

Teoria dei linguaggi

Indice

| | |
|---|----------|
| 1. Introduzione | 2 |
| 1.1. Richiamo sugli insiemi | 2 |
| 1.2. Richiamo sugli alfabeti e i linguaggi | 2 |
| 1.3. Gerarchia di Chomsky | 2 |
| 1.4. Richiamo sulla notazione delle parole | 3 |
| 2. Grammatiche di tipo 3 | 4 |
| 2.1. Automi a stati finiti | 4 |
| 2.1.1. Definizione informale | 4 |
| 2.1.2. Definizione formale | 4 |
| 2.1.3. Linguaggio | 5 |
| 2.1.4. Rappresentazione grafica | 5 |
| 2.2. Automi a stati finiti non deterministici | 5 |
| 2.2.1. Definizione informale | 5 |
| 2.2.2. Definizione formale | 5 |
| 2.3. Distinguibilità | 6 |

1. Introduzione

1.1. Richiamo sugli insiemi

Un **insieme** S è una collezione di elementi di un certo dominio U

Fissato un insieme S , se è finito allora $|S|$ indica la sua **cardinalità**, ovvero il numero di elementi che contiene

Un insieme particolare è l'**insieme vuoto** \emptyset , l'unico insieme che ha cardinalità 0

Sugli insiemi possiamo definire alcune operazioni:

- *intersezione* $A \cap B$ contiene gli elementi comuni di A e B
- *unione* $A \cup B$ contiene gli elementi di A e B assieme
- *differenza* $A - B$ o A/B contiene gli elementi di A che non sono in B
- *complementare* A^c o \bar{A} contiene gli elementi di U che non sono in A
- *sottoinsieme* $A \subset B$ proprio oppure $A \subseteq B$ non proprio

1.2. Richiamo sugli alfabeti e i linguaggi

Un **alfabeto** è un insieme di caratteri e simboli sul quale è possibile definire un **linguaggio**: quest'ultimo infatti è un insieme di parole costruite a partire dall'alfabeto dato

Ogni **parola** è una sequenza di caratteri (detti anche simboli o lettere) dell'alfabeto

Tra le parole di un linguaggio c'è anche la **parola vuota** ϵ

Un linguaggio può essere **finito** (linguaggio composto dalle parole italiane) oppure **infinito** (linguaggio sull'alfabeto $\{ (,) \}$ delle parole ben bilanciate)

Per rappresentare i linguaggi possiamo usare diversi modelli

- **dichiarativo**: date delle regole, si verifica se una parola rispetta o meno queste regole, come nelle *espressioni regolari*
- **generativo**: date delle regole, si parte da *un certo punto* e si generano tutte le parole di quel linguaggio con le regole date
- **ricognoscitivo**: 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 derivare tutte le parole di questo linguaggio

Proviamo a definire alcune regole per 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

I modelli generativi generano le **grammatiche**

- *ricognoscitivo*: abbiamo una *black-box* che prende una parola e ci dice se appartiene o meno al linguaggio (in realtà questa *black-box* potrebbe non terminare mai la sua esecuzione)

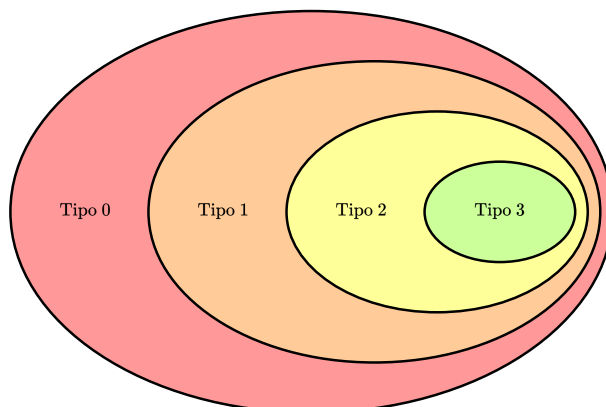
Proviamo a definire alcune regole per riconoscere tutte le parole di questo linguaggio:

- il numero di $($ è uguale al numero di $)$
- per ogni prefisso, il numero di $($ deve essere maggiore o uguale al numero di $)$

1.3. Gerarchia di Chomsky

Negli anni 50 **Noam Chomsky** studia la generazione dei linguaggi formali e crea una **gerarchia di grammatiche formali**

- **tipo 0:** grammatiche che generano tutti i linguaggi, sono senza restrizioni e come modello equivalente hanno le **macchine di Turing**
- **tipo 1:** grammatiche *context-sensitive* (dipendenti dal contesto), come modello equivalente hanno le **Linear Bounded Automata**
- **tipo 2:** grammatiche *context-free* (libere dal contesto), hanno come modello equivalente gli **automi a pila**
- **tipo 3:** grammatiche regolari, hanno come modello equivalente gli **automi a stati finiti**



