia L^* accettato da un DFA $M = (Q, \Sigma, \delta, q_1, F)$.

Consideriamo una grammatica $G = (V, \Sigma, R, S)$, con:

- V in corrispondenza biunivoca con Q (ad ogni q_i corrisp. una var. X_i , e $S = X_1$)
- Se $\delta(q_i, a) = q_j$ per $q_i, q_j \in Q$ e $a \in \Sigma$, allora in R inseriamo la regola

$$X_i \rightarrow a X_j$$

- Se $q_i \in F$, inseriamo la regola $X_i \rightarrow \epsilon$.

(oppure: Se $\delta(q_i, a) \in F$, inserisco $X_i \rightarrow a$)



$$X_i \rightarrow a X_j$$

In generale, una grammatica CF è detta regolare se tutte le sue regole sono di uno tra i seguenti tipi:

- 1) $X \rightarrow aY$ (vaz. produce terminal per vaz.)
- 2) $X \rightarrow a$
- 3) $X \rightarrow \epsilon$

Tutte le grammatiche regolari generano linguaggi regolari.

Esercizio Dato l'alfabeto $\Sigma = \{a, b\}$, dimostrare che il linguaggio di tutte le esp. regolari su Σ è CF.

$$S \rightarrow aXb$$
$$X \rightarrow aXb \mid \epsilon$$

$$\{a^n b^n \mid n \geq 0\}$$

Soluz. esercizio

$$((\underline{a} \cup \underline{b}) \cdot \underline{a})^*$$

$$\hat{\Sigma} = \{\underline{\phi}, \underline{\varepsilon}, \underline{a}, \underline{b}, (,), \cup, \cdot, ^*\}$$

Autami pushdown (to a pile)

Un automa pushdown può leggere e scrivere da una pila (stack LIFO).

Formalmente, un PDA (pushdown automaton) è una 6-pla

$M = (Q, \Sigma, \Gamma, \delta, q_1, F)$ dove Q, Σ e Γ sono insiemi finiti (stati, alfabeto, nastro e alfabeto pile),

$q_1 \in Q$ (stato iniziale), $F \subseteq Q$ (stati terminali o di accettazione) e

$$\delta : Q \times (\underbrace{\Sigma \cup \{\epsilon\}}_{\text{"pop"})} \times (\Gamma \cup \{\epsilon\}) \rightarrow P(\underbrace{Q \times (\Gamma \cup \{\epsilon\})}_{\text{"push")})$$

lettura da pila
 estrazione

scrittura su pila
 inserimento

Data $w \in \Sigma^*$, si ha $w \in L(M)$ se $w = w_1 \dots w_n$, con $n \geq 0$ e $w_i \in \Sigma \cup \{\varepsilon\}$ per $i = 1, \dots, n$, ed esistono n stati $q_1, \dots, q_n \in Q$ e $s_0 = \varepsilon, s_1, \dots, s_n \in \Gamma^*$ tali che:

per $i = 2, \dots, n$, esistono $\alpha, \beta \in \Gamma \cup \{\epsilon\}$ per cui

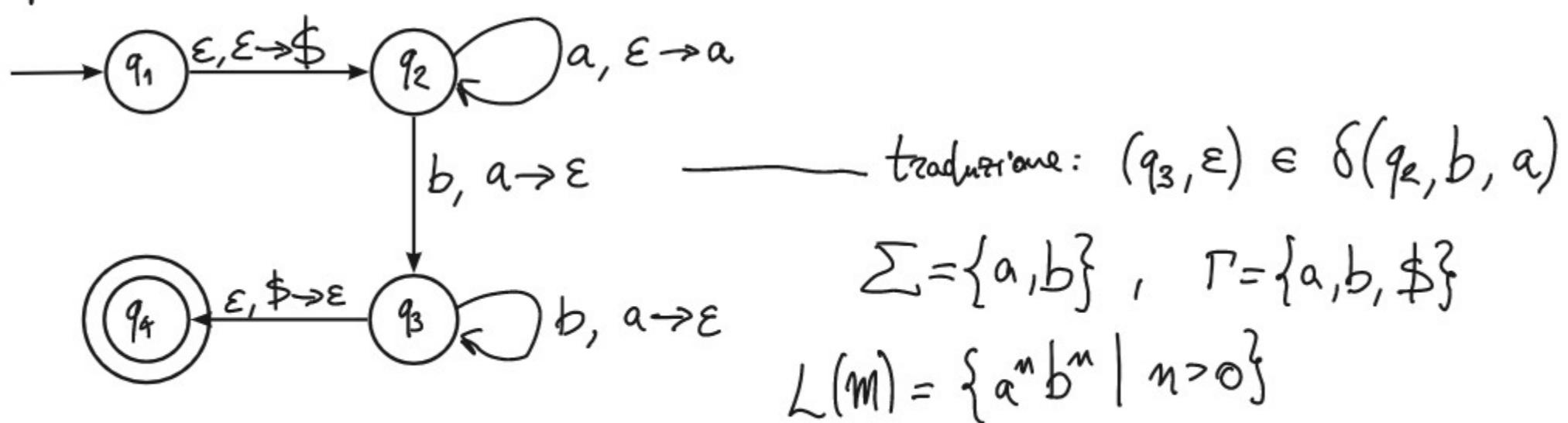
$(q_i, \beta) \in \delta(q_{i-1}, w_i, \alpha)$, dove $s_i = \beta t$, $s_{i-1} = \alpha t$ per qualche $t \in \Gamma^*$, e $q_n \in F$.

