





## Linguaggi formali e Computabilità

### Riassunto Concetti

**Alfabeto:** Insieme finito di caratteri. Si denota con  $\Sigma$  (sigma).

**Stringa:** Sequenza finita di caratteri presi da un alfabeto.

- **Lunghezza:** n° posizioni presenti nella stringa.
- **Stringa vuota:** Si denota con  $\epsilon$  (epsilon minuscola).
- **Alfabeto infinito:** Insieme di tutte le stringhe. Si denota con  $\Sigma^*$

**Linguaggio:** È un insieme di stringhe. Può essere:

- **Riconoscitore:** riconosce validità di stringhe rispetto ad un linguaggio.
- **Generatore:** regole che, attraverso sostituzioni, generano un linguaggio.

**Regola di produzione:** si applica la regola. Si denota con  $\rightarrow$

**Passo di derivazione:** ha applicato una regola di produzione. Si denota con  $\Rightarrow$

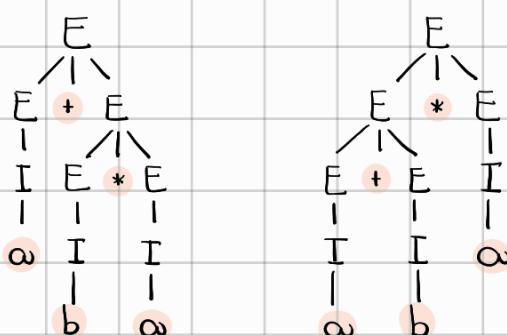
**Grammatica:** Una grammatica è definita come  $G = (V, T, P, S)$  dove:

- $V$  insieme finito di simboli non terminali.
- $T$  insieme finito di simboli terminali.
- $P$  insieme produzioni.
- $S$  start symbol

**Grammatica ambigua:** quando una stringa può essere ottenuta con due alberi sintattici.

Esempio:  $E \rightarrow I \mid E + E \mid E * E$

$I \rightarrow a \mid b$



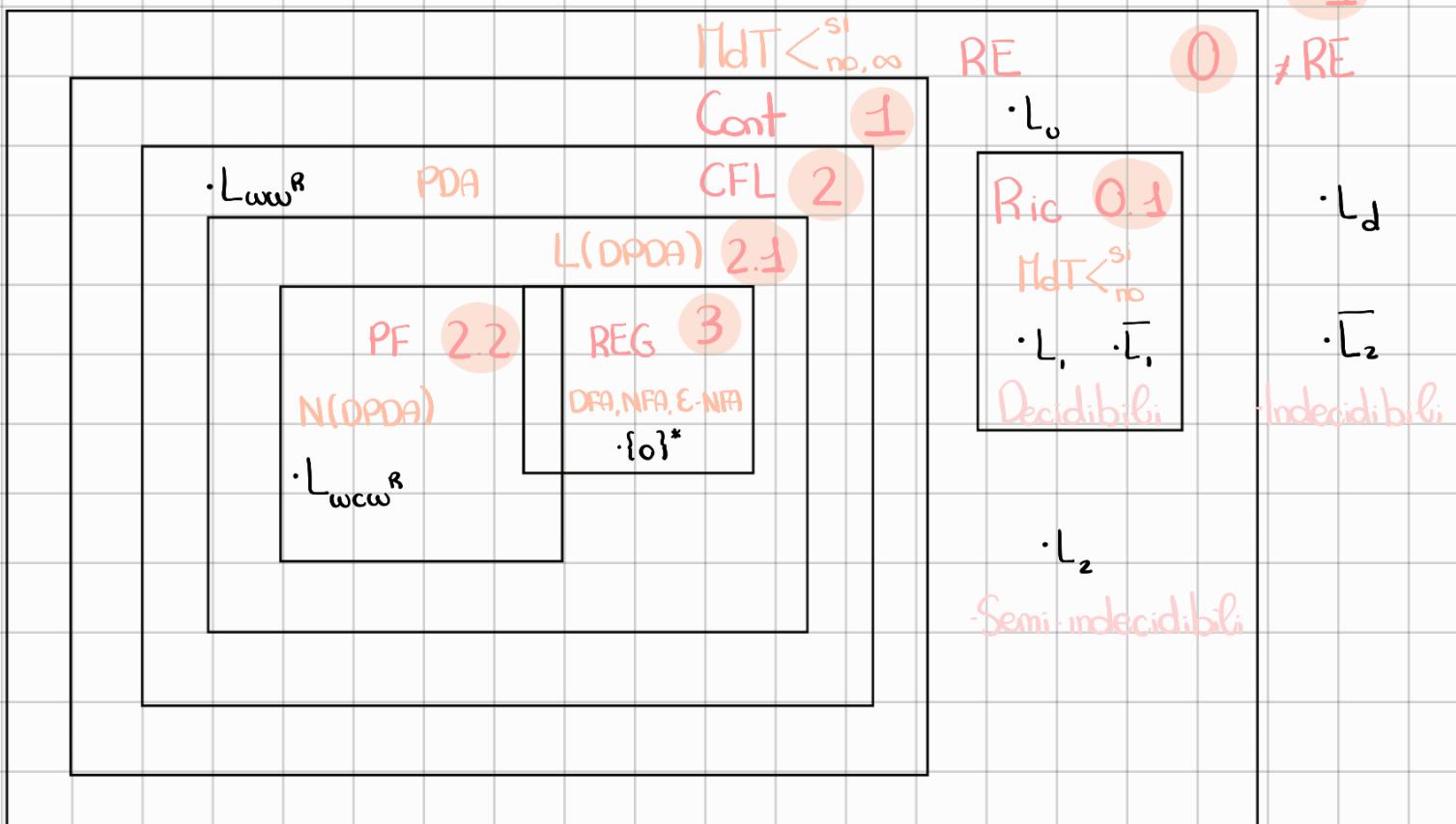


# Linguaggi formali e Computabilità

## Riassunto Concetti

### Gerarchia di Chomsky versione Quack

- **Tipo 0:** Linguaggi Ricorsivamente Enumerabili (RE). Riconosciuti da Macchine di Turing (MdT).
- **Tipo 1:** Linguaggi Contestuali.
- **Tipo 2:** Linguaggi Context-Free. (CFG) Vengono riconosciuti da Automi o Pilas Non Deterministici (PDA).
- **Tipo 3:** Linguaggi Regolari (REG). Sono riconosciuti da automi o stati finiti. (DFA, NFA, E-NFA).
- **Tipo -1:** Linguaggi Non Ricorsivamente Enumerabili. ( $\neq$  RE) Non hanno MdT che li accettano.
- **Tipo 0.1:** Linguaggi Ricorsivi (Ric). Sono RE ma la MdT che li accettano non va mai in loop infinito
- **Tipo 2.1:** Linguaggi accettati da automi o pilas deterministico per stati finiti.
- **Tipo 2.2** Linguaggi Prefix-free. (PF) Sono accettati da automi o pilas deterministico anche per pilas vuote





