DECIDIBILITÀ

RIDUCIBILITÀ MEDIANTE FUNZIONE
Capitolo 5, sezione 5.3 con integrazioni.

Riducibilità: descrizione informale

È importante riuscire a riconoscere che un problema P è indecidibile.

Come? Tre possibilità:

- Supporre l'esistenza di una TM che decide il linguaggio associato a P e provare che questo conduce a una contraddizione.
- Considerare un problema P' di cui sia nota l'indecidibilità del linguaggio associato e dimostrare che P' "non è più difficile" del problema in questione P.
- Teorema di Rice.

Esempio $\Sigma = \{0, 1\}$.

$$EVEN = \{w \in \Sigma^* \mid w \text{ è la rappresentazione binaria di } n \in \mathbb{N} \text{ pari}\}$$

$$ODD = \{w \in \Sigma^* \mid w \text{ è la rappresentazione binaria di } n \in \mathbb{N} \text{ dispari}\}$$

Sia $w \in \Sigma^*$ e sia n il corrispondente decimale di w. È facile costruire la TM *INCR*:

$$w \rightarrow \boxed{INCR} \rightarrow w' \quad (= \text{ rappresentazione binaria di } n+1)$$

- $1000 + 1 = 1001$
  $1001 + 1 = 1010$
  $1011 + 1 = 1100$
- $1111 + 1 = 10000$

- *INCR* = "Sulla stringa di input w:
  1. Trasforma w in $\$w$.
  2. Nello stato q_r scorre l'input da sinistra a destra fino a incontrare il simbolo $\sqcup$.
  3. Passa nello stato q_ℓ, si sposta a sinistra cambiando ogni carattere 1 che vede in 0 fino a leggere un carattere diverso da 1 (Nota: questo può accadere anche senza incontrare alcun 1).
  4. Se questo carattere è 0 lo cambia in 1, si sposta a sinistra fino a leggere \$. Poi elimina \$ e sposta a sinistra di una casella la stringa di caratteri 0 e 1 sul nastro e si ferma.
  5. Se questo carattere diverso da 1 è \$, l'input era una stringa di caratteri uguali a 1. Allora cambia \$ in 1 e si ferma su questo carattere."

- Abbiamo definito una trasformazione (funzione calcolabile)

$$f : w \in \Sigma^* \rightarrow w' \in \Sigma^*$$

tale che

$$w = \langle n \rangle \in EVEN \Leftrightarrow f(w) = w' = \langle n+1 \rangle \in ODD$$

- f è una riduzione di $EVEN$ a ODD.
- Nota: f è definita su tutto Σ^*, non solo su $EVEN$ o ODD.

- *EVEN* "non è più difficile" di *ODD*: se esiste una TM R che decide *ODD*, la TM S decide *EVEN*.

$$S : w \rightarrow \boxed{INCR} \rightarrow w' \rightarrow \boxed{R}$$

- Viceversa se *EVEN* è indecidibile proviamo così che anche *ODD* lo è: se per assurdo esistesse una TM R che decide *ODD*, la TM S deciderebbe *EVEN*.

Riducibilità: descrizione informale

- Idea: convertire le istanze di un problema P nelle istanze di un problema P' in modo che un algoritmo per P', se esiste, possa essere utilizzato per progettare un algoritmo per P: P non è più difficile di P'.
- Sia A il linguaggio associato a P, sia B il linguaggio associato a P'. Allora proveremo che:
  B decidibile $\Rightarrow$ A decidibile,
  A indecidibile $\Rightarrow$ B indecidibile.
- **Nota**: nulla è detto sulla decidibilità di A o B ma solo sulla decidibilità di A assumendo di disporre di un algoritmo per decidere di B.

Definizione

Una proposizione è una frase che dichiara qualcosa e che può essere vera o falsa ma non può essere entrambe.

Siano p, q due proposizioni. Il contronominale di

$$p \to q$$

è

$$\neg q \to \neg p$$

Il contronominale di $p \to q$ è logicamente equivalente a $p \to q$

$$(p \leftrightarrow q) \equiv (p \to q) \wedge (q \to p)$$

$$(p \leftrightarrow q) \equiv (p \to q) \wedge (q \to p)$$

$$\begin{aligned} p \leftrightarrow q &\equiv (p \to q) \wedge (q \to p) \\ &\equiv (\neg q \to \neg p) \wedge (\neg p \to \neg q) \\ &\equiv (\neg p \to \neg q) \wedge (\neg q \to \neg p) \\ &\equiv \neg p \leftrightarrow \neg q \end{aligned}$$

Abbiamo provato che le proposizioni composte $p \leftrightarrow q$ e $\neg p \leftrightarrow \neg q$ sono logicamente equivalenti.

