Indice

Informatica Teorica

Ede Boanini

3 ottobre 2025

1 Introduzione

1.1 Definizioni essenziali

Definizione 1.1 (Grafo)**.** Sia $G = (V, E)$ un grafo non orientato, dove:

- V è l'insieme dei nodi
- E è l'insieme degli archi

Definizione 1.2 (Coppia di nodi)**.** Siano u e v due nodi di un grafo $G = (V, E)$. La coppia $\{u, v\}$ rappresenta un arco che connette i nodi u e v.

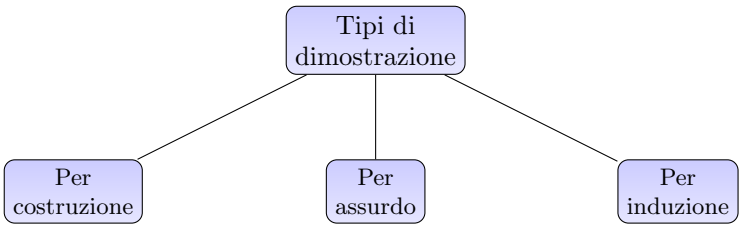
Definizione 1.3 (Grado di un nodo)**.** Numero di archi che collegano un nodo v ad altri nodi.

$$deg(v) = k$$

Definizione 1.4 (Grafo k-regolare)**.** Un grafo $G = (V, E)$ è k-regolare se ogni nodo ha grado k:

$$\forall v \in V, \quad deg(v) = k$$

1.2 Tipi di dimostrazioni



Definizione 1.5 (Dimostrazione per Costruzione)**.** Il teorema afferma che esiste un particolare tipo di oggetto. Un modo per dimostrare un teorema di questo tipo è mostrare come costruire l'oggetto.

★ Idea: vuoi dimostrare che un oggetto esiste? Lo costruisci direttamente.

Esempio

Per ogni numero pari $n > 2$, $\exists$ un grafo 3-regolare con n nodi.

Dimostrazione. Sia n un numero pari maggiore di 2. Costruisco un grafo $G = (V, E)$ con n nodi come segue:
Dispongo i nodi in cerchio. Collego ogni nodo con il successivo $\{i, i+1\}$ formando un ciclo. Dopodichè collego ogni nodo con il suo opposto $\{i, i+n/2\}$. In questo modo, ogni nodo ha 3 archi, quindi G è 3-regolare. □

Ho dimostrato il teorema costruendo un grafo che rispetta l'ipotesi (che il numero di nodi sia un numero pari maggiore di 2) arrivando poi alla tesi: ipotesi → costruzione → tesi.

Definizione 1.6 (Dimostrazione per Assurdo)**.** Assumo che il teorema sia falso e mostro che questa assunzione conduce a una proposizione che è logicamente impossibile, cioè che contraddice un fatto già dimostrato o una proprietà nota. Questa contraddizione implica che l'assunzione iniziale era falsa, quindi il teorema è vero.

★ Idea: supponi che il teorema sia falso (neghi la tesi). Se questa assunzione porta ad un'assurdità, allora il teorema deve essere vero.

Definizione 1.7 (Dimostrazione per Induzione)**.** Metodo usato per mostrare che tutti gli elementi di un insieme infinito possiedono una proprietà specifica. Questa dimostrazione consiste in due fasi:

- **Base:** dimostro che la proprietà $\mathcal{P}$ vale per il primo elemento dell'insieme. Verifico che $\mathcal{P}(1)$ (oppure $\mathcal{P}(0)$, dipende da dove parte l'insieme) è vera.
- **Passo induttivo:** suppongo che, per ogni $k \geq 1$, la proprietà $\mathcal{P}(k)$ sia vera (ipotesi induttiva). Ciò implica che anche $\mathcal{P}(k+1)$ è vera.

★ Idea: dimostri che una proprietà vale per infiniti casi.

Esempio

Per ogni $t \geq 0$, vale la seguente formula:

$$P_t = PM^t - Y\Big(\frac{M^t - 1}{M - 1}\Big)$$

Dimostrazione. **Base:** dimostra che la formula è vera per $t = 0$.

$$P_0 = PM^0 - Y\Big(\frac{M^0 - 1}{M - 1}\Big)$$

Sapendo che $P_0 = P$ e $M^0 = 1$, dunque ottengo:

$$P = P - Y\Big(\frac{0}{M - 1}\Big)$$

$$P = P - Y(0)$$

$$P = P - 0$$

$$P = P$$

Passo induttivo: $\forall k \geq 0$, assumo che la formula è vera per $t = k$ (ipotesi induttiva), ovvero assumo vera che:

$$P_k = PM^k - Y\Big(\frac{M^k - 1}{M - 1}\Big) \qquad \text{(ipotesi induttiva)}$$

questo per dimostrare che:

$$P_{k+1} = PM^{k+1} - Y\Big(\frac{M^{k+1} - 1}{M - 1}\Big) \qquad \text{(tesi)}$$

Se riesco a dimostrare che per $t = k + 1$ è vera, allora automaticamente è vera $\forall t \geq 0$.
Iniziamo:
Per definizione so che,

$$P_{k+1} = P_k M - Y$$

usando l'ipotesi induttiva sostituisco

$$P_{k+1} = P_k M - Y$$

e ottengo

$$P_{k+1} = \Big[PM^k - Y\Big(\frac{M^k - 1}{M - 1}\Big)\Big]M - Y$$

sviluppando alla fine ottengo

$$= PM^{k+1} - Y\Big(\frac{M^{k+1} - 1}{M - 1}\Big)$$

Che è proprio quello che volevo dimostrare. □

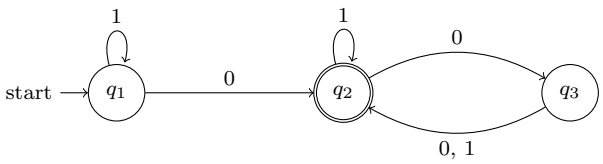
Ho dimostrato il teorema utilizzando l'ipotesi induttiva (che supponevo vera, per questo posso applicarla) nella definizione di P_{k+1}, poi ho sviluppato ed ottenuto la tesi.

IPOTESI	Condizioni che si assumono vere
TESI	Ciò che bisogna dimostrare

1.3 Macchina di Turing

Definizione 1.8 (Macchina deterministica)**.** Esiste una sola scelta possibile per ogni combinazione di stato e simbolo dell'alfabeto.

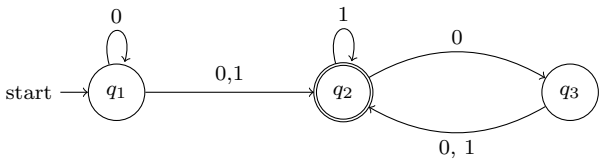
Esempio



Se sono nello stato q_1 e leggo il simbolo 0 ho solo una scelta: cambiare stato in q_2.

Definizione 1.9 (Macchina non deterministica)**.** Esistono più scelte possibili per ogni combinazione di stato e simbolo dell'alfabeto.

Esempio



Se sono nello stato q_1 e leggo il simbolo 0 ho più scelte:

- cambiare stato in q_2
- tornare nello stato q_1

Definizione 1.10 (Linguaggio)**.** Insieme di stringhe costruite a partire da un alfabeto e che rispettano certe regole.

Esempio

- Sia l'alfabeto $\Sigma = \{0, 1\}$
- Sia L il linguaggio

Linguaggio che contiene stringhe con un numero pari di 1:
$L = \{11, 011, 0011, 1100, 0110, 1111, 01111, \dots\}$

Definizione 1.11 (MdT modello standard)**.** Una Macchina di Turing standard è una 7-upla $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ dove Q, Σ, Γ sono insiemi finiti e:

- Q insieme degli stati
- Σ alfabeto di input (non contiene il simbolo * blank); $\Sigma \subseteq \Gamma$
- Γ alfabeto del nastro (tutti i simboli che può leggere e scrivere sul nastro, include anche il simbolo * blank)
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{S, D\}$ funzione di transizione

- $q_0 \in Q$ stato iniziale
- $q_{accept} \in Q$ è lo stato di accettazione
- $q_{reject} \in Q$ è lo stato di rifiuto dove $q_{accept} \neq q_{reject}$

Il modello standard è dunque una macchina deterministica.

