6. RIDUCIBILITÀ

Una ***riduzione*** è un modo di convertire un problema in un altro problema in modo tale che una soluzione al secondo problema può essere usata per risolvere il primo problema.

La riducibilità coinvolge sempre due problemi, chiamati A e B. Se A si riduce a B, possiamo usare una soluzione per B per risolvere A.

Quando A è riducibile a B, trovare la soluzione di A non può essere più difficile di risolvere B perché una soluzione per B offre una soluzione A. In termini di teoria della computabilità, se A riducibile a B e B è decidibile, allora anche A è decidibile.

Equivalentemente, se A è indecidibile e riducibile a B, B è indecidibile.

Il "vero" problema della fermata:

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

Abbiamo $A_{TM} = \{\langle M, w\rangle \mid M \text{ è una MdT che accetta } w\}$. L'unico caso che fa sì che $\langle M, w\rangle$ non sia un'istanza SÍ è il caso in cui la macchina va in loop su *w*.

Vogliamo far vedere che questo $HALT_{TM}$ *è anch'esso indecidibile*, utilizzando una *riduzione* da un problema indecidibile. Se $HALT_{TM}$ fosse decidibile, allora potremmo decidere anche A_{TM} (cioè, se avessimo un decisore per $HALT_{TM}$ allora potremmo costruire un decisore per A_{TM}, il che ci porterà ad una contraddizione per l'indecidibilità di A_{TM}).

Sia *R* una MdT che decide $HALT_{TM}$ (per assurdo). Costruiamo quindi *S* (che può decidere A_{TM}) che sull'input $\langle M, w\rangle$, dove *M* è una MdT e *w* è una stringa, *simula R su* $\langle M, w\rangle$:

- *se R rifiuta*, allora *S* rifiuta (poiché *M* va in loop quando $w \notin L(M)$);
- *se R accetta* (cioè, *M* si ferma su *w*), allora simula *M* finché *M* si arresta su *w*.

Se *M* ha accettato, allora S accetta l'input $\langle M, w\rangle$ ($w \in L(M)$); se *M* ha rifiutato, allora S rifiuta l'input $\langle M, w\rangle$ ($w \notin L(M)$). In definitiva, *S* accetta $\langle M, w\rangle$ se e solo se $\langle M, w\rangle \in A_{TM}$.

Se esistesse *R* che decide $HALT_{TM}$, allora otterremmo *S* che decide A_{TM}. Poiché sappiamo che A_{TM} è indecidibile, allora *R* non può esistere e $HALT_{TM}$ deve essere indecidibile (contraddizione).

In generale, lo **schema di riduzione** dal problema A al problema B si utilizza come segue:

1. Sappiamo che il problema noto A risulta indecidibile;
2. Vogliamo provare che B è indecidibile;
3. Assumiamo (per assurdo) B decidibile ed usiamo questa assunzione per provare A decidibile;
4. La contraddizione ci fa concludere che B è indecidibile.

Di seguito un riepilogo per l'esempio della prova dell'indecidibilità di $HALT_{TM}$:

1. Sappiamo che il problema noto A risulta indecidibile. Abbiamo posto A = A_{TM};

2. Vogliamo provare che B è indecidibile. $HALT_{TM}$ gioca il ruolo di B;

3. Assumiamo (per assurdo) B decidibile ed usiamo questa assunzione per provare A decidibile. Proviamo che se $HALT_{TM}$ è decidibile allora A_{TM} è decidibile;

4. La contraddizione ci fa concludere che B è indecidibile. Giungiamo alla ***contraddizione***.

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

Esempio:

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

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

Un linguaggio *A* è ***riducibile a un linguaggio*** *B* ($\boldsymbol{A \leq_m B}$) se esiste una funzione calcolabile $f: \Sigma^* \to \Sigma^*$ tale che $\forall w,\ w \in A \Leftrightarrow f(w) \in B$.

Una riduzione fornisce un modo per convertire problemi di appartenenza ad *A* in problemi di appartenenza a *B*. Se un problema *A* è riducibile a *B*, e sappiamo risolvere *B*, allora sappiamo risolvere *A*: ciò implica che *A* "non è più difficile" di *B*.

