Azzolini Riccardo 2020-12-22

Indecidibilità del linguaggio universale

1 Complementi di linguaggi decidibili

Teorema: Se L è un linguaggio decidibile, allora anche il suo complemento $\overline{L}$ è decidibile.

Dimostrazione: Siccome L è decidibile, esiste per definizione una MdT M che termina per ogni input ed è tale che $L(M) = L$. Si può dunque scrivere un programma P che simula l'esecuzione di M su un input w e inverte il risultato restituito dalla simulazione:

Input: $w \in \{0,1\}^*$
Output: `ACCETTA` o `RIFIUTA`

```
result = simula(#MdT(M), w)
if (result == ACCETTA)
  return RIFIUTA
else
  return ACCETTA
```

Il programma P termina per ogni input, e accetta w se e solo se $w \notin L(M) = L$, ovvero riconosce il linguaggio $\overline{L}$. Applicando la tesi di Church-Turing, si deduce che esiste anche una macchina di Turing che riconosce $\overline{L}$ e termina per ogni input, quindi $\overline{L}$ è decidibile.

2 Complementi di linguaggi ricorsivamente enumerabili

Teorema: Se un linguaggio L e il suo complemento $\overline{L}$ sono ricorsivamente enumerabili, allora L e $\overline{L}$ sono anche decidibili.

Dimostrazione: Secondo l'ipotesi che L e $\overline{L}$ siano r.e., esistono per definizione due MdT M e $\overline{M}$ tali che $L(M) = L$ e $L(\overline{M}) = \overline{L}$. Si consideri ora un programma P che, data in ingresso una stringa $w \in \{0,1\}^*$, esegue in parallelo `simula`$(\#_{\text{MdT}}(M), w)$ e `simula`$(\#_{\text{MdT}}(\overline{M}), w)$. Siccome per ogni stringa w si ha che $w \in L = L(M) \iff w \notin \overline{L} = L(\overline{M})$, sicuramente una delle MdT accetta la stringa, quindi termina sempre almeno una delle due esecuzioni parallele di `simula`. Il comportamento di P viene poi definito in funzione del risultato restituito dalla prima simulazione che termina:[1]

- se termina prima $\texttt{simula}(\#_{\text{MdT}}(M), w)$, allora

$$P \text{ restituisce } \begin{cases} \texttt{ACCETTA} & \text{se } \texttt{simula}(\#_{\text{MdT}}(M), w) \text{ restituisce } \texttt{ACCETTA} \\ \texttt{RIFIUTA} & \text{se } \texttt{simula}(\#_{\text{MdT}}(M), w) \text{ restituisce } \texttt{RIFIUTA} \end{cases}$$

- se invece termina prima $\texttt{simula}(\#_{\text{MdT}}(\overline{M}), w)$, allora

$$P \text{ restituisce } \begin{cases} \texttt{ACCETTA} & \text{se } \texttt{simula}(\#_{\text{MdT}}(\overline{M}), w) \text{ restituisce } \texttt{RIFIUTA} \\ \texttt{RIFIUTA} & \text{se } \texttt{simula}(\#_{\text{MdT}}(\overline{M}), w) \text{ restituisce } \texttt{ACCETTA} \end{cases}$$

Così, P termina per ogni input, e restituisce `ACCETTA` se e solo se $w \in L$.

Dall'esistenza di P si deduce, tramite la tesi di Church-Turing, che esiste una MdT M' la quale termina per ogni input ed è tale che $L(M') = L$. Ciò dimostra che L è decidibile, e dunque, per il teorema precedente, anche $\overline{L}$ è decidibile.

3 Il linguaggio universale è r.e. ma non decidibile

Si è già dimostrato che il linguaggio universale,

$$L_u = \{\#_{\text{MdT}}(M)111w \mid M \text{ accetta } w\}$$

è ricorsivamente enumerabile. Ora, si vuole dimostrare che esso è indecidibile:

Teorema: Il linguaggio L_u è ricorsivamente enumerabile ma non è decidibile.

3.1 Dimostrazione

Si consideri il linguaggio complemento di L_u:

$$\overline{L_u} = \left\{ \alpha \in \{0,1\}^* \;\middle|\; \begin{array}{l} \alpha \text{ non codifica correttamente una coppia } (M, w), \\ \text{oppure } \alpha = \#_{\text{MdT}}(M)111w \text{ e } M \text{ non accetta } w \end{array} \right\}$$

Se si suppone per assurdo che L_u sia decidibile, segue da uno dei teoremi precedenti che anche $\overline{L_u}$ è decidibile: esiste una MdT $\overline{M}$ che si arresta per ogni input ed è tale che

[1] Si osservi che la prima simulazione che termina non è necessariamente quella che accetta l'input: la computazione che accetta termina sempre, ma la computazione che non accetta potrebbe in alcuni casi terminare per prima (mentre in altri casi potrebbe terminare per ultima, o non terminare mai). Allora, nel definire il comportamento di P, è importante considerare appunto i casi in cui la prima simulazione terminata rifiuta l'input.