# Linguaggi Formali e Computabilità

## Linguaggi Regolari

3

Linguaggi regolari: I linguaggi regolari sono:

- generati: da grammatiche di tipo 3, avveri regolari. (lineari o sx o dx)
- definiti: da espressioni regolari (ER)
- accettati: da automi a stati finiti (DFA, NFA, E-NFA)

### Generazione

#### Vincoli

- $\Sigma$ : compare solo in  $S \rightarrow E$  sse  $S$  start-symbol.
- Lineari: regole di produzione tutte lineari o dx o o sx

#### Esempi

- $L = \{a^n b^m \mid n, m \geq 0\}$  lineare o dx

$$S \rightarrow \epsilon \mid b \mid aS \mid bB$$

$$B \rightarrow b \mid bb$$

- $L = \{ab^n cd^m e \mid n \geq 0, m > 0\}$  lineare o sx

$$S \rightarrow Xe$$

$$X \rightarrow Yd \mid Xd$$

$$Y \rightarrow Zc$$

$$Z \rightarrow a \mid Zb$$



# Linguaggi Formali e Computabilità

## Linguaggi Regolari

3

### Denotazione

**Definizione:** Le espressioni regolari sono un modo dichiarativo di esprimere le stringhe ed accettarle.

**Definizione ricorsiva:**

· **caso base**

- $\epsilon$  e  $\emptyset$  sono ER
- Se  $a \in \Sigma$ ,  $a$  ER
- Variabili che rappresentano linguaggi (e.s. L) sono ER.

· **caso ricorsivo:**

- $L(E+F) = L(E) + L(F)$  (commutativa, associativa, identità ( $\emptyset$ ), idempotenza ( $L+L$ ), distributiva)
- $L(E \cdot F) = L(E) \cdot L(F)$  (associativa, identità ( $\epsilon$ ), annullatore ( $\emptyset$ )).
- $L(E^*) = (L(E))^*$  (idempotenza,  $\emptyset^* = \epsilon$ ,  $\epsilon^* = \epsilon$ ,  $L^+ = L \cdot L^*$ ,  $L^* = L^+ + \epsilon$ )
- $L((E)) = L(E)$

**Esempi:**

- $ER = (0+1)^* = (L(0+1))^* = (L(0) \cup L(1))^* = (\{0\} \cup \{1\})^* = \{0,1\}^*$  tutte le stringhe binarie.
- $ER = ((01)^* \sqcup (0+1)^*)^*$  dire quali stringhe  $w_i \in L$ .
  - $w_1: \underline{0101}$  **No**, genero 01 2 volte con  $(01)^*$  ma poi devo generare 10 ma ho 01.
  - $w_2: 10111$  **Si**, non genero  $(01)^*$ , tra 10 e poi genero 3 volte 1 con  $(0+1)^*$
  - $w_3: 0101010101$  **Si**, genero 2 volte 01 con  $(01)^*$ , tra 10 e poi qualsiasi cosa è ok.



# Linguaggi Formali e Computabilità

## Linguaggi Regolari

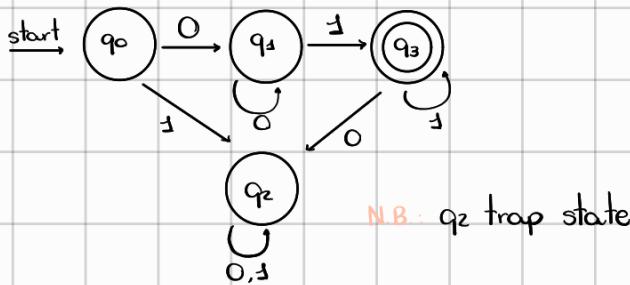
3

### Accettazione

**Definizione DFA.** Un DFA è una quintupla  $A = \{Q, \Sigma, \delta, q_0, F\}$  dove:

- $Q$ : insieme degli stati.
- $\Sigma$ : insieme delle stringhe in input
- $\delta$ : funzione di transizione degli stati:  $\delta: Q \times \Sigma \rightarrow Q$  totale
- $q_0 \in Q$ : stato iniziale
- $F \subseteq Q$ : insieme degli stati finali.

**Esempio DFA.** DFA per  $L = \{0^n 1^m \mid n > 0, m > 0\}$



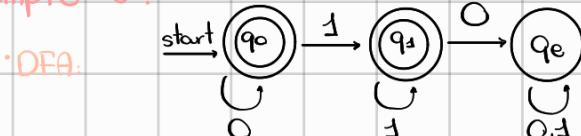
**Definizione  $\hat{\delta}$ :** Dato un DFA, il linguaggio accettato è  $L(A) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}$

**Definizione ricorsiva di  $\hat{\delta}$ :**

**Base:** se  $|w| = 0$ , allora  $\hat{\delta}(q, \epsilon) = q$

**Passo:** se  $|w| > 0$ , allora  $w = ux$  con  $u \in \Sigma$ ,  $x \in \Sigma^*$  t.c.  $\hat{\delta} = (q, w) = \hat{\delta}(q, u)x = \hat{\delta}(\hat{\delta}(q, u), x)$

