



Linguaggi formali e Computabilità

Riassunto Concetti

Alfabeto: Insieme finito di caratteri. Si denota con Σ (sigma).

Stringa: Sequenza finita di caratteri presi da un alfabeto.

• **Lunghezza:** n° posizioni presenti nella stringa.

• **Stringa vuota:** Si denota con ϵ (epsilon minuscola).

• **Alfabeto infinito:** Insieme di tutte le stringhe. Si denota con Σ^* .

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: ho 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.

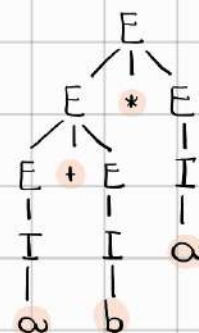
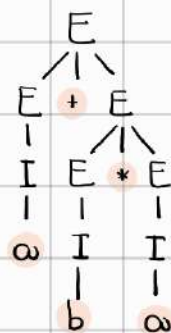
• **P** insieme produzioni.

• **S** start symbol.

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

Es. $E \rightarrow I \mid E + E \mid E * E$

$I \rightarrow a \mid b$



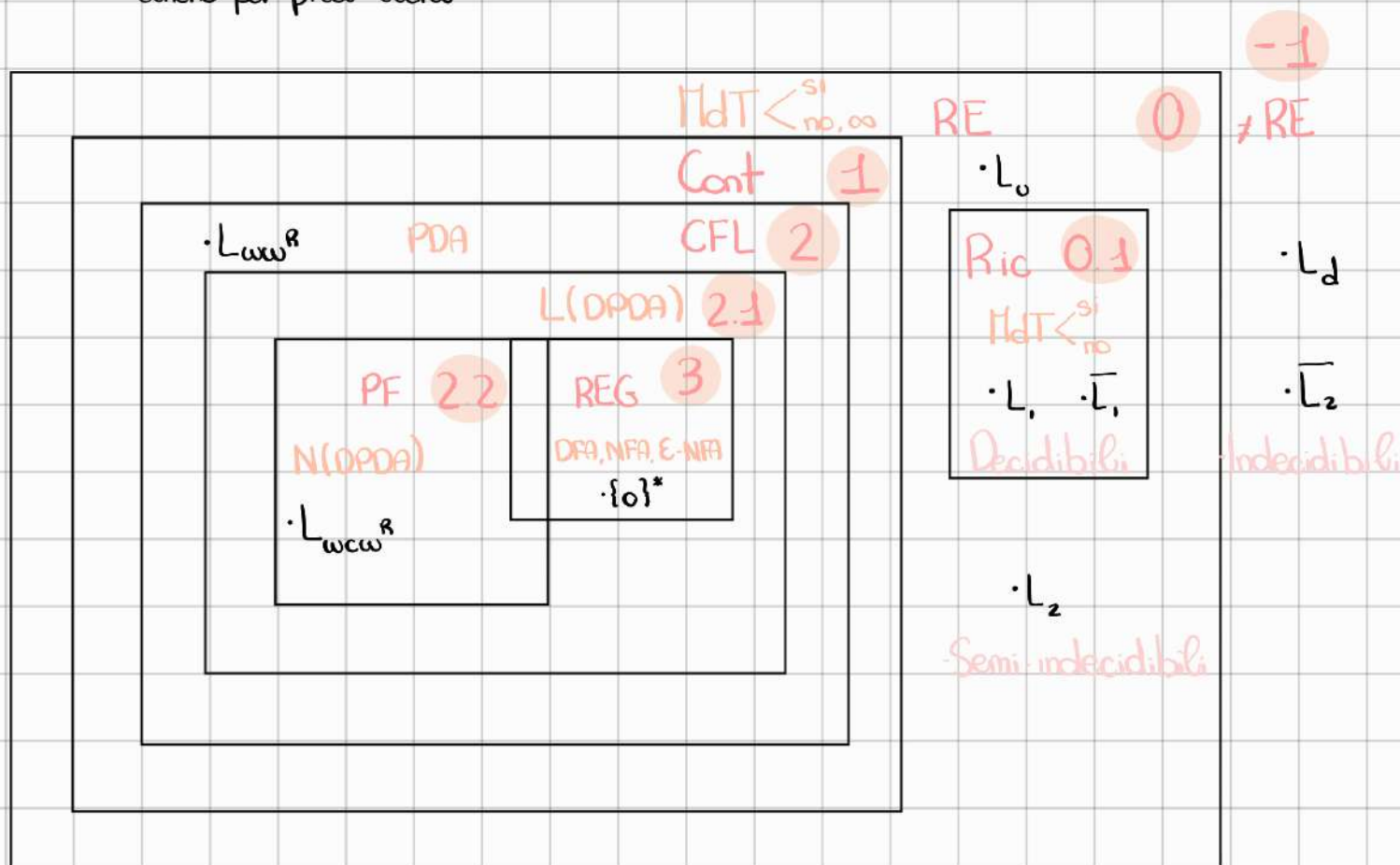


Linguaggi formali e Computabilità

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 a Pila Non Deterministici (PDA).
- Tipo 3: Linguaggi Regolari (REG). Sono riconosciuti da automi a stati finiti (DFA, NFA, E-NFA).
- Tipo 4: 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 accetta non va mai in loop infinito.
- Tipo 2.1: Linguaggi accettati da automi a pila deterministico per stati finali.
- Tipo 2.2: Linguaggi Prefix-free (PF) Sono accettati da automi a pila deterministico anche per pila vuota.





Linguaggi formali e Computabilità

3

Linguaggi Regolari

Linguaggi regolari: I linguaggi regolari sono:

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

Generazione

Vincoli

- ϵ : compare solo in $S \rightarrow \epsilon$ sse S start-symbol.
