Lezione di Informatica Teorica: Decidibilità e Indecidibilità

Appunti da Trascrizione Automatica

30 giugno 2025

Indice

1 Introduzione e Ripasso

Ripassiamo i concetti fondamentali della scorsa lezione, chiarendo alcuni punti sulla Macchina Universale (MU) e introducendo nuovi linguaggi e problemi legati alla decidibilità.

1.1 Macchina Universale (MU)

La Macchina Universale è una Macchina di Turing programmabile, capace di simulare il comportamento di qualsiasi altra Macchina di Turing. Sebbene la sua funzione di transizione sia complessa, la sua progettazione è fattibile e si può realizzare con un numero limitato di stati.

1.2 Linguaggio Diagonale (LD)

La scorsa lezione abbiamo introdotto il linguaggio diagonale LD:

Definizione 1.1 (Linguaggio Diagonale (LD))**.** *Il linguaggio diagonale LD è l'insieme delle codifiche di Macchine di Turing M_i tali che M_i non accetta la propria codifica $\langle M_i \rangle$.*

$$LD = \{\langle M_i \rangle \mid M_i \text{ non accetta } \langle M_i \rangle\}$$

Abbiamo dimostrato che LD non appartiene alla classe dei linguaggi ricorsivamente enumerabili (R_e).

Teorema 1.1. $LD \notin R_e$.

Dimostrazione. La dimostrazione si basa sulla tecnica di diagonalizzazione di Cantor, già vista in precedenza. Assumendo per assurdo l'esistenza di una Macchina di Turing M_D che accetta LD, si può costruire una contraddizione nel comportamento di M_D sulla propria codifica. □

Una conseguenza diretta di questo teorema è che LD non può appartenere nemmeno alla classe dei linguaggi ricorsivi (R).

Proposizione 1.1. *Se $L \in R$, allora $L \in R_e$. Poiché $LD \notin R_e$, allora $LD \notin R$.*

2 Linguaggio Universale (LU)

Introduciamo il Linguaggio Universale.

Definizione 2.1 (Linguaggio Universale (LU))**.** *Il linguaggio universale LU è l'insieme delle coppie $\langle M, w \rangle$, dove M è una Macchina di Turing e w è una stringa, tali che M accetta w. L'operatore "accetta" (indicato anche con $\models$) implica che M si arresta in uno stato accettante su w.*

$$LU = \{\langle M, w \rangle \mid M \text{ accetta } w\}$$

Teorema 2.1. $LU \in R_e$.

Dimostrazione. Per dimostrare che $LU \in R_e$, dobbiamo mostrare l'esistenza di una Macchina di Turing che lo accetti. Tale macchina è la Macchina Universale (MU).

1. La MU prende in input la codifica $\langle M, w \rangle$.
2. La MU simula passo-passo il comportamento di M su w.

3. Se M si arresta in uno stato accettante su w, la MU si arresta e accetta (risponde "sì").
4. Se M si arresta in uno stato non accettante su w, la MU si arresta e rifiuta (risponde "no").
5. Se M non si arresta su w (entra in un loop infinito), la MU simulerà all'infinito e non si arresterà (non darà risposta "sì").

Poiché la MU accetta tutte le istanze di LU e non accetta quelle che non vi appartengono, LU è ricorsivamente enumerabile. □

Ora, la domanda cruciale: $LU \in R$? Intuitivamente, se M va in loop su w, la MU non si arresterà per dare una risposta "no". Questo suggerisce che LU non sia decidibile.

Teorema 2.2. $LU \notin R$.

Dimostrazione. La dimostrazione procede per assurdo, utilizzando una riduzione da LD (di cui sappiamo la non appartenenza a R_e, e quindi a R).

Assunzione per assurdo: Supponiamo che $LU \in R$.

