Teoria dei Linguaggi

Indice

1. Lezione 01 [26/02]

1.1. Cosa faremo

In questo corso studieremo dei sistemi formali che possiamo quindi descrivere a livello matematico. Questi sistemi descrivono dei linguaggi. Ci chiediamo giustamente cosa sono in grado di fare questi sistemi, ovvero cosa sono in grado di descrivere in termini di linguaggi.

Ci occuperemo anche delle risorse utilizzate dal sistema o delle risorse necessarie per descrivere il linguaggio. Per le prime citate, ci occuperemo del tempo come numero di mosse eseguite da una macchina riconoscitrice oppure del numero di stati per descrivere, ad esempio, una macchina a stati finiti oppure dello spazio utilizzato da una macchina di Turing. Queste ultime due questioni rientrano più nella complessità descrizionale di una macchina.

1.2. 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/e 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**. Anche se ci hanno messo l'equivalente di 18 anni/uomo, questa è la prima applicazione dei linguaggi formali. Con l'avvento, negli anni successivi, dei linguaggi Algol, quindi linguaggi con strutture di controllo, la teoria dei linguaggi formali è diventata sempre più importante.

Oggi la teoria dei linguaggi formali sono usati nei compilatori di compilatori, dei tool usati per generare dei compilatori per un dato linguaggio fornendo la descrizione di quest'ultimo.

1.3. Ripasso

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

Una **stringa** x (o **parola**) è una sequenza finita $x = a_1...a_n$ 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 o λ, che, come dice il nome, ha simboli, ovvero $|\varepsilon| = |\lambda| = 0$ (ogni tanto è Λ).

L'insieme di tutte le possibili parole su Σ è detto Σ^*, ed è un insieme infinito.

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 $\frac{|w||w+1|}{2} + 1$, visti i doppioni.
- **sottosequenza**: x si dice sottosequenza di w se x è ottenuta eliminando 0 o più caratteri da w; in poche parole, x si ottiene da w scegliendo dei simboli IN ORDINE; non devono essere caratteri contigui, basta che una volta scelti i caratteri essi siano mantenuti nell'ordine di apparizione della stringa iniziale;
  - un fattore è una sottosequenza contigua.

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

1.4. Gerarchia di Chomsky

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 è ben bilanciata.
- 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. Lezione 02 [28/02]

2.1. 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; sono anche detti meta-simboli;
- 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 e non vuoto delle **regole di produzione**;
- $S \in V$ **simbolo iniziale** o **assioma**, è il punto di partenza della generazione.

2.1.1. Regole di produzione

Soffermiamoci sulle regole di produzione: la forma di queste ultime è $\alpha \longrightarrow \beta$, con $\alpha \in (V \cup \Sigma)^+$ e $\beta \in (V \cup \Sigma)^*$. Non l'abbiamo detto la scorsa volta, ma la notazione con il $+$ è praticamente Σ^* senza la parola vuota.

Una regola di produzione viene letta come «se ho α allora posso sostituirlo con β».

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 $x_{i-1} \Rightarrow x_i \quad \forall i \in [1, k]$.

Teniamo anche il caso $k = 0$ per dire che da x derivo x stesso, ma è solo per comodità.

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 $x \overset{k}{\Rightarrow} y$.

2.1.2. Linguaggio generato da una grammatica

Indichiamo con $L(G)$ il linguaggio generato dalla grammatica G, ed è l'insieme $\left\{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\right\}$. In poche parole, è l'insieme di tutte le stringhe di non terminali che si possono ottenere tramite **derivazioni** a partire dall'assioma S della grammatica.

In questo insieme abbiamo solo stringhe di non terminali che otteniamo tramite derivazioni. Le stringhe intermedie che otteniamo nei vari passi di derivazioni sono dette **forme sintattiche**.

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:

- $\Sigma = \{a, b\}$;
- $V = \{S, A, B\}$;
- $P = \{S \longrightarrow aB \mid bA, A \longrightarrow a \mid aS \mid bAA, B \longrightarrow b \mid bS \mid aBB\}$.

