

Appunti di Automi e Linguaggi Formali

17 Agosto 2023

Rosso Carlo

Contents

1	Introduzione	2
2	Automi a stati finiti	2
2.1	Definizione	2
2.2	Operazioni sui linguaggi regolari	2
2.3	Automa a stati finiti non deterministico	3
2.4	Espressioni regolari	3
2.5	Pumping Lemma	4
3	Grammatica context-free	4
3.1	Forma normale	4
3.2	Automa a pila	5
3.3	Pumping Lemma	5
4	Macchina di Turing	5
5	Decidibilità	6

1 Introduzione

La teoria della computazione si pone una domanda: che cos'è un computer? Idealizziamo un computer come un modello computazionale. Un modello computazionale può essere preciso in qualche aspetto, ma non in altri. Per questo motivo studiamo diversi modelli computazionali, in base alle caratteristiche che ci interessano. Cominciamo con il modello computazionale più semplice: l'automa a stati finiti (DFA, Deterministic Finite Automaton).

2 Automi a stati finiti

2.1 Definizione

Def. 2.1 (Automa a stati finiti) Un automa finito è una quintupla $(Q, \Sigma, \delta, q_0, F)$ dove:

1. Q è un insieme finito chiamato insieme degli stati;
2. Σ è un insieme finito chiamato alfabeto;
3. $\delta : Q \times \Sigma \rightarrow Q$ è la funzione di transizione;
4. $q_0 \in Q$ è lo stato iniziale;
5. $F \subseteq Q$ è l'insieme degli stati finali.

Se A è l'insieme delle stringhe accettate da un DFA M , allora A è il linguaggio di M , e lo indichiamo con $L(M)$. E diciamo che M accetta A .

Def. 2.2 (Computazione) Sia $M = (Q, \Sigma, \delta, q_0, F)$ un DFA e $w = w_1w_2 \dots w_n$ una stringa su Σ . Diciamo che M accetta w se esiste una sequenza di stati (computazione) r_0, r_1, \dots, r_n in Q tale che:

1. $r_0 = q_0$;
2. $r_{i+1} = \delta(r_i, w_{i+1})$ per $i = 0, 1, \dots, n-1$;
3. $r_n \in F$.

Def. 2.3 (Linguaggio regolare) Un linguaggio si dice regolare se qualche DFA lo accetta.

2.2 Operazioni sui linguaggi regolari

Def. 2.4 (Unione) Siano A e B due linguaggi.

$$A \cup B = \{x \mid x \in A \text{ oppure } x \in B\}.$$

Def. 2.5 (Concatenazione) Siano A e B due linguaggi.

$$A \circ B = \{xy \mid x \in A \text{ e } y \in B\}.$$

Def. 2.6 (Star) Sia A un linguaggio.

$$A^* = \{x_1x_2 \dots x_k \mid k \geq 0 \text{ e } x_i \in A \text{ per } i = 1, 2, \dots, k\}.$$

2.3 Automa a stati finiti non deterministico

Definiamo ϵ come la stringa vuota e $\mathcal{P}(A)$ come l'insieme delle parti di A .

Def. 2.7 (Automa a stati finiti non deterministico) Un automa a stati finiti non deterministico (NFA, Nondeterministic Finite Automaton) è una quintupla $(Q, \Sigma, \delta, q_0, F)$ dove:

1. Q è un insieme finito chiamato insieme degli stati;
2. Σ è un insieme finito chiamato alfabeto;
3. $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$ è la funzione di transizione;
4. $q_0 \in Q$ è lo stato iniziale;
5. $F \subseteq Q$ è l'insieme degli stati finali.

NB: δ è una funzione che mappa un elemento di Q e un elemento di $\Sigma \cup \{\epsilon\}$ in un sottoinsieme di Q , quindi anche \emptyset .

Def. 2.8 (Computazione) Sia $M = (Q, \Sigma, \delta, q_0, F)$ un DFA e $w = w_1 w_2 \dots w_n$ una stringa su Σ . Diciamo che M accetta w se esiste una sequenza di stati (computazione) r_0, r_1, \dots, r_n in Q tale che:

1. $r_0 = q_0$;
2. $r_{i+1} \in \delta(r_i, w_{i+1})$ per $i = 0, 1, \dots, n-1$;
3. $r_n \in F$.

Def. 2.9 (Equivalenza tra automi) Diciamo che due automi M_1 e M_2 sono equivalenti se $L(M_1) = L(M_2)$.

Teorema 2.1 (Equivalenza tra DFA e NFA) Ogni automa a stati finiti non deterministico ha un equivalente deterministico.

2.4 Espressioni regolari

Def. 2.10 (Espressione regolare) R è un'espressione regolare se vale una delle seguenti:

- a per qualche $a \in \Sigma$;
- ϵ ;
- \emptyset ;
- $(R_1 \cup R_2)$ dove R_1 e R_2 sono espressioni regolari;
- $(R_1 \circ R_2)$ dove R_1 e R_2 sono espressioni regolari;
- (R_1^*) dove R_1 è un'espressione regolare.

