Informatica Teorica

Stefano Staffolani, Filippo Speziali

Anno accademico 2022-2023

Contents

1 Puntate precedenti

qui metto quello che manca dalle lezioni precedenti....

2 La macchina di Turing

Una macchina di Turing è una tupla $\langle \sum, Q, q_0, H, \delta \rangle$. Dove:

1. $\sum$ è un **alfabeto** di simboli, che include un simbolo speciale $\emptyset$ che indica una cella vuota.
2. Q è un insieme finito di **stati**.
3. $q_0 \in Q$ è lo stato iniziale.
4. $H \subset Q$ è l'insieme degli **stati accettanti (o finali)**.
5. δ è la funzione di transizione $\delta : (Q \backslash H) \times \sum \to Q \times \sum \{\to, \leftarrow\}$

δ esprime il programma che governa il funzionamento della TM, ed è una funzione totale. Possimao scrivere la definizione di δ come un insieme di quintuple.

2.1 Espressività delle macchine di Turing

Molto spesso un problema che vogliamo risolvere usando uno strumento di calcolo può essere espresso come un problema di decisione. Ad esempio

- Dato un grafo, è strettamente connesso?
- Dato un insieme di equazioni, ha una soluzione?

3 WHILE un linguaggio di programmazione di alto livello

Turing-completo: quando il suo potere espressivo e' equivalente ad una macchina di Turing. Quindi quando qualcuno scrive un nuovo linguaggio deve preoccuparsi del fatto che esso sia Turing-complete. Non tutti i linguaggi sono Turning-Completi. Ad esempio i domain specific language. Sintassi in maniera informale:

1. 1) assegnazione di variabili X:=3
2. 2) cicli while while X!=Y do Program
3. 3) sequenziamento di programmi program1; program2; program3
4. 4) altri costrutti come it-then-else possono essere definiti a partire da questi primitivi.

1) assegnazione di variabili X:=3 NB: NON ci sono side-effects(tipo print(42))

3.1 Semantica

Un programma WHILE calcola una funzione paziale da $N^K - > N^K$

3.2 Equivalenza

Theorem 3.1. *Una funzione (parziale) é computabile da un programma WHILE se e solo se é computabile da una macchina di Turing.*

Proof. La dimostrazione e' per induzione sulla struttura di un programmma WHILE, supponiamo basato su variabili $X_1 ... X_k$

1. Caso base Se il programma e' un'assegnazione di 0 ad una variabile: $begin X_j := 0 end$ la TM corrispondente e' quella che su input rimpiazza il valore di X_j con 0. Gli altri casi base sono il programma vuoto e le altre due forme di assegnazione.

2. Caso induttivo Ci sono due casi da considerare: sequenza e ciclo while.

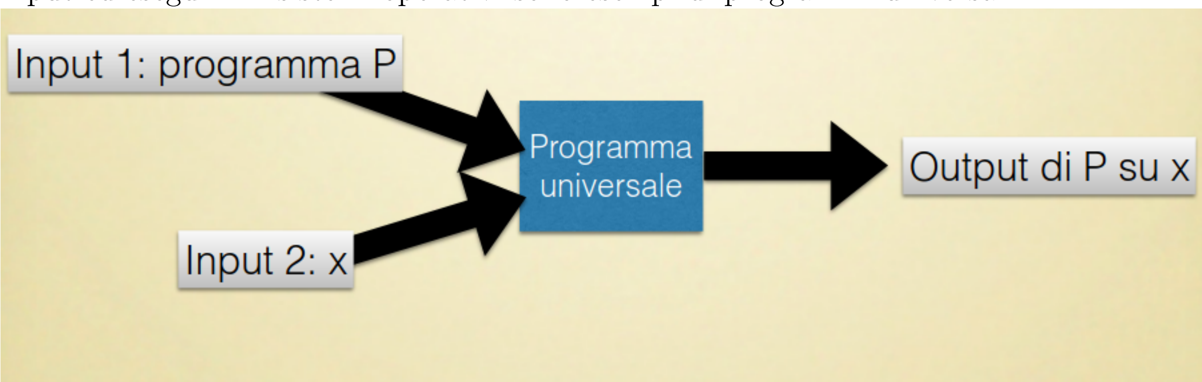
   - sequenza per II abbiamo un programma della forma $begin P_1; P_2; ...; P_j end$ dove $P_1, ...$ sono programmi per i quali abbiamo, da ipotesi induttiva, TM equivalenti M1,... rispettivamente. La TM per $begin P_1; ...; P_j end$ e' definita nel seguente modo a partire da esse, dove l'output di Mi viene dato come input di Mi+1.
   - ciclo while Assumiamo un programma della forma $begin while X_i != X_j do P end$ dove P e' un programma per il quale, da ipotesi induttiva, abbiamo un aTM M equivalente. Possiamo costruire una TM M_{test} che rigetta l'input se il valore di $X_i e X_j$ e' diverso, ed accetta altrimenti. La TM per il nostro programma e' costruita come segue: (snapshot da inserire)