Teorema:

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

Dimostrazione:

Siano *M* il decider per *B* ed f la riduzione da *A* a *B*. Costruiamo un decider *N* per *A* che, su input *w*:

- calcola f(*w*);
- "utilizza" *M* su f(*w*) e dà lo stesso output.

A questo punto: $w \in A \Leftrightarrow f(w) \in B$ (perché *f* è una riduzione da *A* a *B*) $\Leftrightarrow M\ accetta\ f(w)$. Quindi, *N* decide *A*.

Dunque: $w \in A \Rightarrow M\ accetta\ f(w) \Rightarrow N\ accetta\ w$.

D'altronde: $w \notin A \Rightarrow f(w) \notin B \Rightarrow \begin{cases} M\ non\ accetta\ w \\ N\ non\ accetta\ w \end{cases}$.

Teorema:

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

Dimostrazione:

Siano R_A un riconoscitore per *A* e R_B un riconoscitore per *B*; allora:

$$R_A: w \to \boxed{f} \to f(w) \to \boxed{R_B}.$$

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 precedente.

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 precedente.

Esempio1:

Siano definiti i seguenti linguaggi:

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

Un esempio di riduzione è $\boldsymbol{A_{TM} \leq_m \overline{E_{TM}}}$. Il linguaggio E_{TM} prende in input una MdT *M* e si pone come domanda "L(*M*) = ∅?". Questa riduzione ci dirà che decidere se il linguaggio di una MdT è vuoto è un problema indecidibile. Dobbiamo far vedere che esiste la funzione di riduzione, ossia una funzione che mappa stringhe del primo linguaggio in stringhe del secondo linguaggio e, per ogni stringa che non appartiene al primo linguaggio, il risultato sarà una stringa che non appartiene al secondo linguaggio.

Consideriamo $f: \Sigma^* \to \Sigma^*$ tale che $f(\langle M, w\rangle) = \langle M_1\rangle$, dove M_1 su 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 *w*.

f è una riduzione di A_{TM} a $\overline{E_{TM}}$?

Abbiamo che:

$$\langle M, w\rangle \in A_{TM} \Rightarrow w \in L(M) \Rightarrow \begin{cases} x \notin L(M_1) & \text{se } x \neq w \\ w \in L(M_1) & \text{se } x = w \end{cases} \Rightarrow \begin{cases} L(M_1) \neq \emptyset \\ \langle M_1\rangle \in E_{TM} \end{cases}.$$

In questo punto, si ha che la funzione definita mappa una stringa del primo linguaggio (A_{TM}) in una stringa $f(\langle M, w\rangle) = \langle M_1\rangle$ che appartiene al secondo linguaggio; in definitiva, abbiamo mostrato che

$$\langle M, w\rangle \in A_{TM} \Rightarrow f(\langle M, w\rangle) = \langle M_1\rangle \in \overline{E_{TM}}.$$

Ci resta da mostrare il caso in cui la stringa non appartiene ad A_{TM}:

$$\langle M, w\rangle \notin A_{TM} \Rightarrow w \notin L(M) \Rightarrow \begin{cases} x \in L(M_1) & \text{se } x \neq w \\ x \notin L(M_1) & \text{se } x = w \end{cases} \Rightarrow \begin{cases} L(M_1) = \emptyset \\ f(\langle M, w\rangle) = \langle M_1\rangle \notin \overline{E_{TM}} \end{cases}.$$

In sintesi, la funzione *f* è calcolabile, e *M* accetta *w* (cioè, $\langle M, w\rangle \in A_{TM}$) se e solo se $L(M_1) \neq \emptyset$ (cioè, se e solo se $\langle M_1\rangle \in \overline{E_{TM}}$).

□

Per completezza, in base ad uno dei corollari definiti in precedenza, abbiamo che

$$A_{TM} \leq_m \overline{E_{TM}} \text{ e } A_{TM} \text{ indecidibile} \Rightarrow \overline{E_{TM}} \text{ indecidibile.}$$

Quindi, anche E_{TM} è indecidibile (*la decidibilità non risente dalla complementazione*).

