Linguaggi e Computabilità

UniShare

Davide Cozzi
@dlcgold

Gabriele De Rosa
@derogab

Federica Di Lauro
@f_dila

Indice

Capitolo 1

Introduzione

Questi appunti sono presi a lezione. Per quanto sia stata fatta una revisione è altamente probabile (praticamente certo) che possano contenere errori, sia di stampa che di vero e proprio contenuto. Per eventuali proposte di correzione effettuare una pull request. Link: `https://github.com/dlcgold/Appunti`.
Grazie mille e buono studio!

1.1 Definizioni

- un **linguaggio** è un insieme di stringhe che può essere generato mediante un dato meccanismo con delle date caratteristiche; un linguaggio può essere riconosciuto, ovvero dando in input una stringa un meccanismo può dirmi se appartiene o meno ad un linguaggio. I meccanismi che generano linguaggi si chiamano *grammatiche*, quelli che li riconoscono *automi*. I linguaggi formali fanno parte dell'informatica teorica *(TCS)*
- si definisce **alfabeto** come un insieme finito e non vuoto di simbolo (come per esempio il nostro alfabeto o le cifre da 0 a 9). Solitamente si indica con Σ o Γ
- si definisce **stringa** come una sequenza finita di simboli (come per esempio una parola o una sequenza numerica). La stringa vuota è una sequenza di 0 simboli, e si indica con ε o λ
- si definisce **lunghezza di una stringa** il numero di simboli che la compone (ovviamente contando ogni molteplicità). Se si ha $w \in \Sigma^*$ è una stringa w con elementi da Σ^* (insieme di tutte le stringhe di tutte le lunghezze possibili fatte da Σ), allora $|w|$ è la lunghezza di w, inoltre $|\varepsilon| = 0$.

- si definisce **potenza di un alfabeto** Σ^k come l'insieme di tutte le sequenze (espressi come stringhe e non simboli) di lunghezza $k \in \mathbb{N}$, $k > 0$ ottenibili da quell'alfabeto (se Σ^2 si avranno tutte le sequenza di 2 elementi etc...). Se ho $k = 1$ si ha $\Sigma^1 \neq \Sigma$ in quanto ora ho stringhe e non simboli. Se ho $k = 0$ ho $\Sigma^0 = \varepsilon$. Dato k ho $|\Sigma|$ che è la cardinalità dell'insieme Σ (e non la sua lunghezza come nel caso delle stringhe); sia $w \in \Sigma^k = a_1, a_2, ..., a_k$, $a_i \in \Sigma$ e $|\Sigma| = q$ ora:
$$|\Sigma^k| = q^k$$
- si definisce Σ^* come **chiusura di Kleene** che è l'unione infinita di Σ^k ovvero
$$\Sigma* = \Sigma^0 \cup \Sigma^1 \cup ... \cup \Sigma^k$$
- si ha che Σ^+ è l'unione per $k \geq 1$ di Σ^k ovvero:
$$\Sigma+ = \Sigma^1 \cup \Sigma^2 \cup ... \cup \Sigma^k = \Sigma^* - \Sigma^0$$
per esempio, per l'insieme $\{0, 1\}$ si ha:
$$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 100, 000, ...\}$$
- quindi un **linguaggio** L è un insieme di stringhe e:
$$L \subseteq \Sigma^*$$
si hanno sottoinsiemi particolari, come l'insieme vuoto, che resta però un linguaggio, il **linguaggio vuoto** e $\emptyset \in \Sigma^k$, $|\emptyset| = 0$ che è diverso dal linguaggio che contiene la stringa vuota $|\varepsilon| = 1$ (che conta come una stringa). Inoltre $\Sigma^* \subseteq \Sigma^*$ che ha lunghezza infinita. Posso concatenare due stringhe con un punto: $a \cdot b \cdot c = abc$ e $a \cdot \varepsilon = a$. Ovviamente la stringa concatenata è lunga come la somma delle lunghezze delle stringhe che la compongono. Vediamo qualche esempio di linguaggio:
  - il linguaggio di tutte le stringhe che consistono in n 0 seguiti da n 1:
$$\{\varepsilon, 01, 0011, 000111, ...\}$$
  - l'insieme delle stringhe con un uguale numero di 0 e di 1:
$$\{\varepsilon, 01, 10.0011, 0101.1001, ..\}$$
  - l'insieme dei numeri binari il cui valore è un numero primo:
$$\{\varepsilon, 10, 11, 101, 111, 1011, ...\}$$

