Automi e Linguaggi Formali

Indecidibilità, riduzioni e teorema di Rice

Lamberto Ballan
lamberto.ballan@unipd.it

Padova, 26 Maggio 2017

Università degli Studi di Padova

Esercizi

- TM che accetta L delle stringhe binarie palindrome

- Esercizio (9.3.1): mostrare che l'insieme dei codici di TM che accettano tutti gli input palindromi è indecidibile

Il linguaggio universale

- Il linguaggio universale L_u è l'insieme delle stringhe binarie che codificano una coppia (M, w) dove $w \in L(M)$
  - detto diversamente, L_u è RE ed è l'insieme delle stringhe che rappresentano una TM e un input da essa accettato
- Esiste una TM U, detta "TM universale", tale che $L_u = L(U)$
  - U ha tre nastri: uno per la codifica di M e di w, uno per il nastro di M, e uno per la codifica dello stato di M; così U simula M su w, e accetta (M, w) se e solo se M accetta w
- Indecidibilità di L_u: dato che alcune TM M corrispondono a linguaggi non ricorsivi, ma RE, non si fermano quando la stringa w in input non è del linguaggio. Anche U si comporterà allo stesso modo su (M, w), e quindi L_u è RE ma non ricorsivo.

Riduzioni

- Possiamo “servirci” di linguaggi di cui conosciamo lo status in termini di decidibilità ed enumerabilità ricorsiva (come L_u e L_d) per studiare nuovi linguaggi

- Dato un problema noto P_1 indecidibile, vorremmo vedere se un nuovo problema P_2 è a sua volta indecidibile

- Un problema P_1 si *riduce* a P_2 se abbiamo un algoritmo per convertire le istanze P_1 in istanze di P_2 con stessa risposta

    - possiamo quindi avvalerci della riduzione per provare che P_2 è “difficile” almeno quanto P_1

- Teorema (9.7): se esiste una riduzione da P_1 a P_2: (*a*) se P_1 è indecidibile, lo è anche P_2; (*b*) se P_1 è non RE, lo è anche P_2

Macchine di Turing per il linguaggio vuoto

- Vediamo due esempi di riduzioni che coinvolgono le TM (utilizzando la codifica M_i precedentemente introdotta)
- Definiamo i due linguaggi sull'alfabeto binario {0, 1}:
  - $L_e = \{M \mid L(M) = \varnothing\}$
  - $L_{ne} = \{M \mid L(M) \neq \varnothing\}$
- Vogliamo verificare se sono ricorsivi, RE o neanche RE

Macchine di Turing per il linguaggio vuoto

- Teorema (9.8): L_{ne} è ricorsivamente enumerabile

  D: Dobbiamo dimostrare che esiste una TM M che lo accetta.

  Lo si dimostra costruendo una TM non-deterministica M che, data una TM M_i in input, simula in modo non deterministico M_i su tutte le possibili stringhe in input w.

  Detto diversamente, M verifica se M_i accetta w, e per questa parte M simula la TM universale U che accetta il linguaggio L_u. Se M_i accetta w, allora M accetta il proprio input, ovvero la TM M_i.

  Perciò, se M_i accetta anche solo una stringa w, essa sarà tra quelle "congetturate" da M che perciò accetta M_i. Se però $L(M_i)= \emptyset$, nessun w congetturato porta all'accettazione di M_i (e quindi di M).

  Di conseguenza possiamo affermare che $L(M)=L(L_{ne})$, cioè L_{ne} è RE.

Macchine di Turing per il linguaggio vuoto

- Teorema (9.9): L_{ne} non è ricorsivo

D: Lo dimostriamo riducendo L_u a L_{ne}. Essendo L_u non ricorsivo, non potrà esserlo neanche L_{ne}.

Per ridurre L_u a L_{ne} dobbiamo convertire ogni coppia (M, w) per cui $w \in L(M)$ in una nuova TM M' t.c. $w \in L(M)$ se e solo se $L(M') \neq \emptyset$

Sia M' una TM che (ignora il suo input e) simula M su w:

- se M accetta w, allora M' accetta il suo input e quindi L(M') non è vuoto

- se M non accetta w, allora M' non accetta nessun input e quindi L(M') è vuoto

Questo significa che abbiamo ridotto L_u a L_{ne} per via dell'algoritmo che costruisce M' da M e w. Possiamo perciò concludere che, essendo L_u non ricorsivo, anche L_{ne} sarà non ricorsivo.

Macchine di Turing per il linguaggio vuoto

- Teorema (9.10): L_e non è ricorsivamente enumerabile

  D: Dato che L_{ne} non è ricorsivo (per il teorema 9.9), e che L_e è il complemento di L_{ne}, allora L_e non può essere ricorsivamente enumerabile.

  Se lo fosse allora sarebbero entrambi ricorsivi (per il teorema 9.3 che abbiamo dimostrato nella precedente lezione).