□

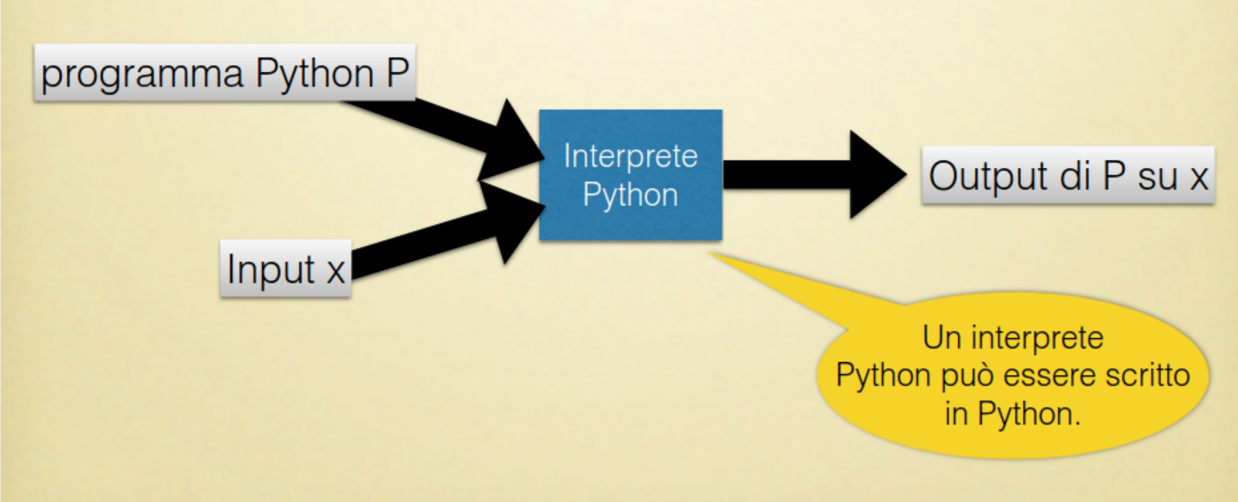
4 Macchina di Turing Universale

4.1 Programmi universali

Un programma universale e' un programma pensato per ricevere altri programmi come input ed eseguirli. I sistemi operativi sono esempi di programmi universali.