Nota: Non si conosce una riduzione da A_{TM} a E_{TM}.

Esempio2:

Siano definiti i linguaggi:

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

Un esempio di riduzione è $\boldsymbol{A_{TM} \leq_m REGULAR_{TM}}$. Il linguaggio $REGULAR_{TM}$ contiene tutte le descrizioni di MdT il cui linguaggio è regolare (cioè, si ricordi, se il linguaggio è accettato da qualche automa finito). Questo linguaggio prende in input una MdT *M* e si pone come domanda "*L*(*M*) è regolare?". Questa riduzione ci dirà che decidere se il linguaggio di una MdT è regolare è un problema indecidibile.

Non abbiamo alcuna restrizione su come costruire la funzione *f*. L'unica cosa importante è che questa funzione sia calcolabile.

Consideriamo la funzione $f: \langle M, w\rangle \to \langle R\rangle$ come riduzione da A_{TM} a $REGULAR_{TM}$, dove *R* su un input *x*:

1. Se $x \in \{0^n1^n \mid n \in \mathbb{N}\}$, allora *R* si ferma e accetta *x*;
2. Se $x \notin \{0^n1^n \mid n \in \mathbb{N}\}$, allora *R* simula *M* su *w* e accetta *x* se *M* accetta *w*.

Posto $L = \{0^n1^n \mid n \in \mathbb{N}\}$ per brevità, abbiamo che:

$$\langle M, w\rangle \in A_{TM} \Rightarrow M \text{ accetta } w \Rightarrow \begin{cases} \text{accettata da } R \text{ (per def.)} & \text{se } x \in L \\ \text{accettata da } R & \text{se } x \notin L \end{cases} \Rightarrow L(R) = \Sigma^*,$$

posto Σ = {0, 1}. Sapendo che Σ^* (= *L*(*R*)) è regolare, allora $f(\langle M, w\rangle) = \langle R\rangle \in REGULAR_{TM}$.

Ci resta da mostrare l'implicazione inversa:

$$\langle M, w\rangle \notin A_{TM} \Rightarrow M \text{ non accetta } w \Rightarrow \begin{cases} \text{accettata da } R \text{ (per def.)} & \text{se } x \in L \\ \text{non accettata da } R & \text{se } x \notin L \end{cases} \Rightarrow L(R) = L,$$

il quale non è regolare (mostrato in passato con il Pumping Lemma).

Questo fatto implica che $f(\langle M, w\rangle) = \langle R\rangle \notin REGULAR_{TM}$. A questo punto, abbiamo dimostrato le due implicazioni.

In sintesi, la funzione *f* è calcolabile, e:

- *L*(*R*) = Σ* (regolare) se *M* accetta *w*;
- *L*(*R*) = {0ⁿ1ⁿ | n ∈ ℕ} (non regolare) altrimenti.

Esempio3:

Siano definiti i linguaggi:

$$E_{TM} = \{\langle M\rangle \mid M \text{ è una MdT e } L(M) = \emptyset\}.$$
$$EQ_{TM} = \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ sono MdT e } L(M_1) = L(M_2)\}.$$

Un esempio di riduzione è $\boldsymbol{E_{TM} \leq_m EQ_{TM}}$. Il linguaggio EQ_{TM} prende in input due MdT M_1 e M_2 e si pone come domanda "*L*(M_1) = *L*(M_2)?". Questa riduzione ci dirà che decidere se il linguaggio di due MdT sono uguali è un problema indecidibile.

Consideriamo la funzione $f: \langle M\rangle \to \langle M_1, M_2\rangle$ come riduzione da E_{TM} a EQ_{TM}, dove $\langle M\rangle \in E_{TM}$ se e solo se $f(\langle M\rangle) \in EQ_{TM}$; occorre dimostrare che la funzione è calcolabile, e che vale il sse precedente.

Sia M_1 una MdT tale che *L*(M_1) = ∅. *La funzione sarà* $f: \langle M\rangle \to \langle M, M_1\rangle$, la quale è calcolabile, e bisogna far vedere che effettivamente questa è una riduzione di E_{TM} a EQ_{TM}.