Definizione 1.12 (Linguaggio decidibile/ricorsivo)**.** Un linguaggio L è **decidibile** se $\exists$ una MdT M tale che, per ogni stringa $w \in \Sigma^*$ in input:

- Se $w \in L \implies$ M si ferma (accettando w) nello stato q_{accept}; Quindi accetta la stringa.
- Se $w \notin L \implies$ M si ferma (rifiutando w) nello stato q_{reject}; Quindi rifiuta la stringa.

Notare che la macchina si ferma sempre. In questo caso si dice che la macchina "decide" L.

Definizione 1.13 (Linguaggio semidecidibile/ricorsivamente enumerabile)**.** Un linguaggio L è **semidecidibile** se $\exists$ una MdT M tale che, per ogni stringa $w \in \Sigma^*$ in input:

- Se $w \in L \implies$ M si ferma (accettando w) nello stato q_{accept}; Quindi accetta la stringa.
- Se $w \notin L \implies$ M va in loop, non fermandosi mai.

Notare che la macchina non si ferma sempre. In questo caso si dice che la macchina "riconosce" L.

Definizione 1.14 (Linguaggio indecidibile)**.** Un linguaggio L è **indecidibile** quando $\nexists$ una MdT M in grado di decidere L. Ma può esistere una MdT in grado di riconoscere L.
Nota: L è indecidibile e potrebbe essere semidecidibile, ma mai decidibile.

Definizione 1.15 (Linguaggio di una macchina)**.** Se A (linguaggio) è l'insieme di tutte le stringhe che la MdT M accetta oppure riconosce, dico che M accetta o riconosce A e lo indico come $L(M) = A$.
Se M non accetta nessuna stringa, lo indico come $L(M) = \emptyset$

Definizione 1.16 (Enumeratore)**.** Un enumeratore è una MdT che genera tutte le stringhe del linguaggio (separandole con il simbolo "#"), una dopo l'altra, senza ricevere nessun input. Infatti, la macchina E inizia a lavorare su nastro vuoto (input vuoto).

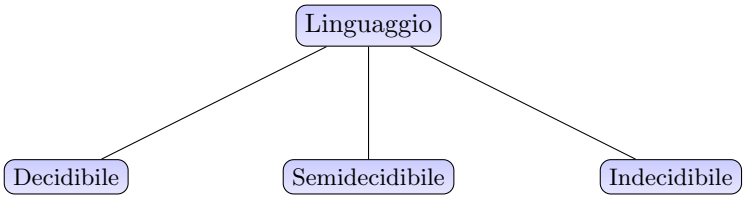
Funzionamento:

1. L'enumeratore viene eseguito inizialmente su nastro vuoto (nessun input)
2. Genera la stringa scrivendola sul nastro

3. Quando ha terminato di scrivere la stringa, la invia al dispostivo di output (stampante)

4. Torna al passo 2

2 Decidibilità



Come provare che un linguaggio è decidibile, semidecidibile o indecidibile?

- L è decidibile: dimostrazione per costruzione
- L è semidecidibile: dimostrazione per costruzione
- L è indecidibile: dimostrazione per assurdo + riduzione ad un problema che sappiamo essere indecidibile (es: Teorema dell'arresto) oppure Teorema di Rice

Definizione 2.1 (Codifica di una MdT)**.** La notazione $R(M)$ indica la codifica di una macchina. Spesso utilizzata nelle dimostrazioni.

2.1 Decidibilità

Questa dimostrazione fa riferimento ad un automa a stati finiti deterministico (DFA). Lo stesso quesito per una MdT non è decidibile (vedi esercizio 2.2). Lo scopo è mostrare come funziona la dimostrazione per costruzione.

Esercizio 1.1

Sia il linguaggio $L_{DFA} = \{(R(M), w) \mid M \text{ accetta } w\}$. Dimostrare che L è decidibile.

Ragionamento. L_{DFA} è l'insieme delle codifiche di automi a stati finiti deterministici che accettano w. Ovvero, siano M_1,M_2,M_3 tre DFA:

- M_1 accetta w, allora $R(M_1) \in L_{DFA}$
- M_2 rifiuta w, allora $R(M_2) \notin L_{DFA}$
- M_3 accetta w, allora $R(M_3) \in L_{DFA}$

Quindi $L_{DFA} = \{R(M_1), R(M_3)\}$
**utilizzo dimostrazione per costruzione.

Dimostrazione.

$$L_{DFA} = \{(R(M), w) \mid M \text{ accetta } w\} \qquad \text{(ipotesi)}$$

$$L_{DFA}\text{è decidibile} \qquad \text{(tesi)}$$

Sapendo che per ipotesi $L_{DFA} = \{(R(M), w) \mid M \text{ accetta } w\}$, allora costruisco una MdT N che decide L_{DFA}. Dato in input $(R(M), w)$, dove $R(M)$ è la codifica di un DFA arbitrario e w una stringa:

1. Controlla che $R(M)$ sia una codifica valida e che w sia una stringa, altrimenti rifiuta.
2. Simula M su input w.

3. Se la simulazione termina:
   - in uno stato accettante $\implies$ N termina accettando $R(M)$;
     Quindi $R(M) \in L_{DFA}$
   - in uno stato di rifiuto $\implies$ N termina rifiutando $R(M)$;
     Quindi $R(M) \notin L_{DFA}$

Ho costruito una macchina N in grado di decidere L_{DFA}; Inoltre, poichè un DFA ha un numero di stati finiti e la stringa w è finita, la simulazione termina in uno stato finale (accettante o di rifiuto), garantendo che anche N si fermi sempre accettando o rifiutando l'input. Pertanto, si dimostra che L_{DFA} è decidibile. □

2.2 Indecidibilità

Un linguaggio L è indecidibile quando $\nexists$ una MdT in grado di fermarsi sempre accettando o rifiutando l'input. Ciò vuol dire che:

- L è indecidibile e anche semidecidibile
- L è indecidibile ma non semidecidibile

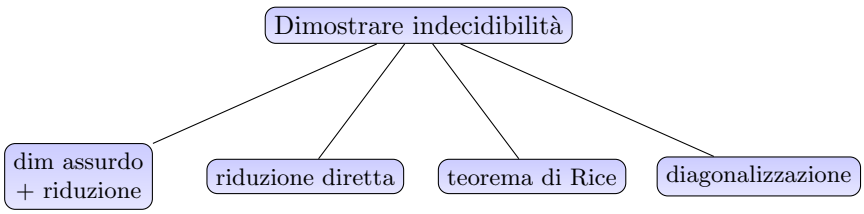
Pertanto, quando viene richiesto di dimostrare l'indecidibilità di un linguaggio:

- Se sospetto che L possa essere indecidibile + semidecidibile, allora:

  1. Applico la dimostrazione per costruzione;
     cioè, costruisco una MdT che riconosce[1] il linguaggio.
  2. Effettuo la riduzione ad un linguaggio noto indecidibile.

  Nel primo punto dimostro la semidecidibilità di L e nel secondo la indecidibilità.

- Se sospetto che il linguaggio è indecidibile ma non semidecidibile, allora posso considerare una di queste tecniche:



  - dimostrazione per assurdo + effettuo la riduzione ad un linguaggio noto indecidibile
  - riduzione diretta
  - uso il Teorema di Rice
  - applico la diagonalizzazione

[1] Turing-reconizable: MdT si ferma accettando le stringhe che appartengono al linguaggio e va in loop per quelle che non appartengono.

2.3 Esercizi

Teorema

Sia il linguaggio $L = \{(R(M), w) \mid M \text{ accetta } w\}$. Dimostrare che L è semidecidibile ma anche indecidibile.

Esercizio 2.1

Sia il linguaggio $L_1 = \{(R(M), w) \mid M \text{ accetta } w\}$. Dimostrare che L è semidecidibile.

Ragionamento. L_1 è l'insieme delle codifiche di MdT che accettano w. Ovvero, siano M_1,M_2,M_3 tre MdT:

- M_1 accetta w, allora $R(M_1) \in L_1$
- M_2 rifiuta w, allora $R(M_2) \notin L_1$
- M_3 accetta w, allora $R(M_3) \in L_1$

Quindi $L_1 = \{R(M_1), R(M_3)\}$
Dato che devo dimostrare la semidecidibilità di L_1, costruisco una MdT che riconosce il linguaggio.