$L(\overline{M}) = \overline{L_u}$. Dunque, la simulazione $\texttt{simula}(\#_{\text{MdT}}(\overline{M}), w)$ termina per ogni $w \in \{0,1\}^*$, e restituisce:

$$\texttt{simula}(\#_{\text{MdT}}(\overline{M}), w) = \begin{cases} \texttt{ACCETTA} & \text{se } w \in L(\overline{M}) = \overline{L_u} \\ \texttt{RIFIUTA} & \text{se } w \notin L(\overline{M}) = \overline{L_u} \end{cases}$$

Si consideri ora il seguente programma P:

Input: $w \in \{0,1\}^*$
Output: `ACCETTA` o `RIFIUTA`

i `=` $\texttt{bin2dec}(1w)$
M_i `=` $e_{\text{MdT}}(i)$
return $\texttt{simula}(\#_{\text{MdT}}(\overline{M}), \#_{\text{MdT}}(M_i)111w_i)$

Per prima cosa, questo programma calcola l'indice i della stringa di input w nell'enumerazione e_{01}, cioè il valore i tale che $w_i = w$. Tale indice viene poi usato per ottenere M_i, la MdT avente indice i nell'enumerazione e_{MdT}. Infine, P simula l'esecuzione di $\overline{M}$ su una stringa di input che codifica la coppia (M_i, w_i).

Il programma P termina per ogni input (perché la simulazione della computazione di $\overline{M}$ si arresta sempre per ipotesi, così come terminano sempre tutte le altre funzioni utilizzate), e in particolare, dato un input $w = w_i$,[2]

$$P \text{ restituisce } \begin{cases} \texttt{ACCETTA} & \text{se } \#_{\text{MdT}}(M_i)111w_i \in L(\overline{M}) = \overline{L_u} \\ \texttt{RIFIUTA} & \text{se } \#_{\text{MdT}}(M_i)111w_i \notin L(\overline{M}) = \overline{L_u} \end{cases}$$

(per la definizione di $\texttt{simula}(\#_{\text{MdT}}(\overline{M}), \#_{\text{MdT}}(M_i)111w_i)$ e per l'ipotesi $L(\overline{M}) = \overline{L_u}$).

Successivamente, ricordando la definizione di $\overline{L_u}$,

$$\overline{L_u} = \left\{ \alpha \in \{0,1\}^* \;\middle|\; \begin{array}{l} \alpha \text{ non codifica correttamente una coppia } (M, w), \\ \text{oppure } \alpha = \#_{\text{MdT}}(M)111w \text{ e } M \text{ non accetta } w \end{array} \right\}$$

si osserva che, nel caso particolare del programma P, la stringa α è sempre della forma $\#_{\text{MdT}}(M_i)111w_i$, ovvero codifica correttamente una coppia (M_i, w_i), quindi $\#_{\text{MdT}}(M_i)$ $111w_i \notin \overline{L_u}$ se e solo se M_i accetta w_i: su un input w_i,

$$P \text{ restituisce } \begin{cases} \texttt{ACCETTA} & \text{se } M_i \text{ non accetta } w_i \\ \texttt{RIFIUTA} & \text{se } M_i \text{ accetta } w_i \end{cases}$$

Per la tesi di Church-Turing, dall'esistenza di P si deduce l'esistenza di una MdT M_P che termina per ogni input e riconosce il linguaggio

$$L(M_P) = \{w_i \in \{0,1\}^* \mid M_i \text{ non accetta } w_i\}$$

[2] Siccome la prima cosa che P fa è determinare l'indice di w in e_{01}, è conveniente rappresentare la stringa di input w già come un elemento w_i dell'enumerazione.

(lo stesso linguaggio riconosciuto da P). Così, a partire dall'ipotesi che L_u sia decidibile, si è dedotto che $L(M_P)$ è decidibile, ma ciò è *assurdo*: $L(M_P) = L_d$ è il linguaggio di diagonalizzazione, e si è dimostrato che L_d non può essere decidibile, in quanto non ricorsivamente enumerabile.

In conclusione, poiché si sa già che L_u è r.e., si deduce che L_u è r.e. ma non è decidibile: il teorema è dimostrato.

4 Halting problem

Un altro problema di decisione importante, simile al problema LM corrispondente al linguaggio universale, è l'**halting problem** (problema dell'arresto), HP:

> *Parametri*: Una macchina di Turing $M = \langle Q, \{0,1\}, \Gamma, \delta, q_1, B, F\rangle$ e un suo input $w \in \{0,1\}^*$.
>
> *Domanda*: M si arresta su input w?

Il linguaggio corrispondente a HP è

$$L(\mathrm{HP}) = \left\{ x \;\middle|\; \begin{array}{l} x \text{ è la codifica di una coppia } (M, w) \\ \text{per cui } M \text{ si arresta su input } w \end{array} \right\}$$

che è ricorsivamente enumerabile ma non è decidibile.