**Esempio  $\hat{\delta}$ :**



$\hat{\delta}$ :  $\hat{\delta}(q_0, 0001) = \hat{\delta}(\delta(q_0, 0)001) = \hat{\delta}(q_1, 001) = \hat{\delta}(\delta(q_1, 0), 01) = \hat{\delta}(q_2, 01) = \hat{\delta}(\delta(q_2, 0), 1) = \hat{\delta}(q_2, 1) = \hat{\delta}(\delta(q_2, 1)\epsilon) = \delta(q_2, \epsilon) = q_2 \in F$



# Linguaggi Formali e Computabilità

## Linguaggi Regolari

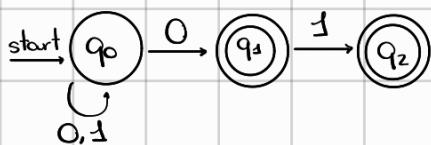
3

### Accettazione

**Definizione NFA:** Un NFA è una quintupla  $A = \{Q, \Sigma, \delta, q_0, F\}$  dove:

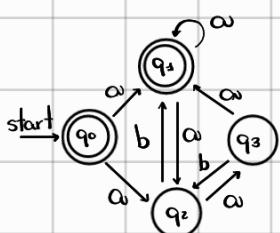
- $Q$  insieme degli stati.
- $\Sigma$  insieme delle stringhe in input
- $\delta$ : funzione di transizione degli stati:  $\delta: Q \times \Sigma \rightarrow 2^Q$  insieme parti di  $Q$  parziale
- $q_0 \in Q$ : stato iniziale
- $F \subseteq Q$ : insieme degli stati finali.

Esempio:



**Trasformazione da NFA a DFA:** trascrivo nella tabella solo gli stati necessari

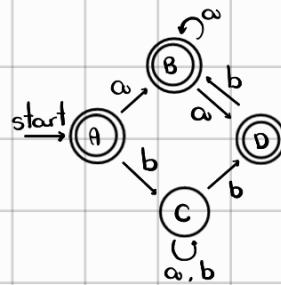
• NFA:



• Tabella di stati:

	a	b	
*	A {q0}	B {q3, q2}	C ∅
*	B {q2, q3}	B {q3, q2}	D {q3}
*	C ∅	C ∅	∅
*	D {q2, q3}	B {q3, q2}	∅

• DFA:



**Definizione  $\epsilon$ -NFA:** come NFA ma  $\delta$  cambia in quanto possiamo muoverci senza consumare mosse:  $\epsilon$ -mosse.

**Eclose(a):** insieme di stati raggiungibili da uno stato con solo  $\epsilon$ -mosse:  $Q \rightarrow 2^a$



# Linguaggi Formali e Computabilità

## Linguaggi Regolari

3

### Accettazione

Definizione ricorsiva eclose(q):

- **Base:**  $q \in \text{eclose}(q)$
- **Passo:** se  $p \in \text{eclose}(q)$  ed esiste transizione etichettata  $\epsilon$  da  $p$  a  $r$  allora  $r \in \text{eclose}(q)$ .

Definizione ricorsiva  $\hat{\delta}: 2^a \times \Sigma^* \rightarrow 2^a$

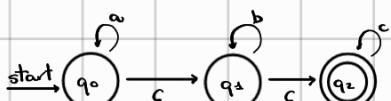
- **caso base:**  $|w| = 0, \hat{\delta}(S, \epsilon) = \text{eclose}(S)$
- **caso passo:**  $|w| > 0, w = \omega x, \omega \in \Sigma, x \in \Sigma^*, \hat{\delta}(S, \omega x) = \hat{\delta}(\hat{\delta}(S, \text{eclose}(S), \omega), x)$

Trasformazione da  $\epsilon$ -NFA a DFA

Dato  $\epsilon$ -NFA otteniamo DFA equivalente, cioè  $L(\epsilon\text{-NFA}) = L(\text{DFA})$ . Avremo:

- $q_d(\text{eclose}_n)$
  - $F_d = \{s \in Q_d \mid s \cap F_0 \neq \emptyset\}$
  - $\forall a \in \Sigma \quad \forall S = \{p_1, p_2, \dots, p_n\} \in Q_d \quad a \subseteq Q_a$ .
- $$\delta_d(S, a) = \text{eclose}(R) \text{ con } R = \{r_1, r_2, \dots, r_m\} = \bigcup_{p \in S} \delta(p, a)$$

•  $\epsilon$ -NFA



•  $\text{eclose}$

$$\text{eclose}(q_0) = \{q_0, q_1, q_2\}$$

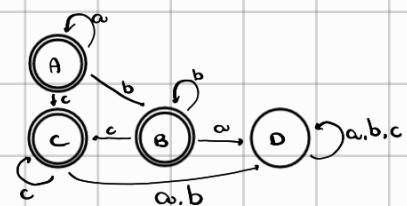
$$\text{eclose}(q_1) = \{q_1, q_2\}$$

$$\text{eclose}(q_2) = \{q_2\}$$

• Tabella start

	a	b	c
A	$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_2\}$
B	$\emptyset$	$\{q_2\}$	$\{q_2\}$
C	$\emptyset$	$\emptyset$	$\{q_2\}$
D	$\emptyset$	$\emptyset$	$\emptyset$

• DFA



$$\begin{aligned} \cdot \hat{\delta}: \hat{\delta}(\{q_0, q_1, q_2\}, a) &= \text{eclose}(\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a)) = \text{eclose}(\{q_0\} \cup \emptyset \cup \emptyset) \\ &= \text{eclose}(\{q_0\}) = \text{eclose}(q_0) = \{q_0, q_1, q_2\} \end{aligned}$$



# Linguaggi Formali e Computabilità

## Linguaggi Regolari

3

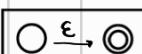
### Accettazione

#### Dai ER ai $\epsilon$ -NFA

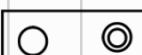
- 1 stato accettante
- nessun arco entrante nello stato iniziale e uscente nell'accettante.