Dimostrazione.

$$L_1 = \{(R(M), w) \mid M \text{ accetta } w\} \qquad \text{(ipotesi)}$$

$$L_1 \text{ è semidecidibile} \qquad \text{(tesi)}$$

Costruisco una MdT N che riconosce L_1. Dato in input $(R(M), w)$, dove $R(M)$ è la codifica di una MdT arbitraria e w una stringa, la MdT N si comporta come segue:

1. Controlla che $R(M)$ sia una codifica valida e che w sia una stringa, altrimenti rifiuta.
2. Simula M su input w.
3. Se la simulazione termina:
   - in uno stato accettante $\implies$ N termina accettando $R(M)$;
     Quindi $R(M) \in L_1$
   - in uno stato di rifiuto $\implies$ N termina rifiutando $R(M)$;
     Quindi $R(M) \notin L_1$

Notare che la macchina N va in loop sull'input $(R(M), w)$ se M va in loop su w.
Un decisore deve sempre fermarsi (in ogni caso), ma qui, se M va in loop su w, N non si fermerà mai. Questo rende N un riconoscitore e non un decisore.
Ho dimostrato che L_1 è semidecidibile costruendo la MdT N che riconosce tale linguaggio.
Pertanto, si dimostra che L_1 è semidecidibile. □

Esercizio 2.2

Sia il linguaggio $L_1 = \{(R(M), w) \mid M \text{ accetta } w\}$. Dimostrare che L è indecidibile.

Ragionamento. Per dimostrarlo, applico:

- Dim per assurdo (suppongo per assurdo (nego la tesi) per poi ottenere una contraddizione, che rende falsa l'assunzione fatta)

Dimostrazione.

$$L_1 = \{(R(M), w) \mid M \text{ accetta } w\} \qquad \text{(ipotesi)}$$

$$L_1 \text{ è indecidibile} \qquad \text{(tesi)}$$

Per applicare la dimostrazione per assurdo nego la tesi, quindi suppongo per assurdo che L_1 sia decidibile. Sapendo che per ipotesi (la mia per assurdo, non quella del teorema) L_1 è decidibile, allora $\exists$ una MdT N che decide L_1 con il seguente funzionamento:

$$N(R(M), w) = \begin{cases} \text{accept} & \text{se } M \text{ accetta } w \\ \text{reject} & \text{se } M \text{ non accetta } w \end{cases}$$

Con "non accetta" si intende che M potrebbe rifiutare w oppure andare in loop.
Dato in input $(R(M), w)$ alla MdT N, dove $R(M)$ è la codifica di una MdT arbitraria e w una stringa:

1. N controlla che $R(M)$ sia una codifica valida e che w sia una stringa, altrimenti rifiuta.
2. Simula M su input w.
3. Se la simulazione termina:
   - in uno stato accettante $\implies$ N termina accettando $R(M)$;
     Quindi $R(M) \in L_1$
   - in uno stato di rifiuto/va in loop $\implies$ N termina rifiutando $R(M)$;
     Quindi $R(M) \notin L_1$

Adesso costruisco una nuova MdT D con N come subroutine. D chiama N per determinare cosa fa M quando l'input per M è la sua stessa codifica (e non la stringa w). Il comportamento di D è l'opposto di N, ovvero:

$$D(R(M)) = \begin{cases} \text{accept} & \text{se } M \text{ non accetta } R(M) \\ \text{reject} & \text{se } M \text{ accetta } R(M) \end{cases}$$

Funzionamento di D:

1. Esegue N su input $(R(M), R(M))$ dove $R(M)$ è la codifica della MdT M:
   - Se N si ferma accettando $\implies$ D rifiuta.
     (ricorda che se N accetta allora vuol dire che M ha accettato $R(M)$)
   - Se N si ferma rifiutando $\implies$ D accetta.
     (ricorda che se N rifiuta allora vuol dire che M non ha accettato $R(M)$)

Cosa succederebbe se fornissimo alla MdT D la propria codifica come input? Otterrei:

$$D(R(D)) = \begin{cases} \text{accept} & \text{se } D \text{ non accetta } R(D) \\ \text{reject} & \text{se } D \text{ accetta } R(D) \end{cases}$$

che è una contraddizione.
Conclusione: Poiché abbiamo ottenuto una contraddizione (D rifiuta $R(D)$ quando D accetta $R(D)$), l'ipotesi che L_1 sia decidibile è falsa. Pertanto, L_1 è indecidibile. □

Teorema

Sia il linguaggio $L = \{(R(M), w) \mid M \text{ accetta } w\}$. Dimostrare che $\overline{L}$ non è semidecidibile.

Esercizio 2.3

Sia il linguaggio $L_1 = \{(R(M), w) \mid M \text{ accetta } w\}$. Dimostrare che $\overline{L_1}$ non è semidecidibile.

Ragionamento. Devo dimostrare che $\overline{L_1} = \{(R(M), w) \mid M \text{ non accetta } w\}$ non è semidecidibile, ovvero che $\nexists$ una MdT in grado di riconoscere $\overline{L_1}$. Procedo per assurdo.

Dimostrazione.

$$L_1 = \{(R(M), w) \mid M \text{ accetta } w\} \qquad \text{(ipotesi)}$$

$$\overline{L_1} \text{ non è semidecidibile} \qquad \text{(tesi)}$$

Sappiamo che L_1 è un linguaggio semidecidibile (esercizio 2.1).
Suppongo per assurdo che anche $\overline{L_1}$ sia semidecidibile (nego la tesi, ipotesi per assurdo). Dato che $L_1, \overline{L_1}$ sono entrambi semidecidibili $\implies$ per definizione, L_1 è decidibile. Ma questo è assurdo perchè abbiamo dimostrato che L_1 è indecidibile (esercizio 2.2). Ciò porta ad una contraddizione e l'ipotesi che $\overline{L_1}$ sia semidecidibile è falsa. Pertanto, $\overline{L_1}$ è non semidecidibile.

□

2.4 Famosi problemi indecidibili

2.4.1 Halting Problem

Definizione 2.2. (Problema dell'arresto) Esiste una MdT H che preso in input (M, w), termina dicendo:

- Sì, se M termina su w
- No, se M va in loop su w

Tale macchina esiste? No.

Definizione 2.3. (Linguaggio Halting Problem) Sia $\mathcal{L}_{\text{Halt}}$ il linguaggio del problema dell'arresto definito come segue:

$$\mathcal{L}_{\text{Halt}} = \{(R(M), w) \mid M \text{ termina su } w\}$$

Notare che $\mathcal{L}_{\text{Halt}}$ è semidecidibile, ovvero $\exists$ una MdT N che dato in input $(R(M), w)$:

- Se M termina su w, allora N accetta
  (Se M termina su $w \implies (R(M), w) \in \mathcal{L}_{\text{Halt}}$)[2].
- Se M non termina su w, allora N va in loop

*Attenzione: "M termina su w" e non dice "M accetta w", che sono due cose differenti. Se M termina su w, allora N accetta; ovvero l'importante è che M termini su w, che sia in uno stato accettante o di rifiuto. (Quindi non ci interessa se M accetta o rifiuta w, quello che ci interessa è che M si fermi su w.)

Pertanto, $\exists$ una MdT che riconosce (ma non decide) $\mathcal{L}_{\text{Halt}}$.

Osservazione

Il problema dell'arresto (o, equivalentemente, il linguaggio $\mathcal{L}_{\text{Halt}}$) è **indecidibile**, cioè non esiste una MdT che termina sempre dando la risposta corretta. Tuttavia, $\mathcal{L}_{\text{Halt}}$ è **semidecidibile**: esiste infatti una MdT universale che, dato in input $(R(M), w)$, termina accettando se M termina su w, mentre può non terminare se M non termina su w.

Il problema dell'arresto è semidecidibile + indecidibile, oppure, equivalentemente scrivo:

$\mathcal{L}_{\text{Halt}}$ è semidecidibile ma non decidibile

[2]Ho scritto "M termina su w" e non "$w \in L(M)$" poichè si sta parlando di terminare su w e non accettare w. La dicitura classica $w \in L(M)$ si scrive solo quando M accetta w.