1.4. Richiamo sulla notazione delle parole

Andremo ad indicare con le lettere maiuscole dell'alfabeto greco un **alfabeto**, come ad esempio Σ o Γ

Sia Σ un alfabeto, indichiamo con Σ^* tutte le parole sull'alfabeto Σ , compresa la parola vuota ε , e con Σ^+ tutte le parole non vuote sull'alfabeto Σ : in poche parole, $\Sigma^+ = \Sigma^* / \{\varepsilon\}$

Vediamo prima alcune operazioni sulle parole

- la **concatenazione** è la composizione di due parole x, y che formano la parola $w = xy$, e in generale $w_1 = xy \neq yz = w_2$
- la **lunghezza** di una parola w si indica con $|w|$
- il **numero di occorrenze** di una lettera $a \in \Sigma$ nella parola w si indica con $|w|_a$

Vediamo ora alcune proprietà delle parole

- **fattore:** y si dice *fattore* di w se esistono $x, z \in \Sigma^*$ tali che $xyz = w$
- **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 $x \neq \varepsilon$
- **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 $y \neq \varepsilon$
- **sottosequenza:** x si dice *sottosequenza* di w se x è ottenuta eliminando 0 o più caratteri da w , anche non in ordine

Terminiamo definendo l'operazione **reversal**, ovvero se $w = a_1 a_2 \dots a_n$, allora $w^R = a_n a_{n-1} \dots a_1$

Una parola w si dice **palindroma** se $w = w^R$

2. Grammatiche di tipo 3

2.1. Automi a stati finiti

Gli **automi a stati finiti** sono un modello riconoscitivo usato per caratterizzare i *linguaggi regolari*

2.1.1. Definizione informale

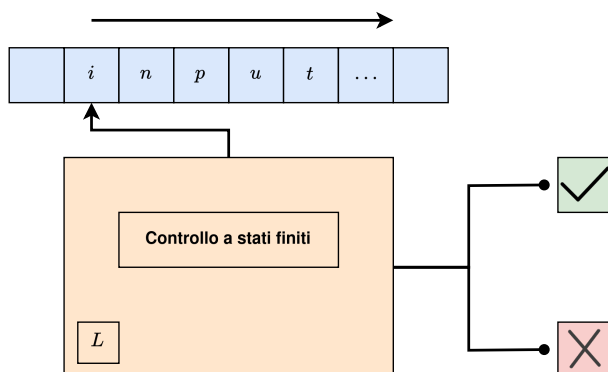
Gli automi a stati finiti sono delle macchine molto semplici: hanno un **controllo a stati finiti** che legge l'input da un **nastro**, formato da una serie di celle, ognuna delle quali contiene un carattere dell'input

Per leggere l'input si utilizza una **testina di lettura**, posizionata inizialmente sulla cella più a sinistra (ovvero sul primo carattere di input), e poi spostata iterazione dopo iterazione da sinistra verso destra

Il controllo a stati finiti, prima della lettura dell'input, è allo **stato iniziale**

Ad ogni iterazione, a partire dallo stato corrente e dal carattere letto dal nastro, ci si muove con la testina a destra sul simbolo successivo e si cambia stato

Quando si arriva alla fine del nastro, in base allo stato corrente dell'automa, quest'ultimo risponde "sì", ovvero la parola in input appartiene al linguaggio, oppure "no", ovvero la parola in input non appartiene al linguaggio



2.1.2. Definizione formale

Un automa è una tupla $\{Q, \Sigma, \delta, q_I, F\}$, con

- Q insieme degli stati
- Σ alfabeto di input
- $\delta : Q \times \Sigma \rightarrow Q$ funzione di transizione
- $q_I \in Q$ stato iniziale
- $F \subseteq Q$ insieme degli stati finali

La parte dinamica dell'automa è la **funzione di transizione** che, dati lo stato iniziale e un simbolo del linguaggio, calcola lo stato successivo

Possiamo estendere la funzione di transizione affinché utilizzi una parola del linguaggio, ovvero definiamo $\bar{\delta} : Q \times \Sigma^* \rightarrow Q$ tale che

- $\bar{\delta}(q, \epsilon) = q \quad \forall q \in Q$
- $\bar{\delta}(q, wa) = \delta(\bar{\delta}(q, w), a) \quad \forall q \in Q, w \in \Sigma^*, a \in \Sigma$