Definizione

Una funzione $f : \Sigma^* \to \Sigma^*$ è calcolabile se esiste una TM M tale che su ogni input w, M si arresta con $f(w)$, e solo con $f(w)$, sul suo nastro.

Definizione

Un linguaggio $A \subseteq \Sigma^*$ *è riducibile mediante funzione a un linguaggio* $B \subseteq \Sigma^*$ ($A \leq_m B$) *se esiste una funzione calcolabile* $f : \Sigma^* \to \Sigma^*$ *tale che*

$$\forall w \in \Sigma^* \quad w \in A \Leftrightarrow f(w) \in B$$

La funzione f *è chiamata una riduzione da* A *a* B*.*

Esempio. $EVEN \leq_m ODD$. Una riduzione è la funzione f da $\{0,1\}^*$ in $\{0,1\}^*$ tale che $f(\langle n \rangle) = \langle n+1 \rangle$.

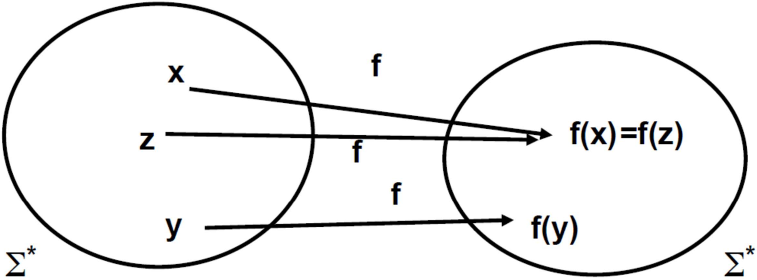

Immagine tratta dalle dispense della Prof.ssa Emanuela Fachini

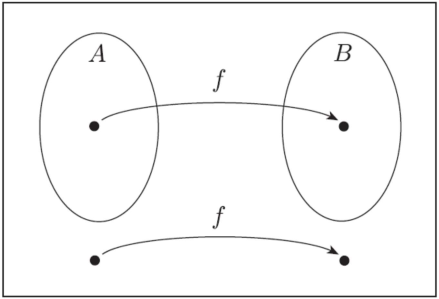

Figura tratta da M. Sipser, Introduzione alla teoria della computazione.

Riducibilità mediante funzione

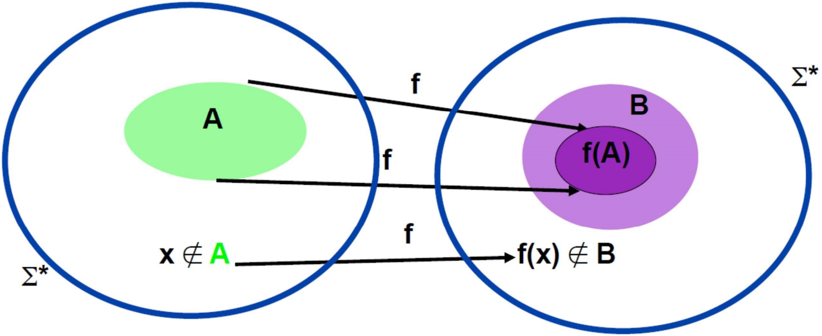

Immagine tratta dalle dispense della Prof.ssa Emanuela Fachini

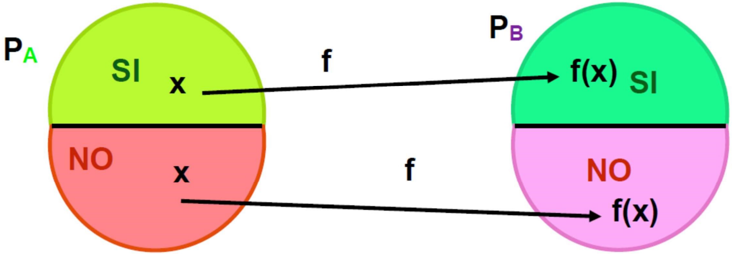

Immagine tratta dalle dispense della Prof.ssa Emanuela Fachini

Una riduzione fornisce un modo per convertire questioni riguardanti l'appartenenza o meno di stringhe ad A in questioni riguardanti l'appartenenza o meno di stringhe a B.

Intuitivamente se un linguaggio A è riducibile a B allora il problema associato ad A "non è più difficile" di quello associato a B.