Teorema : Halting Problem (2.4.1)

Il problema dell'arresto è indecidibile.
$\mathcal{L}_{\text{Halt}} = \{(R(M), w) \mid M \text{ termina su } w\}$ è indecidibile.

Dimostrazione.

$$\mathcal{L}_{\text{Halt}} = \{(R(M), w) \mid M \text{ termina su } w\} \qquad \text{(ipotesi)}$$

$$\mathcal{L}_{\text{Halt}} \text{ è indecidibile} \qquad \text{(tesi)}$$

Suppongo per assurdo che il problema dell'arresto sia decidibile (ipotesi per assurdo, nego la tesi). Quindi, per ipotesi, $\exists$ una MdT H che risolve il problema dell'arresto. H su input $(R(M), w)$ dove $R(M)$ è la codifica di una MdT arbitraria e w una stringa:

- Se M termina su w, allora H accetta
- Se M non termina su w, allora H rifiuta

Modifico la MdT H per costruire H' con il seguente comportamento.

$$H'(R(M), w) = \begin{cases} \text{loop} & \text{se } M \text{ termina su } w \\ \text{reject} & \text{se } M \text{ non termina su } w \end{cases}$$

H' su input $(R(M), w)$:

- Se M termina su w, allora H' va in loop
- Se M non termina su w, allora H' rifiuta

Adesso costruisco un'altra MdT D (che prende in input solo la codifica di una macchina)[a] combinando H' con una procedura.

$$D(R(M)) = \begin{cases} \text{loop} & \text{se } M \text{ termina su } R(M) \\ \text{reject} & \text{se } M \text{ non termina su } R(M) \end{cases}$$

La macchina D su input $(R(M))$:

1. Controlla che $R(M)$ sia un codifica valida, altrimenti rifiuta
2. Esegue varie transizioni che, a partire dall'input, produce la coppia $(R(M), R(M))$
   (legge la codifica della macchina in input e la copia come secondo argomento da fornire a H')
3. Esegue H' sull'input $(R(M), R(M))$:
   - Se H' va in loop, allora anche D va in loop
     (ricorda che se H' va in loop vuol dire che M ha terminato su $R(M)$)
   - Se H' rifiuta, allora D rifiuta
     (ricorda che se H' rifiuta vuol dire che M non ha mai terminato su $R(M)$)

E adesso cosa succederebbe se fornissi alla MdT D la propria codifica come input? Otterrei:

$$D(R(D)) = \begin{cases} \text{loop} & \text{se } D \text{ termina su } R(D) \\ \text{reject} & \text{se } D \text{ non termina su } R(D) \end{cases}$$

una contraddizione.
Conclusione: Poichè ho ottenuto una contraddizione (D va in loop su $R(D) \iff D$ termina su $R(D)$), l'ipotesi che il problema dell'arresto sia decidibile è falsa. Pertanto, il problema dell'arresto è indecidibile (oppure scrivo: Pertanto, $\mathcal{L}_{\text{Halt}}$ è indecidibile). □

[a] invece dell'input $(R(M), w)$, prende in input $(R(M))$

Teorema

$\mathcal{L}_{\text{Halt}}$ è semidecidibile. $\overline{\mathcal{L}_{\text{Halt}}}$ non è semidecidibile.

Dimostrazione. Sappiamo per definizione, che $\mathcal{L}_{\text{Halt}}$ è semidecidibile.
Suppongo per assurdo che $\overline{\mathcal{L}_{\text{Halt}}}$ sia anch'esso semidecidibile. Dato che $\overline{\mathcal{L}_{\text{Halt}}}$,$\mathcal{L}_{\text{Halt}}$ sono entrambi semidecidibili $\implies$ per definizione, $\mathcal{L}_{\text{Halt}}$ è decidibile. Ma questo è assurdo perchè abbiamo dimostrato che $\mathcal{L}_{\text{Halt}}$ è indecidibile (teorema 2.4.1). Ciò porta ad una contraddizione e l'ipotesi che $\overline{\mathcal{L}_{\text{Halt}}}$ sia semidecidibile è falsa.
Pertanto, $\overline{\mathcal{L}_{\text{Halt}}}$ è non semidecidibile. □

2.4.2 Problema del nastro vuoto

Definizione 2.4 (Problema del nastro vuoto)**.** Esiste una MdT E che preso in input $R(M)$, termina dicendo:

- Sì, se M termina su nastro vuoto
- No, se M non termina su nastro vuoto

Tale macchina esiste? No.
*Attenzione: "termina su nastro vuoto" significa "termina quando parte su nastro vuoto".

Definizione 2.5 (Linguaggio Blank-Tape Halting Problem)**.** Sia $\mathcal{L}_{\text{Blank-Tape}}$ il linguaggio del problema del nastro vuoto definito come segue:

$$\mathcal{L}_{\text{Blank-Tape}} = \{R(M) \mid M \text{ termina su nastro vuoto}\}$$

Notare che $\mathcal{L}_{\text{Blank-Tape}}$ è semidecidibile.

Quindi, il problema del nastro vuoto è semidecidibile + indecidibile, oppure, equivalentemente scrivo:

$\mathcal{L}_{\text{Blank-Tape}}$ è semidecidibile ma non decidibile

Teorema : Blank-Tape Halting Problem (versione A)

Il problema del nastro vuoto è indecidibile.
$\mathcal{L}_{\text{Blank-Tape}} = \{R(M) \mid M \text{ termina su nastro vuoto}\}$ è indecidibile.

Ragionamento. Per dimostrarlo, uso la dim per assurdo e poi effettuo una riduzione da un problema noto indecidibile (es: problema dell'arresto) al mio (problema del nastro vuoto).
Dato che sto ragionando per assurdo, non posso usare questa: se $A \leq_m B \ \wedge \ A$ è indecidibile $\implies$ B è indecidibile (teorema 3.2); ma devo usare questa: se $A \leq_m B \ \wedge \ B$ è decidibile $\implies$ A è decidibile (teorema 3.1);
Quindi costruisco una riduzione del tipo: $\mathcal{L}_{\text{Halt}} \leq_m \mathcal{L}_{\text{Blank-Tape}}$.

Dimostrazione.

$$\mathcal{L}_{\text{Blank-Tape}} = \{R(M) \mid M \text{ termina su nastro vuoto}\} \qquad \text{(ipotesi)}$$

$$\mathcal{L}_{\text{Blank-Tape}} \text{ è indecidibile} \qquad \text{(tesi)}$$

Siano $\mathcal{L}_{\text{Halt}}$, $\mathcal{L}_{\text{Blank-Tape}}$ due linguaggi su Σ_H^*, Σ_B^* rispettivamente, dove:

- $\mathcal{L}_{\text{Halt}}$ è il linguaggio del problema dell'arresto
- $\mathcal{L}_{\text{Blank-Tape}}$ è il linguaggio del problema del nastro vuoto

1. Ipotesi per assurdo

Suppongo per assurdo che $\mathcal{L}_{\text{Blank-Tape}}$ sia decidibile (ipotesi per assurdo). Quindi per ipotesi $\exists$ una MdT E che decide $\mathcal{L}_{\text{Blank-Tape}}$. Ovvero:

$$E(R(M)) = \begin{cases} \text{accept} & \text{se } M \text{ termina su nastro vuoto} \\ \text{reject} & \text{se } M \text{ non termina su nastro vuoto} \end{cases}$$

2. Costruisco funzione di riduzione

Costruisco una funzione di riduzione $f : \Sigma_H^* \to \Sigma_B^*$ t.c. $\forall w \in \Sigma_H^*$:

$$w \in \mathcal{L}_{\text{Halt}} \iff f(w) \in \mathcal{L}_{\text{Blank-Tape}}$$

Quindi, costrusco la funzione di riduzione (che sarebbe la MdT R):

- Se w non è della forma $(R(M), w)$, pongo $f(w) = 1$
- Se $w = (R(M), w)$, allora pongo $f(w) = R(M')$

Ovvero, il funzionamento della MdT R è costruire una nuova macchina M', praticamente il seguente:

1. R riceve come input $w = (R(M), w)$ da N
2. R costruisce la nuova macchina M' che ha questo comportamento:
   (a) M' viene avviata su nastro vuoto (condizione del Blank-Tape Halting Problem)
   (b) M' scrive w sul nastro
   (c) M' riporta la testina all'inizio del nastro
   (d) M' esegue M su w
3. R genera l'output $R(M')$ (che è la nuova macchina costruita nel passo 2)