1. Se $LU \in R$, allora per le proprietà delle classi di linguaggi, anche il suo complemento $\overline{LU} \in R$.
2. Se $\overline{LU} \in R$, allora esiste una Macchina di Turing $M_{\overline{LU}}$ che **decide** $\overline{LU}$. Ciò significa che $M_{\overline{LU}}$ si arresta sempre (su ogni input) e dà una risposta corretta (sì/no).
3. Costruiamo una nuova Macchina di Turing M' che prende in input una codifica di macchina $\langle M_i \rangle$. Il comportamento di M' è il seguente:
   (a) Riceve $\langle M_i \rangle$ come input.
   (b) Crea una copia di $\langle M_i \rangle$ e la usa come stringa w. Forma la coppia $\langle M_i, \langle M_i \rangle \rangle$. (Quindi l'input per la fase successiva è $\langle M_i, w \rangle$ dove $w = \langle M_i \rangle$).
   (c) Dà in input la coppia $\langle M_i, \langle M_i \rangle \rangle$ alla macchina $M_{\overline{LU}}$ (la cui esistenza è garantita dalla nostra assunzione).
   (d) M' adotta la risposta di $M_{\overline{LU}}$:
   - Se $M_{\overline{LU}}$ risponde "sì", allora M' risponde "sì".
   - Se $M_{\overline{LU}}$ risponde "no", allora M' risponde "no".

Ora analizziamo il linguaggio deciso da M', $L(M')$:

- **Se M' risponde "sì":** Ciò significa che $M_{\overline{LU}}$ ha risposto "sì" sull'input $\langle M_i, \langle M_i \rangle \rangle$. Poiché $M_{\overline{LU}}$ decide $\overline{LU}$, la risposta "sì" implica che $\langle M_i, \langle M_i \rangle \rangle \in \overline{LU}$. Per definizione di $\overline{LU}$, ciò significa che M_i **non accetta** $\langle M_i \rangle$.
- **Se M' risponde "no":** Ciò significa che $M_{\overline{LU}}$ ha risposto "no" sull'input $\langle M_i, \langle M_i \rangle \rangle$. Poiché $M_{\overline{LU}}$ decide $\overline{LU}$, la risposta "no" implica che $\langle M_i, \langle M_i \rangle \rangle \notin \overline{LU}$, ovvero $\langle M_i, \langle M_i \rangle \rangle \in LU$. Per definizione di LU, ciò significa che M_i **accetta** $\langle M_i \rangle$.

Il comportamento di M' è esattamente quello di una macchina che decide LD. Infatti, M' risponde "sì" se M_i non accetta $\langle M_i \rangle$, e "no" se M_i accetta $\langle M_i \rangle$. Dunque, $L(M') = LD$.

Poiché M' è costruita usando $M_{\overline{LU}}$ (che è un decider e si arresta sempre), M' è anch'essa una Macchina di Turing che si arresta sempre, ovvero un decider. Questo implicherebbe che $LD \in R$.

Contraddizione: Sappiamo che $LD \notin R_e$, e quindi $LD \notin R$. L'assunzione iniziale ($LU \in R$) deve essere falsa.

Conclusione: $LU \notin R$. □

2.1 Il Complemento del Linguaggio Diagonale ($\overline{LD}$)

Consideriamo ora il complemento di LD:

Definizione 2.2 (Complemento del Linguaggio Diagonale ($\overline{LD}$))**.** *Il complemento del linguaggio diagonale $\overline{LD}$ è l'insieme delle codifiche di Macchine di Turing M_i tali che M_i accetta la propria codifica $\langle M_i \rangle$.*

$$\overline{LD} = \{\langle M_i \rangle \mid M_i \text{ accetta } \langle M_i \rangle\}$$

Teorema 2.3. $\overline{LD} \in R_e$.

Dimostrazione. Per dimostrare che $\overline{LD} \in R_e$, dobbiamo costruire una Macchina di Turing $M_{\overline{LD}}$ che accetti $\overline{LD}$.

Costruzione di $M_{\overline{LD}}$:

1. $M_{\overline{LD}}$ prende in input la codifica di una Macchina di Turing $\langle M_i \rangle$.
2. $M_{\overline{LD}}$ crea una copia di $\langle M_i \rangle$ per usarla come stringa w. Forma quindi la coppia $\langle M_i, \langle M_i \rangle \rangle$.
3. $M_{\overline{LD}}$ simula M_i su $\langle M_i \rangle$ usando una Macchina Universale (MU).
4. Se la simulazione di M_i su $\langle M_i \rangle$ si arresta e accetta, allora $M_{\overline{LD}}$ accetta (risponde "sì").
5. Se la simulazione di M_i su $\langle M_i \rangle$ si arresta e rifiuta, o entra in loop, allora $M_{\overline{LD}}$ non accetta (risponde "no" o va in loop).