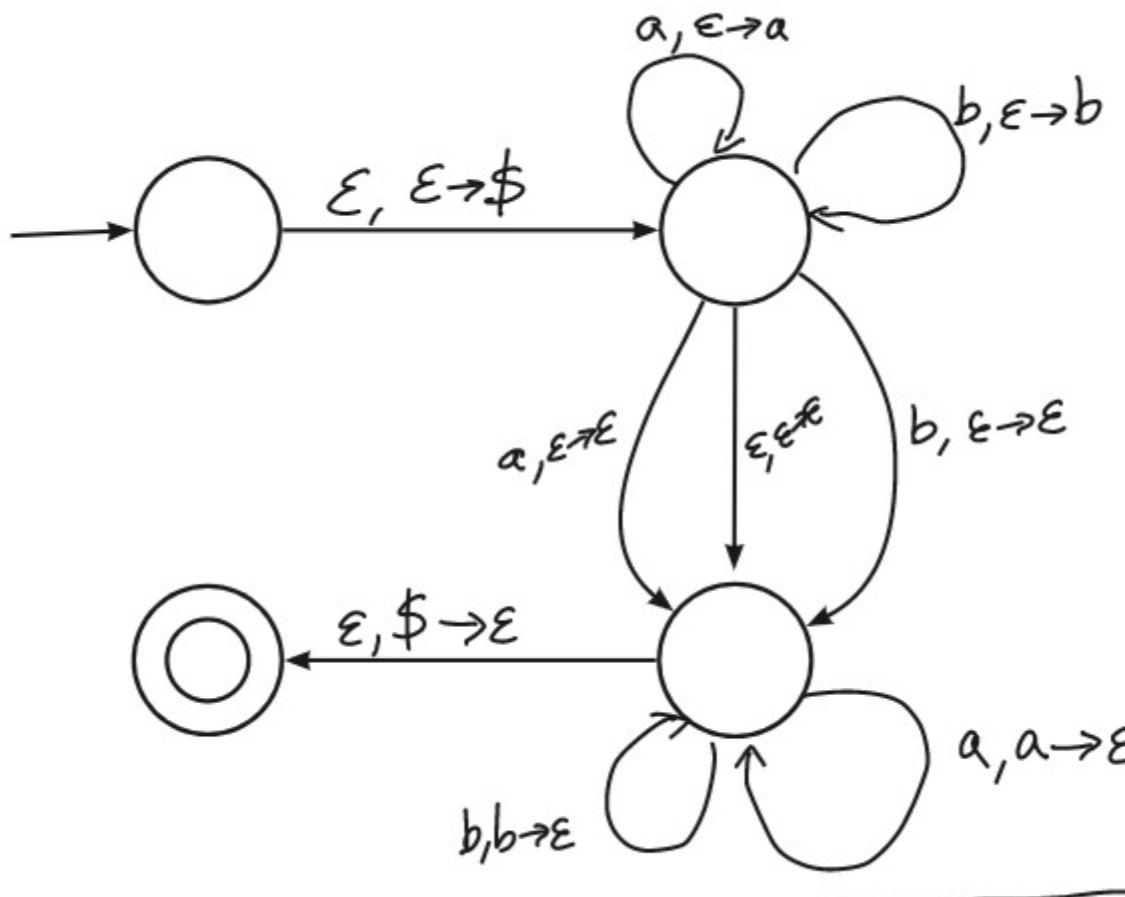
↑
 nuova
lettera in
cima alla
pila (ϵ)

 ↑
 stato
precedente
(ϵ)

 ↑
 lettera
nella
pila
precedente
(ϵ)

Esempio





$L(M) = \{w \in \{a,b\}^* \mid w = \tilde{w}, \text{ cioè } w \text{ palindromo}\}$.

Teorema $L \subseteq \Sigma^*$ è context-free $\Leftrightarrow L$ è accettato da un PDA.

Notazione Se $a \in \Sigma \cup \{\epsilon\}$, $\beta \in \Gamma \cup \{\epsilon\}$, $w = \alpha_1 \dots \alpha_n \in \Gamma^*$, scriviamo

$(q, w) \in \delta^*(q', a, \beta)$ se esistono q_2, \dots, q_m tali che $(q_i, \alpha_i) \in \delta(q', a, \beta)$,
 per $i = 3, \dots, n$ $\delta(q_i, \epsilon, \epsilon) = \{(q_{i-1}, \alpha_{i-1})\}$, e $\delta(q_2, \epsilon, \epsilon) = \{(q_1, \alpha_1)\}$ | $\xrightarrow{q' \xrightarrow{a, \beta \rightarrow w} q}$