- Σ^* è un linguaggio per ogni alfabeto Σ
- $\emptyset$, il linguaggio vuoto, e $\{\varepsilon\}$ sono un linguaggio rispetto a qualunque alfabeto

Prendiamo un alfabeto $\Sigma = \{0, 1\}$ con la sua chiusura di Kleen $\Sigma = \{0, 1\}^*$. Quando si ha un input si può avere un problema di decisione, P, che dia come output "si" o "no". Posso avere un problema di decisione (o *membership*) su $w \in \Sigma = \{0, 1\}^*$, con w stringa, che dia in output "si" o "no". Un linguaggio L sarà:

$$L = \{w \in \{0, 1\}^* \mid P(w) = si$$

quindi si ha che:

$$\Sigma^* \backslash L = \{P(w) = no\}$$

Vediamo ora un esempio di *Context Free Language (CFL)*, costruito a partire da una *Context Free Grammar (CFG)*:

Esempio 1. *Sia* $\Sigma = \{0, 1\}$ *e* L_{pal} = "*stringhe palindrome binarie*". *Quindi, per esempio,* $0110 \in L$, $11011 \in L$ *ma* $10010 \notin L$. *Si ha che* ε*, la stringa vuota, appartiene a* L*. Diamo una definizione ricorsiva:*

- ***base:*** ε, $0 \;\; 1 \in L_{pal}$
- ***passo:*** *se* w *è palindroma allora* $0w0$ *è palindromo e* $1w1$ *è palindromo*

una variabile generica S *può sottostare alle regole di produzione di una certa grammatica. In questo caso si ha uno dei seguenti:*

$$S \to \varepsilon,\ S \to 0,\ S \to 1,\ S \to 0S0,\ S \to 1S1$$

Si ha che una grammatica G è una quadrupla $G = (V, T, P, S$ con:

- V simboli variabili
- T simboli terminali, ovvero i simboli con cui si scrivono le stringhe alla fine
- P regole di produzione
- S variabile di partenza *start*

riprendiamo l'esempio sopra:

Esempio 2.

$$G_{pal} = (V = \{S\},\, T = \{0,1\},\, P,\, S)$$

con:

$$P = \{S \to \varepsilon,\, S \to 0,\, S \to 1,\, S \to 0S0,\, S \to 1S1\}$$

Si può ora costruire un algoritmo per creare una stringa palindroma a partire dalla grammatica G:

$$\underbrace{S}_{\textit{start}} \quad \underbrace{\to}_{\textit{applico una regola}} \quad 1S1 \to 01S10 \to \quad \underbrace{01010}_{\textit{sostituisco variabile}}$$

con S, $1S1$ *e* $01S10$ *che sono forme sentenziali. Posso così ottenere tutte le possibili stringhe. Esiste anche una forma abbreviata:*

$$S \to \varepsilon|o|1|0S0|1S1$$

Non si fanno sostituzioni in parallelo, prima una S *e poi un'altra*

Si hanno 4 grammatiche formali, *gerarchia di Chomsky*:

- **tipo 0:** non si hanno restrizioni sulle regole di produzione, $\alpha \to \beta$. Sono linguaggi ricorsivamente numerabili e sono rappresentati dalle *macchine di Turing*, deterministiche o non deterministiche (la macchina di Turing è un automa)

- **tipo 1:** il lato sinistro della produzione (*testo*) ha lunghezza uguale a quello destro (*corpo*). Sono grammatiche dipendenti dal contesto (*contestuali*) e come automa hanno *la macchina di Turing che lavora in spazio lineare*:
$$\alpha_1 A \alpha_2 \to \alpha_1 B \alpha_2$$
con α_1 e α_2 detti *contesto* e α_1, α_2, $\beta \in (V \cup T)^*$

- **tipo 2:** sono quelle libere dal contesto, context free. Come regola ha $A \to \beta$ con $A \in V$ e $\beta \in V \cup T)^*$ e come automa ha gli *automi a pila non deterministici*

- **tipo 3:** sono le grammatiche *regolari*. Come regole ha $A \to \alpha B$ (o $A \to B\alpha$) e $A \to \alpha$ con $A, B \in V$ e $\alpha \in T$. Come automi ha gli *automi a stato finito deterministici o non deterministici*

Esempio 3. *Sia* $G = (V, T, O, E)$, *con* $V = \{E, I\}$ *e* $T = \{a, b, 0, 1, (,), +, *\}$ *quindi ho le seguenti regole, è di tipo 3:*