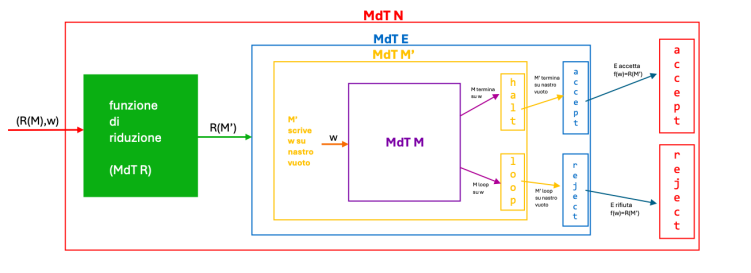
3. Costruisco MdT che decide il problema dell'arresto

Adesso, costrusco una MdT N che decide $\mathcal{L}_{\text{Halt}}$:

1. N su input $w \in \Sigma_H^*$ calcola $f(w) \in \Sigma_B^*$
2. N esegue E su $f(w)$:
   - Se E accetta, allora N accetta
     (ricorda che se E accetta, vuol dire che M' ha terminato su nastro vuoto (se M' termina su nastro vuoto, vuol dire che M ha terminato su w))
     ovvero: Se $R(M') \in \mathcal{L}_{\text{Blank-Tape}}$, allora $(R(M), w) \in \mathcal{L}_{\text{Halt}}$
     ma è più corretto scrivere:
     $w \in \mathcal{L}_{\text{Halt}} \iff f(w) \in \mathcal{L}_{\text{Blank-Tape}}$
   - Se E rifiuta, allora N rifiuta
     (ricorda che se E rifiuta, vuol dire che M' non ha terminato su nastro vuoto (se M' non termina su nastro vuoto (loop), vuol dire che M non ha terminato su w (loop)))
     ovvero: Se $R(M') \notin \mathcal{L}_{\text{Blank-Tape}}$, allora $(R(M), w) \notin \mathcal{L}_{\text{Halt}}$
     ma è più corretto scrivere:
     $w \notin \mathcal{L}_{\text{Halt}} \iff f(w) \notin \mathcal{L}_{\text{Blank-Tape}}$

Quindi N è un algoritmo di decisione per il problema dell'arresto che applica la funzione di riduzione per ogni input e sfrutta la MdT E che decide $\mathcal{L}_{\text{Blank-Tape}}$ per decidere $\mathcal{L}_{\text{Halt}}$. Perchè alla fine N si ferma accettando w se e solo se E si ferma accettando $f(w)$ oppure N si ferma rifiutando w se e solo se E si ferma rifiutando $f(w)$.
La costruzione e il funzionamento della MdT N rende il problema dell'arresto decidibile. Ma questo è assurdo perchè sappiamo che il problema dell'arresto è indecidibile, quindi questo porta ad una contraddizione. Poichè ho ottenuto una contraddizione, l'ipotesi che il problema del nastro vuoto sia decidibile è falsa. Pertanto, il problema del nastro vuoto è indecidibile. □

Disegno:



Teorema : Blank-Tape Halting Problem (versione B)

Il problema del nastro vuoto è indecidibile.
$\mathcal{L}_{\text{Blank-Tape}} = \{R(M) \mid M \text{ termina su nastro vuoto}\}$ è indecidibile.

Ragionamento. Per dimostrarlo, non uso la dim per assurdo ma effettuo **direttamente** una riduzione dal problema dell'arresto al mio. Ovvero, costruisco una riduzione del tipo: $\mathcal{L}_{\text{Halt}} \leq_m \mathcal{L}_{\text{Blank-Tape}}$.
Uso solo la riduzione perchè "sfrutto":

1. La definizione di funzione di riduzione
   ($L_1 \leq_m L_2$ se $\exists$ una funzione di riduzione $f : \Sigma_1^* \to \Sigma_2^*$ t.c. $\forall w \in \Sigma_1^*$:
   $w \in L_1 \iff f(w) \in L_2$)
2. Il teorema $A \leq_m B \ \wedge \ A$ è indecidibile $\implies B$ è indecidibile (teorema 3.2).

Dimostrazione.

$$\mathcal{L}_{\text{Blank-Tape}} = \{R(M) \mid M \text{ termina su nastro vuoto}\} \qquad \text{(ipotesi)}$$

$$\mathcal{L}_{\text{Blank-Tape}} \text{ è indecidibile} \qquad \text{(tesi)}$$

Siano $\mathcal{L}_{\text{Halt}}$, $\mathcal{L}_{\text{Blank-Tape}}$ due linguaggi su Σ_H^*, Σ_B^* rispettivamente, dove:

- $\mathcal{L}_{\text{Halt}}$ è il linguaggio del problema dell'arresto
- $\mathcal{L}_{\text{Blank-Tape}}$ è il linguaggio del problema del nastro vuoto

Sia $\mathcal{L}_{\text{Halt}} = \{(R(M), w) \mid M \text{ termina su } w\}$. È noto che $\mathcal{L}_{\text{Halt}}$ è indecidibile.
1. Riduzione
Definisco una funzione di riduzione[a] $f : \Sigma_H^* \to \Sigma_B^*$ come segue.
Dato in input $(R(M), w)$[b] la funzione f genera come output $R(M')$[c], dove M' è la nuova MdT costruita dalla funzione di riduzione f che ha questo comportamento:

1. M' inizia la computazione su nastro vuoto
2. M' scrive w sul nastro
3. M' riporta la testina all'inizio del nastro
4. M' esegue M su w

La funzione di riduzione f trasforma ogni istanza del problema dell'arresto $(R(M), w)$ in un'istanza $R(M')$ del problema del nastro vuoto, dove M' inizia la computazione

su nastro vuoto, scrive w sul nastro ed esegue M su w. Per definizione di funzione di riduzione, vale:

$$(R(M), w) \in \mathcal{L}_{\text{Halt}} \iff f((R(M), w)) \in \mathcal{L}_{\text{Blank-Tape}}.^{d}$$

Poiché $\mathcal{L}_{\text{Halt}} \leq_m \mathcal{L}_{\text{Blank-Tape}}$ e $\mathcal{L}_{\text{Halt}}$ è indecidibile, segue che $\mathcal{L}_{\text{Blank-Tape}}$ è indecidibile. □

[a] ricorda la funzione di riduzione trasforma un'istanza del problema dell'arresto H in un'istanza del problema del nastro vuoto B

[b] istanza del problema dell'arresto

[c] istanza del problema del nastro vuoto

[d] dove $f((R(M), w)) = R(M')$; precisando che $w = (R(M), w)$ e $f(w) = R(M')$ dove w è l'input dato alla funzione di riduzione e $f(w)$ è l'ouput generato dalla stessa funzione di riduzione, ovvero il valore ottenuto applicando f a w.

2.4.3 Problema del linguaggio vuoto

Teorema : Emptiness Problem (versione lunga)

Il problema del linguaggio vuoto è indecidibile.
$\mathcal{L}_{\text{Emptiness}} = \{R(M) \mid L(M) = \emptyset\}$ è indecidibile.

Ragionamento. Per dimostrarlo, uso la dim per assurdo + costruisco una riduzione dal problema dell'arresto al mio, ovvero $\mathcal{L}_{\text{Halt}} \leq_m \mathcal{L}_{\text{Emptiness}}$.
Uso quindi la MdT E che decide $\mathcal{L}_{\text{Emptiness}}$ per decidere $\mathcal{L}_{\text{Halt}}$. Uso solo la riduzione perchè "sfrutto":

1. La definizione di funzione di riduzione
   ($L_1 \leq_m L_2$ se $\exists$ una funzione di riduzione $f : \Sigma_1^* \to \Sigma_2^*$ t.c. $\forall w \in \Sigma_1^*$:
   $w \in L_1 \iff f(w) \in L_2$)
2. Il teorema $A \leq_m B \ \wedge \ B$ è decidibile $\implies A$ è decidibile (teorema 3.1).