Teorema 1
$A \leq_m B$ se e solo se $\overline{A} \leq_m \overline{B}$

Dimostrazione
Per ipotesi $A \leq_m B$, quindi esiste una riduzione di A a B.

Poiché f è una riduzione, f è calcolabile e inoltre

$$\forall w \in \Sigma^* \quad w \in A \Leftrightarrow f(w) \in B$$

Proviamo che f è anche una riduzione da $\overline{A}$ a $\overline{B}$.

Infatti, poiché f è una riduzione, f è calcolabile e inoltre

$$\forall w \in \Sigma^* \quad w \in A \Leftrightarrow f(w) \in B$$

Quindi

$$\forall w \in \Sigma^* \quad w \notin A \Leftrightarrow f(w) \notin B$$

Cioè

$$\forall w \in \Sigma^* \quad w \in \overline{A} \Leftrightarrow f(w) \in \overline{B}$$

Quindi, per definizione, f è una riduzione da $\overline{A}$ a $\overline{B}$.

$\square$

Teorema 5.22

Se $A \leq_m B$ e B è decidibile, allora A è decidibile.

Dimostrazione

Sia M_B una macchina di Turing che decide B, sia f una riduzione da A a B e sia M_f una macchina di Turing che calcola f.

Consideriamo la macchina di Turing M_A
$M_A =$ "Sull'input w:

1. simula M_f e calcola $f(w)$
2. simula M_B su $f(w)$
3. se M_B accetta $f(w)$, accetta; se M_B rifiuta $f(w)$, rifiuta."

$$M_A : w \rightarrow \boxed{\boxed{M_f} \rightarrow f(w) \rightarrow \boxed{M_B}}$$

M_A decide A.

M_A è un decider. Infatti M_A si ferma su w se si fermano M_f ed M_B. Ora, per ogni w, M_f si ferma con $f(w)$ sul nastro e per ogni w, M_B si ferma su $f(w)$ perché M_B è un decider.

Inoltre M_A riconosce A. Infatti

$$\begin{aligned} w \in L(M_A) &\Leftrightarrow f(w) \in L(M_B) \text{ (per la definizione di } M_A) \\ &\Leftrightarrow f(w) \in B \text{ (perché } M_B \text{ decide } B) \\ &\Leftrightarrow w \in A \text{ (per definizione di riduzione)} \end{aligned}$$

$\square$

Teorema 5.28
Se $A \leq_m B$ e B è Turing riconoscibile, allora A è Turing riconoscibile.

Dimostrazione
Sia M_B una macchina di Turing che riconosce B, sia f una riduzione da A a B e sia M_f una macchina di Turing che calcola f.

Consideriamo la macchina di Turing M_A
$M_A =$ "Sull'input w:

1. simula M_f e calcola $f(w)$
2. simula M_B su $f(w)$
3. se M_B accetta $f(w)$, accetta; se M_B rifiuta $f(w)$, rifiuta.

(Ovviamente se M_B cicla su $f(w)$ anche M_A cicla.)

M_A riconosce A. Infatti

$$
\begin{aligned}
w \in L(M_A) \quad &\Leftrightarrow \quad f(w) \in L(M_B) \text{ (per la definizione di } M_A) \\
&\Leftrightarrow \quad f(w) \in B \text{ (perché } M_B \text{ riconosce } B) \\
&\Leftrightarrow \quad w \in A \text{ (per definizione di riduzione)}
\end{aligned}
$$

$\square$

Corollario
Se $A \leq_m B$ e A è indecidibile, allora B è indecidibile.

Dimostrazione: Se B fosse decidibile lo sarebbe anche A in virtù del Teorema 5.22.

(Ancora il contronominale!)

Corollario

Se $A \leq_m B$ e A non è Turing riconoscibile, allora B non è Turing riconoscibile.

Dimostrazione: Se B fosse Turing riconoscibile lo sarebbe anche A in virtù del Teorema 5.28.

Definizione

Una funzione $f : \Sigma^* \rightarrow \Sigma^*$ *è calcolabile se esiste una TM* M *tale che su ogni input* w*,* M *si arresta con* $f(w)$*, e solo con* $f(w)$*, sul suo nastro.*

- Nota: questa definizione sottolinea la differenza tra definire una funzione f, cioè definire i valori di f e calcolare tali valori di f.