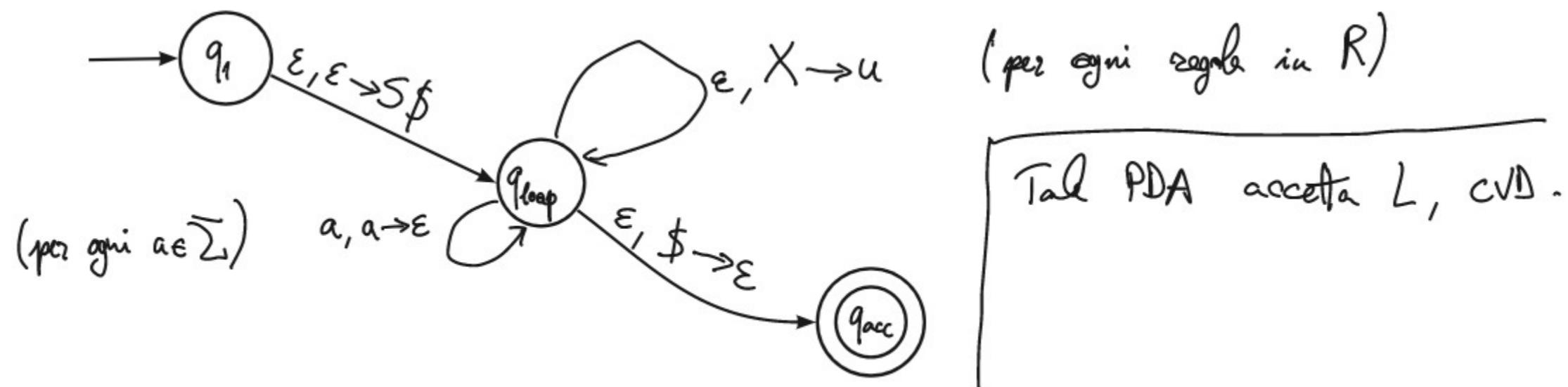
Dim. (solo " \Rightarrow ")

Supponiamo che L sia generato da $G = (V, \Sigma, R, S)$.

Costruiamo un PDA con 3 stati "principali": q_1 , q_{loop} , q_{acc} .

Se R contiene la regola $X \rightarrow u$, facciamo in modo che $(q_{loop}, u) \in \delta^*(q_{loop}, \epsilon, X)$;
inoltre per ogni $a \in \Sigma$, $\delta(q_{loop}, a, a) = \{(q_{loop}, \epsilon)\}$.

Schema del diagramma di transizione risultante:



Es. pag. 163 Sipser , n. 2.5, 2.9, 2.10

5) Descrivere dei PDA che accettano i linguaggi dell'es. 2.4

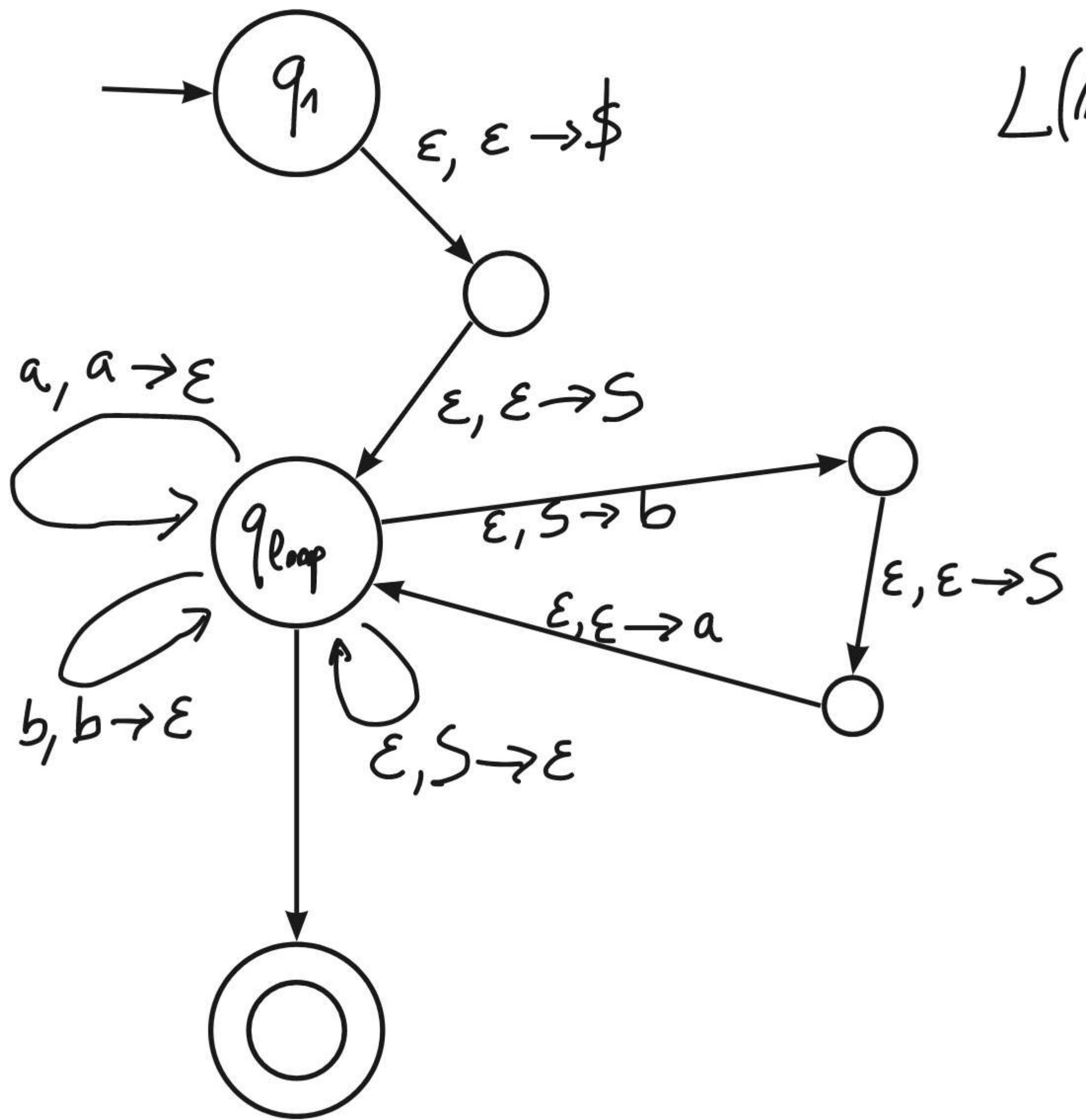
9) Trovare una grammatica CF che genera

$\{a^i b^j c^k \mid i=j \text{ oppure } j=k, i, j, k \geq 0\}$.

È ambigua?

10) Descrivere un PDA che accetti tale linguaggio.

$S \rightarrow aSb \mid \varepsilon$) grammaric CF \mathcal{G} .



$$L(M) = \{a^n b^n \mid n \geq 0\} = L(G)$$

Soluzione es. 2.9

$$L = \{a^i b^j c^k \mid i, j, k \geq 0, \quad i=j \text{ oppure } j=k\}$$

$$S \rightarrow XY \mid WZ$$

$$X \rightarrow aXb \mid \epsilon$$

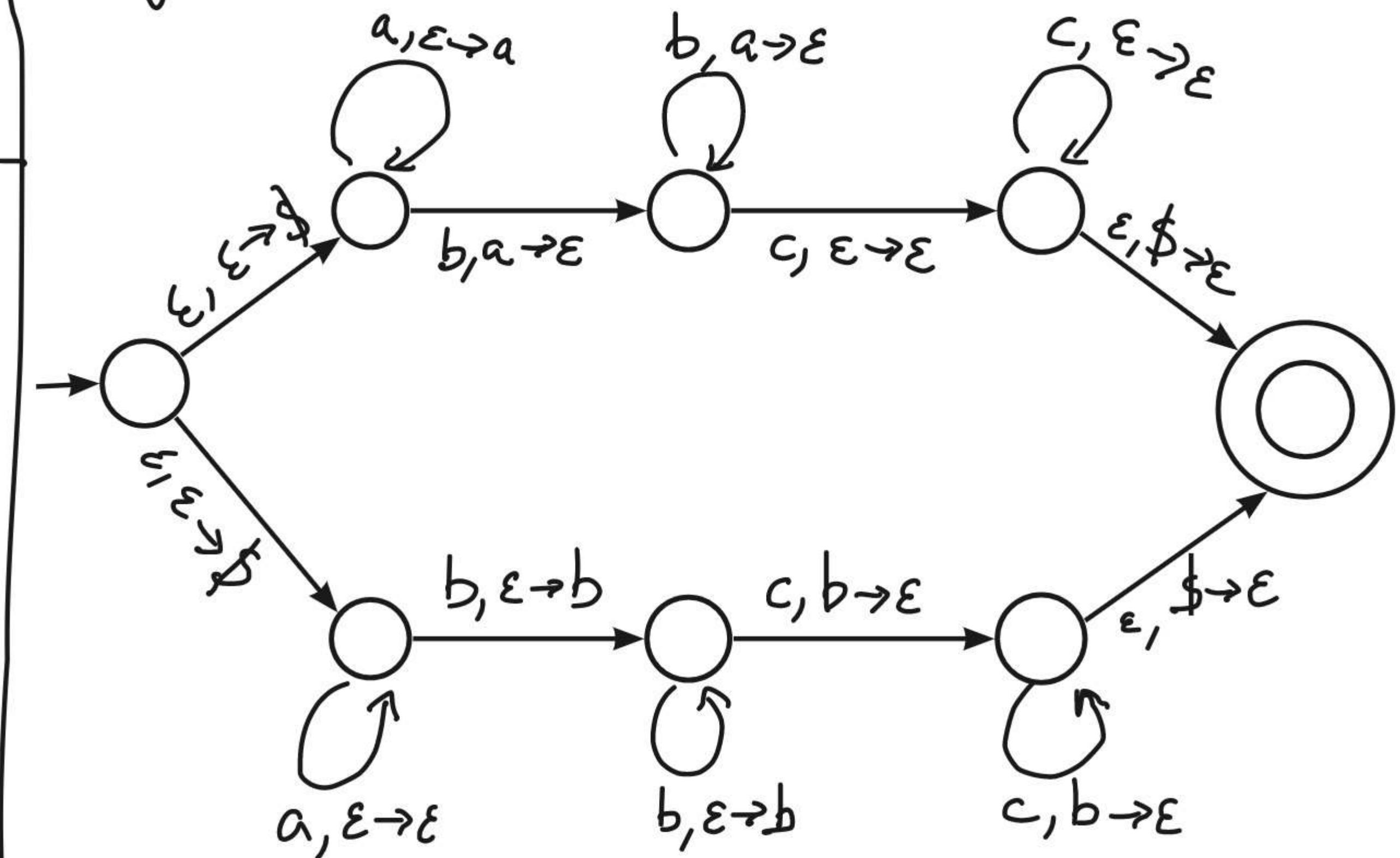
$$Y \rightarrow cY \mid \epsilon$$

$$W \rightarrow aW \mid \epsilon$$

$$Z \rightarrow bZc \mid \epsilon$$

2.10

Diagramma di PDA che accetta L



Conseguenza del teorema (L linguaggio CF $\Leftrightarrow L$ accettato da PDA):

Tutti i linguaggi regolari sono CF.