Dimostrazione.

$$\mathcal{L}_{\text{Emptiness}} = \{R(M) \mid L(M) = \emptyset\} \qquad \text{(ipotesi)}$$

$$\mathcal{L}_{\text{Emptiness}} \text{ è indecidibile} \qquad \text{(tesi)}$$

Siano $\mathcal{L}_{\text{Halt}}$, $\mathcal{L}_{\text{Emptiness}}$ due linguaggi su Σ_H^*, Σ_E^* rispettivamente, dove:

- $\mathcal{L}_{\text{Halt}}$ è il linguaggio del problema dell'arresto
- $\mathcal{L}_{\text{Emptiness}}$ è il linguaggio dell'emptiness problem

1. Ipotesi per assurdo
Suppongo per assurdo che $\mathcal{L}_{\text{Emptiness}}$ sia decidibile (ipotesi per assurdo). Allora $\exists$ una MdT E che decide $\mathcal{L}_{\text{Emptiness}}$. Ovvero:

$$E(R(M)) = \begin{cases} \text{accept} & \text{se } M \text{ non accetta nessuna stringa} \\ \text{reject} & \text{se } M \text{ accetta almeno una stringa} \end{cases}$$

2. Riduzione
Definisco una funzione di riduzione[a] $f : \Sigma_H^* \to \Sigma_E^*$ come segue.
Dato in input $(R(M), w)$[b] la funzione f genera come output $R(M')$[c], dove M' è la nuova MdT costruita dalla funzione di riduzione f che ha il seguente comportamento ($R(M)$ è la codifica di una MdT arbitraria e w una stringa).

M' su una stringa di input x generica:

- se $x \neq w$, rifiuta
- se $x = w$, esegue M su w dove:
  - se M termina su w, allora M' accetta w (perchè $x = w$)
  - se M va in loop su w, allora M' va in loop

In sostanza, osservandolo dal punto di vista del linguaggio accettato da M':

$$L(M') = \begin{cases} \{w\} & \text{se } M \text{ termina su } w \\ \emptyset & \text{se } M \text{ non termina (loop) su } w \end{cases}$$

3. Costruisco MdT che decide il problema dell'arresto

Adesso, costrusco una MdT N che decide $\mathcal{L}_{\text{Halt}}$:

1. N su input $w \in \Sigma_H^*$ calcola $f(w) \in \Sigma_E^*$ (riduzione)
2. N esegue E su $f(w)$:
   - Se E accetta, allora N rifiuta
     (se E accetta, vuol dire che M' ha linguaggio vuoto, cioè M' loop w, quindi $L(M') = \emptyset$ (se M' loop su w, vuol dire che M ha loopato w))
     ovvero: Se $R(M') \in \mathcal{L}_{\text{Emptiness}}$, allora $(R(M), w) \notin \mathcal{L}_{\text{Halt}}$
     ma è più corretto scrivere:
     $w \notin \mathcal{L}_{\text{Halt}} \iff f(w) \in \mathcal{L}_{\text{Emptiness}}$
   - Se E rifiuta, allora N accetta
     (se E rifiuta, vuol dire che M' non ha linguaggio vuoto, cioè M' ha accettato w, quindi $L(M') = \{w\}$ (se M' accetta w, vuol dire che M ha terminato su w))
     ovvero: Se $R(M') \notin \mathcal{L}_{\text{Emptiness}}$, allora $(R(M), w) \in \mathcal{L}_{\text{Halt}}$
     ma è più corretto scrivere:
     $w \in \mathcal{L}_{\text{Halt}} \iff f(w) \notin \mathcal{L}_{\text{Emptiness}}$

*Precisando che $w = (R(M), w)$ e $f(w) = R(M')$ dove w è l'input dato alla funzione di riduzione e $f(w)$ è l'ouput generato dalla stessa funzione di riduzione, ovvero il valore ottenuto applicando f a w.

Poiché $\mathcal{L}_{\text{Halt}} \leq_m \mathcal{L}_{\text{Emptiness}}$ e $\mathcal{L}_{\text{Emptiness}}$ è decidibile (ipotesi per assurdo), segue che $\mathcal{L}_{\text{Halt}}$ è decidibile. Ma questo è assurdo perché è ben noto che $\mathcal{L}_{\text{Halt}}$ è indecidibile (contraddizione). Poichè ho ottenuto una contraddizione, l'ipotesi che $\mathcal{L}_{\text{Emptiness}}$ sia decidibile è falsa. Pertanto, l'emptiness problem è indecidibile. □

[a] ricorda la funzione di riduzione trasforma un'istanza del problema dell'arresto H in un'istanza dell'emptiness problem E
[b] istanza del problema dell'arresto
[c] istanza del problema dell'emptiness problem

Teorema : Emptiness Problem (versione corta)

Il problema del linguaggio vuoto è indecidibile.
$\mathcal{L}_{\text{Emptiness}} = \{R(M) \mid L(M) = \emptyset\}$ è indecidibile.

Ragionamento. Dim per assurdo + riduzione:

1. Nego la tesi (suppongo per assurdo che il nostro problema sia decidibile)
2. Costruisco una funzione di riduzione da un problema noto indecidibile al nostro: $\mathcal{L}_{\text{Halt}} \leq_m \mathcal{L}_{\text{Emptiness}}$

Concludo che, se $\mathcal{L}_{\text{Emptiness}}$ fosse decidibile, allora potrei usare f e il decisore di $\mathcal{L}_{\text{Emptiness}}$ per costruire un decisore per $\mathcal{L}_{\text{Halt}}$. Ma $\mathcal{L}_{\text{Halt}}$ è noto per essere indecidibile: contraddizione. Pertanto, $\mathcal{L}_{\text{Emptiness}}$ è indecidibile.

Dimostrazione. Siano $\mathcal{L}_{\text{Halt}}$, $\mathcal{L}_{\text{Emptiness}}$ due linguaggi su Σ_H^*, Σ_E^* rispettivamente, dove:

- $\mathcal{L}_{\text{Halt}}$ è il linguaggio del problema dell'arresto
- $\mathcal{L}_{\text{Emptiness}}$ è il linguaggio dell'emptiness problem

1. Ipotesi per assurdo

Suppongo per assurdo che $\mathcal{L}_{\text{Emptiness}}$ sia decidibile. Quindi $\exists$ una MdT E che decide $\mathcal{L}_{\text{Emptiness}}$.

2. Riduzione

Costruisco una funzione di riduzione $f : \Sigma_H^* \to \Sigma_E^*$ tale che dato $(R(M), w)^a$ definisco

$$f((R(M), w)) = (R(M'))^b,$$

dove M' su una stringa di input x generica:

- se $x \neq w$, rifiuta
- se $x = w$, esegue M su w dove:
  - se M termina su w, allora M' accetta w $(x = w)$
  - se M non termina (loop) su w, allora M' non termina

Dalla costruzione di f segue che: $w \in \mathcal{L}_{\text{Halt}} \iff f(w) \notin \mathcal{L}_{\text{Emptiness}}$. Concludo che, se $\mathcal{L}_{\text{Emptiness}}$ fosse decidibile, allora potrei usare f e il decisore di $\mathcal{L}_{\text{Emptiness}}$ per costruire un decisore per $\mathcal{L}_{\text{Halt}}$. Ma $\mathcal{L}_{\text{Halt}}$ è noto per essere indecidibile: contraddizione. Pertanto, $\mathcal{L}_{\text{Emptiness}}$ è indecidibile.

□

3. Costruisco MdT che risolve problema dell'arresto (opzionale)

Adesso, costrusco una MdT N che decide $\mathcal{L}_{\text{Halt}}$:

1. N su input $(R(M), w) \in \Sigma_H^*$ calcola $f((R(M), w))^c \in \Sigma_E^*$ (riduzione)
2. N esegue E su $f((R(M), w))$:
   - se E accetta, N rifiuta
     (se E accetta vuol dire che $L(M') = \emptyset$, ovvero che M' loop perchè M loop su w; dato che M loop su w allora N rifiuta[d])
     $w \notin \mathcal{L}_{\text{Halt}} \iff f(w) \in \mathcal{L}_{\text{Emptiness}}$
   - se E rifiuta, N accetta
     (se E rifiuta vuol dire che $L(M') = \{w\}$, ovvero che M' ha accettato $x = w$ perchè M ha terminato su w; dato che M ha terminato su w allora N accetta[e])
     $w \in \mathcal{L}_{\text{Halt}} \iff f(w) \notin \mathcal{L}_{\text{Emptiness}}$

---

[a] istanza del problema dell'arresto

[b] istanza dell'emptiness problem

[c] $f((R(M), w)) = R(M')$

[d] (perchè il problema dell'arresto, per definizione, è definito in modo tale che se se M loop su w, allora rifiuta)

[e] (perché il problema dell'arresto, per definizione, è definito in modo tale che se M termina su w, allora accetta)

2.4.4 Problema dell'equivalenza dei linguaggi

Teorema : Equivalence Problem

Il problema di determinare se i linguaggi di due MdT coincidono è indecidibile.
$\mathcal{L}_{\text{Equivalence}} = \{(R(M_1), R(M_2)) \mid L(M_1) = L(M_2)\}$ è indecidibile.