Questa grammatica genera il linguaggio $L(G) = \{w \in \Sigma^* \mid \#_a(w) = \#_b(w)\}$: infatti, ogni volta che inserisco una a inserisco anche una B per permettere poi di inserire una b. Il discorso vale lo stesso a lettere invertite.

Vediamo un esempio ancora più complesso. Siano:

- $\Sigma = \{a, b\}$;
- $V = \{S, A, B, C, D, E\}$;
- $P = \{S \longrightarrow ABC, AB \longrightarrow \varepsilon \mid aAD \mid bAE, DC \longrightarrow BaC, EC \longrightarrow BbC, Da \longrightarrow aD, Db \longrightarrow bD, Ea \longrightarrow aE, Eb \longrightarrow bE, C \longrightarrow \varepsilon, aB \longrightarrow Ba, bB \longrightarrow Bb\}$.

Questa grammatica genera il linguaggio pappagallo $L(G) = \{ww \mid w \in \Sigma^*\}$: infatti, eseguendo un paio di derivazioni si nota questo pattern.

2.2. Gerarchia di Chomsky

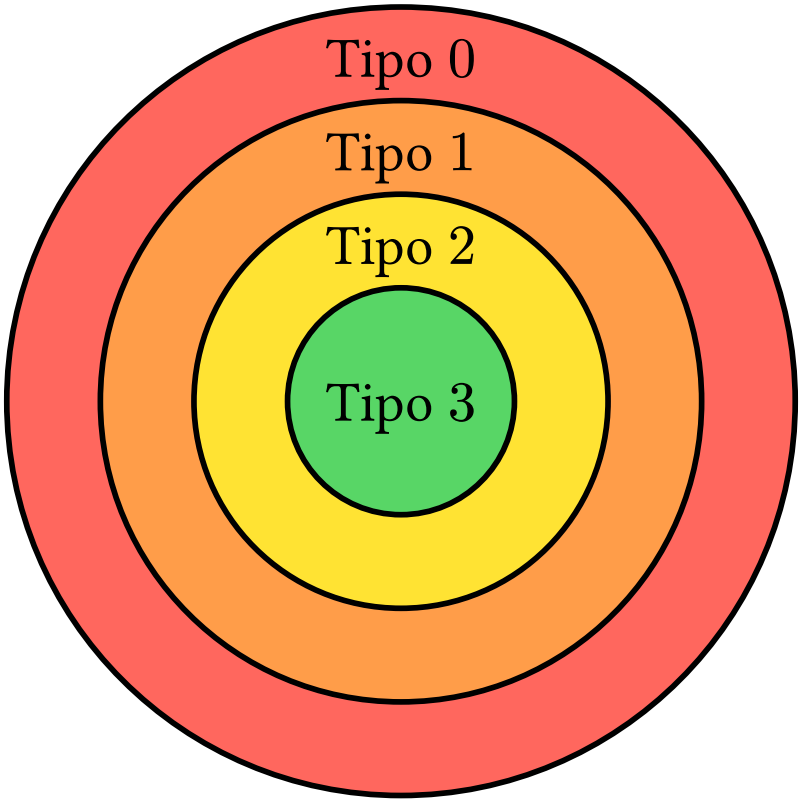
Negli anni ′50 Noam Chomsky studia la generazione dei linguaggi formali e crea una **gerarchia di grammatiche formali**. La classificazione delle grammatiche viene fatta in base alle regole di produzione che definiscono la grammatica.

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 $\alpha_1 A \alpha_2 \longrightarrow \alpha_1 B \alpha_2$, con $\alpha_1, \alpha_2 \in (V \cup \Sigma)^*$ che rappresentano il contesto, $A \in V$ e $B \in (V \cup \Sigma)^+$	**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. Esistono infatti linguaggi che non sono descrivibili in maniera finita con le grammatiche.



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.

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. Questa macchina è detta **verificatore** e si dice che le grammatiche di tipo 1 sono **decidibili**.