Le seguenti funzioni aritmetiche sono calcolabili (dove $n, m \in \mathbb{N}$):

- $incr(n) = n + 1$
- $dec(n) = \begin{cases} n - 1 & \text{se } n > 0; \\ 0 & \text{se } n = 0 \end{cases}$
- $(m, n) \to m + n$;
- $(m, n) \to m - n$;
- $(m, n) \to m \cdot n$

Le funzioni possono essere anche trasformazioni di codifiche di macchine di Turing.

Stabilire se funzioni di questo tipo sono calcolabili può non essere facile, soprattutto se le macchine di Turing sono descritte ad alto livello.

Esempio. Data una macchina di Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, in questo esempio denotiamo con M' la macchina di Turing che accetta le stringhe rifiutate da M e rifiuta quelle accettate (in generale M' **NON** riconosce il complemento di $L(M)$).

Consideriamo la funzione $f : \Sigma^* \to \Sigma^*$ così definita.

$$f(y) = \begin{cases} \epsilon & \text{se } y \neq \langle M \rangle,\ M \text{ TM;} \\ \langle M' \rangle & \text{se } y = \langle M \rangle \end{cases}$$

Una tale funzione è calcolabile?

Consideriamo la MT F che, sull'input y:

1. Se $y \neq \langle M \rangle$, restituisce ϵ
2. Se $y = \langle M \rangle$, "costruisce" la MT M' così definita
   $M' =$ sull'input x
   1. "simula" M su x
   2. se M accetta, rifiuta
   3. se M rifiuta, accetta
3. Fornisce in output $\langle M' \rangle$

Ma F come "costruisce"?

Ed M' come "simula"?

F inizia la sua computazione scorrendo l'input per verificare se è "legale", cioè se è la codifica di una macchina di Turing.

Se non lo è cancella l'input e si ferma.

Se l'input è la codifica di una macchina di Turing M, F scorre nuovamente l'input ed effettua le seguenti operazioni.

Chiamiamo δ la funzione di transizione di M e δ' quella della macchina M' che sarà costruita.

F cerca nella codifica di M la codifica delle transizioni della forma $\delta(q, a) = (q_{accept}, a', D)$, $D \in \{L, R\}$, $a, a' \in \Gamma$, $q \in Q$ e cambia ognuna di esse con la codifica di $\delta'(q, a) = (q_{reject}, a', D)$, $D \in \{L, R\}$, $a, a' \in \Gamma$, $q \in Q$.

Fa un'azione analoga sulla codifica delle transizioni della forma $\delta(q, a) = (q_{reject}, a', D)$, che cambia con la codifica di $\delta'(q, a) = (q_{accept}, a', D)$.

Lascia le altre transizioni inalterate e si ferma.

Esiste una tale MT F?

Si perché F deve solo scorrere l'input, verificare se è "legale" e poi cambiare dei caratteri in esso contenuti!

La stringa in output di F è la codifica $\langle M' \rangle$ della macchina di Turing M' che "simula" M e accetta se M rifiuta mentre rifiuta se M accetta.

Ma non avviene alcuna esecuzione o simulazione di M!

Teorema 5.22

Se $A \leq_m B$ e B è decidibile, allora A è decidibile.

Funzioni non calcolabili - Esempio

Consideriamo A_{TM} e $B = \{ab\}$.

Consideriamo la funzione $f : \Sigma^* \to \Sigma^*$, dove $a, b \in \Sigma$, così definita.

$$f(y) = \begin{cases} ab & \text{se } y = \langle M, w \rangle \in A_{TM}; \\ a & \text{altrimenti} \end{cases}$$

Quindi f è una funzione tale che $f(y) = a$ se y non è della forma $\langle M, w\rangle$, oppure se $y = \langle M, w\rangle$ con $\langle M, w\rangle \notin A_{TM}$.

Invece $f(y) = ab$ se $y = \langle M, w\rangle$ con $\langle M, w\rangle \in A_{TM}$.

Quindi per ogni $y \in \Sigma^*$,

$$y \in A_{TM} \Leftrightarrow f(y) \in \{ab\}$$

Provare che f non è calcolabile.

Teorema 5.22

Se $A \leq_m B$ e B è decidibile, allora A è decidibile.

Nota. Se $A \leq_m B$ e A è decidibile non possiamo dedurre nulla su B.

Ad esempio consideriamo $A = \{ab\}$ e A_{TM}.

$A = \{ab\}$ è decidibile e A_{TM} è indecidibile.

Ma possiamo provare che $\{ab\} \leq_m A_{TM}$.

Mostriamo che $\{ab\} \leq_m A_{TM}$.

