Linguaggi e Computabilità

Daniele De Micheli

2019

Indice

Parte I
Prima Parte

1 Alfabeti e Linguaggi

Si definisce **alfabeto** un insieme finito e non vuoto di simboli. Ad esempio, l'alfabeto $\{A, B, C, ..., Z\}$ potremmo definirlo come l'insieme delle lettere maiuscole dell'alfabeto italiano. Altri esempi intuitivi sono l'inseme delle cifre che compongono i numeri arabi $\{1, 2, 3, ...9, 0\}$ o l'insieme $\{0, 1\}$ che rappresenta i numeri binari. Un alfabeto si indica con una lettera maiuscola greca:

- $A = \{A, B, C, ..., Z\}$;

- $\Gamma = \{0, 1\}$;

Si definisce invece una **stringa** un insieme vuoto, finito o infinito di simboli presi da un dato alfabeto. Una stringa vuota si indica con $\epsilon o \lambda$.

1.1 Stringhe

Una stringa, come abbiamo già visto, si rappresenta con una lettera greca maiuscola. Nel caso volessimo indicare invece la lunghezza di una stringa bisogna utilizzare la seguente notazione:

$$|\Gamma| = x$$

dove Γ rappresenta la stringa e x la sua lunghezza.

Le stringhe possono inoltre essere "manipolate", o meglio esse si possono *concatenare* per ottenere una nuova stringa. Tale operazione si può indicare così: $\Gamma \circ A$ e si legge "Γ *concatenata ad* A". La concatenazione **non** è un'operazione commutativa. Infatti

$$\Gamma \circ A \neq A \circ \Gamma$$

Se prendiamo ad esempio $A = \{010\}$ e $\Gamma = \{11\}$, allora $A \circ \Gamma = \{01011\}$ mentre $\Gamma \circ A = \{11010\}$ che non sono assolutamente uguali come si può ben vedere.

1.2 Alfabeti

Gli alfabeti, come abbiamo già visto, sono insieme finiti di simboli. Su tali insiemi è possibile definire delle operazioni che generano delle stringhe a partire dall'alfabeto stesso.

Potenza di un alfabeto : dato un alfabeto E, si definisce *potenza di* E la stringa di lunghezza k contenente tutti gli elementi dell'alfabeto E.

$$dato\ E, k \geq 0 \in \mathbb{Z}, \Rightarrow E^k = E \times E \times E \times E \times ... \times E, k\ volte$$

Se $|E| = q$, allora $|E^k| = q^k$. Ad esempio, prendiamo l'alfabeto $E = \{0, 1\}$. Allora:

- $E^1 = \{0, 1\}$
- $E^2 = \{00, 01, 10, 11\}$

- $E^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

Come si può intuire dall'esempio qui sopra, il risultato della potenza di un alfabeto non è altro che l'insieme delle *combinazioni* di numero k degli elementi dell'alfabeto.

Chiusura di Kleene : la chiusura di Kleene è un'operazione che riguarda le potenze di un alfabeto. Infatti tale operazione non è altro che l'unione di tutte le potenze di un alfabeto fino a k. Formalmente:

$$E^\star = \cup E^k = E^0 \cup E^1 \cup E^2 ... \cup E^k,\ t.c.\ k \geq 0$$

Un'operazione derivata da quest'ultima è la chiusura di Kleene ma senza l'elemento 0:

$$E^+ = E^\star \setminus E^0$$

1.3 Linguaggio

Possiamo definire un *linguaggio L su E* un sottoinsieme di $E^\star$ tale che $L \subseteq E^\star$. Per esempio, preso $E = \{a, b, c\}$, un linguaggio L potrebbe essere $L_1 = \{aa, cbc\}$. Un linguaggio può essere finito (vedi L_1), oppure infiniti (es. $L_2 = \{w \in E^\star \mid w\ contiene\ lo\ stesso\ numero\ di\ a\ e\ c\}$).

Preso un linguaggio $L \subseteq E^\star$, possiamo affermare che:

1. $\emptyset \subseteq L$;
2. $\varepsilon \subseteq L$;
3. $E^\star \subseteq L$;

sono tutti linguaggi. La principale caratteristica di un linguaggio è che esso deve essere riconosciuto e interpretato da una macchina (o automa) ed essa deve anche essere in grado di generarlo tramite una ***grammatica***.

Problema di Decisione. *Il problema di decisione si presenta nel momento in cui, dato un quesito, le possibili risposte sono sempre e sole "sì" o "no".*

Problema di Membership. *Il problema di Memebership è legato al concetto di stringa (come input), di linguaggio e di appartenenza ad un determinato linguaggio. Data una stringa w in input, una determinata macchina deve essere in grado di dire se essa appartiene ad un linguaggio oppure no.*

DEFINIZIONI Una **forma sentenziale** è una stringa di simboli terminali e non terminali: $\gamma \in (V \cup T)^\star$

Concatenazione di linguaggi : Dati due linguaggi $L_1, L_2 \subseteq E^\star$ allora

$$L_1 \circ L_2 = \{w | w = w_1 \circ w_2, w_1 \in L_1, w_2 \in L_2\}$$

1.4 Grammatiche

La descrizione grammaticale di un linguaggio consiste di quattro componenti:

1. Un insieme finito di simboli che formano le stringhe del linguaggio. Questi sono chiamati *terminali* o *simboli terminali*.