Infatti un NDFA è un caso particolare di PDA (in cui la pila non è usata e in ogni transizione si legge una lettera dal mostro).

Pumping lemma per linguaggi CF

Teorema. Sia $L \subseteq \Sigma^*$ un linguaggio CF. Allora esiste $p \geq 0$ per cui ogni parola w di L lunga almeno p si può scrivere come $w = uvxyz$, con

- 1) $uv^ixy^iz \in L$ per $i \geq 0$;
- 2) $|vy| > 0$ (cioè v e y non sono entrambe vuote)
- 3) $|vxy| \leq p$

Dimostrazione

Sia L generato da $G = (V, \Sigma, R, S)$, con $V = \{X_1, X_2, \dots, X_R\}$ e sia β la massima lunghezza di una stringa presente come parte destra di una regola.

Possiamo supporre $\beta \geq 2$: se $\beta \leq 1$, $L \subseteq \Sigma \cup \{\epsilon\}$, quindi basta scegliere $p=2$ dato che L non ha parole lunghe almeno 2.

Scegliamo $p = \beta^{R+1}$.

Gli alberi di derivazione di altezza (^{max} lung. cammino da radice a foglia) 1 corrispondono a parole di lunghezza al più β .



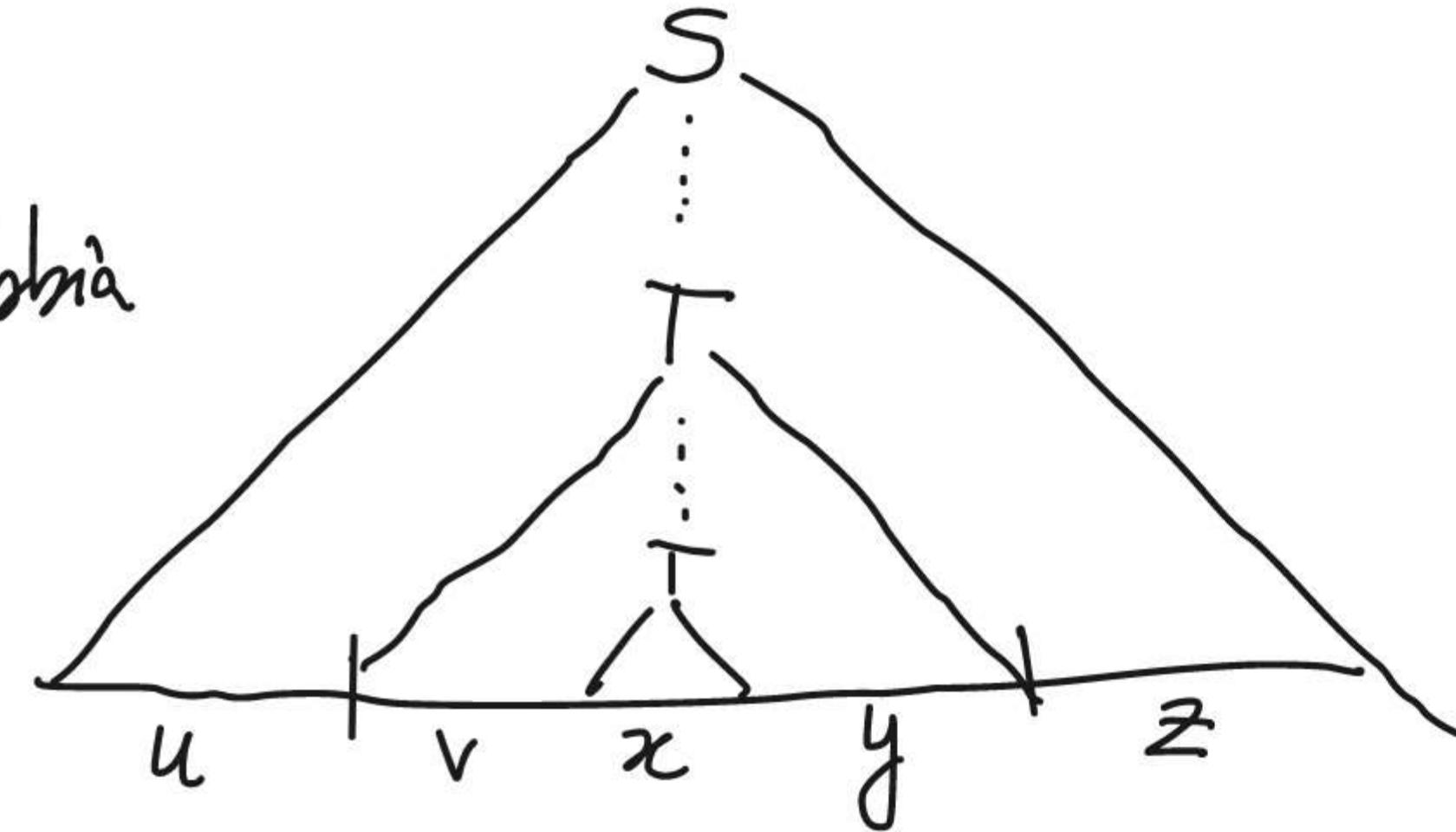
Più in generale, alberi di altezza h corrispondono a parole di lung. al più β^h .