#### Dai ER ai $\epsilon$ -NFA per induzione

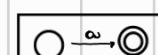
base: se  $R = \epsilon$ ,  $L(R) = \epsilon$ , avremo



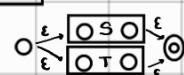
• se  $R = \emptyset$ ,  $L(R) = \emptyset$  avremo



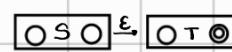
• se  $R = a$ ,  $L(R) = \{a\}$  avremo



passo: se  $R = S + T$ ,  $L(R) = L(S) \cup L(T)$  avremo



• se  $R = ST$ ,  $L(R) = L(S) \cdot L(T)$  avremo



• se  $R = S^*$ ,  $L(R) = (L(S))^*$  avremo



• se  $R = (S)$ ,  $L(R) = L(S)$  l'  $\epsilon$ -NFA per  $S$  vale anche per  $R$ .

#### Esempio ER $\rightarrow$ $\epsilon$ -NFA

$$ER = (0+1)^* \cdot 1 (0+1)$$





# Linguaggi formali e Computabilità

3

## Linguaggi Regolari

### Accettazione

Trasformazione da DFA a ER: Caso base

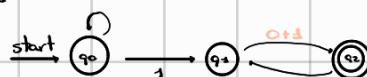
- $\xrightarrow{\text{start}} \textcircled{0} \xrightarrow{\text{ER}} \textcircled{1}$
- $\xrightarrow{\text{start}} \textcircled{0} \xrightarrow{\text{ER}} \textcircled{0}$

Dfa:  $q_0$  stato iniziale,  $q_1, \dots, q_n$  stati finali, resto da eliminare.

Esempio DFA a ER



- riscrivo archi



- elimino  $q_1$

· predecessori:  $q_0, q_2$  ER:  $q_0 \rightarrow q_2 : 1 \quad q_2 \rightarrow q_2 : 0 + 1$

· successori:  $q_2$  ER:  $q_1 \rightarrow q_2 : 0 + 1$

· no self-loop ER:  $\emptyset$

· passaggi:  $R_{0,2} : \emptyset, R_{2,2} : \emptyset$

· Inseriamo passaggi con formula: passaggio diretto +  $x \rightarrow y, y \rightarrow z$  dove  $R_{xz}$  e  $y$  da eliminare.

$$R_{0,2} = \emptyset + \emptyset \emptyset^* (0+1) = \emptyset (0+1)$$

$$R_{2,2} = \emptyset + (0+1) \emptyset^* (0+1) = (0+1)(0+1)$$

· Ottieniamo l'automa:

· Riprendo schema e formula generale

· Riscrivo la formula:  $ER = (0+1(0+1))((0+1)(0+1))^* \emptyset^* + (0+1)((0+1)(0+1))^*$

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



# Linguaggi formali e Computabilità

3

## Linguaggi Regolari

### Accettazione

#### Algoritmo riempì-tabella

1. Se trovi uno distinguibile, segno con X.
  2. Se dopo 1 giro ho messo una X, ripeto sui quadranti bianchi metto una X se due stati vanno in due stati distinguibili.
- I quadrati che rimangono vuoti sono le classi di equivalenza. Se un letterale non ne ha allora è classe d'equivalenza.

DFA minimizzato: le classi di equivalenza sono gli stati.

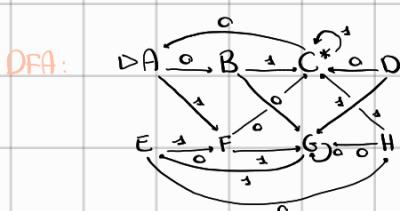


Tabella:

B	X					
C	X	X				
D	X	X	X			
E		X	X	X		
F	X	X	X		X	
G	X	X	X	X	X	X
H	X		X	X	X	X
A B C D E F G H						

Legenda:

- Denoto con X quelli che ottengo senza mosse (gli stati da cui parto sono già diversi!)
- Denoto con X quelli che ottengo con casa base.
- Denoto con X quelli che ottengo con casa passo.

(Classi equivalenza):  $\{A, E\}$ ,  $\{B, H\}$ ,  $\{C, F\}$ ,  $\{D\}$ ,  $\{G\}$  eliminiamo D in quanto non ha archi entranti.





# Linguaggi formali e Computabilità

3

## Linguaggi Regolari

### Accettazione

**Pumping lemma:** Si usa per dimostrare che un linguaggio regolare.

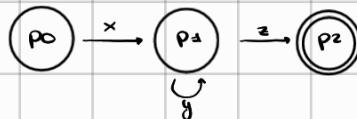
Sia  $L$  un linguaggio regolare.  $\exists$  una costante  $n$  (dipendente da  $L$ ) tc.  $\forall w \in L$  con  $|w| \geq n$

$w$  può essere scomposto come  $w = xyz$  tc:

- $y \in \Sigma^*$
- $|xy| \leq n$
- $\forall k \geq 0$ , anche  $xy^kz \in L$

**Dimostrazione:** se  $L \in \text{Reg.}$ , allora  $\exists$  DFA che lo accetta. Suppongo che  $A$  abbia  $n$  stati e consideriamo  $w = a_1a_2a_3 \dots a_m$  con  $m \geq n$ .  $\forall i = 0, 1, \dots, n$  sia  $p_i = \hat{S}(a_0, a_1, \dots, a_i)$  con  $p_0 = q_0$ .  $p_0, p_1, \dots, p_n$  sarebbero  $n+1$  stati, allora  $\exists i, j$  tc  $0 \leq i \leq j \leq n$  tc  $p_i = p_j$ . Scomponiamo allora  $w = xyz$  come segue.

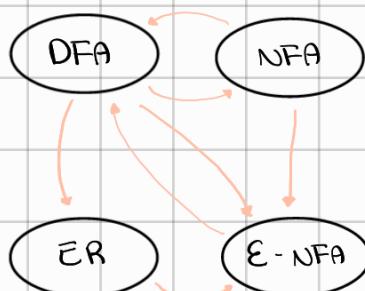
- $x = a_1, \dots, a_i$
- $y = a_{i+1}, \dots, a_j$
- $z = a_{j+1}, \dots, a_m$



**Esempio:** Dim che  $L_{0s} = \{x^n y^n \mid n \geq 1\}$  non è regolare. Suppongo lo sia, allora  $n \in \mathbb{N}$  ha costante di PL. Sia  $w = 0^n 1^n \in L_{0s}$ . Scriviamo  $x = wyz$ :

$x = 0^{n-1}$   $y = 0$   $z = 1^n$  allora per PL deve valere  $xy^kz \in L_{0s}$ ,  $\forall k \geq 0$ . Ma per  $k=0$ ,  $xz \notin L_{0s}$ .