Ragionamento. Uso dim per assurdo + riduzione e come conclusione il teorema 3.1. Effettuo una riduzione da un problema noto indecidibile al mio: $\mathcal{L}_{\text{Emptiness}} \leq_m \mathcal{L}_{\text{Equivalence}}$

Dimostrazione.

$$\mathcal{L}_{\text{Equivalence}} = \{(R(M_1), R(M_2)) \mid L(M_1) = L(M_2)\} \qquad \text{(ipotesi)}$$

$$\mathcal{L}_{\text{Equivalence}} \text{ è indecidibile} \qquad \text{(tesi)}$$

Siano $\mathcal{L}_{\text{Emptiness}}$, $\mathcal{L}_{\text{Equivalence}}$ due linguaggi su Σ_E^*, Σ_Q^* rispettivamente, dove:

- $\mathcal{L}_{\text{Emptiness}}$ è il linguaggio dell'emptiness problem
- $\mathcal{L}_{\text{Equivalence}}$ è il linguaggio dell'equivalence problem

1. Ipotesi per assurdo

Suppongo per assurdo che $\mathcal{L}_{\text{Equivalence}}$ sia decidibile. Quindi $\exists$ una MdT Q che decide $\mathcal{L}_{\text{Equivalence}}$ con il seguente comportamento:

$$Q((R(M_1), R(M_2))) = \begin{cases} accept & \text{se } L(M_1) = L(M_2) \\ reject & \text{se } L(M_1) \neq L(M_2) \end{cases}$$

2. Riduzione

Definisco una funzione di riduzione[a] $f : \Sigma_E^* \to \Sigma_Q^*$ come segue.

Dato in input $R(M)$[b] la funzione f genera come output $(R(M), R(M_1))$[c], dove M_1 è la nuova MdT costruita dalla funzione di riduzione f che ha il seguente comportamento ($R(M)$ è la codifica di una MdT arbitraria):

M_1 per ogni stringa di input w: rifiuta; quindi $L(M_1) = \emptyset$

3. Costruisco una MdT che decide l'emptiness problem

Adesso, costruisco una MdT E che decide $\mathcal{L}_{\text{Emptiness}}$:

1. E su input $w \in \Sigma_E^*$ calcola $f(w) \in \Sigma_Q^*$ (riduzione)
2. E esegue Q su $f(w)$:
   - Se Q accetta, allora E accetta
     (se Q accetta vuol dire che i linguaggi delle MdT in input M e M_1 coincidono). $w \in \mathcal{L}_{\text{Emptiness}} \iff f(w) \in \mathcal{L}_{\text{Equivalence}}$
   - Se Q rifiuta, allora E rifiuta
     (se Q rifiuta vuol dire che i linguaggi delle MdT in input M e M_1 sono diversi). $w \notin \mathcal{L}_{\text{Emptiness}} \iff f(w) \notin \mathcal{L}_{\text{Equivalence}}$

*Precisando che $w = R(M)$ e $f(w) = (R(M), R(M_1))$ dove w è l'input dato alla funzione di riduzione e $f(w)$ è l'ouput generato dalla stessa funzione di riduzione, ovvero il valore ottenuto applicando f a w.

Poiché $\mathcal{L}_{\text{Emptiness}} \leq_m \mathcal{L}_{\text{Equivalence}}$ e $\mathcal{L}_{\text{Equivalence}}$ è decidibile (ipotesi per assurdo), segue che $\mathcal{L}_{\text{Emptiness}}$ è decidibile. Ma questo è assurdo perché è ben noto che $\mathcal{L}_{\text{Emptiness}}$ è indecidibile (contraddizione). Poichè ho ottenuto una contraddizione, l'ipotesi che $\mathcal{L}_{\text{Equivalence}}$ sia decidibile è falsa. Pertanto, l'equivalence problem è indecidibile. □

[a] ricorda la funzione di riduzione trasforma un'istanza del problema dell'emptiness problem E in un'istanza dell'equivalence problem Q
[b] istanza dell'emptiness problem
[c] istanza dell'equivalence problem

2.4.5 Problema della terminazione totale

Teorema : Total Halting Problem

Il problema della terminazione totale è indecidibile.
$\mathcal{L}_{\text{Total-Halt}} = \{R(M) \mid M \text{ termina su ogni input } w\}$ è indecidibile.

Ragionamento. Dim per assurdo + riduzione:

1. Nego la tesi (suppongo per assurdo che il nostro problema sia decidibile)
2. Costruisco una funzione di riduzione da un problema noto indecidibile al nostro: $\mathcal{L}_{\text{Halt}} \leq_m \mathcal{L}_{\text{Total-Halt}}$

Concludo che, se $\mathcal{L}_{\text{Total-Halt}}$ fosse decidibile, allora potrei usare f e il decisore di $\mathcal{L}_{\text{Total-Halt}}$ per costruire un decisore per $\mathcal{L}_{\text{Halt}}$. Ma $\mathcal{L}_{\text{Halt}}$ è noto per essere indecidibile: contraddizione. Pertanto, $\mathcal{L}_{\text{Total-Halt}}$ è indecidibile.

Dimostrazione.

$$\mathcal{L}_{\text{Total-Halt}} = \{R(M) \mid M \text{ termina su ogni input } w\} \qquad \text{(ipotesi)}$$

$$\mathcal{L}_{\text{Total-Halt}} \text{ è indecidibile} \qquad \text{(tesi)}$$

Siano $\mathcal{L}_{\text{Total-Halt}}$, $\mathcal{L}_{\text{Halt}}$ due linguaggi su Σ_T^*, Σ_H^* rispettivamente, dove:

- $\mathcal{L}_{\text{Total-Halt}}$ è il linguaggio del total halting problem
- $\mathcal{L}_{\text{Halt}}$ è il linguaggio del problema dell'arresto

1. Ipotesi per assurdo
Suppongo per assurdo che $\mathcal{L}_{\text{Total-Halt}}$ sia decidibile. Quindi $\exists$ una MdT T che decide $\mathcal{L}_{\text{Total-Halt}}$.
2. Riduzione
Costruisco una funzione di riduzione $f : \Sigma_H^* \to \Sigma_T^*$ tale che dato $(R(M), w)^a$ definisco

$$f((R(M), w)) = (R(M'))^b,$$

dove M' su una stringa di input x generica:

- se $x \neq w$, rifiuta
- se $x = w$, esegue M su w dove:
  - se M termina su w, allora M' accetta o termina? w $(x = w)$
  - se M non termina (loop) su w, allora M' non termina

Dalla costruzione di f segue che: $w \in \mathcal{L}_{\text{Halt}} \iff f(w) \notin \mathcal{L}_{\text{Emptiness}}$. Concludo che, se $\mathcal{L}_{\text{Emptiness}}$ fosse decidibile, allora potrei usare f e il decisore di $\mathcal{L}_{\text{Emptiness}}$ per costruire un decisore per $\mathcal{L}_{\text{Halt}}$. Ma $\mathcal{L}_{\text{Halt}}$ è noto per essere indecidibile: contraddizione. Pertanto, $\mathcal{L}_{\text{Emptiness}}$ è indecidibile.