Mostriamo le implicazioni:

$$\langle M\rangle \in E_{TM} \Rightarrow L(M) = \emptyset \Rightarrow L(M) = L(M_1) \Rightarrow \langle M, M_1\rangle = f(\langle M\rangle) \in EQ_{TM},$$
$$\langle M\rangle \notin E_{TM} \Rightarrow L(M) \neq \emptyset \Rightarrow L(M_1) = \emptyset \Rightarrow \langle M, M_1\rangle = f(\langle M\rangle) \notin EQ_{TM}.$$

Esempio4:
Siano definiti i linguaggi:

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ è una MdT e } w \in L(M)\},$$
$$EQ_{TM} = \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ sono MdT e } L(M_1) = L(M_2)\}.$$

Un esempio di riduzione è $\boldsymbol{A_{TM} \leq_m EQ_{TM}}$. Procediamo come nell'esempio precedente.
Consideriamo la funzione $f: \langle M, w\rangle \to \langle M_1, M_2\rangle$ come riduzione da A_{TM} a EQ_{TM}, dove $\langle M, w\rangle \in A_{TM}$ se e solo se $f(\langle M, w\rangle) \in EQ_{TM}$; occorre dimostrare che la funzione è calcolabile, e che vale il sse precedente.
L'idea è la seguente. Data $\langle M, w\rangle$, consideriamo le MdT M_1 e M_2 tali che:
- M_1 accetta x (qualunque input) $\Rightarrow L(M_1) = \Sigma^*$;
- M_2 simula M su w. Se M accetta w, allora M_2 accetta $x \Rightarrow L(M_2) = M$.

La funzione è calcolabile, in quanto costruire una macchina che accetta tutte le stringhe è semplice, così come la seconda parte di f è M stessa, quindi è l'input (basta concatenare la descrizione di M_1 a M stessa, ed otteniamo il risultato della funzione). Ora, mostriamo che la funzione f è una riduzione.
Mostriamo le implicazioni:

$$\langle M, w\rangle \in A_{TM} \Rightarrow w \in L(M) \Rightarrow \begin{cases} L(M_1) = \Sigma^* \\ L(M_2) = \Sigma^* \end{cases} \Rightarrow L(M_1) = L(M_2) \Rightarrow \langle M_1, M_2\rangle = f(\langle M\rangle) \in EQ_{TM},$$
$$\langle M, w\rangle \notin A_{TM} \Rightarrow w \notin L(M) \Rightarrow \begin{cases} L(M_1) = \Sigma^* \\ L(M_2) = \emptyset \end{cases} \Rightarrow L(M_1) \neq L(M_2) \Rightarrow \langle M_1, M_2\rangle = f(\langle M\rangle) \notin EQ_{TM}.$$

In sintesi, $f: \langle M, w\rangle \to \langle M_1, M_2\rangle$ è una riduzione da A_{TM} a EQ_{TM}.

Teorema:

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

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

$$A_{TM} \leq_m \overline{EQ_{TM}} \Rightarrow \overline{A_{TM}} \leq_m EQ_{TM}.$$

Quindi $\overline{A_{TM}}$ sarebbe Turing riconoscibile: questo è assurdo.
Supponiamo per assurdo che EQ_{TM} sia co-Turing riconoscibile, cioè che $\overline{EQ_{TM}}$ sia Turing riconoscibile. Allora

$$A_{TM} \leq_m EQ_{TM} \Rightarrow \overline{A_{TM}} \leq_m \overline{EQ_{TM}}.$$

Quindi $\overline{A_{TM}}$ sarebbe Turing riconoscibile: questo è assurdo.

6.1 TEOREMA DI RICE

Questo teorema è uno strumento generale che permette di stabilire l'indecidibilità di una vasta classe di problemi.
Afferma che, per ogni proprietà non banale delle funzioni calcolabili, il problema di decidere quali funzioni soddisfino tale proprietà e quali no, è indecidibile.

Proprietà banale:

Proprietà che non effettua alcuna discriminazione tra le funzioni calcolabili, cioè che vale o per tutte o per nessuna.