Analisi del comportamento di $M_{\overline{LD}}$:

- **Se $\langle M_i \rangle \in \overline{LD}$:** Per definizione, M_i accetta $\langle M_i \rangle$. La simulazione della MU si arresterà e accetterà. Di conseguenza, $M_{\overline{LD}}$ accetterà.
- **Se $\langle M_i \rangle \notin \overline{LD}$:** Per definizione, M_i non accetta $\langle M_i \rangle$. Ciò significa che M_i o rifiuta o va in loop su $\langle M_i \rangle$.
  - Se M_i rifiuta $\langle M_i \rangle$, la simulazione della MU si arresterà e rifiuterà. $M_{\overline{LD}}$ non accetterà.
  - Se M_i va in loop su $\langle M_i \rangle$, la simulazione della MU andrà in loop. $M_{\overline{LD}}$ non accetterà.

Poiché $M_{\overline{LD}}$ accetta esattamente le stringhe che appartengono a $\overline{LD}$, si conclude che $\overline{LD} \in R_e$. □

Proposizione 2.1. $\overline{LD} \notin R$.

Dimostrazione. Se $\overline{LD}$ fosse in R, allora per la proprietà che R è chiusa rispetto al complemento, anche LD sarebbe in R. Ma sappiamo che $LD \notin R_e$, e quindi $LD \notin R$. Questo è una contraddizione. □

3 Problema dell'Arresto (HALT)

Introduciamo il famoso Problema dell'Arresto.

Definizione 3.1 (Problema dell'Arresto (HALT))**.** *Il linguaggio $HALT$ è l'insieme delle coppie $\langle M, w \rangle$, dove M è una Macchina di Turing e w è una stringa, tali che M si arresta su w (indipendentemente dal fatto che accetti o rifiuti).*

$$HALT = \{\langle M, w \rangle \mid M \text{ si arresta su } w\}$$

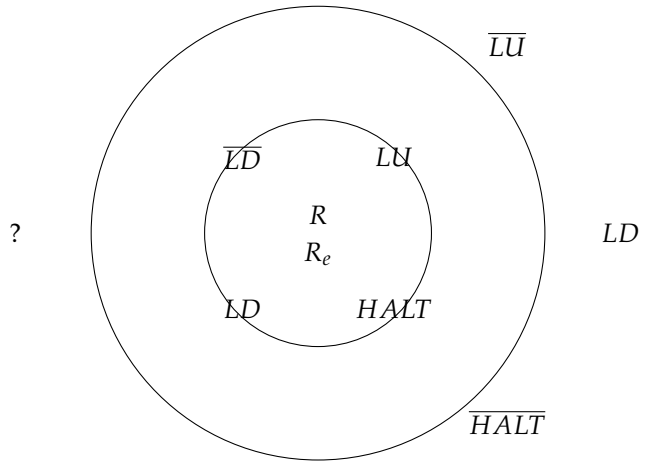

Figura 1: Relazioni tra classi di linguaggi R e R_e e posizione di alcuni linguaggi

La differenza con LU è sottile ma cruciale: LU richiede l'accettazione, $HALT$ richiede solo l'arresto.

Teorema 3.1. $HALT \in R_e$.

Dimostrazione. Per dimostrare che $HALT \in R_e$, dobbiamo costruire una Macchina di Turing M_{HALT} che accetti $HALT$.

Costruzione di M_{HALT}:

1. M_{HALT} prende in input la coppia $\langle M, w \rangle$.
2. M_{HALT} simula M su w usando una Macchina Universale (MU).
3. Se la simulazione di M su w si arresta (sia in uno stato accettante che non accettante), allora M_{HALT} accetta (risponde "sì").
4. Se la simulazione di M su w entra in loop infinito, allora M_{HALT} entra in loop (non risponde "sì").

Analisi del comportamento di M_{HALT}:

- **Se $\langle M, w \rangle \in HALT$:** Per definizione, M si arresta su w. La simulazione della MU si arresterà. Di conseguenza, M_{HALT} accetterà.
- **Se $\langle M, w \rangle \notin HALT$:** Per definizione, M non si arresta su w (va in loop). La simulazione della MU andrà in loop. Di conseguenza, M_{HALT} non accetterà.