**Schema riassuntivo:**





## Linguaggi formali e Computabilità Context free

2

Linguaggi context free : I linguaggi context free sono:

- generati da grammatiche di tipo 2, ovvero context free grammar (CFG).
- accettati da automi a pilas non deterministici

(Generazione)

Regole: Due modi per costruire un sottobilinguaggio.

- concatenare
- contenitore

Definizione CFG: Un linguaggio generato da una grammatica  $G = (V, T, P, S)$  è  $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$ . Se  $G$  è una CFG allora  $L$  è una CFL.

Derivazione: La derivazione avviene con leftmost o rightmost

Esempio: CFG per  $L = \{a^n b^k c^l d^m \mid n > 0, k \geq 0\}$

- $S \rightarrow aSd \mid aCd$
- $C \rightarrow \epsilon \mid bCc$
- $S \Rightarrow aCd \Rightarrow ad$
- $S \Rightarrow aSd \Rightarrow aaCd \Rightarrow aad$



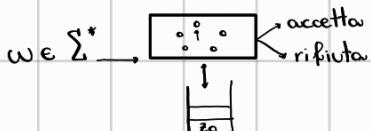
# Linguaggi formali e Computabilità

## Context free

2

### Accettazione

Automa su pila: E-NFA + stack.



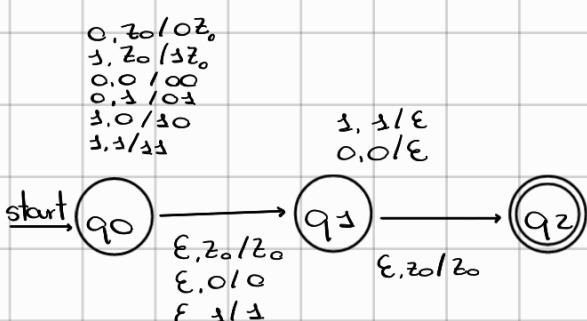
Definizione PDA: È una settepla  $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

- $Q$ : insieme finito e non vuoto di **stati**.
- $\Sigma$ : alfabeto di simboli in **input**.
- $\Gamma$ : alfabeto di simboli di **stack**.
- $\delta: Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow P(Q \times \Gamma^*)$
- $q_0 \in Q$ : stato **iniziale**.
- $z_0 \in \Gamma$ : simbolo **inizialmente** presente nello **stack**.
- $F \subseteq Q$ : insieme stati **finali**.

Esempio PDA:  $L_{wwr} = \{wwr \mid w \in \{0,1\}^*\}$  palindromi pari. CFG:  $S \rightarrow 0S0 \sqcup S \sqcup \epsilon$

Si può fare in 3 stati: riempio la pila, matcho e svuoto, accetto.

$$P = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, z_0\}, \delta, q_0, z_0, \{q_2\})$$



Descrizione istantanea (10): È una tripla  $(q, w, \gamma)$  dove:

- $q \in Q$  è lo stato **attuale**.
- $w \in \Sigma$  è l'**input** residuo.
- $\gamma \in \Gamma$  è il **contenuto attuale** dello **stack**.



# Linguaggi formali e Computabilità

## Context free

2

### Accettazione

**Definizione massiva:** Sia  $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$  un PDA e supponiamo che  $(p, \alpha) \in \delta(q, \omega, x)$

Allora  $\forall w \in \Sigma^* \ \forall \beta \in \Gamma^*, (q, \alpha w, x \beta) \vdash_p (p, w, \alpha \beta)$ .

**Definizione ricorsiva:**  $I_P^*$

base:  $I \vdash^* I$

passo:  $I \vdash^* J \quad \exists I D K \text{ tc } I \vdash K, K \vdash^* J$ .

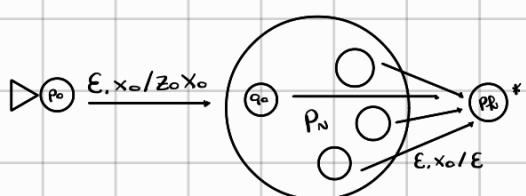
**Linguaggi accettati da PDA**

**Stato finale:** Sia  $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ , Paccetta per stato finale il linguaggio  $L(P) = \{w \in \Sigma^* \mid (q_0, w, z_0) \vdash_p^* (q, \varepsilon, x) \text{ con } q \in F \text{ e } x \in \Gamma^*\}$

**Pila vuota:** Sia  $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ , Paccetta per pila vuota il linguaggio

$N(P) = \{w \in \Sigma^* \mid (q_0, w, z_0) \vdash_p^* (q, \varepsilon, \varepsilon) \text{ con } q \in Q\}$

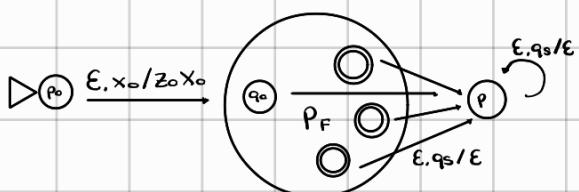
**Pila vuota o stato finale:** Se  $L = N(P_N)$  per PDA  $P_N$  allora  $\exists P_F \text{ tc } L = L(P_F)$



$P_F = (Q \cup \{p_0, p_F\}, \Sigma, \Gamma \cup \{x_0\}, \delta_F, p_0, x_0, \{p_F\})$ .

dove  $\delta_F = \delta_N + \delta$  di  $p_0$  +  $\delta$  di tutti i passi per  $p_F$ .