Formalizzato in termini di linguaggio, e quindi decidibilità di linguaggi:
Teorema di Rice:

Sia ***L_P = {⟨M⟩ | M è una MdT che verifica la proprietà P}*** un linguaggio che soddisfa le seguenti due condizioni: 1. L'appartenenza di M a ***L_P*** dipende solo da L(M), cioè: **$\forall M_1, M_2$ MdT tali che $L(M_1)=L(M_2)$, $\langle M_1\rangle \in L_P \leftrightarrow \langle M_2\rangle \in L_P$** 2. ***$L_P$*** è un ***problema non banale***, cioè: **$\exists M_1, M_2$ MdT tali che $\langle M_1\rangle \in L_P$, $\langle M_2\rangle \notin L_P$** allora ***$L_P$ è indecidibile***.

Quindi, ogni proprietà non banale del linguaggio di una MdT è indecidibile.
Informalmente, il punto 1) significa che la proprietà P è una proprietà del linguaggio della MdT in considerazione, significa che se 2 MdT hanno lo stesso linguaggio, per ogni coppia M_1 e M_2 MdT tali che $L(M_1) = L(M_2)$, allora o entrambe appartengono al linguaggio L_P o nessuna delle due appartiene.
Il punto 2) significa che (non banale significa che non vale né per tutte le macchine né per nessuna macchina) deve esistere almeno una MdT che gode della proprietà P ed almeno una MdT che non gode della proprietà P. In termini del linguaggio L_P devono esistere 2 MdT M_1 ed M_2 dove la descrizione di M_1 appartiene al linguaggio e la descrizione di M_2 non appartiene al linguaggio.
Ricapitolando:
Se abbiamo un insieme di descrizioni di MdT che soddisfano la data proprietà, e quest'ultima è non banale che dipende solo dal linguaggio della macchina e non dalla macchina stessa, allora si ha che il linguaggio L_P è indecidibile. Il che significa che ogni proprietà non banale del linguaggio di una MdT, la verifica di P è un problema indecidibile.

Nota: La differenza tra una proprietà di L(M) e una proprietà di M è che: Proprietà del linguaggio significa che dipende dal linguaggio, L(M), quindi dalle stringhe accettate dalla macchina e non dalla macchina M.

Esempio:

- L(M)=Ø è una proprietà del linguaggio;
- "M ha almeno 1000 stati" è una proprietà della MdT;

Quindi, "L(M) =Ø" è indecidibile, mentre "M ha almeno 1000 stati" è facilmente decidibile, basta guardare alla codifica di M e contare.

Dimostrazione Teorema di Rice:

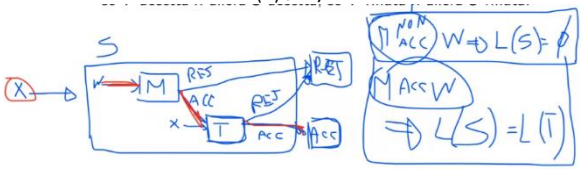
Vogliamo dimostrare che se abbiamo una proprietà P e consideriamo il linguaggio di tutte le descrizioni di MdT che verificano la P, allora il linguaggio L_P=***{⟨M⟩ | M è una MdT che verifica la proprietà P}*** gode delle due proprietà del teorema, L_P è indecidibile.

Mostriamo $A_{TM} \leq_m L_P$, e mostriamo una funzione calcolabile ∀x x∈A_{TM} sse f(x)∈L_P, quindi andiamo a trasformare la nostra riduzione in una funzione calcolabile che ci permette di trasformare elementi di A_{TM} in elementi di L_P, mentre se invece abbiamo una stringa che non appartiene a A_{TM} allora non otteniamo un elemento di L_P. In A_{TM} abbiamo coppie ⟨M, w⟩ e in L_P abbiamo delle MdT M', quindi la funzione deve trasformare una coppia ⟨M, w⟩ in una stringa che è una descrizione di una macchina M', dove ⟨M, w⟩∈A_{TM} sse M'∈L_P. Formalmente: f: ⟨M, w⟩ → ⟨M'⟩.