più piccolo possibile, come min. di nodi

Sia $w \in L$ con $|w| \geq p = \beta^{k+1}$. Un albero di deriva^Vzione per w ha lunghe altezza $\geq k+1$; quindi un cammino di lunghezza massima da radice a foglia contiene almeno $k+1$ variabili. Per il principio della piccionaia, almeno una variabile compare almeno 2 volte in tale cammino.

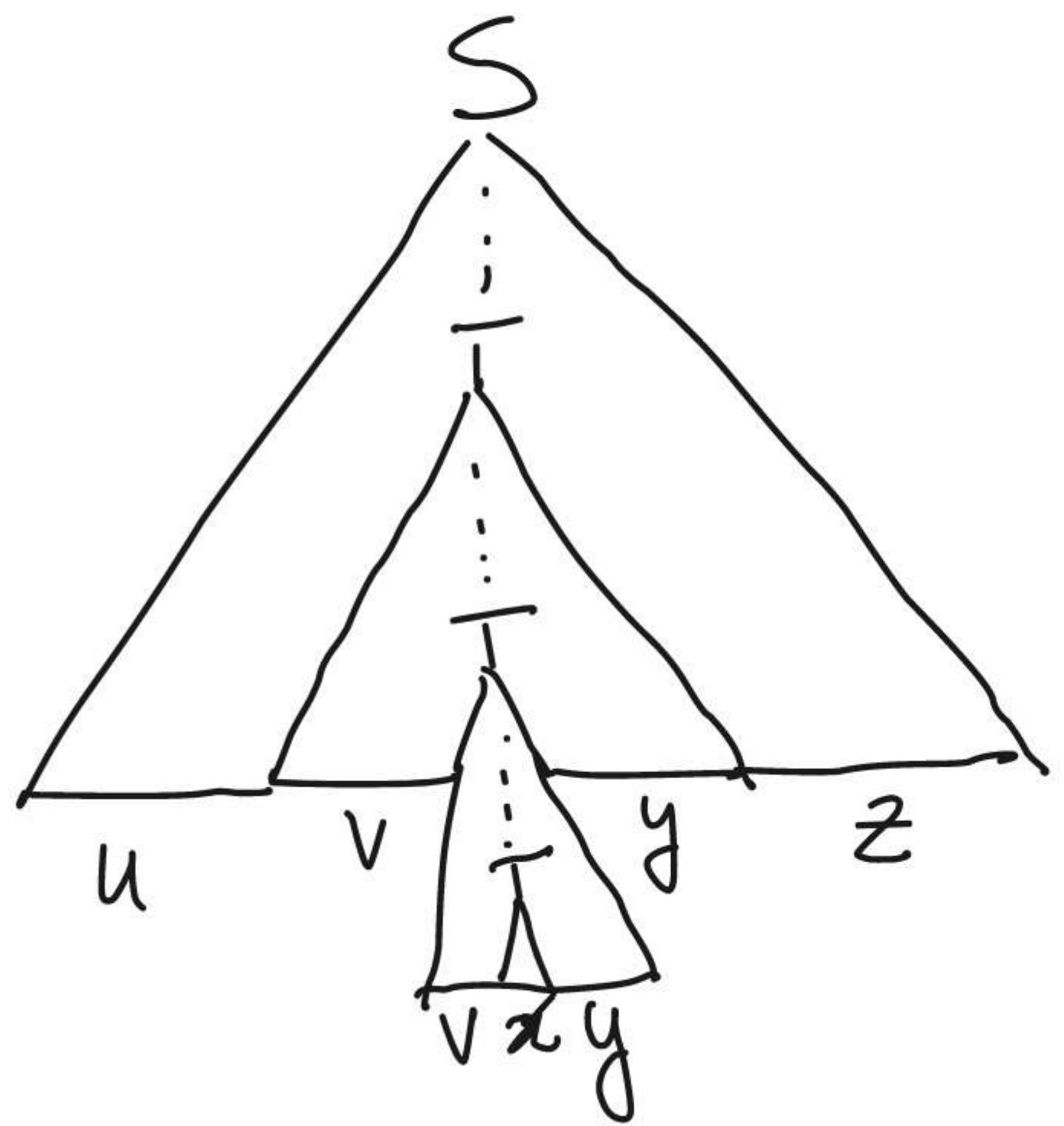
Possiamo rappresentare l'albero così:

dove T è l'ultima variabile nel cammino che abbia almeno 2 occorrenze (e scegliamo le ultime 2)

