Teoria dei linguaggi

Indice

1. Introduzione **2**
1.1. Storia 2
1.2. Ripasso 2
2. Gerarchia di Chomsky **3**
2.1. Rappresentazione 3
2.2. Grammatiche 3
2.2.1. Regole di produzione 3
2.2.2. Linguaggio generato da una grammatica 3
2.3. Gerarchia 4
2.4. Potenza computazionale 5

1. Introduzione

1.1. Storia

Un **linguaggio** è *uno strumento di comunicazione usato da membri di una stessa comunità*, ed è composto da due elementi:
- **sintassi**: insieme di simboli (o *parole*) che devono essere combinati con una serie di regole;
- **semantica**: associazione frase-significato.

Per i linguaggi naturali è difficile dare delle regole sintattiche: vista questa difficoltà, nel 1956 **Noam Chomsky** introduce il concetto di **grammatiche formali**, che si servono di regole matematiche per la definizione della sintassi di un linguaggio.

Il primo utilizzo dei linguaggi risale agli stessi anni con il **compilatore Fortran**, ovvero un traduttore da un linguaggio di alto livello ad uno di basso livello, ovvero il *linguaggio macchina*.

1.2. Ripasso

Un **alfabeto** è un insieme *non vuoto* e *finito* di simboli, di solito indicato con Σ o Γ.

Una **stringa** (o **parola**) è una sequenza *finita* di simboli appartenenti a Σ.

Data una parola w, possiamo definire:
- $|w|$ *numero di caratteri* di w;
- $|w|_a$ *numero di occorrenze* della lettera $a \in \Sigma$ in w.

Una parola molto importante è la **parola vuota** ε, che, come dice il nome, ha simboli, ovvero $|\varepsilon| = 0$.

L'insieme di tutte le possibili parole su Σ è detto Σ^*.

Un'importante operazione sulle parole è la **concatenazione** (o *prodotto*), ovvero se $x, y \in \Sigma^*$ allora la concatenazione w è la parola $w = xy$.

Questo operatore di concatenazione:
- *non è commutativo*, infatti $w_1 = xy \neq yz = w_2$ in generale;
- *è associativo*, infatti $(xy)z = x(yz)$.

La struttura $(\Sigma^*, \cdot, \varepsilon)$ è un **monoide** libero generato da Σ.

Vediamo ora alcune proprietà delle parole:
- **prefisso**: x si dice *prefisso* di w se esiste $y \in \Sigma^*$ tale che $xy = w$;
  - **prefisso proprio** se $y \neq \varepsilon$;
  - **prefisso non banale** se $x \neq \varepsilon$;
  - il numero di prefissi è uguale a $|w| + 1$.
- **suffisso**: y si dice *suffisso* di w se esiste $x \in \Sigma^*$ tale che $xy = w$;
  - **suffisso proprio** se $x \neq \varepsilon$;
  - **suffisso non banale** se $y \neq \varepsilon$;
  - il numero di suffissi è uguale a $|w| + 1$.
- **fattore**: y si dice *fattore* di w se esistono $x, z \in \Sigma^*$ tali che $xyz = w$;
  - il numero di fattori è al massimo $\frac{|w| \cdot |w+1|}{2} + 1$.
- **sottosequenza**: x si dice *sottosequenza* di w se x è ottenuta eliminando 0 o più caratteri da w;
  - un *fattore* è una sottosequenza ordinata.

Un **linguaggio** L definito su un alfabeto Σ è un qualunque sottoinsieme di Σ^*.

2. Gerarchia di Chomsky

2.1. Rappresentazione

Vogliamo rappresentare in maniera finita un oggetto infinito come un linguaggio.

Abbiamo a nostra disposizione due modelli molto potenti:
- **generativo**: date delle regole, si parte da *un certo punto* e si generano tutte le parole di quel linguaggio con le regole date; parleremo di questi modelli tramite le *grammatiche*;
- **riconoscitivo**: si usano dei *modelli di calcolo* che prendono in input una parola e dicono se appartiene o meno al linguaggio.

Considerando il linguaggio sull'alfabeto $\{(,)\}$ delle parole ben bilanciate, proviamo a dare due modelli:
- *generativo*: a partire da una sorgente S devo applicare delle regole per derivate tutte le parole appartenenti a questo linguaggio;
  - la parola vuota ε è ben bilanciata;
  - se x è ben bilanciata, allora anche (x) è ben bilanciata;
  - se x, y sono ben bilanciate, allora anche xy sono ben bilanciate.
- *riconoscitivo*: abbiamo una *black-box* che prende una parola e ci dice se appartiene o meno al linguaggio (in realtà potrebbe non terminare mai la sua esecuzione);
  - $\#(\ = \ \#)$;
  - per ogni prefisso, $\#(\ \geq \ \#)$.

