

Concetti di base sugli automi e sui linguaggi formali

Andrea Burattin

18 marzo 2005

Sommario

Piccolo insieme di concetti sul funzionamento degli automi (a stati finiti, a pila, ...), delle grammatiche libere dal contesto, le macchine di Turing e i linguaggi (ricorsivi, ricorsivamente enumerabili, in P , in NP , gli NP -completi).

Indice

1	Introduzione	1
2	Automi e macchine di Turing	2
2.1	Automi a stati finiti ed espressioni regolari	2
2.1.1	Esempio di automa a stati finiti	2
2.2	Grammatiche libere dal contesto (CFG)	2
2.3	Automi a pila	3
2.4	Problemi decidibili ed indecidibili	3
2.5	Macchine di Turing	3
3	Tipi di linguaggi	4
3.1	Linguaggi ricorsivi	4
3.2	Linguaggi ricorsivamente enumerabili	4
3.3	Linguaggi in P ed in NP	4
3.4	Linguaggi NP -completi	4
4	Riduzione	5

1 Introduzione

Un automa è un dispositivo astratto, utilizzato per realizzare delle computazioni. Gli automi sono utilizzati come modello nella progettazione di circuiti digitali, negli analizzatori lessicali di un compilatore, per la ricerca di parole chiave su file o (in maniera pi estesa) sul web, o per software per la verifica di sistemi a stati finiti, come i protocolli di comunicazione.

Turing ha studiato e definito le *macchine di Turing* prima che esistessero i calcolatori. Le macchine di Turing possono essere idealmente rappresentate come computer astratti.

2 Automi e macchine di Turing

2.1 Automi a stati finiti ed espressioni regolari

Un automa a stati finiti è una quintupla formata da $A = (Q, \Sigma, \delta, q_0, F)$:

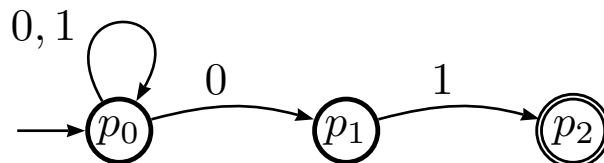
1. Q : insieme finito degli stati dell'automa
2. Σ : alfabeto del linguaggio
3. δ : insieme delle transizioni di stato
4. q_0 : unico stato iniziale
5. F : insieme finito di stati finali

2.1.1 Esempio di automa a stati finiti

Si voglia costruire un automa in grado di identificare le stringhe caratterizzate da:

$$L = \{\text{tutte le stringhe che finiscono con } 01\}$$

Eccone una rappresentazione grafica:



Un automa a stati finiti, espresso in maniera formale ed in forma algebrica prende il nome di **espressione regolare**. Ecco un esempio di espressione regolare che possa generare le stringhe del linguaggio visto precedentemente (tutte le stringhe formate da 0 e 1, che terminano con 01):

$$(0 + 1) * 01$$

2.2 Grammatiche libere dal contesto (CFG)

Una grammatica libera dal contesto è formata da:

- un insieme finito di simboli che formano stringhe del linguaggio. Questi elementi vengono comunemente detti *terminali*;
- un insieme finito di variabili (detti simboli *non terminali*) che rappresentano un linguaggio (insieme di stringhe);
- un simbolo iniziale che rappresenta il linguaggio da definire;
- un insieme delle produzioni che rappresentano la definizione ricorsiva del linguaggio. Una produzione è formata da:

1. la *testa della produzione*: la variabile definita dalla produzione;
2. il *simbolo di produzione* (\rightarrow);
3. il *corpo*: rappresenta un modo per formare un linguaggio della variabile di testa.

L'estensione del simbolo iniziale, tramite le sue produzioni, fino ad ottenere una stringa formata esclusivamente da simboli terminali è detta **derivazione**. Se in una grammatica è possibile trovare una stringa terminale utilizzando pi alberi sintattici, allora la grammatica viene definita **ambigua**.