Andiamo ad indicare con $T_\emptyset$ una MdT tale che L($T_\emptyset$)= ∅, a questo punto possiamo assumere che ⟨$T_\emptyset$⟩∉L_P, altrimenti si potrebbe procedere con $\overline{L_P}$.

Poiché L_P è non banale, per ipotesi del teorema di Rice, esiste una MdT T tale che ⟨T⟩∈L_P (ed esiste una MdT tale che ⟨T⟩∉L_P).

A questo punto, la funzione f sarà: **f(⟨M,w⟩)=⟨S⟩**, dove S è una MdT su input x:



- Simula M con input w:
  - Se M si ferma e rifiuta, allora S rifiuta x;
  - Se M accetta, allora S simula T su input x:
    se T accetta x allora S accetta, se T rifiuta x allora S rifiuta.

Quindi se M rifiuta w il risultato sarà reject, mentre se M accetta w la macchina chiede cosa fa T su input x, quello che abbiamo è che se M non accetta w allora il linguaggio di S sarà ∅, L(S)= ∅. Ma se M accetta w il linguaggio accettato da S sarà il linguaggio accettato da T, L(S)=L(T).

Abbiamo quindi un doppio funzionamento, andando a creare due macchine con due linguaggi ben precisi che dipendono dal fatto che se la coppia ⟨M,w⟩ appartiene o meno ad A_{TM}.

Mostriamo che la funzione ***f è una riduzione***:

- f è calcolabile, perché abbiamo costruito da M e w la macchina S, come se S fosse un programma che utilizza due sottoprogrammi, uno M e uno T.
- ⟨M,w⟩∈A_{TM} ⟺ ⟨S⟩∈L_P, è vero perché:
  (⇒) ⟨M,w⟩∈A_{TM} → w∈L(M) → L(S)=L(T), ma T∈L_P e sfruttando la *proprietà 1 del teorema di Rice* → S∈L_P,
  (⇐) ⟨M,w⟩∉A_{TM} → M non accetta w → L(S)=∅=L($T_\emptyset$) → ma sappiamo per ipotesi ⟨$T_\emptyset$⟩∉L_P e *proprietà 1 del teorema di Rice* → S∉L_P,

Poiché sappiamo che A_{TM} è indecidibile allora L_P è indecidibile.

Conseguenze del Teorema di Rice:

Si può dimostrare col teorema di Rice che non possiamo decidere se una MdT:

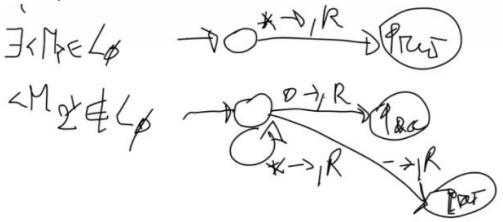
- Accetta ∅:

  Esempio:

  $L_\emptyset$= {⟨M⟩ | M è TM tale che L(M)=∅}, quindi la proprietà P dal teorema di Rice sarà P: L(M)=∅ (non banale perché dipende solo dal linguaggio).

  Verifichiamo adesso questa proprietà tramite i 2 punti del teorema di Rice:

  1. ∀M_1, M_2 MdT tali che L(M_1)=L(M_2), ⟨M_1⟩∈P ↔ ⟨M_2⟩∈P, se queste due macchine hanno lo stesso linguaggio entrambe sono il L(M)=∅ o nessuna: lo vediamo perché se L(M_1)=L(M_2)=∅ e quindi ⟨M_1⟩,⟨M_2⟩∈$L_\emptyset$, mentre se L(M_1)=L(M_2)≠∅ e quindi ⟨M_1⟩,⟨M_2⟩∉$L_\emptyset$.
  2. ∃M_1, M_2 MdT tali che ⟨M_1⟩∈P, ⟨M_2⟩∉P, per far ciò possiamo esibire 2 macchine:



  E quindi le condizioni 1 e 2 ci dicono che $L_\emptyset$ è indecidibile, senza necessità della funzione calcolabile.

- Accetta un linguaggio finito:
- Accetta un linguaggio regolare
- …