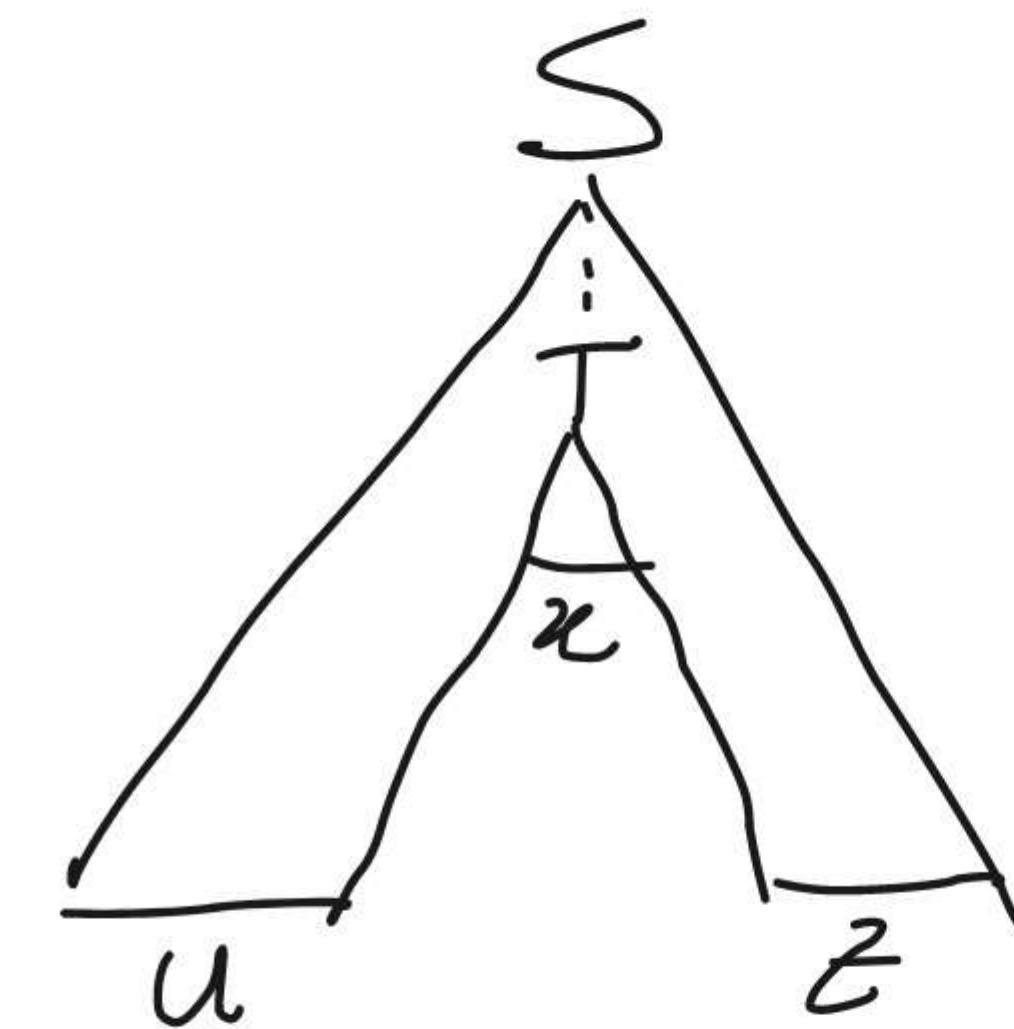


Allora $w = uvxyz$ come in figura.

La proprietà 1 si ottiene sostituendo opportunamente i sottoalberi al di sotto delle occorrenze di T .



$uv^2xyz^2, \dots, uv^ixy^iz$ per $i > 0$



$$uxz = uv^ixy^iz \in L$$

-
- 2) Se per assurdo $v=y=\epsilon$, allora $w=uxz$, l'albero mostrato per uxz avrebbe meno nodi di quello inizialmente scelto, contro la ipotesi.

3) La parte del cammino successiva alla prima occorrenza di T ha lunghezza al più k : altimenti (per piccionaia) dovremmo avere altre variabili ripetute; dunque $|vxy| \leq \beta^{k+1} = p$, CVD.

$$L = \{a^n b^n c^n \mid n \geq 0\} \text{ NON è CF.}$$

Infatti, poniamo p.a. che lo sia e consideriamo $w = a^p b^p c^p$. Dal pumping lemma dovremmo avere $w = uvxyz$ con $|vy| > 0$ e $uv^i xy^i z \in L$ per ogni i .

Se v e y contengono lettere solo di un tipo, $uv^2 xy^2 z$ non avrà lo stesso num. di occorrenze di a , b e c .

Se v (o y) contiene lettere di più tipi, $uv^2 xy^2 z \notin L$.

Ad es. se $v = a^i b^j$, $uv^2 xy^2 z$ contiene ba .

Conseguenza: La famiglia dei linguaggi; CF su Σ NON è chiusa rispetto a intersezione e complemento.

Infatti $\{a^i b^i c^j \mid i, j \geq 0\}$ e $\{a^i b^j c^j \mid i, j \geq 0\}$ sono CF, ma la loro intersezione è \emptyset visto poco fa.

Segue che non c'è chiusura neanche risp. al complemento, da de Morgan.