Sia M la macchina di Turing tale che $L(M) = L(a^*)$.

Consideriamo la funzione $f : \Sigma^* \to \Sigma^*$, dove $a, b \in \Sigma$, così definita.

$$f(y) = \begin{cases} \langle M, a \rangle & \text{se } y = ab; \\ \langle M, b \rangle & \text{altrimenti} \end{cases}$$

Ovviamente $\langle M, a \rangle \in A_{TM}$ mentre $\langle M, b \rangle \notin A_{TM}$.

Vogliamo dimostrare che f è una riduzione da $\{ab\}$ ad A_{TM}.

Osservazione

$f : \Sigma^* \to \Sigma^*$, dove $a, b \in \Sigma$.

$$f(y) = \begin{cases} \langle M, a \rangle & \text{se } y = ab; \\ \langle M, b \rangle & \text{altrimenti} \end{cases}$$

La funzione f è calcolabile. La MdT F che calcola f sull'input y:

- se $y = ab$ cancella ab, scrive la stringa $\langle M, a \rangle$ e si ferma;
- se $y \neq ab$ cancella y, scrive la stringa $\langle M, b \rangle$ e si ferma.

Inoltre

$$y \in \{ab\} \Rightarrow y = ab \Rightarrow f(y) = \langle M, a \rangle \in A_{TM}$$

$$y \notin \{ab\} \Rightarrow y \neq ab \Rightarrow f(y) = \langle M, b \rangle \notin A_{TM}$$

Quindi, per ogni stringa y,

$$y \in \{ab\} \Leftrightarrow f(y) \in A_{TM}$$

In conclusione $\{ab\} \leq_m A_{TM}$.

Nei teoremi seguenti proveremo l'esistenza di riduzioni da A_{TM} (o da un altro linguaggio indecidibile) ad alcuni linguaggi B associati a problemi di decisione sulle macchine di Turing.

Una conseguenza importante di tali teoremi è che ognuno di questi linguaggi B è indecidibile.

Inoltre, quando descriviamo una macchina di Turing che calcola una riduzione da A a B, assumiamo che agli input che non sono "della forma corretta" sia associata una stringa al di fuori di B.

Più precisamente, se A è il linguaggio associato a un problema di decisione, definiamo la riduzione solo sulle codifiche delle istanze del problema (**Questo non significa "sugli elementi di A"**).

Ad esempio se $A = A_{TM}$, definiremo la riduzione (e la macchina di Turing che calcola la riduzione) sulle stringhe della forma $\langle M, w \rangle$, dove M è una TM e w è una stringa. (**questo non significa "sugli elementi di A_{TM}"**).

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ è una TM e } w \in L(M)\}$$

$$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ è una TM e } M \text{ si arresta su } w\}$$

Teorema

$A_{TM} \leq_m HALT_{TM}$

Dimostrazione

Per provare che $A_{TM} \leq_m HALT_{TM}$ dobbiamo definire una funzione calcolabile $f : \Sigma^* \to \Sigma^*$ tale che, per ogni stringa $\langle M, w\rangle$, con M macchina di Turing e w stringa,

$$\langle M, w\rangle \in A_{TM} \Leftrightarrow \langle M', w\rangle \in HALT_{TM}$$

$A_{TM} \leq_m HALT_{TM}$

Consideriamo la MT F che, sull'input $\langle M, w\rangle$:

1. Costruisce la macchina M',
   $M' =$ "sull'input x
   1. simula M su x
   2. se M accetta, accetta
   3. se M rifiuta, cicla"
2. Fornisce in output $\langle M', w\rangle$

Nota: la macchina M' si ferma su un input x se e solo se M accetta x.

La funzione f calcolata da F, che associa a $\langle M, w \rangle$ la stringa $\langle M', w \rangle$, è una riduzione da A_{TM} a $HALT_{TM}$.

Infatti, f è calcolabile (cioè è possibile definire una macchina di Turing F che ha il comportamento input/output descritto prima).

Inoltre

$$\langle M, w \rangle \in A_{TM} \Leftrightarrow M \text{ accetta } w \Leftrightarrow M' \text{ si arresta su } w$$

$$\Leftrightarrow \langle M', w \rangle \in HALT_{TM}.$$

$\square$

Teorema

$$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ è una } TM \text{ e } M \text{ si arresta su } w\}$$

è indecidibile.

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ è una TM e } w \in L(M)\}$$

$$E_{TM} = \{\langle M\rangle \mid M \text{ è una } TM \text{ e } L(M) = \emptyset\}$$

Proviamo che $A_{TM} \leq_m \overline{E_{TM}}$.