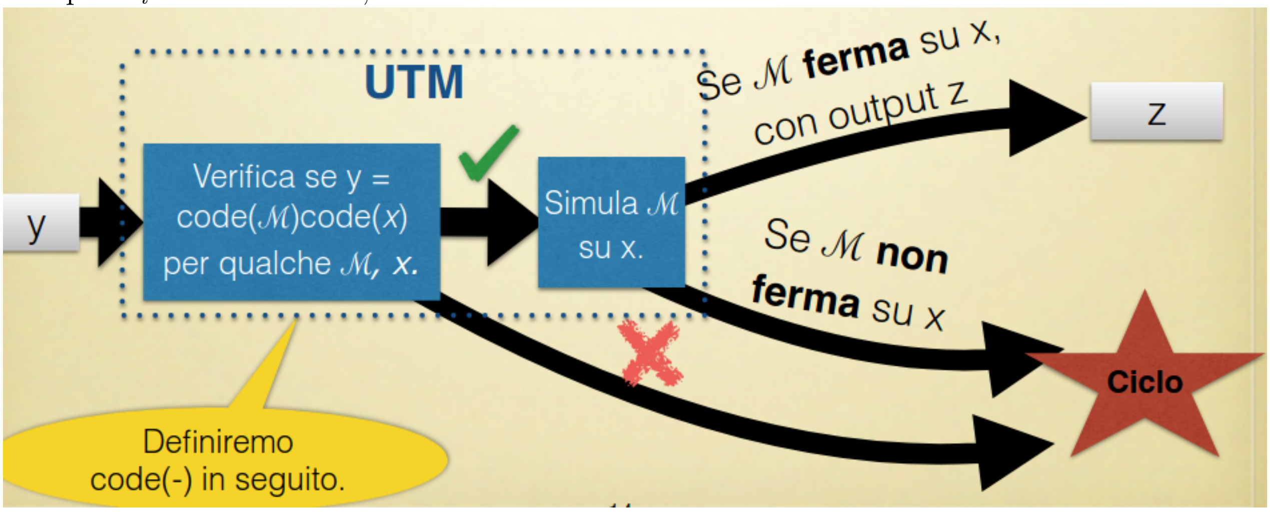


Anche gli interpreti sono programmi universali.



4.2 La macchina di Turing universale (UTM)

La UTM prende in input una stringa y, e per prima cosa verifica che y sia della forma code(M)code(x), dove code() e' una codifica, M e' un TM, e x una stringa nell'alfabeto di input $\sigma_i di M$. Se e' cosi', allora la UTM simula l'esecuzione di M siu x.



4.3 Codificare una TM

$M = (\sigma, Q, q_0, H, \delta)$ Introduciamo alcune convezioni. Gli stati in Q sono ordinati come $q_0, q_1, \ldots$ con q_0 iniziale. Ordiniamo i simboli che possono apparire nella definizione di δ

$$\sigma 0 = \emptyset \quad \sigma 1 = \rightarrow \quad \sigma 2 = \leftarrow$$

e gli altri simboli in σ come $\sigma 4, ..$ Possiamo codificare gli stati ed i simboli come stringhe unarie:

$$code(q_i) = 11...1 \quad code(\sigma_i) = 11...1$$

Codifichiamo una tupla t di δ come:

$$code(t) = code(q_i)0code(\sigma_n)0code(q_j)0code(\sigma_m)0code(\sigma_0)0$$

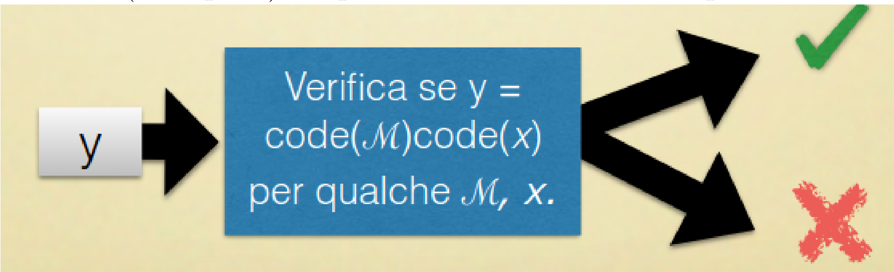
E la funzione di transizione $\delta = t_1, t_2, ..., t_k$ come

$$code(\delta) = code(t_1)0code(t_2)0...0code(t_k)0$$

Possiamo dedurre quali sono gli stati finali di H: sono quelli su cui δ non e' definita (= non occorrono mai in terza posizione in una tupla). INSERIRE ESEMPIO

4.4 Osservazioni sulla codifica

E' possibile che ci siano due o piu' TM che computino la stessa funzione, ma codificate come stringhe differenti (intuitivamente: se esprimono un diverso algoritmo). Nondimeno, la codifica e' *iniettiva*: due macchine differenti saranno codificate da stringhe differenti. Data una stringa su 0,1, e' possibile determinare se sia o meno il codice di una TM (e di quale). In particolare: si tratta di un problema **decidibile**.



4.5 Costruzione di una UTM

La UTM e' definita come una macchina di Turing con tre nastri.

1. Nastro 1 mantiene il nastro di M in forma codificata.
2. Nastro 2 manterra' code(M).
3. Nastro 3 manterra' o stato corrente di M in forma codificata.

Un passo della simulazione di M da parte della UTM funziona nel seguente modo.

1. Cerca in code(M) una tupla $\langle q_i, \sigma_n, q_j, \sigma_m, \sigma_o \rangle$ dove qi concida con lo stato sul nastro 3 e σ_n coincida con il simbolo attualmente esaminato da M.
2. Aggiorna nastro 1 con il nuovo simbolo σ_0 e sposta la testina nella direzione σ_m.
3. Aggiorna nastro 3 con lo stato q_j. Se é finale, fermati.