2. Un insieme finito di *variabili*, dette anche *non terminali* o *categorie sintattiche*. Ognuna di esse rappresenta un linguaggio.

3. Una variabile, chiamata *simbolo iniziale*, che rappresenta il liunguaggio da definire.

4. Un insieme finito di *produzioni*, o *regole*, che rappresentano la definizione ricorsiva di un linguaggio. Ogni produzione consiste di tre parti:

   - Una variabile che viene definita dalla produzione ed è spesso detta la **testa** della produzione.
   - Il simbolo di produzione $\rightarrow$.
   - Una stringa di zero o più terminali e variabili. detta il **corpo** della produzione, il quale rappresenta un modo di formare le stringhe nel linguaggo della variabile di testa.

1.4.1 Grammatica libere dal contesto -CFG-

Una grammatica context-free è una grammatica che non prevede l'incrocio dei simboli terminali per cui è necessario utilizzare delle regole differenti. Possiamo rappresentare una CFG per mezzo dei quattro componenti descritti sopra, ossia $G = (V, T, P, S)$, dove V è l'insieme delle variabili, T i terminali, P l'insieme delle produzioni ed S il simbolo iniziale.

Stringhe palindrome : le stringhe palindrome sono un esempio semplice di linguaggio che utilizza una grammatica context-free. Abbiamo il l'alfabeto $E = \{0, 1\}$ e il linguaggio costruito su esso $L_{pal} \subseteq E^\star$. Da questo alfabeto e con questo linguaggio possiamo costruire una stringa w palondroma come

$$w = \{0110\}$$

Essa può essere definita per induzione come segue:

1. Passo base: $\varepsilon, 0, 1 \in L_{pal}$
2. Passo induttivo: se $w \in L_{pal}$, allora $0w0, 1w1, \varepsilon \in L_{pal}$

Un esempio di regole del linguaggio possono essere:

$$\begin{aligned} S &\to \varepsilon \\ S &\to 0 \\ S &\to 1 \\ S &\to 0S0 \\ S &\to 1S1 \end{aligned} \tag{1}$$

in cui la **testa** può essere sostituita dal **corpo**.

Queste regole possono essere applicate tramite le due *relazioni*:

1. $\Rightarrow$
2. $\Rightarrow^\star$

La prima (1) possiamo definirla come segue:

Prima relazione. *Sia* $G = (V, T, P, S)$ *una CFG e sia* $\alpha A \beta$ *tale che* $\alpha, \beta \in (V \cup T)^\star$ *e* $A \in V$. *Sia* $A \to \gamma \in P$. *Allora* $\alpha A \beta \Rightarrow \alpha \gamma \beta$.

Seconda relazione. *Si ha che* $\alpha \Rightarrow^\star \beta$, *con* $\alpha, \beta \in (V \cup T)^\star$, *se e solo se* $\exists\, \gamma_1, \gamma_2, ..., \gamma_n \in (V \cup T)^\star$ *tale che* $\alpha \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow \gamma_3 \Rightarrow ... \Rightarrow \gamma_n \Rightarrow \beta$ *con* $n \geq 1$. *Se* $n = 1$, *allora* $\alpha = \beta$ *e vale* $\alpha \Rightarrow^\star \beta$ *cioè* $\alpha \Rightarrow^\star \gamma$.

Le produzioni di una CFG si applicano per dedurre se una stringa può appartenere al linguaggio oppure no. Per fare questo si possono utilizzare due metodi differenti. Prese le seguenti regole

1. $E \to I$
2. $E \to E + E$
3. $E \to E * E$
4. $E \to (E)$
5. $I \to a$
6. $I \to b$
7. $I \to Ia$
8. $I \to Ib$
9. $I \to I0$
10. $I \to I1$

e questa stringa $w = a * (a + b00)$. Possiamo a questo punto intraprendere due strade per affermare se la stringa w appartiene al linguaggio, quella della *inferenza ricorsiva* o quella della *derivazione*. La prima funzina così: creo una tabella in cui in ogni passaggio applico una regola specificando per quale linguaggio e quali altre precedenti stringhe già ottenute sto riutilizzando. L'altro metodo, quello per *derivazione* prevede invece l'uso delle *relazioni* viste nel paragrafo (1.4.1). La stessa stringa è quindi ottenuta nel seguente modo:

$$E \Rightarrow_{lm} E * E \Rightarrow_{lm} I * E \Rightarrow_{lm} a * E \Rightarrow_{lm}$$
$$\Rightarrow_{lm} a * (E) \Rightarrow_{lm} a * (E + E) \Rightarrow_{lm} a * (I + E) \Rightarrow_{lm} a * (a + E) \Rightarrow_{lm}$$
$$\Rightarrow_{lm} a * (a + I) \Rightarrow_{lm} a * (a + I0) \Rightarrow_{lm} a * (a + I00) \Rightarrow_{lm} a * (a + b00)$$

