Definizioni da imparare

Grammatica: *complesso di regole* necessarie alla costruzione di *frasi, sintagmi* e *parole* di un determinato linguaggio.
Correttezza (**sintattica**): una stringa è *corretta* se *può essere generata* (*derivata*) applicando le *regole della grammatica*.
Derivazione: il processo di *generazione* di una *frase* usando la *grammatica*.
Linguaggio formale: insieme di *tutte le stringhe* che possono essere *derivate* dalle *regole della grammatica*, a prescindere dal significato.

Alfabeto: un insieme X *finito* e *non vuoto* di simboli.
Parola: sequenza **finita** $x_1x_2\dots x_n$, dove ogni x_i è preso da uno *stesso alfabeto* X. Si ottiene *concatenando* o *giustapponendo* simboli dello stesso alfabeto.
X^*: l'insieme di *tutte le stringhe* di lunghezza finita sull'alfabeto X (comprende λ). Ha un numero di elementi che è un *infinito numerabile* (cioè è possibile *elencarli*).
Concatenazione: sia $\alpha \in X^*$ una stringa di lunghezza m e $\beta \in X^*$ una stringa di lunghezza n, la concatenazione di α e β, denotata con $\alpha\beta$ o $\alpha \cdot \beta$, è definita come la stringa di lunghezza $m+n$, i cui primi m simboli costituiscono una *stringa uguale* a α ed i cui ultimi n simboli costituiscono una *stringa uguale* a β. La concatenazione è un'*operazione binaria* su X^*:

$$\cdot : X^* \times X^* \to X^*$$

Prefisso, suffisso: se $\gamma \in X^*$ è della forma

$$\gamma = \alpha\beta$$

ove $\alpha, \beta \in X^*$, allora α è *prefisso* di γ e β è *suffisso* di γ.
Sottostringa: se $\delta, \beta \in X^*$ e δ è della forma

$$\delta = \alpha\beta\gamma$$

ove $\alpha, \beta \in X^*$ allora β è una *sottostringa* di δ.
Potenza di una stringa: Data una stringa α su X, la *potenza h-esima* di α è definita (induttivamente) come segue:

$$\alpha^h = \begin{cases} \lambda & \text{se h} = 0 \\ \alpha\alpha^{h-1} & \text{altrimenti} \end{cases}$$

è un tipo speciale di concatenazione, poiché è ottenuta concatenando la stringa h volte con se stessa.
Potenza di un alfabeto: definizione ricorsiva:

$$X^h = \begin{cases} \{\lambda\} & \text{se h} = 0 \\ X \cdot X^{h-1} & \text{altrimenti} \end{cases}$$

X^k indica l'insieme di tutte le stringhe di lunghezza k. Unendo tutti i possibili X^k, con $k > 0$ ottengo un nuovo insieme X^+:

$$X^+ = \bigcup_{i=1}^{+\infty} X^i$$

ovvero l'insieme di **tutte le stringhe di lunghezza maggiore di 0** costruibili a partire dall'alfabeto X, (quindi escludendo λ).
Dato X^+ possiamo ricavare anche X^*:

$$X^* = \{\lambda\} \cup X^*$$

Linguaggi formali

Linguaggio formale: un linguaggio formale L su un alfabeto X è un sottoinsieme di X^*

$$L \subseteq X^*$$

Un linguaggio L infatti è l'insieme di stringhe di lunghezza arbitraria *ben costruite sintatticamente* (che rispettano le regole di una grammatica G).
Come faccio a descrivere gli elementi appartenenti ad un linguaggio?

- se il linguaggio è *finito*: elenchiamo gli elementi;
- se il linguaggio è *infinito* utilizziamo una notazione insiemistica, detta *descrittore del linguaggio*.

Una **grammatica generativa** è una quadrupla:

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

ove:

- X è l'alfabeto terminale;
- V è l'alfabeto non terminale;
- S è il simbolo di partenza;
- P è l'insieme delle produzioni.
  devono inoltre valere le seguenti condizioni:

$$X \cap V = \emptyset \text{ e } S \in V$$

Una **produzione** è una coppia (v, w) nella forma:

$$v \in (X \cup V)^+ \text{ e } w \in (X \cup V)^*$$

e v contiene un NT.