4.6 Considerazioni finali

Questa costruzione mostra l'esistenza di una macchina di Turing universale, M_U. Niente impedisce a M_U di ricevere la sua stessa codifica code(M_U) come parte dell'input! Questa forma di autoreferenzialita' sara' utilizzata nella prossima lezione per dimostrare che esiste un problema indecidibile.

5 Cosa non possono fare le TM

Introduciamo il nostro primo problema indecidibile: il problema della **fermata**(*halting problem*). Questo risultato ci informa, più in generale, sui limiti della computazione per algoritmi. Abbiamo visto che l'essere calcolabile da una procedura algoritmica implica essere calcolabile da una TM. Dunque **non** essere calcolabile da una TM implica non essere calcolabile da nessuna procedura algoritmica.

5.1 Ripasso: Linguaggi e TM

Theorem 5.1. *Una TM M **decide** un linguaggio **L** se:*

1. *Quando $x \in L$, allora M accetta x(= ferma nello stato Y).*
2. *Quando $x \notin L$, allora M rigetta x(= ferma nello stato N).*

Theorem 5.2. *Una TM M **riconosce** un linguaggio **L** se:*

1. *Quando $x \in L$, allora M termina.*
2. *Quando $x \notin L$, allora M non termina.*

5.2 Gradi di (in)calcolabilità

1. Decidibile da una TM se e solo se è calcolabile($\exists$ un algoritmo che risponde "Si" o "No").
2. Non decidibile da nessuna TM ma riconoscibile da una qualche TM se e solo se non è calcolabile(Semi-decidibile: nessun algoritmo saprà calcolare tutte le risposte "No").
3. Non riconoscibile da nessuna TM se e solo se non è calcolabile(del tutto non calcolabile: qualsiasi algoritmo fallirà nel dare sia le risposte "Si" che "No").

5.3 Un problema indecidibile: Halting Problem

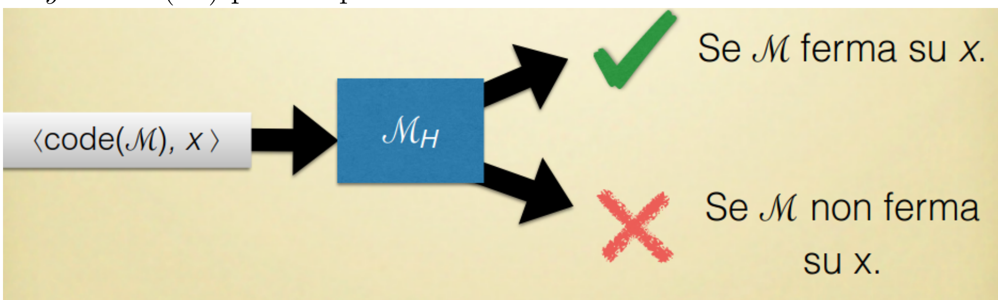
Supponiamo che esista un a codifica *code*$(-)$ che presa una TM su alfabeto $\sum$ restituisce le stringhe $x \in \sum^*$. La codifica usata per definire la macchina di Turing universale è un esempio di tale procedura. Definiamo il linguaggio del **problema della fermata**:

$$\text{HALT} = \{(\text{y,x}) \in \textstyle\sum^* \text{x} \sum^* \mid \text{y} = \text{code}(M) \ e \ M \text{ ferma su x. } \}$$

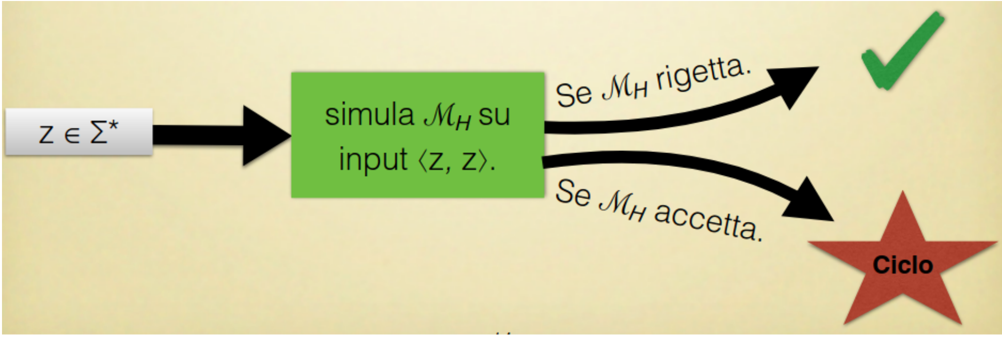
Theorem 5.3. *Il problema della fermata è riconoscibile ma non è decidibile.*

Proof. Dimostriamo che HALT è riconoscibile[1] e che non è decidibile[2].