**Dai stati finali a pila vuota:** se  $L = L(P_F)$  per un PDA  $P_F$  allora  $\exists$  PDA  $P_N$  tc  $L = N(P_N)$



$P(N) = \{Q \cup \{p_0, p_N\}, \Sigma, \Gamma \cup \{x_0\}, \delta_N, p_0, x_0\}$ .

dove  $q_S$  è  $\forall$  simbolo di  $\Gamma \cup \{x_0\}$



# Linguaggi formali e Computabilità

## Context free

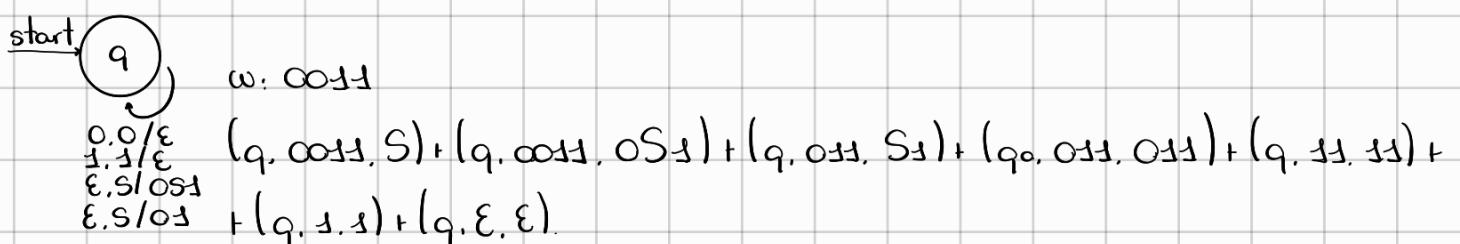
2

### Accettazione

Da CFG a PDA per pilawata: Dato  $G = (V, T, Q, S)$  costruiamo PDA  $P = (Q, T, V \cup T, \delta, q_0, S)$

- $\forall a \in V, \delta(q, \epsilon, A) = \{(q, \beta) | A \rightarrow B \text{ produzione di } G\}$ .
- $\forall a \in T, \delta(q, a, a) = \{(q, \epsilon)\}$

Esempio:  $S \rightarrow 0S110s$



PDA deterministico: Un PDA  $P$  è deterministico (DPA) deve soddisfare:

- $|\delta(q, a, x)| \leq 1$
- se  $|\delta(q, a, x)| = 1$  per  $a \in \Sigma$ ,  $|\delta(q, \epsilon, x)| = 0$

Linguaggio accettato DPA: Un DPA accetta meno dei PDA perché non riconosce tutti i CFL come  $L_{wwr}$  in quanto necessita di un valore "sentinella" per cambiare stato.

Prefix-free: Un linguaggio è prefix-free se  $\exists x, y \in L$  t.c  $x \neq y$  e  $x$  è prefissario di  $y$ .

Esempio:  $L_{wcwr}$  è prefix-free? sì perché togliendo un elemento  $\notin L$ .  $\{0\}^*$  non è prefix-free

Teorema: Un DPA accetta per pilawata sse  $L$  è prefix-free.



# Linguaggi formali e Computabilità

## Contestuali

**Definizione:** I linguaggi contestuali hanno meno vincoli dei CFL. Vengono accettati da MdT su nastro lineare.

## Esempio:

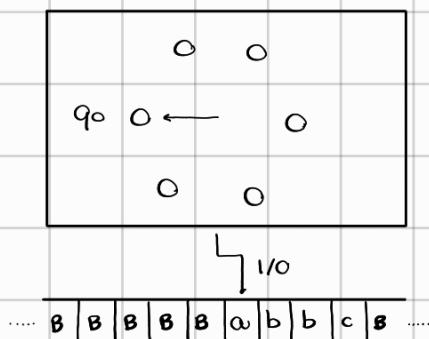
- $S \rightarrow aSBC$
  - $S \rightarrow aBC$
  - $CB \rightarrow BC$
  - $aB \rightarrow ab$
  - $bB \rightarrow bb$
  - $bC \rightarrow bc$
  - $cC \rightarrow cc$



# Linguaggi formali e Computabilità

## Ricorsivamente Enumerabili

O



$B$  = blank, logicamente vuoto

$S(q_0, a) = (p, b, L/R)$  con delta= funzione parziale (finisce quando è su stato non definito)

**Definizione MdT:** MdT è una settepla  $M = (Q, \Sigma, \Gamma, S, q_0, B, F)$  dove

- $Q$  è un insieme finito e non vuoto di **stringhe**
- $\Sigma$  è l'alfabeto di simboli di **input**
- $\Gamma$  è l'alfabeto di simboli del **mastro**. ( $B \in \Gamma \wedge B \notin \Sigma$ )
- $S: Q \times \Gamma \rightarrow Q \times \Sigma \times \{L, R\}$  è la **funzione parziale di transizione**
- $q_0 \in Q$  stato **iniziale**
- $B \in \Gamma$  simbolo di **blank**
- $F \subseteq Q$  stati **finali**

$L(M) = \{ w \in \Sigma^* \mid M \text{ accetta } w \}$

**Definizione T:** relazione binaria tra  $I \vdash J$ .  $x_1, x_2, \dots, x_{i-1}, q, x_i, x_{i+1}, \dots, x_n$

Suppongo che  $S(q, x_i) = (p, y, L)$ . Allora  $x_1, x_2, \dots, x_{i-1}, q, x_i, x_{i+1}, \dots, x_n \vdash x_1 x_2 \dots x_{i-1} p x_{i+1} y x_{i+2} \dots x_n$



# Linguaggi Formali e Computabilità

## Ricorsivamente Enumerabili

0

### Casi particolari:

- $i=1 \quad q \times_1 x_2 \dots x_n \vdash p \beta y_{x_2 \dots x_n}$
- $i=n, Y=B \quad x_1 \dots x_{n-1} q \times_n \vdash x_1 \dots x_{n-2} p x_{n-1}$
- $i=n \quad x_1 \dots x_{n-1} p x_n \vdash x_1 \dots x_{n-1} y_p B$
- $i=1, Y=B \quad q \times_1 \dots x_n \vdash p x_2 \dots x_n$

