Automi e Linguaggi Formali

Indecidibilità, linguaggi ricorsivi e linguaggio universale

Lamberto Ballan
lamberto.ballan@unipd.it

Padova, 23 Maggio 2017

Università degli Studi di Padova

Correzione esercizi

- TM che accetta L delle stringhe con stesso numero di 0 e 1
- TM che accetta L delle stringhe binarie palindrome

Indecidibilità

- Un linguaggio *L* è **ricorsivamente enumerabile** (RE) se *L*=*L*(*M*) per una macchina di Turing *M*
  - *i.e.* esiste una TM che si arresta se la stringa è accettata (ma potrebbe non fermarsi se non la accetta)

- Abbiamo poi introdotto in modo formale cosa significa che un problema é *indecidibile*
- Abbiamo visto un esempio di linguaggio non RE (il linguaggio di diagonalizzazione L_d) e lo abbiamo dimostrato

Indecidibilità

- Cerchiamo adesso di chiarire la struttura dei linguaggi RE, ossia quelli accettati da una TM, individuando due classi:
  - linguaggi *L* in cui abbiamo una TM che non solo riconosce il linguaggio ma che segnala anche se *w* non è in *L*
  - linguaggi *L* che non sono accettati da alcuna TM che garantisca di arrestarsi

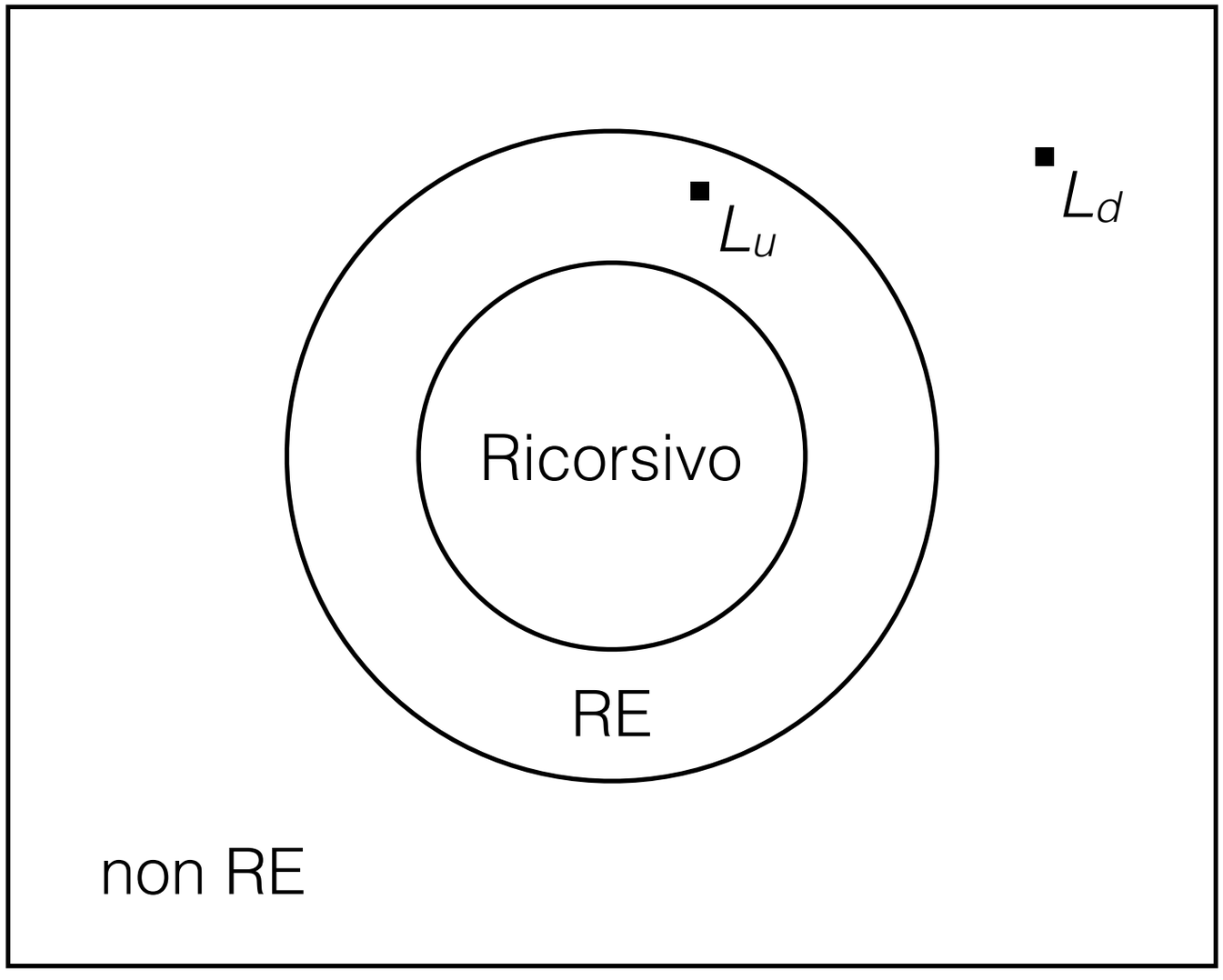
Linguaggi ricorsivi

- Un linguaggio *L* è **ricorsivo** se *L*=*L*(*M*) per una macchina di Turing *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

- Una TM d questo tipo corrisponde alla nozione informale di *algoritmo*, cioè una sequenza ben definita di passi che termina sempre e produce una risposta

- Se consideriamo *L* come un problema, allora il problema *L* è detto *decidibile* se si tratta di un linguaggio ricorsivo, mentre è detto *indecidibile* se si tratta di un linguaggio non ricorsivo

Relazione tra linguaggi ricorsivi, RE, non RE



Complementi di linguaggi ricorsivi e RE

- Teorema (9.3): se L é un linguaggio ricorsivo, lo è anche il suo complementare L^c

D: Sia L=L(M) per una TM che si arresta sempre.