Poiché M_{HALT} accetta esattamente le istanze di $HALT$, si conclude che $HALT \in R_e$. □

Teorema 3.2. $HALT \notin R$.

Dimostrazione. La dimostrazione procede per assurdo, utilizzando una riduzione da LU (di cui sappiamo la non appartenenza a R).

Assunzione per assurdo: Supponiamo che $HALT \in R$.

1. Se $HALT \in R$, allora esiste una Macchina di Turing M^*_{HALT} che **decide** $HALT$. Ciò significa che M^*_{HALT} si arresta sempre e dà una risposta corretta (sì/no).
2. Costruiamo una nuova Macchina di Turing M' che prende in input una coppia $\langle M, w \rangle$. Il comportamento di M' è il seguente:
   (a) Riceve $\langle M, w \rangle$ come input.
   (b) Dà in input $\langle M, w \rangle$ alla macchina M^*_{HALT} (la cui esistenza è garantita dalla nostra assunzione).
   (c) M' verifica la risposta di M^*_{HALT}:
   - **Se** M^*_{HALT} **risponde "no"**: (Significa che M non si arresta su w). Allora M' risponde "no". (In questo caso, M non può accettare w, quindi $\langle M, w \rangle \notin LU$).
   - **Se** M^*_{HALT} **risponde "sì"**: (Significa che M si arresta su w). Allora M' simula M su w usando una Macchina Universale (MU).
     – Se la simulazione di M su w si arresta e accetta, allora M' risponde "sì".
     – Se la simulazione di M su w si arresta e rifiuta, allora M' risponde "no".

Ora analizziamo il linguaggio deciso da M', $L(M')$:

- **Se** M' **risponde "sì":** Ciò accade solo se M^*_{HALT} ha risposto "sì" (cioè M si arresta su w) **e** la simulazione di M su w ha accettato. Questo significa che M accetta w. Quindi, $\langle M, w \rangle \in LU$.
- **Se** M' **risponde "no":** Ciò può accadere in due scenari:
  – Scenario 1: M^*_{HALT} ha risposto "no". Questo significa che M non si arresta su w. Se M non si arresta, non può accettare w. Quindi, $\langle M, w \rangle \notin LU$.
  – Scenario 2: M^*_{HALT} ha risposto "sì", ma la simulazione di M su w ha rifiutato. Questo significa che M si è arrestata su w ma non ha accettato w. Quindi, $\langle M, w \rangle \notin LU$.

  In entrambi gli scenari, M' risponde "no" se $\langle M, w \rangle \notin LU$.

Il comportamento di M' è esattamente quello di una macchina che decide LU. Dunque, $L(M') = LU$.

Poiché M' è costruita usando M^*_{HALT} (che è un decider e si arresta sempre), e la parte di simulazione dopo la risposta "sì" di M^*_{HALT} è garantita arrestarsi, M' è anch'essa una Macchina di Turing che si arresta sempre, ovvero un decider. Questo implicherebbe che $LU \in R$.

Contraddizione: Sappiamo che $LU \notin R$. L'assunzione iniziale ($HALT \in R$) deve essere falsa.
Conclusione: $HALT \notin R$. □

Proposizione 3.1. *Il complemento di $HALT$, $\overline{HALT}$, non appartiene a R.*

Dimostrazione. Se $\overline{HALT}$ fosse in R, allora per la proprietà che R è chiusa rispetto al complemento, anche $HALT$ sarebbe in R. Ma abbiamo appena dimostrato che $HALT \notin R$. Questo è una contraddizione. □

4 Problema dell'Arresto su Stringa Vuota (HALT_ϵ)

Questa è una variante specifica del problema dell'arresto.

Definizione 4.1 (Problema dell'Arresto su Stringa Vuota (HALT_ϵ))**.** *Il linguaggio $HALT_\epsilon$ è l'insieme delle codifiche di Macchine di Turing M tali che M si arresta quando le viene data in input la stringa vuota ϵ.*

$$HALT_\epsilon = \{\langle M \rangle \mid M \text{ si arresta su } \epsilon\}$$

Teorema 4.1. $HALT_\epsilon \in R_e$.

Dimostrazione. Per dimostrare che $HALT_\epsilon \in R_e$, dobbiamo costruire una Macchina di Turing M_{HALT_ϵ} che accetti $HALT_\epsilon$.