Data una MT M e una stringa w, sia M_1 la macchina di Turing tale che, su un input x:

1. Se $x \neq w$ allora M_1 si ferma e rifiuta x.
2. Se $x = w$ allora M_1 simula M su w e accetta x se M accetta $x = w$.

Quindi

$$L(M_1) = \begin{cases} \{w\} & \text{se } \langle M, w \rangle \in A_{TM}; \\ \emptyset & \text{altrimenti} \end{cases}$$

La funzione f, che associa a $\langle M, w \rangle$ la stringa $\langle M_1 \rangle$, è una riduzione da A_{TM} a $\overline{E_{TM}}$.

Infatti, f è calcolabile: possiamo costruire una macchina di Turing F che sull'input $\langle M, w \rangle$, fornisce in output $\langle M_1 \rangle$.

Inoltre

$$\langle M, w \rangle \in A_{TM} \Rightarrow M \text{ accetta } w \Rightarrow L(M_1) \neq \emptyset \Rightarrow \langle M_1 \rangle \in \overline{E_{TM}}$$

e

$$\langle M, w \rangle \notin A_{TM} \Rightarrow M \text{ non accetta } w \Rightarrow L(M_1) = \emptyset$$
$$\Rightarrow \langle M_1 \rangle \in E_{TM} \Rightarrow \langle M_1 \rangle \notin \overline{E_{TM}}$$

Quindi

$$\langle M, w \rangle \in A_{TM} \Leftrightarrow M \text{ accetta } w \Leftrightarrow L(M_1) \neq \emptyset \Leftrightarrow \langle M_1 \rangle \in \overline{E_{TM}}$$

$\square$

Teorema 5.27

$\overline{E_{TM}}$ è indecidibile.

Corollario

E_{TM} è indecidibile.

Prova. La classe dei linguaggi decidibili è chiusa rispetto al complemento. Se E_{TM} fosse decidibile lo sarebbe anche $\overline{E_{TM}}$ e questo è in contraddizione con il Teorema 5.27.

$\square$

NOTA: **non esiste nessuna riduzione mediante funzione da A_{TM} a E_{TM}.**

$$REGULAR_{TM} = \{\langle M \rangle \mid M \text{ è una } TM \text{ e } L(M) \text{ è regolare}\}$$

Proviamo che $A_{TM} \leq_m REGULAR_{TM}$.

Data una MT M e una stringa w, sia R la macchina di Turing tale che, su un input x:

1. Se $x \in \{0^n 1^n \mid n \in \mathbb{N}\}$, allora R si ferma e accetta x.
2. Se $x \notin \{0^n 1^n \mid n \in \mathbb{N}\}$ allora R simula M su w e accetta x se M accetta w.

Quindi

$$L(R) = \begin{cases} \Sigma^* & \text{se } \langle M, w \rangle \in A_{TM}; \\ \{0^n 1^n \mid n \in \mathbb{N}\} & \text{altrimenti} \end{cases}$$

$A_{TM} \leq_m REGULAR_{TM}$

La funzione f, che associa a $\langle M, w \rangle$ la stringa $\langle R \rangle$, è una riduzione da A_{TM} a $REGULAR_{TM}$.

Infatti, f è calcolabile: possiamo costruire una macchina di Turing F che sull'input $\langle M, w \rangle$, fornisce in output $\langle R \rangle$.

Inoltre

$$\langle M, w\rangle \in A_{TM} \Rightarrow M \text{ accetta } w$$

$$\Rightarrow L(R) = \Sigma^* \text{ è regolare } \Rightarrow \langle R\rangle \in REGULAR_{TM}.$$

$$\langle M, w\rangle \notin A_{TM} \Rightarrow M \text{ non accetta } w \Rightarrow$$

$$L(R) = \{0^n 1^n \mid n \in \mathbb{N}\} \text{ non è regolare } \Rightarrow \langle R\rangle \notin REGULAR_{TM}.$$

In conclusione

$$\langle M, w\rangle \in A_{TM} \Leftrightarrow M \text{ accetta } w$$

$$\Leftrightarrow L(R) \text{ è regolare } \Leftrightarrow \langle R\rangle \in REGULAR_{TM}.$$

$\square$

Teorema

$$REGULAR_{TM} = \{\langle M \rangle \mid M \text{ è una } TM \text{ e } L(M) \text{ è regolare}\}$$

è indecidibile.