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. $I \to I1$

voglio ottenere $a * (a + b00)$ *sostituisco sempre a destra (right most derivation)*

$$E \to E * E \to E * (E) \to E * (E + E) \to E * (E + I) \to E + (E + I0)$$

$$\to R + (I + b00) \to E * (a + b00) \to I * (a + b00) \to a * (a + b00)$$

usiamo ora l'inferenza ricorsiva:

passo	*stringa ricorsiva*	*var*	*prod*	*passo stringa impiegata*
1	*a*	*I*	*5*	\
2	*b*	*I*	*6*	\
3	*b0*	*I*	*9*	*2*
4	*b00*	*I*	*9*	*3*
5	*a*	*E*	*1*	*1*
6	*b00*	*E*	*1*	*4*
7	*a+b00*	*E*	*2*	*5,6*
8	*(a+b00)*	*E*	*4*	*7*
9	*a*(a+b00)*	*E*	*3*	*5, 8*

definisco formalmente la derivazione $\to$:

Definizione 1. *Prendo una grammatica $G = (V, T, P, S)$, grammatica CFG. Se $\alpha A\beta$ è una stringa tale che $\alpha, \beta \in (V \cup T)^*$, appartiene sia a variabili che terminali. Sia $A \in V$ e sia $a \to \gamma$ una produzione di G. Allora scriviamo:*

$$\alpha A\beta \to \alpha\gamma\beta$$

con $\gamma \in (V \cup T)^*$.
Le sostituzioni si fanno indipendentemente da α e β. Questa è quindi la definizione di derivazione.

Definizione 2. *Definisco il simbolo $\to +$, ovvero il simbolo di derivazioni in 0 o più passi. Può essere definito in modo ricorsivo. Per induzione sul numero di passi.*

- *la base dice che $\forall\alpha \in (V \cup T)^*$, $\alpha \to *\alpha$*
- *il passo è: se $\alpha \to_G *\beta$ e $\beta \to_G *\gamma$ allora $\alpha \to *\gamma$*

Si può anche dire che $\alpha \to_G +\beta$ sse esiste una sequenza di stringhe $\gamma_1, ..., \gamma_n$ con $n \geq 1$ tale che $\alpha = \gamma_1$, $\beta = \gamma_n$ e $\forall i$, $1 < i < n-1$ si ha che $\gamma_1 \to \gamma_{i+1}$ la derivazione in 0 o più passi è la chiusura transitiva della derivazione

Definizione 3. *avendo ora definito questi simboli possiamo definire una forma sentenziale. Infatti è una stringa α tale che:*

$$\forall\alpha \in (V \cup T)^* \text{ tale che } S \to_G *\alpha$$

Definizione 4. *data $G = (V, T, P, S)$ si ha che $L(G) = \{w \in T^* | \, S \to_G *w\}$ ovvero composto da stringhe terminali che sono derivabili o 0 o più passi.*

Esempio 4. *formare una grammatica CFG per il linguaggio:*

$$L = \{0^n1^n | n \geq 1\} = \{01, 0011, 000111, ...\}$$

con x^n intendo una concatenazione di n volte x (che nel nostro caso sono 0 e 1).
posso scrivere:

$$0^n1^n = 00^{n-1}1^{n-1}1$$

il nostro caso base sarà la stringa 01, Poi si ha: $G = /V, T, P, S)$, $T = \{0, 1\}$, $V = \{S\}$, il caso base $S \to 01$ e $S \to 0S1$ il caso passo è quindi: se $w = 0^{n-1}1^{n-1} \in L$ allora $0w1 \in L$.
Ora voglio dimostare che $000111 \in L$, ovvero $S \to *000111$:

$$S \to 0S1 \to 00S11 \to 000S111$$

Teorema 1. *data la grammatica $G = \{V, T, P, S)$ CFG e $\alpha \in (V \cup T)^*$. Si ha che vale $S \to *\alpha$ sse $S \to_{lm} *\alpha$ sse $S \to_{rm} *\alpha$. Con $to_{lm}*$ simbolo di left most derivation e $to_{rm}*$ simbolo di right most derivation*

Esempio 5. *formare una grammatica CFG per il linguaggio:*

$$L = \{0^n 1^n | n \geq 0\} = \{\varepsilon, 01, 0011, 000111, ...\}$$

stavolta abbiamo anche la stringa vuota. Il caso base stavolta è $S \to \varepsilon|\, 0S1$

Esempio 6. *Fornisco una CFG per $L = \{a^n | n \geq 1\} = \{a, aa, aaa, ...\}$. La base è a*
il passo è che se $a^{n-1} \in L$ allora $a^{n-1}a \in L$ (o che $aa^{n-1} \in L$).
Si ha la grammatica $G = \{V, T, P, S)$, $V = \{S\}$, $T = \{a\}$ e si hanno $S \to a|\, Sa$ (o $S \to a|\, aS$). Dimostro che $a^3 \in L$.