Esempio di grammatica del linguaggio L_1 :

$$L_1 = \{a^n b^n, \text{ con } n \geq 0\}$$

$$S \rightarrow aSb | \epsilon$$

2.3 Automi a pila

Una automa a pila è una settupla formata da:

1. Q : insieme finito degli stati dell'automa;
2. Σ : alfabeto del linguaggio;
3. Γ : insieme finito dei simboli accettati dalla pila;
4. δ : insieme delle transizioni di stato;
5. q_0 : unico stato iniziale;
6. z_0 : simbolo iniziale della pila;
7. F : insieme finito di stati finali.

2.4 Problemi decidibili ed indecidibili

Un problema si dice **indecidibile** se non esiste alcun programma che lo possa risolvere. Si dice **decidibile** se esiste un programma che lo possa risolvere.

Esistono anche i problemi **semi-decidibili** che consistono dei problemi che danno soluzione in tempo finito solo se abbiamo esito positivo.

2.5 Macchine di Turing

Una macchina di Turing (in seguito chiamata pi semplicemente TM) è un automa a stati finiti con un nastro di lunghezza infinita su cui può leggere e scrivere.

Una TM fa una mossa in funzione del suo stato e del simbolo sotto la testina di lettura. In una mossa, la TM

1. cambia stato;
2. scrive un simbolo nel nastro;
3. muove la testina a destra $\{R\}$ o a sinistra $\{L\}$.

Formalmente una macchina di Turing può essere descritta come una settupla formata da:

1. Q : insieme finito degli stati dell'automa;
2. q_0 : stato iniziale;
3. $F \subseteq Q$: insieme finito di stati finali;
4. Σ : insieme finito simboli di input;
5. Γ : insieme finito simboli di nastro;
6. $B \in \Gamma$: simbolo del blank;
7. δ : insieme delle transizioni da $Q \times \Gamma$ a $Q \times \Gamma \times \{L, R\}$.

Se una TM entra in uno stato q_n in cui non è definita alcuna δ , allora si dice che la TM si arresta; se una TM accetta una stringa, si può assumere che si arresti.

3 Tipi di linguaggi

3.1 Linguaggi ricorsivi

Tutti quei linguaggi per cui è possibile costruire una TM che si arresta **sempre**. Questi linguaggi sono detti anche decidibili. Se L (linguaggio) è ricorsivo allora anche \bar{L} (il complementare di L) è ricorsivo.

3.2 Linguaggi ricorsivamente enumerabili

Un linguaggio si dice ricorsivamente enumerabile (in seguito chiamato più semplicemente RE) se esiste una TM che si arresta in caso di risultato positivo e continua all'infinito in caso negativo.

- se L è ricorsivo, allora anche \bar{L} è ricorsivo
- se L è RE ma \bar{L} non è RE, allora L non può essere ricorsivo
- se L e \bar{L} sono RE, allora L è ricorsivo

3.3 Linguaggi in P ed in NP

Un linguaggio che è ricorsivo, quindi decidibile, può essere verificato in tempo polinomiale. La classe di linguaggi P identifica questi linguaggi. Qualora un linguaggio, per essere accettato impieghi tempo polinomiale ma implichi il non determinismo (parallelismo di operazioni) della TM, allora questo linguaggio si definisce NP (non deterministico, polinomiale). Per ottenere il non determinismo su una TM è necessario lavorare contemporaneamente su più nastri.

3.4 Linguaggi NP -completi

Un linguaggio si dice NP -completo se è in NP e se, per ogni altro linguaggio in NP , esiste una riduzione polinomiale che lo possa ricondurre al linguaggio che si vuole dichiarare NP -completo.

4 Riduzione

Sia P_1 indecidibile, vogliamo capire se P_2 lo è a sua volta.

Convertendo qualunque stringa di P_1 a stringa di P_2 e qualunque non-stringa di P_1 a non-stringa di P_2 allora si riduce P_1 a P_2 e si verifica che sono indecidibili.

Qualora una riduzione avvenga in tempo polinomiale, essa verr definita **riduzione polinomiale**.

Riferimenti bibliografici

[HMU03] Hopcroft John E., Motwani Rajeev, Ullman Jeffrey D. *Automi, linguaggi e calcolabilità*, 2003. Pearson Education Italia, 1 edition, 2003.