- $(X \cup V) \rightarrow$ Insieme di simboli terminali e non terminali;
- $(X \cup V)^+ \rightarrow$ Insieme di stringhe composte da *simboli terminali* e *non terminali* (*escluso* λ);
- $(X \cup V)^* \rightarrow$ Insieme di stringhe composte da simboli *terminali* e *non terminali* (*incluso* λ);

La definizione dice che a **sinistra di una regola**:

- ho una stringa;
- ci deve essere *necessariamente* un *non terminale*;
- non può esserci λ.

A **destra di una regola** invece:

- posso avere *qualsiasi* stringa (anche λ).
  scriviamo quindi la definizione di una produzione:

$$P = \left\{ v \rightarrow w \mid v \in (X \cup V)^+ \text{ e } v \text{ contiene almeno un NT}, w \in (X \cup V)^* \right\}$$

Una **derivazione diretta** è il risultato dell'applicazione di **una** regola di produzione:

$$y \Rightarrow z$$

Una **derivazione** è il risultato dell'applicazione di una **sequenza di applicazione di regole** di produzione:

$$y \overset{*}{\Rightarrow} z$$

Linguaggio generato da G: sia $G = (X, V, S, P)$ una grammatica. Il linguaggio generato da G, denotato con $L(G)$, è l'insieme delle stringhe di terminali derivabili dal simbolo di partenza S.

$$L(G) = \left\{ w \in X^* \mid S \underset{\text{G}}{\overset{*}{\Longrightarrow}} w \right\}$$

ricorda che:

$$L \neq L(G)$$

una grammatica è **corretta** se L e $L(G)$ coincidono, ma questo passaggio *non è automatico* e la dimostrazione non è algoritmica, ma per *induzione*.

Gerarchia di Chomsky

A seconda delle restrizioni imposte sulle regole di produzione, si distinguono le *varie classi di grammatiche* (tipo 0, 1, 2 e 3).

Una **grammatica** $G = (X, V, S, P)$ è **libera da contesto** (o di *tipo 2*) se, per ogni produzione $v \to w$, v è un non terminale. Formalmente:

$$G \text{ è libera da contesto} \overset{def}{\iff} \forall v \to w \in P : v \in V$$

(in parole semplici, se a sinistra hai un carattere non terminale come S, A ecc..)

Un **linguaggio** L su un alfabeto X è **libero da contesto** se *può essere generato* da una grammatica libera da contesto. Formalmente:

$$L \text{ libero da contesto} \overset{def}{\iff} \exists G \text{ libera da contesto tale che } L(G) = L$$

Una grammatica $G = (X, V, S, P)$ è *lineare destra* (o di *tipo 3*) se le sue produzioni sono limitate alla forma:

$$\begin{array}{ll} (1)\ A \to bC & \text{con } A, C \in V \text{ e } b \in X; \\ (2)\ A \to b & \text{con } A \in V \text{ e } b \in X \cup \{\lambda\}. \end{array}$$

nota che si tratta di "specializzazione" dei linguaggi di tipo 2.

Un **linguaggio** L su un alfabeto X è **lineare destro** se *può essere generato* da una grammatica lineare destra. Formalmente:

$$L \text{ lineare destro} \overset{def}{\iff} \exists G \text{ lineare destra tale che } L(G) = L$$

(Una grammatica di tipo 3 è detta lineare destra perché il NT, se c'è, compare a destra)

Una grammatica $G = (X, V, S, P)$ è *dipendente da contesto* (o di *tipo 1*) se ogni produzione è in una delle seguenti forme:

$$(1)\ yAz \to ywz \text{ con } A \in V,\ y, z \in (X \cup V)^*,\ w \in (X \cup V)^+$$

dove y è *contesto sinistro* e z è *contesto destro*.

$$(2)\ S \to \lambda \text{ purché } S \text{ non compaia nella parte destra di alcuna produzione}$$

Un **linguaggio** L è **dipendente da contesto** se *può essere generato* da una grammatica dipendente da contesto.

Poiché un linguaggio può essere generato da più grammatiche (anche di diverso tipo), esso si etichetta col tipo più "specifico" che è possibile associare.