- **Lineari**: regole di produzione tutte lineari a dx o a sx

Esempi

- $L = \{a^n b^m \mid n, m \geq 0\}$ lineare a dx
 $S \rightarrow \epsilon \mid b \mid aS \mid bB$
 $B \rightarrow b \mid bB$

- $L = \{a b^n c d^m e \mid n \geq 0, m \geq 0\}$ lineare a sx
 $S \rightarrow X e$
 $X \rightarrow Y d \mid X d$
 $Y \rightarrow Z c$
 $Z \rightarrow a \mid Z b$



Linguaggi formali e Computabilità

3

Linguaggi Regolari

Dimostrazione

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

Definizione ricorsiva:

caso base

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

caso ricorsivo

- $L(E + F) = L(E) \cup L(F)$ (commutativa, associativa, identità (\emptyset), idempotenza ($L + L$), distributiva)
- $L(E \cdot F) = L(E) \cdot L(F)$ (associativa, identità (ϵ), annichilatore (\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.

• $ER: ((01)^* 10(0+1)^*)^*$ dire quali stringhe $w_i \in L$.

• $w_1: 0101$ **no**, genero 01 1 volta con $(01)^*$ ma poi devo generare 10 ma ho 01.

• $w_2: 10111$ **si**, non genero $(01)^*$, trovo poi 10 e poi genero 3 volte 1 con $(0+1)^*$

• $w_3: 01011010101$ **si**, genero 2 volte 01 con $(01)^*$, trovo 10 e poi qualsiasi cosa è ok.



Linguaggi formali e Computabilità

3

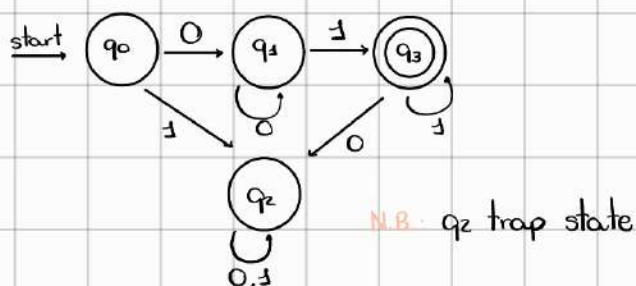
Linguaggi Regolari

Accettazione

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

- Q insieme degli stati
- Σ insieme delle stringhe in input
- δ : 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 $L = \{0^n 1^m \mid n > 0, m > 0\}$