$$S \to Sa \to Saa \to aaa$$

oppure

$$S \to aS \to aaS \to aaa$$

Esempio 7. *trovo una CFG per $L = \{(ab)^n | n \geq 1\} = \{ab, abab, ababab, ...\}$*
La base è ab
il passo è che se $(ab)^{n-1} \in L$ allora $(ab)^{n-1}ab \in L$.
Si ha la grammatica $G = \{V, T, P, S)$, $V = \{S\}$, $T = \{a, b\}$ (anche se in realtà $T = \{ab\}$) e si hanno $S \to ab|\, Aab$. Poi dimostro come l'esempio sopra

Esempio 8. *trovo una CFG per $L = \{a^n cb^n | n \geq 1\} = acb, aacbb, aaacbbb, ...\}$*
Il caso base è acb il passo è che se $a^{n-1}cb^{n-1} \in L$ allora $a^{n-1}cb^{n-1}acb \in L$
Si ha la grammatica $G = \{V, T, P, S)$, $V = \{S\}$, $T = \{a, b, c\}$ e si hanno $S \to aSb|acb$.
dimostro che $aaaacbbbbb \in L$:

$$S \to aSb \to aaSbb \to aaaaSbbb \to aaaacbbbb$$

provo a usare anche una grammatica regolare, con le regole $S \to aS|c$, $c \to cB$ e $B \to bB|b$;

$$S \to aS \to aaS \to aaC \to aacB \to aacb...$$

non si può dimostrare in quanto non si può imporre una regola adatta

Esempio 9. *$L = \{a^n cb^{n-1} | n \geq 2\}$, con $a^n cb^{n-1} = a^{n-1}acb^{n-1}$. $S \to aSb|aacb$. Quindi:*

$$S \to aSb \to aaaccbb \in L$$

Esempio 10. *cerco CFG per* $L = \{a^n c^k b^n | \, n, \, k > 0\}$*. a e b devono essere uguali, uso quindi una grammatica context free, mentre c genera un linguaggio regolare.*
Si ha la grammatica $G = \{V, T, P, S)$, $V = \{S, C\}$, $T = \{a, b, c\}$ *e si hanno* $S \to aSb|aCb$ *e* $C \to cC|c$*. dimostro che* $aaaccbbb \in L, n = 3, \, k = 2$*:*

$$S \to aSb \to aaSbb \to aaaCbbb \to aaacCbbb \to aaaccbbb$$

Esempio 11. *scrivere CFG per* $L = \{a^n b^n c^k b^k | \, n, \, k \geq 0\}$

$$= \{w \in \{a, b, c, d\}^* | \, a^n b^n c^k b^k | \, n, \, k \geq 0\}$$

quindi L concatena due linguaggi L1 e L2, $X = \{a^n b^n\}$ *e* $Y = \{c^k d^k\}$*:*

$$X \to aXb|\varepsilon$$

$$Y \to cYd|\varepsilon$$

$$S \to XY$$

voglio derivare abcd:

$$S \to XY \to XcYd \to aXbcYd \to aXbc\varepsilon d \to a\varepsilon bc\varepsilon d \to abcd$$

voglio derivare cd

$$S \to XY \to Y \to cYd \to cd$$

Quindi se ho $w \in L1, L2$, ovvero appartenente ad una concatenazione di linguaggi prima uso le regole di un linguaggio, poi dell'altro e infine ottengo il risultato finale.

Esempio 12. *scrivere CFG per* $L = \{a^n b^k c^k d^n | \, n > 0, \, k \geq 0\}$.

$$S \to aSd| \, aXd$$

$$X \to bXc|\varepsilon$$

derivo aabcdd:

$$S \to aSd \to aaXdd \to aabXcdd \to aabcdd$$

Esempio 13. *scrivere CFG per* $L = \{a^n cb^n c^m ad^m | \, n > 0, \, m \geq 1\}$.

$$S \to XY$$

$$X \to aXb|c$$

$$Y \to cUd|cad$$

$$S \to XY \to cY \to ccad$$

Esempio 14. *scrivere CFG per* $L = \{a^{n+m}xc^nyd^m |\, n,\, m \geq 0\}$. $a^{n+m} = a^na^m$ *o* a^ma^n. *Si hanno 2 casi:*