Per semplicità useremo δ al posto di $\bar{\delta}$ nella notazione

2.1.3. Linguaggio

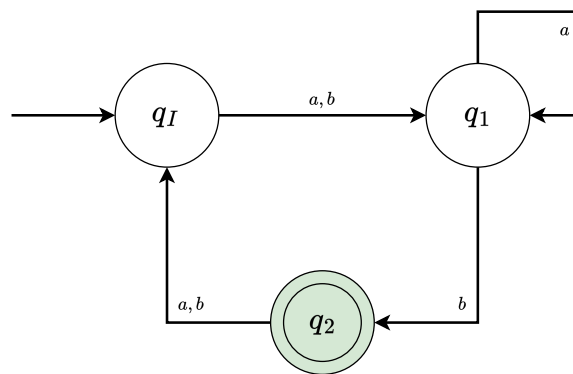
Chiamiamo $L(A) = \{w \in \Sigma^* \mid \delta(q_I, w) \in F\}$ il **linguaggio riconosciuto dall'automa**, ovvero l'insieme delle parole che applicate alla funzione di transizione, a partire dallo stato iniziale, mi mandano in uno stato finale

2.1.4. Rappresentazione grafica

Possiamo vedere un automa come un grafo, dove

- gli **stati** sono nodi etichettati con il nome dello stato
- le **transizioni** sono archi orientati ed etichettati con la lettera dell'alfabeto che causa quella transizione

Lo **stato iniziale** è indicato con una freccia entrante nello stato, mentre gli **stati finali** sono nodi doppiamente cerchiati



2.2. Automi a stati finiti non deterministici

Questi particolari automi sono utili nei linguaggi R_n (fatti come esempio a lezione) perché necessitano di molto meno stati ($n - 1$) rispetto ad un automa deterministico (2^n)

2.2.1. Definizione informale

Gli **automi a stati finiti non deterministici** (NFA) sono particolari automi che hanno *almeno* uno stato dal quale escono 2 o più archi con la stessa lettera

Negli automi **deterministici** (DFA) invece da *ogni* stato esce al più un arco con la stessa lettera

La differenza principale sta nella complessità computazionale: se negli automi deterministici devo controllare se la parola ci porta in uno stato finale, negli automi non deterministici devo controllare se tra *tutti* i possibili cammini dell'**albero di computazione** ne esiste uno che ci porta in uno stato finale

Possiamo vedere il non determinismo come *una scommessa che va sempre a buon fine*

2.2.2. Definizione formale

Un automa non deterministico differisce da un automa deterministico solo per la funzione di transizione: infatti, quest'ultima diventa $\delta : Q \times \Sigma \longrightarrow \mathbb{P}(Q)$, ovvero ritorna un elemento dell'**insieme delle parti** di Q , quindi un sottoinsieme di stati nei quali possiamo finire applicando un carattere di Σ allo stato corrente

Come prima, definiamo l'estensione della funzione di transizione come la funzione $\bar{\delta} : Q \times \Sigma^* \longrightarrow \mathbb{P}(Q)$ tale che

- $\bar{\delta}(q, \varepsilon) = \{q\}$
- $\bar{\delta}(q, wa) = \bigcup_{r \in \bar{\delta}(q, w)} \delta(r, a)$

Cambia anche il linguaggio riconosciuto dall'automa: infatti, $L(A)$ diventa $\{w \in \Sigma^* \mid \bar{\delta}(q_I, w) \cap F \neq \emptyset\}$

2.3. Distinguibilità

Dato un linguaggio $L \subseteq \Sigma^*$, due parole $x, y \in \Sigma^*$ sono **distinguibili** rispetto ad L se $\exists z \in \Sigma^*$ tale che $(xz \in L \wedge yz \notin L) \vee (xz \notin L \wedge yz \in L)$

Teorema 2.3.1 Siano $L \subseteq \Sigma^*$ un linguaggio, $X \subseteq \Sigma^*$ un insieme di parole tutte distinguibili tra loro rispetto ad L e A un automa DFA per L , allora A non può avere meno di $|X|$ stati

Dimostrazione

Per assurdo, sia $X = \{x_1, \dots, x_k\}$ insieme di k parole distinguibili tra loro rispetto ad L e che A abbia meno di k stati

Partendo dallo stato corrente q_I ho due situazioni

- leggendo il carattere x_i , con $1 \leq i \leq k - 1$, l'automa finisce nello stato q_i
- leggendo i caratteri x_{k-1} e x_k l'automa finisce nello stato q_t , visto che A ha meno di k stati

Prendiamo ora una parola z e la applichiamo allo stato q_t , che ci porta in uno stato r qualsiasi, ma questo è assurdo: infatti, abbiamo due parole distinguibili (x_{k-1} e x_k) che però sono entrambe accettate o entrambe rifiutate, in base allo stato r □