Appunti del corso di Automi e Linguaggi Formali

Paolo Broglio

June 27, 2017

Contents

1 La macchina di Turing

1.1 La definizione

Una *Macchina di Turing*, chiamata **TM (Turing Machine)** viene descritta come una settupla

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

i cui componenti hanno i seguenti significati:

- Q: insieme di *stati* di controllo;
- Σ: insieme dei *simboli di input*, ed è sempre sottinsieme di Γ;
- Γ: insieme dei *simboli di nastro*;
- δ: la *funzione di transizione*;
- q_0: è lo *stato iniziale*, appartiene a Q;
- B: simbolo detto *blank*;
- F: insieme degli *stati finali o accettanti*, è sottinsieme di Q.

Un esempio di una TM che accetta il linguaggio $\{0^{\mathrm{n}}1^{\mathrm{n}}|n \geq 1\}$.

$$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$$

Questo è un esempio di come può essere rappresentata una macchina di Turing. Per capire come funziona bisogna utilizzare i **diagrammi di transizione** per δ oppure costruendo il **grafo** correlato.

1.2 Descrizioni istantanee

Le descrizioni istantanee (**ID, Istantaneous Descriptions**), sono usate per illustrare un'esecuzione di una certa TM che ha ricevuto un certo input in ingresso. Di seguito è riportato un esempio di come utilizzare le ID mostrando l'esecuzione della TM definita sul linguaggio precedentemente descritto(numero uguale di 0 e 1).
L'*input* è la stringa 0010.

$$q_0 0010 \vdash Xq_1 010 \vdash X0q_1 101 \vdash Xq_2 0Y0 \vdash q_2 X0Y0 \vdash \tag{1}$$

$$Xq_0 0Y0 \vdash XXq_1 Y0 \vdash XXYq_1 0 \vdash XXY0q_1 B \tag{2}$$

In questo caso l'esecuzione termina con la TM che non sa cosa fare nel momento in cui legge B nell'ultimo stato indicato. In effetti 0010 non si trova nel linguaggio.

2 Teoria del capitolo 8

2.1 Linguaggio RE

I linguaggi che possiamo accettare usando una macchina di Turing sono denominati **Linguaggi Recorsivamente Enumerabili o RE**. Più formalmente un linguaggio è **RE** se, data una TM M vale che

$$L = L(M)$$

3 Teoria del capitolo 9

3.1 Linguaggio di diagonalizzazione

Data una TM M e un linguaggio L,

- il linguaggio L_{d}, detto **linguaggio di diagonalizzazione**, è l'insieme dell stringhe w_{i} tali che w_{i} *non è* in $L(M_{\mathrm{i}})$.

Quindi L_{d} consiste di tutte le stringhe w tali che la TM M con codice w non accetta quando riceve w come input.

3.1.1 Teorema su $L_{\mathbf{d}}$

L_{d} **non è** un linguaggio ricorsivamente enumerabile. In altre parole non esiste alcuna macchina di Turing che accetta L_{d}.

3.2 Linguaggio ricorsivo

Dato un linguaggio L, si definisce **ricorsivo** se $L = L(M)$ per una TM M tale che:

- se w è in L, allora M accetta(e dunque si **arresta**);
- se w non è in L, allora M si arresta pur non entrando in uno stato accettante.

Diciamo inoltre che un problema L è:

- **decidibile** se *si tratta* di un linguaggio ricorsivo;
- **indecidibile** se *non si tratta* di un linguaggio ricorsivo.

3.3 Teoremi sui linguaggi ricorsivi

- Se L è un linguaggio ricorsivo, allora lo è anche $\overline{L}$;
- Se L e $\overline{L}$ sono RE, allora L è ricorsivo.

3.4 Linguaggio universale

Definiamo L_{u}, il **linguaggio universale**, come l'insieme delle stringhe binarie che codificano una coppia (M, w), dove M è una TM con alfabeto di input binario e w è una stringa in $(\mathbf{0} + \mathbf{1})*$ tale che w sia in $L(M)$.
Esiste anche la TM U, detta **macchina di Turing universale**, tale che
$L_{\mathrm{u}} = L(U)$.

3.4.1 Teorema su $L_{\mathbf{u}}$

L_{u} è RE ma non ricorsivo.

3.5 Riduzioni

Se esiste una **riduzione** da P_1 a P_2, allora:

1. se P_1 è indecidibile, lo è anche P_2;
2. se P_1 non è RE, lo è anche P_2.