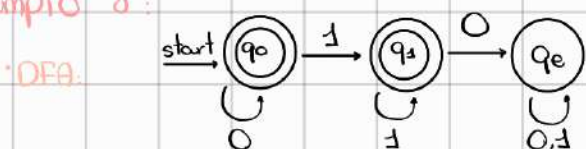
Definizione $\hat{\delta}$: Dato un DFA, il linguaggio accettato è $L(A) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}$

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

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

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

Esempio $\hat{\delta}$:



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



Linguaggi formali e Computabilità

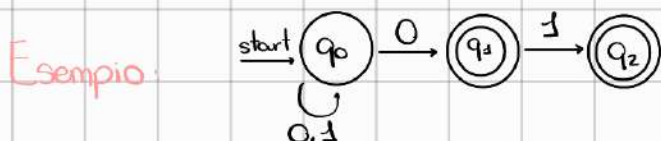
3

Linguaggi Regolari

Accettazione

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

- Q insieme degli stati
- Σ insieme delle stringhe in input
- δ : funzione di transizione degli stati: $\delta: Q \times \Sigma \rightarrow 2^Q$ *$P(a)$ insieme parti Q . parentale*
- $q_0 \in Q$: stato iniziale
- $F \subseteq Q$: insieme degli stati finali.

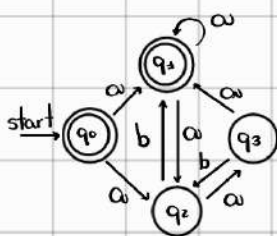


Trasformazione da NFA a DFA: trascrivo nella tabella solo gli stati necessari

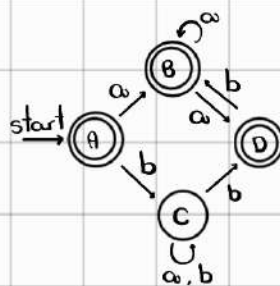
• NFA:

• Tabella stati:

• DFA:



		a	b
*	A {q0}	B {q1, q2}	C ∅
*	B {q1, q2}	B {q1, q2}	D {q2, q3}
	C ∅	C ∅	C ∅
*	D {q2, q3}	B {q1, q2}	C ∅



Definizione E-NFA: come NFA ma δ cambia in quanto possiamo muoverci senza consumare mosse: *ϵ -mosse.*

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



Linguaggi formali e Computabilità

3

Linguaggi Regolari

Accettazione

Definizione ricorsiva $\text{eclose}(q)$:

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

Definizione ricorsiva $\hat{S}: 2^Q \times \Sigma^* \rightarrow 2^Q$

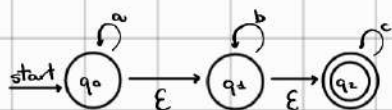
- **caso base**: $|w|=0, \hat{S}(S, \epsilon) = \text{ECLOSE}(S)$
- **caso passo**: $|w|>0, w = ax, a \in \Sigma, x \in \Sigma^*, \hat{S}(S, ax) = \hat{S}(\hat{S}(\text{ECLOSE}(S), a), x)$

Trasformazione da E-NFA a DFA

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

- $q_d(\text{ECLOSE}_n)$
- $F_d = \{s \in Q_d \mid s \cap F_0 \neq \emptyset\}$
- $\forall a \in \Sigma \text{ e } \forall S = \{p_1, p_2, \dots, p_k\} \in Q_d \text{ e } s \in Q_d,$
 $\delta_d(S, a) = \text{ECLOSE}(R) \text{ con } R = \{r_1, r_2, \dots, r_m\} = \bigcup_{i=1}^k \delta_e(p_i, a)$