### Eseuzione MdT:

$$\delta(q_0, 0) = (q_3, x, R) \quad \delta(q_0, 1) = (q_3, 1, L)$$

$$\delta(q_3, 0) = (q_3, 0, R) \quad \delta(q_3, 1) = (q_3, 1, R) \quad \delta(q_3, B) = (q_2, Q, L)$$

$$\delta(q_2, 0) = (q_2, 0, L) \quad \delta(q_2, 1) = (q_2, 1, L) \quad \delta(q_2, x) = (q_3, B, R)$$

$$q_0 0 \xrightarrow{q_3} 0 + x q_3 \xrightarrow{q_3} 0 + x 0 \xrightarrow{q_3} q_3 0 + x 0 \xrightarrow{q_3} q_3 B + x 0 \xrightarrow{q_3} q_2 0 + x 0 q_2 \xrightarrow{q_2} 0 + x q_2 \xrightarrow{q_2} 0 + x q_3 \xrightarrow{q_3} 0 + x q_3 \xrightarrow{q_3} 0 +$$

$q_2 \times 0 \xrightarrow{q_2} 0 + q_3 0 \xrightarrow{q_3} 0$

**Accettazione MdT** Linguaggio accettato da MdT: Sia  $M$  una MdT, allora il ling accettato da  $M$  è  $L(M) = \{\omega \in \sum^* \mid q_0 \omega \xrightarrow{*} \alpha \beta \text{ dove } \alpha \in F, \epsilon \alpha, \beta \in \Gamma^*\}$

**Definizione ricorsivamente enumerabile:** se  $\exists$  una MdT  $M$  tc  $L = L(M)$ . Se un linguaggio è ricorsivamente enumerabile, esiste un algoritmo che elenca tutte le stringhe che ne appartengono, ma non in ordine, dunque non si può sapere in anticipo.

**Accettazione MdT:** Una MdT accetta per arresto, ovvero quando si trova in uno stato non definito da  $\delta$ .

**Non-accettazione MdT:** Se  $L \notin \text{MdT}$  allora possono verificarsi due eventi:

- si ferma in uno stato non accettante
- la macchina va in loop-infinito.



O

# Linguaggi Formali e Computabilità

## Ricorsivamente Enumerabili

**Linguaggi ricorsivi:** I linguaggi ricorsivi, che fanno parte dei RE, non vanno mai in loop infinito.



### Estensioni Macchine di Turing:

Possiamo ipotizzare lo stato stay ma non è utile, in quanto quello normale può simularlo con più passi:  $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\} = \delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, S, R\}$

MdT con due nastri avrà  $\delta(q, a, b) = (p, b, L, c, R)$  è come la macchina standard.

$\delta(q, a) = \{(p_1, b_1, L), \dots, (p_n, b_n, R)\}$ , è meglio della MdT normale? **No**, bastano due nastri.

**Teorema:** Ogni ling accettato da una MdT M<sub>2</sub> è accettato da una MdT M<sub>1</sub> in cui:

- La testina di M<sub>1</sub> non va mai su s<sub>0</sub> della pos iniziale
- M<sub>1</sub> non scrive mai un blank (B), lo denota con un altro simbolo.

**Macchine multi-stack:** DFA + K pile.  $w \in \Sigma^* \xrightarrow{\quad} \boxed{\cdot \cdot \cdot} \xleftarrow{\quad} \text{S}^i$



• 1 pila  $\rightarrow$  DFA

• 2 pile  $\rightarrow$  si simula MdT  $\rightarrow$  RE

• K pile  $\rightarrow$  RE

**Teorema:** Se utilizziamo le multi-stack dove le 2 pile indicano il nr di caratteri che contengono, dove le operazioni sono: **valore > 0 = 0**, **decremento**, **somma**; una macchina con 3 contatori lo simula e quindi accetta RE. Il primo contatore contiene il contenuto di una pila, il secondo un'altra pila e il terzo esegue le operazioni. Questo si può semplificare a sua volta con solo 2 contatori.



# Linguaggi formali e Computabilità

-1

## Capitolo 11: Indecidibilità

$$x \in N \rightarrow \underbrace{M}_{\epsilon N \rightarrow x_0} \longrightarrow y \in N$$

Enumerazione delle stringhe binarie:  $\epsilon, 0, 1, 00, 11, \dots \in \{0,1\}^*$  con posizioni, ha corrispondenza biunivoca:

se aggiungo  $\epsilon$  davanti alla stringa ottengo la posizione in binario.

Codifica di MdT:  $M \rightarrow \text{cod}(M) \in \{0,1\}^*$

$$M = (Q, \{0,1\}, \Gamma, \delta, q_1, B, \{q_2\})$$

$Q$  insieme degli stati dove  $Q = \{q_1: \text{iniziale}, q_2: \text{finale}, q_3, \dots, q_n\}$

$$\Gamma \{ x_1, x_2, x_3, \dots, x_n \}$$

$\begin{matrix} \downarrow & \downarrow & \downarrow \\ 0 & 1 & B \end{matrix}$

$$\delta(q_i, x_j) = (q_k, x_l, D_m)$$

$(i, j, k, l, m)$  memorizza solo gli indici

$0^i 1 0^j 1 0^k 1 0^l 1 0^m = c_1$  e separo le varie  $c_k$  con  $11: c_1 11 c_2 11 \dots 11 c_n \in \{0,1\}^*$

Ese:  $M = (\{q_1, q_2, q_3\}, \{0,1\}, \{0,1, B\}, \delta, q_1, B, \{q_2\})$  dove  $\delta$  è:

$$\begin{aligned} \delta(q_1, 1) &= (q_3, 0, R), & \delta(q_3, 0) &= (q_3, 0, R), & \delta(q_2, 1) &= (q_2, 0, R), & \delta(q_3, B) &= (q_3, 1, L) \\ 0 &\nearrow 1 \quad \nearrow 00 \quad \nearrow 1 \quad \nearrow 000 \quad \nearrow 1 \quad \nearrow 0 \quad \nearrow 1 \quad \nearrow 00 \quad \nearrow 11 & 1 & 1 & 1 & 1 & 1 \\ && & & & & \end{aligned}$$

Nb: Ad ogni stringa cui vengono associati gli indici possiamo associare una MdT

$$\epsilon, 0, 1, 00, 11, \dots \in \{0,1\}^*$$

$$1 \quad 2 \quad 3 \quad 4 \quad 5 \quad \dots \quad i$$

$$M_1 \quad M_2 \quad M_3 \quad M_4 \quad M_5 \quad \dots \quad \text{MdT } M$$

Possa dare una MdT che prende in input un'altra MdT.