Costruzione di M_{HALT_ϵ}:

1. M_{HALT_ϵ} prende in input la codifica $\langle M \rangle$.
2. M_{HALT_ϵ} simula M sulla stringa vuota ϵ usando una Macchina Universale (MU).
3. Se la simulazione di M su ϵ si arresta (sia accettando che rifiutando), allora M_{HALT_ϵ} accetta (risponde "sì").
4. Se la simulazione di M su ϵ entra in loop infinito, allora M_{HALT_ϵ} entra in loop (non risponde "sì").

Analisi del comportamento di M_{HALT_ϵ}:

- **Se $\langle M \rangle \in HALT_\epsilon$:** Per definizione, M si arresta su ϵ. La simulazione della MU si arresterà. Di conseguenza, M_{HALT_ϵ} accetterà.
- **Se $\langle M \rangle \notin HALT_\epsilon$:** Per definizione, M non si arresta su ϵ (va in loop). La simulazione della MU andrà in loop. Di conseguenza, M_{HALT_ϵ} non accetterà.

Poiché M_{HALT_ϵ} accetta esattamente le istanze di $HALT_\epsilon$, si conclude che $HALT_\epsilon \in R_e$. □

Teorema 4.2. $HALT_\epsilon \notin R$.

Dimostrazione. La dimostrazione procede per assurdo, utilizzando una riduzione da $HALT$ (di cui sappiamo la non appartenenza a R).

Assunzione per assurdo: Supponiamo che $HALT_\epsilon \in R$.

1. Se $HALT_\epsilon \in R$, allora esiste una Macchina di Turing $M^*_{HALT_\epsilon}$ che **decide** $HALT_\epsilon$. Ciò significa che $M^*_{HALT_\epsilon}$ si arresta sempre e dà una risposta corretta (sì/no).
2. Costruiamo una nuova Macchina di Turing M' che prende in input una coppia $\langle M, w \rangle$. Il comportamento di M' è il seguente:
   (a) Riceve $\langle M, w \rangle$ come input.
   (b) M' costruisce (mediante un "modulo di reshaping") una nuova Macchina di Turing, che chiamiamo $M_{M,w}$ (o M^{tilde}_w), la cui codifica $\langle M_{M,w} \rangle$ viene passata al passo successivo.
   (c) La Macchina $M_{M,w}$ è definita come segue:
      - Quando $M_{M,w}$ viene avviata su un qualsiasi input (ad esempio, la stringa vuota ϵ), ignora l'input.

- Cancella il suo nastro di input.
- Scrive la stringa w (ottenuta dalla coppia $\langle M, w \rangle$ iniziale) sul proprio nastro.
- Simula la Macchina di Turing M (ottenuta dalla coppia $\langle M, w \rangle$ iniziale) sul contenuto attuale del nastro, ovvero su w.
- $M_{M,w}$ accetta se M accetta w, rifiuta se M rifiuta w, va in loop se M va in loop su w. In sintesi, $M_{M,w}$ si arresta su ϵ se e solo se M si arresta su w.

(d) M' dà in input la codifica $\langle M_{M,w} \rangle$ alla macchina $M^*_{HALT_\epsilon}$.

(e) M' adotta la risposta di $M^*_{HALT_\epsilon}$:

- Se $M^*_{HALT_\epsilon}$ risponde "sì", allora M' risponde "sì".
- Se $M^*_{HALT_\epsilon}$ risponde "no", allora M' risponde "no".

Ora analizziamo il linguaggio deciso da M', $L(M')$:

- **Se M' risponde "sì":** Ciò significa che $M^*_{HALT_\epsilon}$ ha risposto "sì" sull'input $\langle M_{M,w} \rangle$. Poiché $M^*_{HALT_\epsilon}$ decide $HALT_\epsilon$, la risposta "sì" implica che $M_{M,w}$ si arresta su ϵ. Per come abbiamo costruito $M_{M,w}$, questa si arresta su ϵ se e solo se M si arresta su w. Quindi, M si arresta su w. Ciò significa che $\langle M, w \rangle \in HALT$.
- **Se M' risponde "no":** Ciò significa che $M^*_{HALT_\epsilon}$ ha risposto "no" sull'input $\langle M_{M,w} \rangle$. Poiché $M^*_{HALT_\epsilon}$ decide $HALT_\epsilon$, la risposta "no" implica che $M_{M,w}$ non si arresta su ϵ. Per come abbiamo costruito $M_{M,w}$, questa non si arresta su ϵ se e solo se M non si arresta su w. Quindi, M non si arresta su w. Ciò significa che $\langle M, w \rangle \notin HALT$.