• E-NFA



• ECLOSE

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

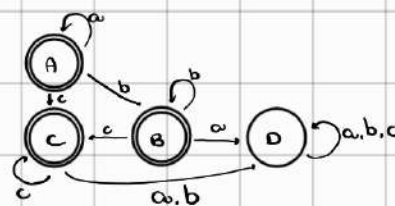
$$\text{ECLOSE}(q_1) = \{q_1, q_2\}$$

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

• Tabella stati

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

• DFA



- $\hat{S}: \delta_d(\{q_0, q_1, q_2\}, a) = \text{ECLOSE}(\delta_e(q_0, a) \cup \delta_e(q_1, a) \cup \delta_e(q_2, a)) = \text{ECLOSE}(\{q_0\} \cup \emptyset \cup \emptyset)$
 $= \text{ECLOSE}(\{q_0\}) = \text{ECLOSE}(q_0) = \{q_0, q_1, q_2\}$



Linguaggi formali e Computabilità

Linguaggi Regolari

3

Accettazione

Dav ER a E-NFA

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

Dav ER a E-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'E-NFA per S vale anche per R.

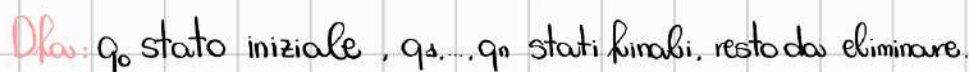
Esempio ER \rightarrow E-NFA

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

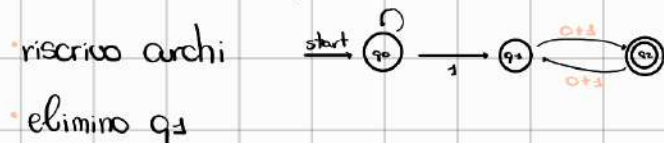




Trasformazione da DFA a ER: Casi base



Esempio DFA a ER



precedence: q_0, q_2 ER: $q_0 \rightarrow q_3: 1$ $q_2 \rightarrow q_3: 0 + 1$

* Successori: q_2 ER: $q_1 \rightarrow q_2: 0+1$

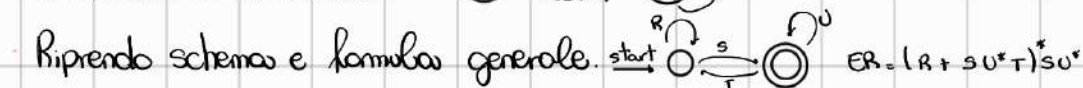
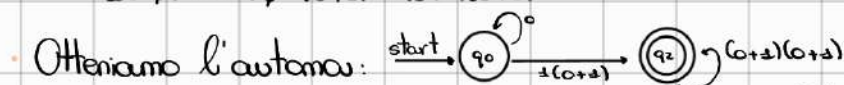
no self-loop $ER: \emptyset$

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

- Inseriamo passaggi con formule: passaggio diretto + $x \rightarrow y, y^2, y \rightarrow z$ dove R_{xz} e y da eliminare.

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

$$R_{22} = \phi + (0+3)\phi^*(0+3) = (0+3)(0+3)$$



Riservo las formulas:
$$ER = (0 + \underbrace{1}_{\text{S}})(\underbrace{(0 + \underbrace{1}_{\text{S}})}_{\text{U}})(\underbrace{(0 + \underbrace{1}_{\text{S}})}_{\text{U}})^* \underbrace{0}_{\text{T}} \underbrace{(0 + \underbrace{1}_{\text{S}})}_{\text{U}}(\underbrace{(0 + \underbrace{1}_{\text{S}})}_{\text{U}})^*$$

$$= 0^* \underbrace{1}_{\text{S}}(\underbrace{(0 + \underbrace{1}_{\text{S}})}_{\text{U}})(\underbrace{(0 + \underbrace{1}_{\text{S}})}_{\text{U}})^*$$



Linguaggi formali e Computabilità

3

Linguaggi Regolari

Accettazione

Algoritmo riempi-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.