2.2. Grammatiche

Una **grammatica** è una tupla (V, Σ, P, S), con:
- V *insieme finito e non vuoto* delle **variabili**; queste ultime sono anche dette *simboli non terminali* e sono usate durante il processo di generazione delle parole del linguaggio;
- Σ *insieme finito e non vuoto* dei **simboli terminali**; questi ultimi appaiono nelle parole generate, a differenza delle variabili che invece non possono essere presenti;
- P *insieme finito* delle **regole di produzione**;
- $S \in V$ **simbolo iniziale** o **assioma**, è il punto di partenza della generazione.

2.2.1. Regole di produzione

Soffermiamoci sulle regole di produzione: la forma di queste ultime è $\alpha \longrightarrow \beta$, con $\alpha \in (V \cup \Sigma)^+$ e $\beta \in (V \cup \Sigma)^*$.

Una regola di produzione viene letta come "$se\ ho\ \alpha\ allora\ posso\ sostituirlo\ con\ \beta$".

L'applicazione delle regole di produzione è alla base del **processo di derivazione**: esso è formato infatti da una serie di **passi di derivazione**, che permettono di generare una parola del linguaggio.

Diciamo che x deriva y in un passo, con $x, y \in (V \cup \Sigma)^*$, se e solo se $\exists(\alpha \longrightarrow \beta) \in P$ e $\exists \eta, \delta \in (V \cup \Sigma)^*$ tali che $x = \eta\alpha\delta$ e $y = \eta\beta\delta$.

Il passo di derivazione lo indichiamo con $x \Rightarrow y$.

La versione estesa afferma che x deriva y in $k \geq 0$ passi, e lo indichiamo con $x \overset{k}{\Rightarrow} y$, se e solo se $\exists x_0, ..., x_k \in (V \cup \Sigma)^*$ tali che $x = x_0$, $x_k = y$ e $x_{i-1} \Rightarrow x_i \ \forall i \in [1, k]$.

Se non ho indicazioni sul numero di passi k posso scrivere:
- $x \overset{*}{\Rightarrow} y$ per indicare un numero generico di passi, e questo vale se e solo se $\exists k \geq 0$ tale che $x \overset{k}{\Rightarrow} y$;
- $x \overset{+}{\Rightarrow} y$ per indicare che serve almeno un passo, e questo vale se e solo se $\exists k > 0$ tale che $x \overset{k}{\Rightarrow} y$.

2.2.2. Linguaggio generato da una grammatica

Indichiamo con $L(G)$ il linguaggio generato dalla grammatica G, ed è l'insieme $\left\{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\right\}$.

Due grammatiche G_1, G_2 sono **equivalenti** se e solo se $L(G_1) = L(G_2)$.

Se consideriamo l'esempio delle parentesi ben bilanciate, possiamo definire una grammatica per questo linguaggio con le seguenti regole di produzione:
- $S \longrightarrow \varepsilon$;
- $S \longrightarrow (S)$;
- $S \longrightarrow SS$.

Vediamo un esempio più complesso. Siano:
- $\Sigma = \{a, b, c\}$;
- $V = \{S, B\}$;
- $P = \{S \longrightarrow aBSc \mid abc, Ba \longrightarrow aB, Bb \longrightarrow bb\}$.

Questa grammatica genera il linguaggio $L(G) = \{a^n b^n c^n \mid n \geq 1\}$: infatti, il "caso base" genera la stringa *abc*, mentre le iterazioni "maggiori" generano il numero di *a* e *c* corretti, con i primi che vengono ordinati prima di inserire anche il numero corretto di *b*.

2.3. Gerarchia

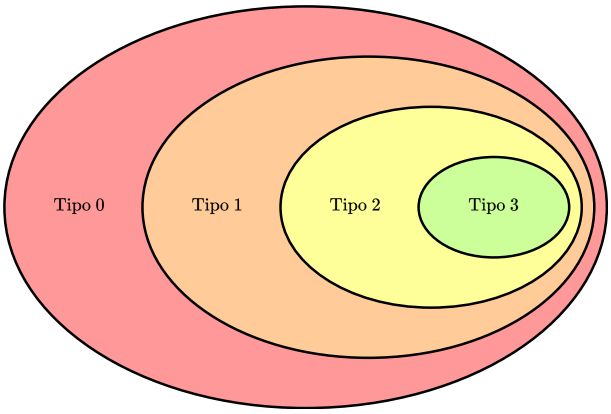
Negli anni 50 Noam Chomsky studia la generazione dei linguaggi formali e crea una **gerarchia di grammatiche formali**. La classificazione delle grammatiche viene fatta in base alle regole di produzione che definiscono la grammatica.