Operazioni sui linguaggi

Le operazioni sui linguaggi ci permettono di **combinare** tra loro linguaggi più semplici al fine di ottenere linguaggi più "complessi".
I linguaggi possono essere combinati mediante operazioni di:

- Unione
- Concatenazione
- Iterazione
- Intersezione
- Complemento

Siano L_1 ed L_2 due linguaggi definiti su uno stesso alfabeto X ($L_1, L_2 \subseteq X^*$):

- L'**unione** di L_1 ed L_2 è:

$$L_1 \cup L_2 = \{w \mid w \in L_1 \vee w \in L_2\}$$

- La **concatenazione** (o prodotto) di L_1 e L_2 è:

$$L_1 \cdot L_2 = \{w \mid w = w_1 w_2,\ w_1 \in L_1,\ w_2 \in L_2\}$$

- L'**iterazione** di L_1 (o chiusura riflessiva e transitiva di L_1 rispetto all'operazione di concatenazione) è:

$$L_1^* = \{w \mid w = w_1 w_2 \dots w_n,\ n \geq 0 \text{ e } \forall i : w_i \in L_1\}$$

- Il **complemento** di L_1 è:

$$\overline{L_1} = X^* - L_1$$

  (tutte le stringhe che NON appartengono ad L)
- L'**intersezione** di L_1 ed L_2 è:

$$L_1 \cap L_2 = \{w \mid w \in L_1 \ \wedge \ w \in L_2\}$$

- La **potenza** n-esima di L, e si denota con $L^n,\ n \geq 0$:

$$L^n = \begin{cases} \{\lambda\} \text{ se } \ n = 0 \\ \\ L^{n-1} \cdot L \text{ altrimenti} \end{cases}$$

  Sia L un linguaggio definito su un alfabeto X. Si ha:

$$L^* = \bigcup_{i \geq 0} L^i = \{\lambda\} \cup L \cup L^2 \cup L^3 \cup \ \ldots$$

- Sia w una parola su un alfabeto $X = \{x_1, X_2, \ldots, x_k\}, w = x_{i_1} x_{i_2} \ldots x_{i_{n-1} x_{i_n}}$.
  Dicesi **stringa riflessa** di w la stringa:

$$w^R = x_{i_n} x_{i_{n-1} \ldots x_{i_2} x_{i_1}}$$

- **Operazione di riflessione**: l'operazione che trasforma w in w^R.
- **Palindromo**: una parola la cui lettura a ritroso riproduce la *parola di partenza*:

$$w \text{ palindromo } \overset{def}{\iff} \ w = w^R$$

  essi possono essere di lunghezza *pari* (asse di simmetria su λ) o *dispari* (asse di simmetria su un simbolo dell'alfabeto). w è palindroma se e solo se:

$$w = \alpha x \alpha^R,\ x \in X \cup \{\lambda\}$$

Proprietà di chiusura delle classi di linguaggio

$\mathcal{L}$ classe di linguaggi su X.
Definizione di chiusura: date α (operazione binaria) e β (operazione unaria):

- $\mathcal{L}$ è *chiusa* rispetto ad $\alpha \ \overset{def}{\iff} \ \forall L_1, L_2 \in \mathcal{L} : \alpha(L_1, L_2) \in \mathcal{L}$
- $\mathcal{L}$ è *chiusa* rispetto a $\beta \ \overset{def}{\iff} \ \forall L_1 \in \mathcal{L} : \beta(L_1) \in \mathcal{L}$

Teorema di chiusura

- La classe dei linguaggi di tipo $i,\ i = 0, 1, 2, 3$ è *chiusa* rispetto alle operazioni di *unione*, *concatenazione* ed *iterazione*.
- La classe dei linguaggi *lineari destri* (tipo 3) è *chiusa* rispetto al *complemento* e all'*intersezione*. La classe di linguaggi di tipo 2 **NON** è chiusa rispetto al *complemento* e all'*intersezione*.

Automi

Un automa è una macchina in grado di operare in modo *autonomo*.
In sistemi di questo tipo, in ogni istante ci si trova in uno **stato** preso da un *insieme finito*.