$E_{TM} \leq_m EQ_{TM}$

$$E_{TM} = \{\langle M \rangle \mid M \text{ è una } TM \text{ e } L(M) = \emptyset\}$$

$$EQ_{TM} = \{\langle M_1, M_2 \rangle \mid M_1, M_2 \text{ sono TM e } L(M_1) = L(M_2)\}$$

Proviamo che $E_{TM} \leq_m EQ_{TM}$.

Sia M_1 una macchina di Turing tale che $L(M_1) = \emptyset$.

Quindi, data una macchina di Turing M, avremo $L(M) = L(M_1)$ se e solo se $L(M) = \emptyset$.

La funzione f, che associa a $\langle M \rangle$ la stringa $\langle M, M_1 \rangle$, è una riduzione da E_{TM} a EQ_{TM}.

Infatti, f è calcolabile: possiamo costruire una macchina di Turing F che sull'input $\langle M \rangle$, fornisce in output $\langle M, M_1 \rangle$.

Inoltre

$$\langle M \rangle \in E_{TM} \Leftrightarrow L(M) = \emptyset \Leftrightarrow L(M) = L(M_1)$$

$$\Leftrightarrow \langle M, M_1 \rangle \in EQ_{TM}.$$

$\square$

Teorema

$$EQ_{TM} = \{\langle M_1, M_2 \rangle \mid M_1, M_2 \textit{ sono TM e } L(M_1) = L(M_2)\}$$

è indecidibile.

Un linguaggio che non è né Turing riconoscibile né co-Turing riconoscibile

Abbiamo provato che A_{TM} è Turing riconoscibile ma non decidibile.

Abbiamo provato che $\overline{A_{TM}}$ non è Turing riconoscibile ma è co-Turing riconoscibile (perché il suo complemento, A_{TM}, è Turing riconoscibile).

Ora proveremo che EQ_{TM} non è né Turing riconoscibile né co-Turing riconoscibile.

$A_{TM} \leq_m EQ_{TM}$

Proviamo che $A_{TM} \leq_m EQ_{TM}$.

Idea: Data $\langle M, w \rangle$, considerare una MT M_1 che **riconosce** Σ^* e una macchina M_2 che riconosce Σ^* se M accetta w:

Per ogni input x:
M_1 accetta x,
M_2 simula M su w. Se M accetta, M_2 accetta.

Quindi $L(M_1) = L(M_2)$ se e solo se M accetta w.

Consideriamo la seguente macchina di Turing G.

$G =$ "Su input $\langle M, w \rangle$, dove M è una TM e w è una stringa:

1. Costruisce le seguenti due macchine M_1 e M_2
   $M_1 =$ "Su ogni input:
   1. Accetta"

   $M_2 =$ "Su ogni input:
   1. Esegue M su w.
   2. Se M accetta, accetta."
2. Restituisce $\langle M_1, M_2 \rangle$."

G calcola una funzione g.

La funzione g associa a $\langle M, w \rangle$ la stringa $\langle M_1, M_2 \rangle$.

La funzione g è una riduzione da A_{TM} a EQ_{TM}.

Infatti

$$\langle M, w \rangle \in A_{TM} \Leftrightarrow M \text{ accetta } w \Leftrightarrow$$

$$L(M_1) = \Sigma^* = L(M_2) \Leftrightarrow \langle M_1, M_2 \rangle \in EQ_{TM}.$$

$\square$

Il prossimo risultato si basa sull'osservazione che per ogni linguaggio X, il linguaggio X^* è sempre diverso dall'insieme vuoto.

In particolare, per ogni alfabeto Σ, l'insieme Σ^* delle stringhe su Σ è sempre diverso dall'insieme vuoto.

Proviamo che $A_{TM} \leq_m \overline{EQ_{TM}}$.

Idea: Data $\langle M, w \rangle$, considerare una MT M_1 che **riconosce l'insieme vuoto** e una macchina M_2 che riconosce Σ^* se M accetta w:

Per ogni input x:
M_1 rifiuta x,
M_2 simula M su w. Se M accetta, M_2 accetta.

Quindi $L(M_1) \neq L(M_2)$ se e solo se M accetta w.

Consideriamo la seguente macchina di Turing F.

$F =$ "Su input $\langle M, w \rangle$, dove M è una TM e w è una stringa:

1. Costruisce le seguenti due macchine M_1 e M_2
   $M_1 =$ "Su ogni input:
   1. Rifiuta"

   $M_2 =$ "Su ogni input:
   1. Esegue M su w.
   2. Se M accetta, accetta."
2. Restituisce $\langle M_1, M_2 \rangle$."