Grammatica	Regole	Modello riconoscitivo
Tipo 0.	Nessuna restrizione, sono il tipo più generale.	*Macchine di Turing.*
Tipo 1, dette **context-sensitive** o *dipendenti dal contesto.*	Se $(\alpha \longrightarrow \beta) \in P$ allora $\lvert\beta\rvert \geq \lvert\alpha\rvert$, ovvero devo generare parole che non siano più corte di quella di partenza. Sono dette *dipendenti dal contesto* perché ogni regola $(\alpha \longrightarrow \beta) \in P$ può essere riscritta come $\alpha_1 A \alpha_2 \longrightarrow \alpha_1 B \alpha_2$, dove α_1 e α_2 rappresentano il *contesto.*	*Automi limitati linearmente.*
Tipo 2, dette **context-free** o *libere dal contesto.*	Le regole in P sono del tipo $\alpha \longrightarrow \beta$, con $\alpha \in V$ e $\beta \in (V \cup \Sigma)^+$.	*Automi a pila.*
Tipo 3, dette **grammatiche regolari**	Le regole in P sono del tipo $A \longrightarrow aB$ oppure $A \longrightarrow a$, con $A, B \in V$ e $a \in \Sigma$. Vale anche il simmetrico.	*Automi a stati finiti.*

Nella figura successiva vediamo una rappresentazione grafica della gerarchia di Chomsky: notiamo come sia una gerarchia propria, ovvero

$$L_3 \subset L_2 \subset L_1 \subset L_0,$$

ma questa gerarchia non esaurisce comunque tutti i linguaggi possibili.



Sia $L \subseteq \Sigma^*$, allora L è di tipo i, con $i \in [0, 3]$, se e solo se esiste una grammatica G di tipo i tale che $L = L(G)$, ovvero posso generare L a partire dalla grammatica di tipo i.

2.4. Potenza computazionale

Se una grammatica é di tipo 1 allora possiamo costruire una macchina che sia in grado di dire, in tempo finito, se una parola appartiene o meno al linguaggio generato da quella grammatica.

Teorema 2.4.1 Una grammatica di tipo 1 è **decidibile**.

Dimostrazione
Siano G una grammatica di tipo 1 e $w \in \Sigma^*$, ci chiediamo se $w \in L(G)$.
Sia $h = |w|$, ma allora essendo G di tipo 1 ogni forma sentenziale che compare in P non deve superare la lunghezza h, altrimenti potremmo ridurre il numero di caratteri presenti nella forma sentenziale e andare contro la definizione di grammatica di tipo 1.

Sia $T_i = \left\{\gamma \in (V \cup \Sigma)^{\leq n} \mid S \overset{\leq i}{\Rightarrow} \gamma\right\}$ l'insieme di tutte le parole generate dalla grammatica G che hanno al massimo n caratteri e sono generate in massimo i passi di derivazione.
Data questa definizione di T_i possiamo affermare che:

- $T_0 = \{S\}$;
- $T_i = T_{i-1} \cup \left\{\gamma \in (V \cup \Sigma)^{\leq n} \mid \exists \beta \in T_{i-1} \text{ tale che } \beta \Rightarrow \gamma\right\}$.

Per come sono costruiti gli insiemi T_i possiamo affermare che

$$T_0 \subseteq T_1 \subseteq \ldots \subseteq (V \cup \Sigma)^{\leq n},$$

ma quest'ultimo insieme è un insieme *finito*.
Prima o poi non si potranno più generare delle stringhe, ovvero $\exists k$ tale che $T_{k-1} = T_k$.
Una volta individuato questo valore k basta controllare se $w \in T_k$. □

Questo non vale invece per le grammatiche di tipo 0: infatti, queste sono dette **semidecidibili**, in quanto un sistema riconoscitivo potrebbe non terminare mai l'algoritmo di riconoscimento e finire quindi in un loop infinito.

Teorema 2.4.2 Una grammatica di tipo 0 è **semidecidibile**.

Dimostrazione
Siano G una grammatica di tipo 0 e $w \in \Sigma^*$, ci chiediamo se $w \in L(G)$.
Non essendo G di tipo 1 non abbiamo il vincolo $|\beta| \geq |\alpha|$ nelle regole di produzione.

Sia $U_i = \left\{ \gamma \in (V \cup \Sigma)^* \mid S \overset{\leq i}{\Rightarrow} \gamma \right\}$ l'insieme di tutte le parole generate dalla grammatica G in massimo i passi di derivazione.
Data questa definizione di U_i possiamo affermare che:
- $U_0 = \{S\}$;
- $U_i = U_{i-1} \cup \left\{ \gamma \in (V \cup \Sigma)^* \mid \exists \beta \in U_{i-1} \text{ tale che } \beta \Rightarrow \gamma \right\}$.

Per come sono costruiti gli insiemi U_i possiamo affermare che

$$U_0 \subseteq U_1 \subseteq ... \subseteq (V \cup \Sigma)^*,$$

ma quest'ultimo insieme è un insieme *infinito*.
Vista questa caratteristica, nessuno garantisce l'esistenza di un k tale che $U_{k-1} = U_k$ e quindi non si ha la certezza di terminare l'algoritmo di riconoscimento. □

Le grammatiche di tipo 0 generano i **linguaggi ricorsivamente enumerabili**: per stabilire se $w \in L(G)$ devo *elencare* con un programma tutte le stringhe del linguaggio e controllare se w compare in esse.

Questa operazione di elencazione in poche parole è la generazione degli insiemi U_i, che poi vengono ispezionati per vedere se la parole w è presente o meno.