1. $L = \{a^na^mxc^nyd^m |\, n,\, m \geq 0\}$
2. $L = \{a^ma^nxc^nyd^m |\, n,\, m \geq 0\}$

ma solo $L = \{a^ma^nxc^nyd^m |\, n,\, m \geq 0\}$ *può generare una CFG (dove non si possono fare incroci, solo concatenazioni e inclusioni/innesti).*

$$S \to aSd|Y$$

$$Y \to Xy$$

$$X \to aXc|x$$

si può fare in 2:

$$S \to aSd|Xy$$

$$X \to aXc|x$$

derivo con $m = n = 1$, *aaxcyd:*

$$S \to aSd \to aXyd \to aaXcyd \to aaxcyd$$

Esempio 15. *scrivere CFG per* $L = \{a^nb^m |\, n \geq m \geq 0\}$.

$$L = \{\varepsilon, a, ab, aa, aab, aabb, aaa, aaab, aaabb, aaabbb, ...\}$$

Se $n \geq m$ *allora* $\exists k \geq 0 \to n = m + k$. *Quindi:*

$$l = \{a^{m+k}b^m|m, k \geq 0\}$$

si può scrivere in 2 modi:

1. $l = \{a^ma^kb^m|m, k \geq 0\}$ *quindi con innesto*
2. $l = \{a^ka^mb^m|m, k \geq 0\}$*quindi con concatenazione*

entrambi possibili per una CFG:

1.

$$S \to XY$$

$$X \to aX|\varepsilon \text{ si può anche scrivere } X \to Xa|\varepsilon$$

$$Y \to aYb|\varepsilon$$

oppure

$$S \to aS|X$$

$$X \to aXb|\varepsilon$$

2.

$$S \to aSb|\varepsilon$$

$$X \to aX|\varepsilon$$

Esempio 16. *scrivere CFG per* $L = \{a^n b^{m+n} c^h |\, m > h \geq 0,\, n \geq 0\}$.
Se $n > h$ *allora* $\exists k \to n = h + k$*, quindi:*

$$L = \{a^n b^{m+h+k} c^h |\, m > h \geq 0,\, n \geq 0\}$$

. *ovvero:*

$$L = \{a^n b^n b^k b^h c^h |\, m \geq 0, k > 0, h \geq 0\}$$

si ha:

$$S \to XYZ$$

$$X \to aXb|\varepsilon$$

$$Y \to Yb|b$$

$$Z \to bZc|\varepsilon$$

si può anche fare:

$$S \to XY$$

$$X \to aXb|\varepsilon$$

$$Y \to bYc|Z$$

$$Z \to bZ|b$$

Esempio 17. *scrivere CFG per* $L = \{a^n b^m c^k |\, k > n + m,\, n, m \geq 0\}$.
per $n = m = 0$, $k = 1$ *avrò la stringa* c*. se* $k > n + m$ *allora* $\exists l > 0 \to k = n + m + l$ *quindi:*

$$L = \{a^n b^m c^{n+m+l} |\, l > 0,\, n, m \geq 0\}$$

$$= L = \{a^n b^m c^n c^m c^l |\, l > 0,\, n, m \geq 0\}$$

sistemando:

$$= L = \{a^n b^m c^l c^m cnl |\, l > 0,\, n, m \geq 0\}$$

quindi:

$$S \to aSc|X$$

$$X \to bXc|Y$$

$$Y \to cY|c$$

Esempio 18. *scrivere CFG per* $L = \{a^n x c^{n+m} y^h z^k d^{m+h} | \, n, m, k, h \geq 0\}$.
ovvero:

$$L = \{a^n x c^n c^m y^h z^k d^h d^m | \, n, m, k, h \geq 0\}$$

quindi avrò:

$$S \to XY$$

$$X \to aXc|x$$

$$Y \to cYd|W$$

$$W \to yWd|X$$

$$Z \to zZ|\varepsilon$$

Esempio 19. *vediamo un esempio di grammatica dipendente dal contesto:*

$$L = \{a^n b^n c^n | \, n \geq 1\}$$

$G = \{V, T, P, S\} = \{(S, B, C, X)\} = \{(a, b, c), P, S\}$ *ecco le regole di produzione (qui posso scambiare variabili a differenza delle context free):*

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. $cC \to cc$

vediamo un esempio di derivazione: per $n = 1$ *ho abc ovvero:*

$$S \to aBC \to abC \to abc$$

con $n = 2$ *ho aabbcc:* $S \to aSBC \to aaBCBC \to aaBXBC \to aaBXCC \to aaBBCC \to aabBCC \to aabbCC \to aabbcC \to aabbcc$