- Uno stato ha lo scopo di ricordare una parte *pertinente* della storia del sistema e a determinare quali sono le **possibili azioni** che si possono effettuare in quel momento;
- Un sistema va progettato attentamente affinché **ricordi** ciò che è **importante** e dimentichi ciò che **non lo è**.
  Uno degli stati è detto *iniziale*, cioè lo stato in cui il sistema si trova inizialmente.
  Spesso è necessario indicare uno o più stati come *finali* o *accettanti*.

Automi a Stati Finiti Deterministici

È un **modello di calcolo** che useremo per riconoscere un linguaggio L:
Sia X un alfabeto. Un *automa a stati finiti* (FSA) è una *quadrupla*:

$$M = (Q,\ \delta,\ q_0,\ F)$$

ove:

- X è detto *alfabeto di ingresso*;
- Q è un insieme finito e non vuoto di *stati*;
- δ è una funzione da in $\mathcal{Q}$, detta funzione di transizione:

  $$\delta : \mathcal{Q} \times X \to \mathcal{Q}$$

  talora i valori della funzione di transizione δ non sono definiti per tutte le coppie (q, x) . In tal caso, si dice che δ è una **funzione parziale**. La funzione può essere trasformata in **funzione totale** creando uno stato apposito (detto *stato pozza*) dove vanno a finire tutte le transizioni che *non portano* a una parola accettata dal linguaggio.
- q_0 è lo *stato iniziale*;
- $F \subseteq \mathcal{Q}$ è l'insieme degli *stati di accettazione* o *finali*.
  Un FSA può essere rappresentato mediante:
- Grafo degli Stati;
- Diagramma di Transizione.

Estensione della funzione di transizione: la funzione di transizione, nella sua definizione di base, prende come input uno *stato* e un *carattere dell'alfabeto*, e restituisce un nuovo stato. Normalmente però un automa riceve come input *un'intera stringa*, non un carattere, perciò *estendiamo* la funzione di transizione a questo scopo.

Dato un FSA $M = (Q,\ \delta,\ q_0,\ F)$ con alfabeto di ingresso X, definiamo per induzione la funzione:

$$\delta^* : \mathcal{Q} \times X^* \to \mathcal{Q}$$

Questa funzione ci dice in quale stato arriviamo dando in input una *stringa* (non un carattere!)
La definizione di questa funzione è **ricorsiva**:

$$\begin{cases} \delta^*(q, \lambda) = q \\ \\ \delta^*(q, wx) = \delta(\delta^*(q, w), x) \end{cases}$$

Parola accettata o riconosciuta: sia $M = (Q,\ \delta,\ q_0,\ F)$ un FSA con alfabeto di ingresso X. Una parola $w \in X^*$ è *accettata* (o *riconosciuta*) da M se, partendo dallo stato iniziale q_0, lo stato q in cui l'automa si porta *alla fine* della sequenza di ingresso w è uno stato finale.

$$w \text{ accettata} \overset{def}{\iff} \delta^*(q_0, w) \in F$$

Linguaggio accettato o riconosciuto: sia $M = (Q,\ \delta,\ q_0,\ F)$ un FSA con alfabeto di ingresso X. Il linguaggio accettato o riconosciuto da M è il seguente sottoinsieme di X^* :

$$T(M) = \{w \in X^* \mid \delta^*(q_0, w) \in F\}$$

FSA equivalenti: siano $M_1 = (Q_1, \delta_1, q_1, F_1)$ ed $M_2 = (Q_2, \delta_2, q_2, F_2)$ due FSA di alfabeto di ingresso X. M_1 ed M_2 si dicono *equivalenti* se:

$$T(M_1) = T(M_2)$$

Linguaggi a stati finiti: Dato un alfabeto X, un linguaggio L su X è un linguaggio a stati finiti (FSL) se esiste un FSA M con alfabeto di ingresso X tale che $L = T(M)$.
Classe di linguaggi a stati finiti:

$$\mathcal{L}_{FSL} = \left\{L \in 2^{X^*} \mid \exists M,\ M \text{ è un FSA} : L = T(M)\right\}$$

dove 2^{X^*} è l'insieme di tutti i *possibili linguaggi*.