Teorema 2.2 Un linguaggio è regolare se e solo se qualche espressione regolare lo genera.

2.5 Pumping Lemma

Teorema 2.3 (Pumping Lemma) *Sia A un linguaggio regolare. Allora esiste un numero p (detto pumping length) tale che per ogni stringa $s \in A$ con $|s| \geq p$ esiste una suddivisione $s = xyz$ tale che:*

1. $|y| > 0$;
2. $|xy| \leq p$;
3. per ogni $i \geq 0$, $xy^iz \in A$.

3 Grammatica context-free

Def. 3.1 (Grammatica context-free) *Una grammatica context-free (CFG, Context-Free Grammar) è una quadrupla (V, Σ, R, S) dove:*

1. V è un insieme finito chiamato insieme delle variabili;
2. Σ è un insieme finito, disgiunto da V , chiamato insieme dei terminali;
3. R è un insieme finito di regole, ciascuna delle quali è una coppia (v, w) dove $v \in V$ e $w \in (V \cup \Sigma)^*$;
4. $S \in V$ è la variabile iniziale.

È intuitivo, ma vediamo come costruire una grammatica CF a partire da un DFA:

1. $\forall q_i \in Q \exists r_i \in R$;
2. $\Sigma_{CFG} = \Sigma_{DFA}$;
3. $r_i \rightarrow ar_j \in R$ per ogni $q_j = \delta(q_i, a)$, tale che $a \in \Sigma_{DFA}$;
4. $r_i \rightarrow \epsilon \in R$ per ogni $q_i \in F$;
5. $S = q_0$.

Def. 3.2 (Ambiguità) *Una grammatica è ambigua se esiste una stringa w tale che w ha due alberi di derivazione distinti. In questo caso si dice che w si deriva in modo ambiguo.*

3.1 Forma normale

Def. 3.3 (Forma normale di Chomsky) *Una grammatica è in forma normale di Chomsky se tutte le sue regole sono della forma:*

- $A \rightarrow BC$;
- $A \rightarrow a$.

dove $A, B, C \in V$ e $a \in \Sigma$ e $B, C \neq S$.

Teorema 3.1 (Forma normale di Chomsky) *Ogni linguaggio context-free è generato da una grammatica context-free in forma normale di Chomsky.*

3.2 Automa a pila

Def. 3.4 (Automa a pila) Un automa a pila (PDA, pushdown automaton) è una 6-upla $(Q, \Sigma, \Gamma, \delta, q_0, F)$, dove $Q, \Sigma, \Gamma \in F$ sono insiemi finiti e:

- Q è l'insieme degli stati;
- Σ è l'alfabeto di input;
- Γ è l'alfabeto dello stack;
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \epsilon) \rightarrow \mathcal{P}(Q \times (\Gamma \cup \{\epsilon\}))$ è la funzione di transizione;
- $q_0 \in Q$ è lo stato iniziale;
- $F \subseteq Q$ è l'insieme degli stati finali.

Def. 3.5 (Computazione) Un PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ accetta l'input w se w può essere scritta come $w = w_1 w_2 \dots w_n$ dove $w_i \in \Sigma$, la sequenza di stati $r_0, r_1, \dots, r_n \in Q$ e le stringhe $s_0, s_1, \dots, s_n \in \Gamma^*$ esistono tali che valgono le seguenti condizioni:

1. $r_0 = q_0$ e $s_0 = \epsilon$;
2. $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$ e $s_{i+1} = bt$ e $s_i = at$ per qualche $t \in \Gamma^*$ e $a, b \in \Gamma \cup \{\epsilon\}$;
3. $r_n \in F$.

Teorema 3.2 (Equivalenza tra PDA e CFG) Un linguaggio è context-free se e solo se è accettato da un PDA.

Dal precedente teorema otteniamo il corollario: ogni linguaggio regolare è context-free.

3.3 Pumping Lemma

Teorema 3.3 (Pumping Lemma) Sia A un linguaggio context-free. Allora esiste un numero p (detto pumping length) tale che per ogni stringa $s \in A$ con $|s| \geq p$ esiste una suddivisione $s = wxyz$ tale che:

1. $|vy| > 0$;
2. $|vxy| \leq p$;
3. per ogni $i \geq 0$, $wv^i x y^i z \in A$.

4 Macchina di Turing

Def. 4.1 (Macchina di Turing) Una macchina di Turing (TM, Turing Machine) è una 7-upla $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$, dove Q, Σ, Γ sono insiemi finiti e valgono le seguenti condizioni:

1. Q è l'insieme degli stati;
2. Σ è l'alfabeto di input, non contiene il carattere speciale \sqcup ;
3. Γ è l'alfabeto del nastro, tale che $\Sigma \subseteq \Gamma$ e $\sqcup \in \Gamma$;
4. $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ è la funzione di transizione;
5. $q_0 \in Q$ è lo stato iniziale;

6. $q_{acc} \in Q$ è lo stato di accettazione;
7. $q_{rej} \in Q$ è lo stato di rifiuto, dove $q_{rej} \neq q_{acc}$.