DFA:

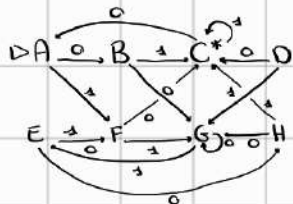


Tabella:

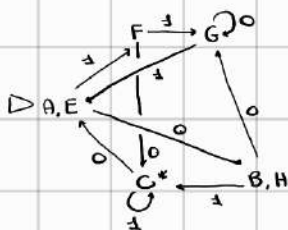
B	X						
C	X	X					
D	X	X	X				
E		X	X	X			
F	X	X	X		X		
G	X	X	X	X	X	X	
H	X		X	X	X	X	X
	A	B	C	D	E	F	G

Legenda:

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

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

DFA minimizzato:





Linguaggi formali e Computabilità

3

Linguaggi Regolari

Accettazione

Pumping lemma: si usa per dimostrare che un linguaggio regolare.

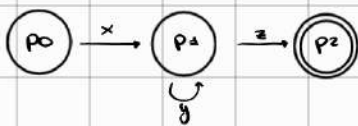
Sia L un linguaggio regolare. \exists una costante n (dipendente da L) t.c. $\forall w \in L$ con $|w| \geq n$

w può essere scomposto come $w = xyz$ t.c.:

- $y \in \Sigma$
- $|xy| \leq n$
- $\forall k \geq 0$, 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 $w = a_1 a_2 \dots a_m$ con $m \geq n$. $\forall i = 0, 1, \dots, n$ sia $p_i = \hat{\delta}(q_0, a_1 \dots a_i)$ con $p_0 = q_0$. p_0, p_1, \dots, p_n sarebbero $n+1$ stati, allora $\exists i, j$ t.c. $0 \leq i < j \leq n$ t.c. $p_i = p_j$. Scomponiamo allora $w = xyz$ come segue.

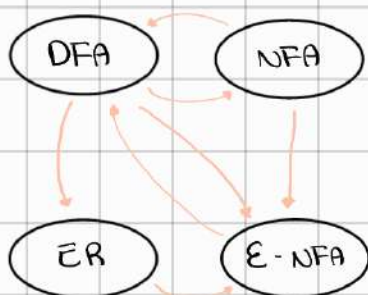
- $x = a_1 \dots a_i$
- $y = a_{i+1} \dots a_j$
- $z = a_{j+1} \dots a_m$



Esempio: Dim che $L_{01} = \{x^n y^n \mid n \geq 1\}$ non è regolare. Suppongo lo sia, allora $n \in \mathbb{N}$ la costante di PL. Sia $w = 0^n 1^n \in L_{01}$. Scriviamo $w = xyz$:

$x = 0^{n-1}$ $y = 0$ $z = 1^n$ allora per PL deve valere $xy^k z \in L_{01}, \forall k \geq 0$. Ma per $k=0$, $xz \notin L_{01}$.

Schemi 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 pila non deterministici

Generazione

Regole: Due modi per costruire un sottoinsieme.

- concatenare
- contenitore

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

Derivazione: La derivazione avviene con *leftmost* o *rightmost*

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

- $S \rightarrow aSd \mid aCd$
- $C \rightarrow \epsilon \mid bCc$
- $S \Rightarrow aCd \Rightarrow ad$
- $S \Rightarrow aSd \Rightarrow aaCdd \Rightarrow aadd$

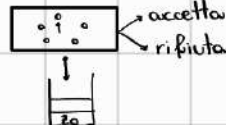


Linguaggi formali e Computabilità

Context free

2

Accettazione

Automato a pila: E-NFA + stack. $w \in \Sigma^*$ → 

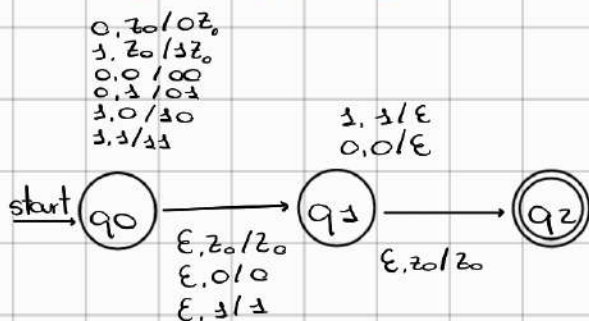
Definizione PDA: È una settupla $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