Proprietà dei linguaggi RE

- Tutte le proprietà non banali di una macchina di Turing (cioè dei linguaggi RE) sono indecidibili
  - cioè è impossibile riconoscere per mezzo di una TM le stringhe che rappresentano codici di TM il cui *L* soddisfa la proprietà
  - un esempio di proprietà è “essere un linguaggio libero dal contesto” (quindi il problema se una particolare TM accetti un linguaggio libero dal contesto è indecidibile)
  - un altro esempio di proprietà è “essere un linguaggio vuoto”
- Formalmente, una proprietà dei linguaggi RE è semplicemente un insieme di linguaggi RE
  - quindi, ad esempio, la proprietà di essere “libero dal contesto” è l’insieme di tutti i linguaggi CFL

Proprietà dei linguaggi RE

- Una proprietà è *banale* se viene soddisfatta da tutti i linguaggi RE oppure da nessuno (cioè è vuota), altrimenti è *non banale*

- Se P è una proprietà dei linguaggi RE, il linguaggio L_p è l'insieme dei codici di TM M_i tali che $L(M_i)$ è un linguaggio in P

    - quando parliamo di decidibilità di una proprietà P, ci riferiamo alla decidibilità del linguaggio L_p

Il teorema di Rice

- Teorema di Rice (9.10): ogni proprietà non banale dei linguaggi RE è indecidibile

D: Sia P una proprietà non banale dei linguaggi RE.

Allora deve esistere un linguaggio RE L che abbia la proprietà P. Sia quindi M_L una TM che accetta L.

Riduciamo L_u a L_p, dimostrando che L_p è indecidibile, dato che come abbiamo già dimostrato L_u è indecidibile.

L'input della riduzione è una coppia (M, w), mentre l'output è una TM M' t.c. L(M') è vuoto se M non accetta w, e L(M')=L se M accetta w.

M' simula M su w. Se M non accetta w, M' non accetta nessuna stringa x in input. Se M accetta w, M' simula M_L su x e perciò accetterà esattamente il linguaggio L.

Poiché L ha la proprietà P, M' è in L_p.

Alcuni problemi indecidibili

- Tutti i problemi sulle TM che riguardano i linguaggi accettati sono indecidibili. Vediamo alcuni esempi:
  - il linguaggio accettato da una TM è vuoto?
  - il linguaggio accettato da una TM è finito?
  - il linguaggio accettato da una TM è regolare?
  - il linguaggio accettato da una TM è libero da contesto?
  - il linguaggio accettato da una TM contiene la stringa ‘ab’?
  - il linguaggio accettato da una TM contiene tutti i numeri pari?

Non tutti i problemi sono indecidibili…

- Problemi che riguardano gli stati di una TM, e non il linguaggio accettato, possono essere decidibili. Alcuni esempi:
  - dire se una TM ha un certo numero *k* di stati (es. *k*=5)
  - dire se esiste un input tale che una TM faccia almeno *k* passi prima di fermarsi
  - dire se una TM esegue una certa transizione
  - dire se una TM passa per lo stato *p* (o, viceversa, se non passa mai per lo stato *p*)
  - dire se una TM passa dallo stato *p* allo stato *q* in al più *k* passi

Un esempio “concreto” di problema indecidibile

- Vediamo un esempio “concreto” di problema indecidibile, svincolato dall’astrazione della TM (che riguarda sempre stringhe)
- Il PCP (“problema di corrispondenza di Post”) consiste in due liste di stringhe A e B, di uguale lunghezza k, sullo stesso alfabeto Σ
  - siano $A=w_1,w_2,\ldots,w_k$ e $B=x_1,x_2,\ldots,x_k$; per ogni intero i, la coppia (w_i, x_i) si dice coppia corrispondente
  - l’istanza di PCP ha soluzione se esiste una sequenza $i_1,i_2,\ldots,i_m$ di indici per A e B che danno la stessa stringa $w_{i_1}w_{i_2}\ldots w_{i_m} = x_{i_1}x_{i_2}\ldots x_{i_m}$
  - in questo caso diciamo che la sequenza $i_1,i_2,\ldots,i_m$ è una soluzione dell’istanza di PCP

Un esempio “concreto” di problema indecidibile

- Il problema di corrispondenza di Post è perciò definito così: data una istanza di PCP stabilire se essa ha soluzione

- Vediamo un primo esempio di istanza di PCP:

	Lista A	Lista B
i	w_i	x_i
1	1	111
2	10111	10
3	10	0

Un’istanza di PCP
(con soluzione)

La lista 2,1,1,3 è soluzione.

Analogamente, 2,1,1,3,2,1,1,3 è un’altra soluzione.

Un esempio “concreto” di problema indecidibile

- Vediamo adesso un secondo esempio di istanza di PCP:

i	Lista A w_i	Lista B x_i
1	10	101
2	011	11
3	101	011

Un’istanza di PCP
(senza soluzione)

Abbiamo necessariamente i_1=1 e quindi le stringhe inizieranno così: A: 10… e *B*: 101…

Per i_2 solo 3 è possibile. Le stringhe saranno perciò: A: 10101… e *B*: 101011…

Nulla vieta che le stringhe si estendano portando a soluzione. Però si può provare che non è possibile farlo perché ci siamo riportati nella stessa situazione iniziale successiva a i_1=1. Quindi dovremo scegliere sempre i=3 per mantenere corrispondenza.