Il comportamento di M' è esattamente quello di una macchina che decide $HALT$. Dunque, $L(M') = HALT$.

Poiché M' è costruita usando $M^*_{HALT_\epsilon}$ (che è un decider e si arresta sempre), M' è anch'essa una Macchina di Turing che si arresta sempre, ovvero un decider. Questo implicherebbe che $HALT \in R$.

Contraddizione: Sappiamo che $HALT \notin R$. L'assunzione iniziale ($HALT_\epsilon \in R$) deve essere falsa.

Conclusione: $HALT_\epsilon \notin R$. □

Proposizione 4.1. *Il complemento di $HALT_\epsilon$, $\overline{HALT_\epsilon}$, non appartiene a R.*

Dimostrazione. Se $\overline{HALT_\epsilon}$ fosse in R, allora per la proprietà che R è chiusa rispetto al complemento, anche $HALT_\epsilon$ sarebbe in R. Ma abbiamo appena dimostrato che $HALT_\epsilon \notin R$. Questo è una contraddizione. □

5 Riepilogo e Conclusioni

Abbiamo esplorato la decidibilità di diversi linguaggi fondamentali per la teoria della computazione:

- LD: Non ricorsivamente enumerabile ($LD \notin R_e \implies LD \notin R$).
- $\overline{LD}$: Ricorsivamente enumerabile ($\overline{LD} \in R_e$), ma non ricorsivo ($\overline{LD} \notin R$).
- LU: Ricorsivamente enumerabile ($LU \in R_e$), ma non ricorsivo ($LU \notin R$).
- $HALT$: Ricorsivamente enumerabile ($HALT \in R_e$), ma non ricorsivo ($HALT \notin R$).

- $HALT_\epsilon$: Ricorsivamente enumerabile ($HALT_\epsilon \in R_e$), ma non ricorsivo ($HALT_\epsilon \notin R$).

Tutti i problemi di non appartenenza a R sono stati dimostrati mediante riduzione ad altri problemi di cui era già nota la non decidibilità. Questo è un metodo standard in teoria della computazione per dimostrare l'indecidibilità.

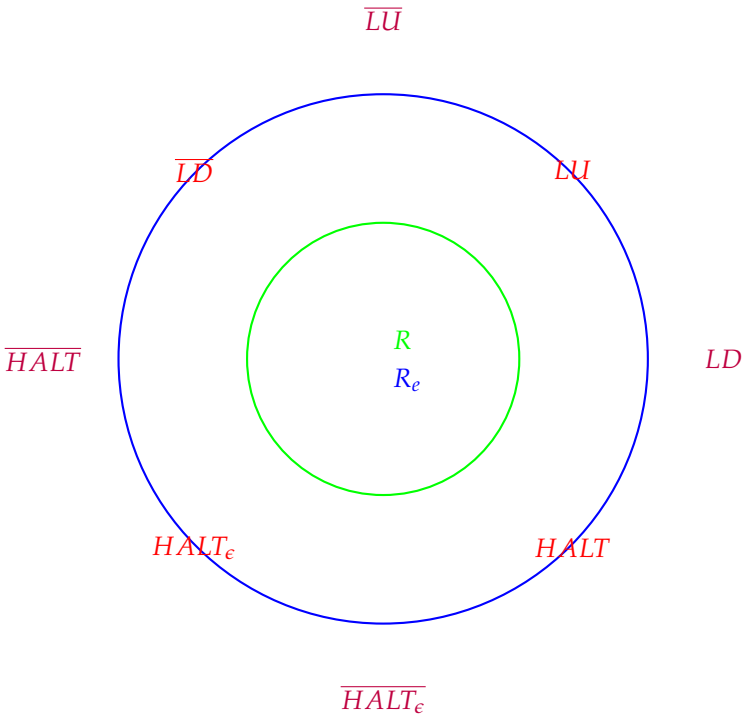

Figura 2: Mappa aggiornata delle classi di linguaggi R e R_e con i linguaggi discussi.