Costruiamo una macchina M^c tale che $L^c=L(M^c)$. Per formare M^c modifichiamo M nel modo seguente:

- gli stati accettanti di M diventano stati non accettanti di M^c senza transizioni; in questi stati M^c si arresta senza accettare.

- M^c ha un nuovo stato accettante q_f, da cui non esistono transizioni uscenti.

- per ogni combinazione di stato non accettante e simbolo di nastro per cui M non ha regole di transizione (quindi si arresta senza accettare) aggiungiamo una transizione verso q_f.

M si arresta per definizione, e quindi anche M^c. Inoltre M^c accetta le stringhe w che M non accetta. Quindi M^c accetta il linguaggio L^c.

Complementi di linguaggi ricorsivi e RE

- Teorema (9.4): se L e L^c sono linguaggi RE, allora L è ricorsivo

D: Siano $L=L(M_1)$ e $L^c=L(M_2)$, dove M_1 e M_2 sono simulate in parallelo da una TM a due nastri M (nota: gli stati di M_1 e M_2 sono componenti dello stato di M).

- se l'input w è in L, allora M_1 accetta in tempo finito e quindi M accetta e si arresta.

- se l'input w non è in L, allora è in L^c, perciò M_2 prima o poi dovrà accettare w; a quel punto M si arresta senza accettare.

Dunque M si arresta su tutti gli input e L(M)=L, e quindi possiamo concludere che L è ricorsivo.

Proprietà dei linguaggi ricorsivi e RE

- Quindi dove possono stare L e L^c?
  - sia L sia L^c sono ricorsivi, cioè si trovano nel cerchio interno
  - né L né L^c sono RE
  - L è RE ma non ricorsivo, e L^c non è RE
  - L^c è RE ma non ricorsivo, e L non è RE (caso analogo al precedente ma L e L^c sono scambiati)
- Non è possibile che un linguaggio (L o L^c) sia ricorsivo e l'altro sia RE o neanche RE (per il primo teorema)
- Non è possibile che siano entrambi RE ma non ricorsivi (per il secondo teorema)

Il linguaggio universale

- Il linguaggio universale L_u è l'insieme delle stringhe binarie che codificano una coppia (M, w) dove $w \in L(M)$

- Esiste una TM U, detta "TM universale", tale che $L_u = L(U)$
- Descriviamo U come TM multinastro (3 nastri):
  - uno per il codice di M (i.e. le transizioni) e l'input w
  - uno per il nastro simulato di M (usando la stessa codifica di M)
  - uno per la codifica dello stato di M
- Così U simula M su w, e accetta (M, w) sse M accetta w

Indecidibilità del linguaggio universale

- Teorema: L_u é RE ma non ricorsivo

D: Abbiamo già visto che L_u è RE.

Supponiamo che L_u sia ricorsivo. Quindi per il T9.3 anche L_u^c sarà ricorsivo. Se abbiamo una M che accetta L_u^c allora si può costruire una TM che accetta L_d. Ma L_d non è RE e quindi siamo in contraddizione.

Supponiamo ora che L(M)= L_u^c. Possiamo modificare M in una TM M' che accetta L_d.

- data una stringa w, M' la trasforma in w111w (si usa un'altro nastro per copiare w)

- M' simula M sul nuovo input; se nella "nostra" enumerazione w è w_i allora M' determina se M_i accetta w_i. Poiché M accetta L_u^c, accetterà sse M_i non accetta w_i, quindi w_i é in L_d.

Perciò M' accetta w sse è in L_d. Ma M' non può esistere quindi L_u non è un linguaggio ricorsivo.

Il problema dell'arresto

- Data una TM *M*, definiamo *H*(*M*) come l'insieme delle stringhe *w* tale che *M* si arresta con input *w* (anche se *M* non accetta)
- Il problema dell'arresto è quindi definito come il linguaggio che contiene le coppie (*M*, *w*) tali che w è in *H*(*M*)
  - questo è un altro problema simile a L_u, e quindi si può definire una TM che simula il comportamento di *M* su *w*, mostrando che è un linguaggio RE ma non ricorsivo
- Quindi non esiste un algoritmo che possa dire se un dato programma termina o no; esiste però un algoritmo che, se il programma in input termina, si ferma, altrimenti non si arresta

Riduzioni

- 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
  - ridurre P_1 a P_2 significa convertire ogni stringa di P_1 in una stringa di P_2, e ogni stringa non in P_1 in una stringa non in P_2
- Supponiamo che esista un algoritmo che risolve P_2. Data una stringa w per P_1, la convertiamo in un'altra stringa x per P_2. Usiamo quindi l'algoritmo di soluzione per P_2 per decidere se x è o meno in P_2. Qualunque sia la risposta, è valida anche per w in P_1. Perciò, a partire dall'algoritmo che risolve P_2, abbiamo costruito un algoritmo che risolve P_1.

Riduzioni

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

D: fare per esercizio