F calcola una funzione f.

La funzione f associa a $\langle M, w \rangle$ la stringa $\langle M_1, M_2 \rangle$.

La funzione f è una riduzione da A_{TM} a $\overline{EQ_{TM}}$.

Infatti

$$\langle M, w \rangle \in A_{TM} \Leftrightarrow M \text{ accetta } w \Leftrightarrow$$

$$L(M_2) = \Sigma^* \neq \emptyset = L(M_1) \Leftrightarrow \langle M_1, M_2 \rangle \in \overline{EQ_{TM}}.$$

$\square$

Il teorema che vogliamo provare utilizza le due precedenti riduzioni e i seguenti risultati.

Teorema 1
$A \leq_m B$ se e solo se $\overline{A} \leq_m \overline{B}$

Corollario 4.23
$\overline{A_{TM}}$ non è Turing riconoscibile.

Teorema 5.28
Se $A \leq_m B$ e B è Turing riconoscibile, allora A è Turing riconoscibile.

Un linguaggio che non è né Turing riconoscibile né co-Turing riconoscibile

Teorema

EQ_{TM} non è né Turing riconoscibile né co-Turing riconoscibile.

Un linguaggio che non è né Turing riconoscibile né co-Turing riconoscibile

Dimostrazione. Supponiamo per assurdo che EQ_{TM} sia Turing riconoscibile.

Siccome $A_{TM} \leq_m \overline{EQ_{TM}}$ allora $\overline{A_{TM}} \leq_m EQ_{TM}$ (Teorema 1).

Quindi, per il Teorema 5.28, $\overline{A_{TM}}$ sarebbe Turing riconoscibile, in contraddizione con il Corollario 4.23.

Supponiamo per assurdo che EQ_{TM} sia co-Turing riconoscibile, cioè che $\overline{EQ_{TM}}$ sia Turing riconoscibile.

Siccome $A_{TM} \leq_m EQ_{TM}$ allora $\overline{A_{TM}} \leq_m \overline{EQ_{TM}}$ (Teorema 1).

Quindi, per il Teorema 5.28, $\overline{A_{TM}}$ sarebbe Turing riconoscibile, in contraddizione con il Corollario 4.23.

$\square$

- Nota: nei precedenti esempi non si è mostrato che le riduzioni esibite sono funzioni calcolabili.
- Riflessioni. Per ognuno dei precedenti esempi, dare una descrizione (a livello di algoritmo) della MdT F che calcola la riduzione.
- Nota. Data una stringa input, F deve solo scrivere la stringa output. Ad esempio, per la riduzione da A_{TM} a EQ_{TM}, F riceve in input una stringa $\langle M, w \rangle$ e scrive la codifica $\langle M_1, M_2 \rangle$ delle macchine M_1, M_2. F **non deve** eseguire le azioni che definiscono le MdT M_1, M_2.

Teorema di Rice.
Sia $L =$
$\{\langle M \rangle \mid M$ è una macchina di Turing che verifica una proprietà $\mathcal{P}\}$
un linguaggio che soddisfa le seguenti tre condizioni:

1. L'appartenenza di $\langle M \rangle$ a L dipende solo da $L(M)$, cioè $\forall M_1, M_2$ macchine di Turing tali che $L(M_1) = L(M_2)$
$$\langle M_1 \rangle \in L \Leftrightarrow \langle M_2 \rangle \in L$$
$\mathcal{P}$ è "non banale", cioè L non è vuoto e non contiene tutte le codifiche delle macchine di Turing:
2. $\exists$ una macchina di Turing M_1 tale che $\langle M_1 \rangle \in L$,
3. $\exists$ una macchina di Turing M_2 tale che $\langle M_2 \rangle \notin L$

Allora L è indecidibile.

Teorema di Rice.
Sia $L =$
$\{\langle M\rangle \mid M$ è una macchina di Turing che verifica una proprietà $\mathcal{P}\}$
un linguaggio che soddisfa le seguenti tre condizioni:

1. Prese comunque due macchine di Turing M_1, M_2 tali che $L(M_1) = L(M_2)$, risulta
$$\langle M_1\rangle \in L \Leftrightarrow \langle M_2\rangle \in L$$
2. $\exists$ una macchina di Turing M_1 tale che $\langle M_1\rangle \in L$,
3. $\exists$ una macchina di Turing M_2 tale che $\langle M_2\rangle \notin L$

Allora L è indecidibile.