- Q è insieme finito e non vuoto di stati.
- Σ : alfabeto di simboli in input.
- Γ : 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: $L_{ww^r} = \{ww^r \mid w \in \{0,1\}^*\}$ palindromo pari CFG: $S \rightarrow 0S0 \mid 1S1 \mid \epsilon$

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

$P = (\underbrace{\{q_0, q_1, q_2\}}_Q, \underbrace{\{0,1\}}_\Sigma, \underbrace{\{0,1,z_0\}}_\Gamma, \underbrace{\delta}_{\delta}, \underbrace{q_0}_{q_0}, \underbrace{z_0}_{z_0}, \underbrace{\{q_2\}}_F)$



Descrizione istantanea (10): È una tripla (q, w, s) dove:

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



Linguaggi formali e Computabilità

Context free

2

Accettazione

Definizione mosca: Sia $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ un PDA e supponiamo che $(p, \alpha) \in \delta(q, w, x)$
 Allora $\forall w \in \Sigma^* \text{ e } \forall \beta \in \Gamma^*, (q, w, x\beta) \vdash_p (p, w, \alpha\beta)$.

Definizione ricorsiva: \vdash_p^*

base: $I \vdash^* \forall ID I$

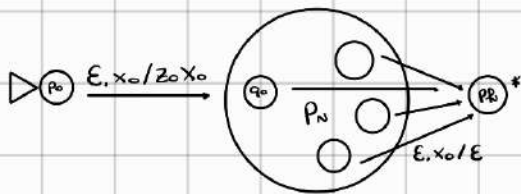
passo: $I \vdash^* J \exists ID K \text{ t.c. } I \vdash K, K \vdash^* J$.

Linguaggi accettati da PDA

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

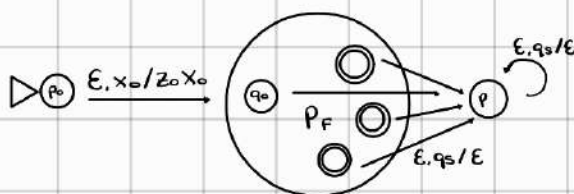
Pila vuota: Sia $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$, P accetta per pila vuota il linguaggio
 $N(P) = \{w \in \Sigma^* \mid (q_0, w, z_0) \vdash_p^* (q, \epsilon, \epsilon) \text{ con } q \in Q\}$

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



$P_F = (Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{x_0\}, \delta_F, p_0, x_0, \{p_f\})$
 dove $\delta_F = \delta_N \cup \delta$ di p_0 e δ di tutti i passi per p_f .

Da stati finali a pila vuota: se $L = L(P_F)$ per un PDA allora \exists PDA P_N t.c. $L = N(P_N)$



$P_N = (Q \cup \{p_0, p\}, \Sigma, \Gamma \cup \{x_0\}, \delta_N, p_0, x_0)$
 dove q_s è \forall simbolo di $\Gamma \cup \{x_0\}$



Linguaggi formali e Computabilità Contestabili

1

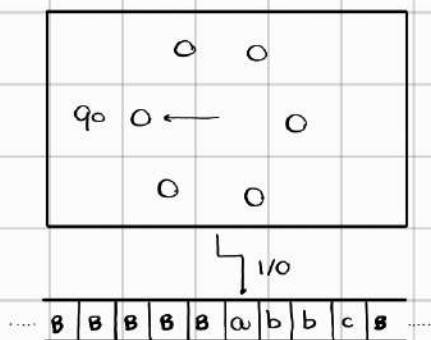
Definizione: I linguaggi contestabili hanno meno vincoli dei CFL. Vengono accettati da MdT a nastro lineare.

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



B = blank, logicamente vuoto

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

Definizione MdT: MdT è una settopla $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ dove

- Q è un insieme finito e non vuoto di **stringhe**
- Σ è l'alfabeto di simboli di **input**
- Γ è l'alfabeto di simboli del **nastro**. ($B \in \Gamma \wedge B \notin \Sigma$)
- $\delta: Q \times \Gamma \rightarrow Q \times \Sigma \times \{L, R\}$ 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 I: relazione binaria tra $ID \vdash J$. $x_1, x_2, \dots, x_{i-1}, q, x_i, x_{i+1}, \dots, x_n$

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



• la macchina va in loop-infinito.