3.6 Linguaggi vuoto e non vuoto

3.6.1 Linguaggio L_e

L_e è il linguaggio formato da tutte le TM codificate in cui il linguaggio è vuoto. Vale quindi:

$$L_e = \{M | L(M) = \emptyset\}$$

3.6.2 Linguaggio L_{ne}

L_{ne} è il linguaggio di tutti i codici delle macchine di Turing che accettano almeno una stringa di input. Vale quindi:

$$L_{ne} = \{M | L(M) \neq \emptyset\}$$

3.6.3 Teoremi sui linguaggi L_e e L_{ne}

- L_{ne} è ricorsivamente enumerabile;
- L_{ne} non è ricorsivo;
- L_e non è ricorsivamente enumerabile.

3.7 Teorema di Rice

Ogni proprietà non banale dei linguaggi RE è indecidibile. Si ricorda che una *proprietà* dei linguaggi RE è semplicemente un insieme di linguaggi RE, e che una proprietà *banale* è tale se è *vuota*.

3.8 Problema di corrispondenza di Post

Un'istanza del **problema di corrispondenza di Post (PCP)**, consiste in due liste di stringhe sullo stesso alfabeto Σ; le due liste devono avere la stessa lunghezza.
L'istanza di PCP **ha soluzione** se esiste una sequenza di uno o più interi che, interpretati come indici per le stringhe di A e B, producono la stessa stringa.

3.8.1 Indecidibilità di PCP

PCP è un problema indecidibile.

4 Teoria del capitolo 10

Nel capitolo 10 si studia la teoria dell'**intrattabilità**, ossia le tecniche per dimostrare che un problema non è risolvibile in tempo polinomiale. I risultati sui problemi intrattabili si fondano tutti su un'ipotesi *non dimostrata*, ma fortemente creduta, la cosiddetta ipotesi $\mathbf{P} \neq \mathbf{NP}$.

4.1 La classe P

Un linguaggio L è nella classe P(**polinomiale deterministica**) se esiste un polinomo $T(n)$ tale che $L = L(M)$ per una TM *deterministica* M di complessità in tempo $T(n)$.

4.2 La classe NP

Un linguaggio L è nella classe NP(**polinomiale non deterministica**) se esiste una TM non deterministica M e una complessità polinomiale in tempo $T(n)$ tale che $L = L(M)$ e che, quando a M viene dato un input di lunghezza n, in M non ci sono sequenze di mosse più lunghe di $T(n)$.

4.2.1 Traveling Salesman Problem, TSP

Il **problema del commesso viaggiatore**(*TSP*, traveling salesman problem) è un esempio di problema che sta nella classe NP.

4.3 Problemi NP-completi

Un problema è **NP-completo** se è definito da un linguaggio L tale che:

- L è in NP;
- per ogni altro L' in NP esiste una riduzione polinomiale di L' a L.

Se un problema NP-completo P è in P, allora $\mathbf{P} = \mathbf{NP}$.

4.4 Problemi NP-hard

Se un linguaggio L è così "arduo"(**hard**) da non poterne dimostrare la condizione che L è in NP, allora L è chiamato problema **NP-hard**.

4.5 Teorema di Cook

Il problema *SAT(problema della soddisfacibilità)* è NP-completo.

4.6 Forma Normale Congiuntiva (CNF)

Diamo due definizioni fondamentali:

- **Letterale**: una variabile o una variabile negata. Ad esempio x e $\neg x$. Notare che una variabile e la sua negazione sono due variabili **diverse**;
- **Clausola**: disgiunzione logica (**OR**) di uno o più letterali. Ad esempio x, $x \lor y$, $x \lor \neg y \lor z$.

Ora possiamo definire **CNF**:
Un'espressione booleana si dice in **CNF(forma normale congiuntiva)**, se è la *congiunzione logica(AND)* di una o più clausole.

4.6.1 k-CNF

Un'espressione booleana si dice in **forma normale k-congiuntiva(k-CNF)**, se è la *congiunzione logica* di clausole, ognuna delle quali è una *disgiunzione logica* di k letterali distinti.

4.6.2 Esempi su CNF

- $(x \lor \neg y) \land (\neg x \lor z)$ è in CNF. Infatti è la congiunzione di clausole che contengono letterali distinti messi in disgiunzione tra loro;
- $(x \lor y \land \neg z) \land (x \lor y \lor z)$ non è in CNF. Infatti la prima clausola non è tale poiché y e $\neg z$ sono in congiunzione e non in disgiunzione;
- $x \land y \land z$ è in CNF. Si ricordi infatti che la clausola può essere costituita anche da un singolo letterale.

4.7 Problema CSAT

Il problema *CSAT*(CircuitSAT) è NP-completo.

4.8 Problema 3SAT

Il problema *3SAT*() è NP-completo.