□

3. Costruisco MdT che risolve problema dell'arresto (opzionale)
Adesso, costrusco una MdT N che decide $\mathcal{L}_{\text{Halt}}$:

1. N su input $(R(M), w) \in \Sigma_H^*$ calcola $f((R(M), w))^c \in \Sigma_T^*$ (riduzione)
2. N esegue E su $f((R(M), w))$:
   - se T accetta, N accetta
     (se T accetta vuol dire che che M' ha accettato o terminato? su w perchè M ha terminato su w; dato che M ha terminato su w allora N accetta[d])
     $w \notin \mathcal{L}_{\text{Halt}} \iff f(w) \in \mathcal{L}_{\text{Emptiness}}$
   - se T rifiuta, N accetta
     (se E rifiuta vuol dire che M' loop su $x = w$ perchè M loop su w; dato che M ha terminato su w allora N accetta[e])
     $w \in \mathcal{L}_{\text{Halt}} \iff f(w) \notin \mathcal{L}_{\text{Emptiness}}$

---

[a] istanza del problema dell'arresto
[b] istanza dell'total halting problem
[c] $f((R(M), w)) = R(M')$
[d] (perchè il problema dell'arresto, per definizione, è definito in modo tale che se se M loop su w, allora rifiuta)
[e] (perché il problema dell'arresto, per definizione, è definito in modo tale che se M termina su w, allora accetta)

2.5 Proprietà non banali delle MdT

2.5.1 Teorema di Rice

2.5.2 Problema del linguaggio vuoto

2.5.3 Problema della finitezza del linguaggio

2.5.4 Problema della terminazione di programmi su tutti gli input

3 Riducibilità

Ricorda la notazione f: input $\to$ output

Definizione 3.1 (Funzione di riduzione)**.** Sia L_1,L_2 due linguaggi su Σ_1^*, Σ_2^*, rispettivamente. Si dice che L_1 è riducibile a L_2, e si scrive $L_1 \leq_m L_2$ se $\exists$ una funzione computabile totale $f : \Sigma_1^* \to \Sigma_2^*$ chiamata **funzione di riduzione** t.c. $\forall w \in \Sigma_1^*$

$$w \in L_1 \iff f(w) \in L_2$$

Spiegazione informale:
Per effetturare una riduzione da L_1 a L_2, deve esistere una MdT R (macchina di riduzione) che prende in input una qualunque stringa $w \in \Sigma_1^*$ e la trasforma in una stringa $f(w) \in \Sigma_2^*$. Ovvero, se la MdT R prende in input un'istanza di L_1 allora produce come output un'istanza di L_2.
La MdT R calcola la funzione di riduzione f.
In questo modo, se avessimo una MdT che decide L_2, potremmo decidere L_1 applicando f.

Teorema : 3.1

Se $A \leq_m B$ e B è decidibile $\implies$ A è decidibile.

Dimostrazione 3.1

Se $A \leq_m B$ e B è decidibile $\implies$ A è decidibile.

Ragionamento. A,B sono due linguaggi su Σ_A^*,Σ_B^* rispettivamente.
Per ipotesi:

- $A \leq_m B$, quindi $\exists$ una funzione di riduzione (totale e calcolabile da una MdT) da A a B.
- B è decidibile, quindi $\exists$ una MdT che decide B.

Dimostro che A è decidibile costruendo una MdT che decide A.

Dimostrazione.

$$A \leq_m B \ \wedge \ B \text{ è decidibile} \qquad \text{(ipotesi)}$$

$$A \text{ è decidibile} \qquad \text{(tesi)}$$

Per ipotesi, posso definire M la MdT di decisione per B e $f : \Sigma_A^* \to \Sigma_B^*$ la funzione di riduzione da A a B (calcolabile dalla MdT R).
Costruisco la MdT N che decide A:

1. N su input $w \in \Sigma_A^*$, calcola $f(w) \in \Sigma_B^*$
   (N esegue R su w che effettua la riduzione producendo come output $f(w)$)
2. N esegue M su $f(w)$:
   - Se M accetta $f(w)$, allora N accetta w.
     $f(w) \in B \iff w \in A$
   - Se M rifiuta $f(w)$, allora N rifiuta w.
     $f(w) \notin B \iff w \notin A$

Conclusione: Ho costruito un algoritmo di decisione per A che applica la riduzione per ogni input e sfrutta la MdT che decide B per A. Inoltre, la MdT N si arresta sempre per ogni input w, perchè f è una funzione totale computabile (la funzione di riduzione) e M è un algoritmo di decisione (MdT che si ferma per ogni input) per B.
Pertanto, A è decidibile.

□

*Notare che scrivo "accetta" o "rifiuta" e non "termina in uno stato accettante" o "termina in uno stato di rifiuto" perchè ho indicato "costruisco la MdT N che decide A" e una MdT che decide un linguaggio si ferma per ogni input, quindi sarebbe ridondante scriverlo.

Spiegazione extra della dimostrazione:
Ovvero, la macchina N ha due MdT al suo interno (prima esegue R e poi M):

- MdT R che effettua la riduzione da A a B:
  1. R riceve in input w (istanza di A)
  2. R effettua la riduzione; calcola la funzione di riduzione su w (applica f a w che scrivo come $f(w)$)
  3. R produce come output $f(w)$ (istanza di B)
- MdT M che decide B:
  1. M riceve in input $f(w)$
  2. Se la computazione di M termina:
     - in uno stato accettante, allora $f(w) \in B$
     - in uno stato di rifiuto, allora $f(w) \notin B$

Quindi, N termina accettando w se e solo se M termina accettando $f(w)$. Oppure, N termina rifiutando w se e solo se M termina rifiutando $f(w)$. Come ben sappiamo, M decide solo istanze di B, per questo è necessario trasformare un'istanza di A in una di B. La riduzione serve perchè M non può ricevere un'istanza di A ma solo di B (perchè per ipotesi B è decidibile e M è la macchina di decisione per B), quindi è necessario che "qualcuno" effettui la trasformazione, che è proprio quello che fa la MdT R.
Potendo trasformare un'istanza di A in B e sapendo che B è decidibile allora posso "decidere" A.

Teorema : 3.2

Se $A \leq_m B$ e A è indecidibile $\implies$ B è indecidibile.

Dimostrazione 3.2

Se $A \leq_m B$ e A è indecidibile $\implies$ B è indecidibile.

Ragionamento. Per dimostarlo, suppongo per assurdo che B sia decidibile (nego la tesi, ipotesi per assurdo) e poi applico la riduzione e poi le definizioni che mi porteranno ad una contraddizione che rende falsa la mia ipotesi per assurdo rendendo poi vero il teorema.

Dimostrazione.

$$A \leq_m B \ \wedge \ A \text{ è indecidibile} \qquad \text{(ipotesi)}$$

$$B \text{ è indecidibile} \qquad \text{(tesi)}$$

Suppongo per assurdo che B sia decidibile (ipotesi per assurdo).
Quindi per ipotesi $\exists$ una MdT M che decide B. Inoltre, per ipotesi (del teorema), $\exists$ una

funzione di riduzione $f : \Sigma_A^* \to \Sigma_B^*$ t.c. $\forall w \in \Sigma_A^*$:

$$w \in A \iff f(w) \in B$$

Costruisco una MdT N che decide A:

1. N su input $w \in \Sigma_A^*$ calcola $f(w) \in \Sigma_B^*$
2. N esegue M su $f(w)$:
   - Se M accetta, allora N accetta
     (Se $f(w) \in B$, allora $w \in A$)
   - Se M rifiuta, allora N rifiuta
     (Se $f(w) \notin B$, allora $w \notin A$)

Ho costruito un algoritmo di decisione per A. Ma quindi se $A \leq_m B$ e B è decidibile, allora per definizione (teorema 3.1) A è decidibile. Ma questo è assurdo (contraddizione) perchè A è indecidibile (secondo l'ipotesi del teorema). Poichè abbiamo ottenuto una contraddizione, l'ipotesi che B sia decidibile è falsa. Pertanto, B è indecidibile. □

Teorema : 3.3

Se $A \leq_m B$ e B è semidecidibile $\implies$ A è semidecidibile.

Teorema : 3.4

Se $A \leq_m B$ e A non è semidecidibile $\implies$ B non è semidecidibile.

3.0.1 Riduzione non sempre funziona

4 Complessità Temporale

4.1 P

4.1.1 Famosi problemi

4.2 NP

4.2.1 NP-Difficile

4.2.2 NP-Completo

4.2.3 Famosi problemi

5 Complessità Spaziale

¡¡¡¡¡¡¡ HEAD eccomiiii:Dfffhhhhhh ======= ¿¿¿¿¿¿¿ 51a7b4f4411c15c836fb6fda8d75410b24dc3762