Per ricavare la stringa abbiamo utilizzato una derivazione sinistra, ma avremmo potuto eseguire la derivazione destra e il risultato sarebbe stato il medesimo. Infatti, *qualunque derivazione ha una derivazione a sinistra e una a destra* ***equivalenti***.

Tabella 1: Inferenza Ricorsiva

n	Stringa ricavata	Linguaggio	Produzione impiegata	Stringhe impiegate
(i)	a	I	5	–
(ii)	b	I	6	–
(iii)	b0	I	9	(ii)
(iv)	b00	I	9	(iii)
(v)	a	E	1	(i)
(vi)	b00	E	1	(iv)
(vii)	a + b00	E	2	(v),(vi)
(viii)	(a + b00)	E	4	(vii)
(ix)	a * (a + b00)	E	3	(v),(viii)

1.4.2 Grammatica NON context-free

Il linguaggio di esempio (di tipo 2): $L = \{w \in \{a, b, c\}^\star | w = a^n b^n c^n, n \geq 1\}$ è generato dalla seguente grammatica (NON context-free):

$$G = (\{S, X, B, C\}, \{a, b, c\}, P, S)$$

e dove le regole di produzione sono:

1. $S \to aSBC$
2. $S \to aBC$
3. $CB \to XB$
4. $XB \to XC$
5. $XC \to BC$
6. $aB \to ab$
7. $bB \to bb$
8. $bC \to bc$
9. $cC \to cc$

Le grammatiche 3,4,5 possono essere "collassate" in $CB \to BC$ Si può dimostrare , usando il Pumping Lemma per i CFL, che non è context-free.

Esempio di Derivazione:

Deriviamo la stringa abc (corrispondente a n = 1), indicando anche ad ogni passo la regola usata.

$$S(2) \to aBC(6) \to abC(8) \to abc$$

Deriviamo la stringa aabbcc (corrispondente a n = 1), indicando anche ad ogni passo la regola usata.

$$S(1) \to aSBC(2) \to aaBCBC(3) \to aaBXBC(4) \to aaBXCC(5) \to$$

$$\to aaBBCC(6) \to aabBCC(7) \to aabbCC(8) \to aabbcC(9) \to aabbcc$$

In generale, per derivare $a^n b^n c^n$, per n < 1:

$$S(n-1\ volte \to (1))a^{n-1}S(BC)^{n-1} \to (2)a^n(BC)^n(n(n-1)/2\ volte\ la$$

$$sequenza \to (3), \to (4), \to (5))a^n B^n C^n ....slide$$

Esercizio: creo una CFG su $L = \{a^{n+m}xc^n yd^m,\ conn, m \geq 0\}$:

2 Alberi Sintattici

Un albero sintattico è una rappresentazione grafica (ad albero) che mostra come una forma sentenziale o una stringa è stata ottenuta tramite le regole di derivazione.

Albero Sintattico. *Data una CFG definita come*

$$G = (V, T, P, S)$$

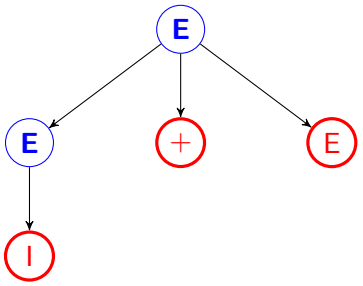
l'albero sintattico è un albero tale che

1. *Ogni nodo interno è* ***etichettato*** *da una variabile;*
2. *Ogni foglia è etichettata da una variabile, oppure un simbolo terminale o ancora da ε. Se è etichettata con ε allora è l'unico figlio riscontrato.*

3. *Se un nodo è etichettato con A (variabile) e i rispettivi figli sono etichettati da sinistra verso destra con $X_1, X_2, X_3, ..., X_k$, allora $A \to X_1, X_2, X_3, ..., X_k \in P$ (ovvero A è una produzione della grammatica).*

Un esempio pratico di albero sintattico: data la CFG definita come

$$E \to I \mid E + E \mid E * E \mid (E) \;\; e \;\; I \to a \mid b \mid Ia \mid I0 \mid I1 \tag{2}$$



abbiamo che l'albero sintattico ottenuto è un albero **radicato** e **ordinato**. Si può notare che i nodi interni rappresentano i passaggi per arrivare alle foglie. Difatti è possibile ricostruire il processo di derivazione:

$$E \to E \; + \; E \to I \; + \; E$$