# Linguaggi formali e Computabilità

-1

## Capitolo 11: Indecidibilità

Dimostrazione di esistenza di indecidibilità: Posso vedere una MdT come un dispositivo che prende  $x \in N$  e restituisce  $y \in N$ , ma le funzioni sono  $x_0^y$  dove  $|N|=x_0$  e ho solo  $x_0$  MdT.

	1	2	3	4	5	6	$\dots$
	$w_1$	$w_2$	$w_3$	$w_4$	$w_5$	$w_6$	
1	M <sub>1</sub>	0 1	1 0	1 0	1 0	1 0	
2	M <sub>2</sub>	1 0	0 1	0 1	0 1	0 1	
3	M <sub>3</sub>	0 0	0 0	0 0	0 0	0 0	
4	M <sub>4</sub>	0 1	0 1	0 1	0 1	0 1	
5	M <sub>5</sub>	1 1	1 1	1 1	1 1	1 1	
6	M <sub>6</sub>	0 1	0 1	0 1	0 1	0 1	
$\vdots$							

stringhe

$0 = \text{non accetta}$

$1 = \text{accetta}$

$111010$

MdT

Prendo la diagonale e la complemento. La complementazione è RE? Quero se esiste a un certo punto la riga uguale al complemento. NO, perché essendo complemento avrà sempre almeno un elemento opposto.

Def. Il complemento viene definito  $L_d$  ling. di diagonalizzazione.  $L_d = \{w_i \in \{0,1\}^* \mid w_i \notin L(i)\}$

Ricorsivi $L \cdot \bar{L}$	RE	$\neq$ RE
$\cdot L_d$		



# Linguaggi formali e Computabilità

-1

## Capitolo 11: Indecidibilità

### MdT universale:

La  $M_u$  prende in input la codifica di una MdT  $M$  (quindi stringa binaria) e poi prende una stringa  $w$  da dare in input a  $M$  e simula  $M$  sulla stringa  $w$ . Dato che entrambe sono sullo stesso nastro (il primo) per sapere quando finisce  $M$  e inizia  $w$  si mette in mezzo "###" mai presente su  $\text{cod}(M)$ .

Di conseguenza si verificano 3 casi:

- se  $w \notin \text{cod}(M)$ , allora  $L_u$  rifiuta.
- se  $w \in \text{cod}(M)$ , allora  $L_u$  accetta.
- se  $M$  va in loop infinito, anche  $L_u$ .

Scrive  $w$  sul secondo nastro e lo stato della macchina sul terzo nastro.  $L_u$  dopo cerca nella codifica di  $M$  la funzione delta indicizzata dallo stato (3<sup>o</sup> nastro) e dal simbolo di input sul 2<sup>o</sup> nastro. Fa il caso della delta e sposta le testine, usando aux per conti. Continua fino a quando il caso della delta è definito.

### Linguaggio Universale

Il linguaggio accettato da  $M_u$  è  $L_u$  ovvero l'insieme delle stringhe binarie che rappresentano le coppie  $M_j, w_i$ , dove  $M_j$  è MdT generica e  $w_i$  stringa, t.c.  $w_i \in L(M_j)$ .

$$L_u = \{(M_j, w_i) \mid w_i \in L(M_j)\}$$

Essendo  $L_u$  accettato da MdT (avendo la  $M_u$ ), allora anche  $M_u$  può essere codificato in binario e quindi compare nelle righe della matrice  $M_j, w_i$ .  $L_u$  dunque è RE/Ric.

Si dimostra che non è ricorsivo in quanto se  $M$  va in loop infinito, anche  $M_u$  andrà.

Non si può sapere in anticipo se  $M$  andrà in loop infinito o terminerà rifiutando Holting.

Problema: problema indecidibile. Stesso discorso RE, altro problema indecidibile: RE enumera le stringhe del linguaggio. Se una stringa non è ancora stampata, non possiamo sapere in anticipo se verrà stampata o meno.



# Linguaggi formali e Computabilità

1

## Capitolo 11: Indecidibilità

### Appartenenza L e $\bar{L}$ :

Considero il caso in cui  $L \in \text{RE/Ric}$ , allora  $\exists \text{MdT}$  tc:

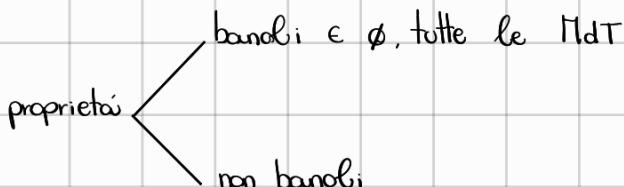
$$w \rightarrow \boxed{\text{M}b} \begin{cases} \text{si se } w \in L \\ \text{no. no se } w \notin L \end{cases}$$

Allora il suo complemento si comporta allo stesso modo. Ma se prendo una MdT che simula le due MdT che combinano le due MdT, se viene accettata  $w$ ,  $\bar{w}$  non viene accettata, ma non troviamo il caso in cui va in loop accetta ricorsiva. Dunque  $L$  e  $\bar{L}$  sono entrambi Ric o uno RE e uno  $\neq$  RE. (si dimostra col complemento di  $L$ ) oppure entrambi fuori.

Def:  $L_e = \{M \mid L(M) = \emptyset\}$  empty  $\in \text{non RE}$

$L_{ne} = \{M \mid L(M) \neq \emptyset\}$  non-empty  $\in \text{RE/Ric}$

Def: Insieme di MdT = proprietà di MdT.



Teorema di Rice: Ogni proprietà non banale è indecidibile o semi-indecidibile (RE/Ric o  $\neq$  RE).