Def. 4.2 (Configurazione) Una configurazione di una TM è una tripla (q, x, y) dove $q \in Q$ è lo stato corrente, $x \in \Gamma^*$ è la stringa sul nastro che precede la posizione della testina e $y \in \Gamma^*$ è la stringa non vuota che segue x , la testina si trova nel primo carattere di y . Un altro modo per rappresentare una configurazione è xqy .

NB: la configurazione iniziale è q_0w dove w è la stringa di input. In ogni configurazione di accettazione, lo stato corrente è q_{acc} . In ogni configurazione di rifiuto, lo stato corrente è q_{rej} . Le configurazioni di accettazione e di rifiuto sono dette configurazioni finali e bloccano la TM.

Def. 4.3 (Computazione) Si dice che una TM M accetta l'input w se esiste una sequenza di configurazioni C_1, C_2, \dots, C_k tale che:

1. $C_1 = q_0w$;
2. C_i genera C_{i+1} per $i = 1, \dots, k-1$;
3. $C_k = q_{acc}$;

L'insieme di stringhe che M accetta è detto linguaggio di M e si indica con $L(M)$.

Def. 4.4 (Turing riconoscibile) Un linguaggio è Turing riconoscibile se è accettato da una TM.

Quando facciamo partire una TM su un input, ci sono tre possibili risultati:

1. la TM si blocca in uno stato di accettazione;
2. la TM si blocca in uno stato di rifiuto;
3. la TM entra in un loop infinito.

Def. 4.5 (Decisore) Una TM è un decisore se si blocca in uno stato di accettazione o di rifiuto per ogni input.

Def. 4.6 (Turing decidibile) Un linguaggio è Turing decidibile se è accettato da un decisore.

5 Decidibilità

Teorema 5.1 (A_{DFA}) Sia $A_{DFA} = \{\langle B, w \rangle \mid B \text{ è un DFA che accetta } w\}$. Allora A_{DFA} è decidibile.

NB: Si simula la computazione del DFA su w e si controlla se si controlla in quale stato si arriva alla fine della computazione.

Teorema 5.2 (A_{NFA}) Sia $A_{NFA} = \{\langle B, w \rangle \mid B \text{ è un NFA che accetta } w\}$. Allora A_{NFA} è decidibile.

NB: Si converte un NFA in DFA e si applica il teorema precedente.

Teorema 5.3 (A_{REGEX}) Sia $A_{REGEX} = \{\langle R, w \rangle \mid R \text{ è una regex che genera } w\}$. Allora A_{REGEX} è decidibile.

NB: Si converte la regex in un DFA e si applica il teorema precedente.

Teorema 5.4 (E_{DFA}) Sia $E_{DFA} = \{\langle A \rangle \mid A \text{ è un DFA tale che } L(A) = \emptyset\}$. Allora E_{DFA} è decidibile.

NB: Costruiamo la TM T che decide E_{DFA} :

$T =$ "Sulla stringa di input $\langle A \rangle$, dove A è un DFA:

1. Segna lo stato iniziale di A ;
2. Ripeti finché nessuno stato nuovo viene segnato:
 - (a) Segna ogni stato raggiungibile da uno stato segnato;
3. Se uno stato finale non è segnato, allora accetta, altrimenti rifiuta.

Teorema 5.5 ($E_{Q_{DFA}}$) Sia $E_{Q_{DFA}} = \{\langle A, B \rangle \mid A \text{ e } B \text{ sono DFA e } L(A) = L(B)\}$. Allora $E_{Q_{DFA}}$ è decidibile.

NB: Se $L(C) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B)) = \emptyset$ allora $L(A) = L(B)$.

Teorema 5.6 (A_{CFG}) Sia $A_{CFG} = \{\langle G, w \rangle \mid G \text{ è una CFG che genera } w\}$. Allora A_{CFG} è decidibile.

Teorema 5.7 (E_{CFG}) Sia $E_{CFG} = \{\langle G \rangle \mid G \text{ è una CFG tale che } L(G) = \emptyset\}$. Allora E_{CFG} è decidibile.

NB: Costruiamo la TM T che decide E_{CFG} :

$T =$ "Sulla stringa di input $\langle G \rangle$, dove G è una CFG:

1. Segna tutti i simboli terminali di G ;
2. Ripeti finché nessuna nuova variabile viene segnata:
 - (a) Segna ogni variabile V tale che G abbia una regola del tipo $V \rightarrow U_1 U_2 \dots U_k$ tali che ogni U_i è già stato segnato;
3. Se la variabile di partenza di G non è segnata, allora accetta, altrimenti rifiuta.