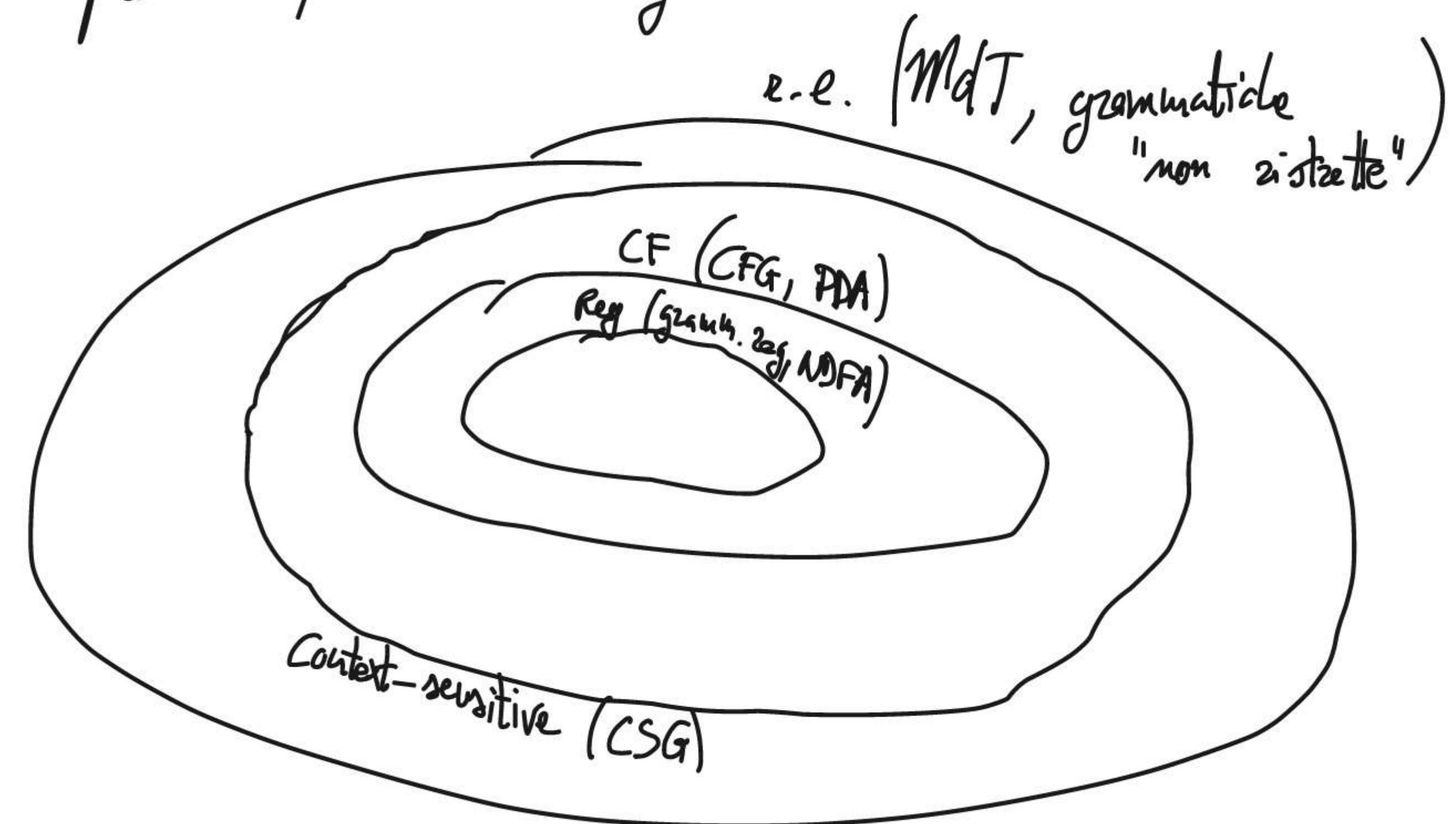
Gerarchia di Chomsky - Schützenberger

Nelle grammatiche CF le regole sono del tipo

$$X \rightarrow u$$

Nelle grammatiche CS invece

$$\alpha X \beta \rightarrow \alpha u \beta, \text{ con } \alpha, \beta \in (\Sigma \cup V)^*$$



Es. pag. 162-3 n. 2.3, 2.11, 2.12

3) Sia G la grammatica individuata dalle regole:

$$R \rightarrow XRX \mid S$$

$$S \rightarrow a\bar{T}b \mid b\bar{T}a$$

$$T \rightarrow XTX \mid X \mid \epsilon$$

$$X \rightarrow a \mid b$$

a) Quali sono le sue variabili?

b) Quali i terminali?

c) Qual è l'assieme? (Risp.: R, non S!)

d) Dare 3 esempi di stringhe in $L(G)$

e) " " " NON in $L(G)$

f) $\bar{T} \Rightarrow aba$, vero o falso?

Altro vero/falso:

g) $T \xrightarrow{*} aba$

h) $\bar{T} \Rightarrow \bar{T}$

j) $XXX \xrightarrow{*} aba$

l) $T \xrightarrow{*} XX$

m) $S \xrightarrow{*} \epsilon$

i) $\bar{T} \xrightarrow{*} \bar{T}$

k) $X \xrightarrow{*} aba$

n) $T \xrightarrow{*} XXX$

o) Descrivere $L(G)$ a parole.

- 11) Convertire la grammatica dell'eserc. 2.1 (su esp. aritmetiche, con \bar{E} , \bar{T} e F)
in un PDA equivalente.
- 12) Lo stesso per la grammatica G dell'es. 2.3.