Linguaggi formali e Computabilità

Ricorsivamente Enumerabili

0

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



Estensioni Macchine di Turing:

- Posso 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 rz è accettato da una MdT rz in cui:

- La testina di M1 non va mai a sx della posizione iniziale
- M1 non scrive mai un blank (B), lo denota con un altro simbolo.

Macchine multi-stack: DFA + K pile. $w \in \Sigma^* \rightarrow \begin{matrix} \boxed{\begin{matrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{matrix}} < \begin{matrix} S_i \\ N_0 \end{matrix}$

- 1 pile \rightarrow DPDA
- 2 pile \rightarrow si simula MdT \rightarrow RE
- K pile \rightarrow RE

Teorema: Se utilizziamo le multi-stack dove le 2 pile indicano il n° 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 pile, il secondo un'altra pile e il terzo esegue le operazioni. Questo si può semplificare a sua volta con solo 2 contatori.



Linguaggi formali e Computabilità

Capitolo 11: Indecidibilità

1

$$x \in \mathbb{N} \rightarrow \underbrace{\boxed{M}}_{\in \mathbb{N} \rightarrow \mathbb{N}} \rightarrow y \in \mathbb{N}$$

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

se aggiungo 1 davanti alla stringa ottengo la posizione in binario.

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

$$M = (Q, \{0, 1\}, \Gamma, \delta, q_s, B, \{q_e\})$$

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

$$\Gamma = \left\{ \begin{array}{c} x_1, x_2, x_3, \dots, x_n \\ \downarrow \quad \downarrow \quad \downarrow \\ 0 \quad 1 \quad B \end{array} \right\}$$

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

$$\hookrightarrow (i, j, k, l, m) \quad \text{memorizzo solo gli indici}$$

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

Es. $M = (\{q_s, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_s, B, \{q_e\})$ dove δ è:

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

$$0 \quad 1 \quad 00 \quad 1 \quad 000 \quad 1 \quad 0 \quad 1 \quad 00 \quad 11 \quad \dots \quad 11 \quad \dots \quad 11 \quad \dots$$

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

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

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

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

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



Linguaggi formali e Computabilità

Capitolo 31: Indecidibilità

1

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

	1	2	3	4	5	6	...
	w_1	w_2	w_3	w_4	w_5	w_6	...
1 Π_1	0	1	1	0	1	0	
2 Π_2	1	0	0	1	0	0	
3 Π_3	0	0	0	0	0	0	
4 Π_4	0	1	0	1	0	0	
5 Π_5	1	1	1	0	0	1	
6 Π_6	0	1	0	1	0	1	

stringhe

0 = non accettato

1 = accettato

111010

MdT

Prendo la diagonale e la complemento. La complementazione è RE? Ovvero 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 **Ld** ling. di diagonalizzazione. $L_d = \{w_i \in \{0,1\}^* \mid w_i \notin L(\Pi_i)\}$

Ricorsivi	RE	\neq RE
L, \bar{L}	L_u	L_d



Linguaggi formali e Computabilità

1

Capitolo 31: 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 "111" 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 indicizzato dallo stato (3° nastro) e dal simbolo di input sul 2° nastro. Fa il caso della delta e sposta le testine, usando aux per i 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, tale $w_i \in L(M_j)$.

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

Essendo L_u accettato da MdT (ovvero da 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: Halting

Problem: 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 31: Indecidibilità

Appartenenza L e \bar{L} :

Considero il caso in cui $L \in RE/Ric$, allora \exists MdT t_L :

$w \rightarrow \boxed{t_L} \begin{cases} \text{si se } w \in L \\ \text{no, os se } w \notin L \end{cases}$

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 accettato w , \bar{w} non viene accettato, ma non troviamo il caso in cui va in loop, accetta ricorsivo. 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 non RE

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

Def: Insieme di MdT = proprietà di MdT.

proprietà $\begin{cases} \text{banali } \in \emptyset, \text{ tutte le MdT} \\ \text{non banali} \end{cases}$

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