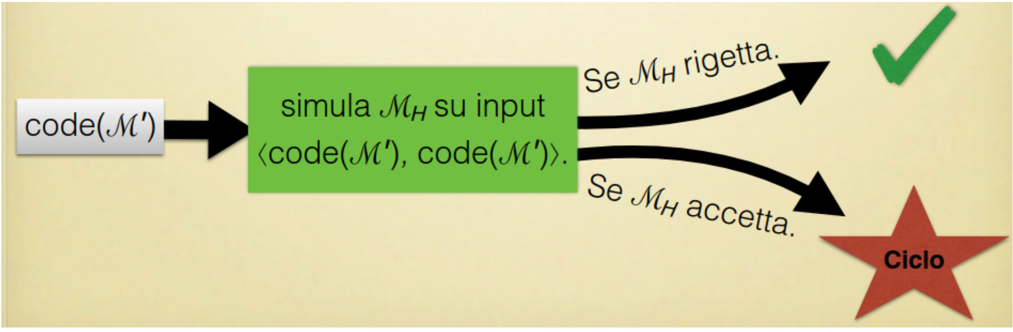
1. Dimostriamo che HALT è riconoscibile. Dobbiamo costruire una TMU M_H che prende in input una coppia (y, x), se y ferma su x M_H si ferma altrimenti cicla.
2. dimostriamo che HALT non è decidibile. Assumiamo che HALT sia decidibile, e chiamiamo M_H la TM che decide HALT. Perciò M_H si comporterà come segue. Se $y = code(M)$ per un qualche M:



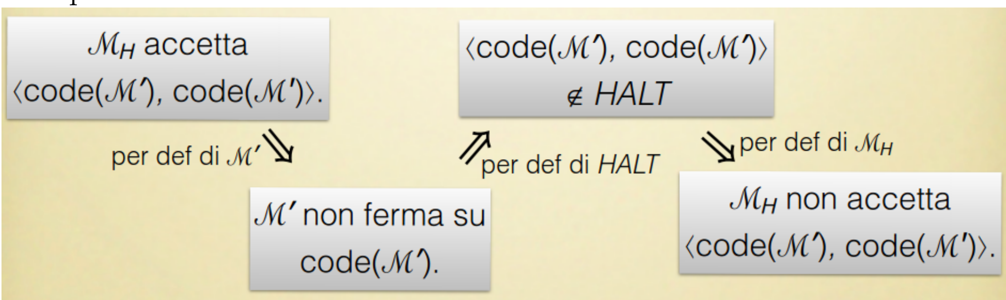
Possiamo definire una nuova TM M' come segue.



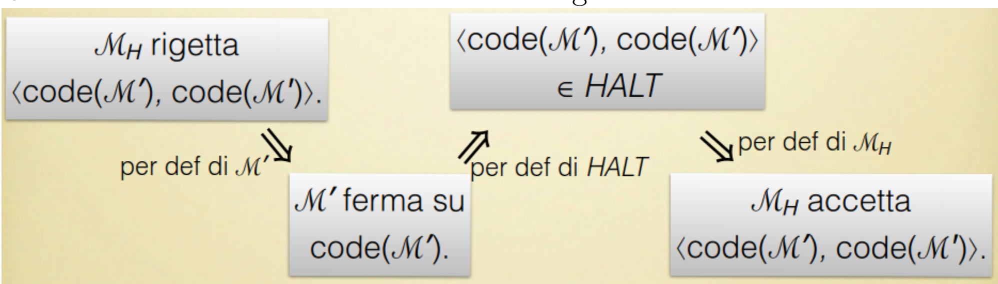
Proviamo ora ad eseguire M' su input $code(M')$.



Dunque



Contraddizione. Proviamo il caso in cui rigetta.



Anche questo caso è in contraddizione. L'unica assunzione utilizzata nel costruire M' è che $\exists M_H$ che decide HALT. Perciò M_H non può esistere: HALT è indecidibile.

$\square$

5.4 Problemi non riconoscibili

Theorem 5.4. *Il complemento $HALT^{-}$ del problema della fermata non è riconoscibile da nessuna TM.*

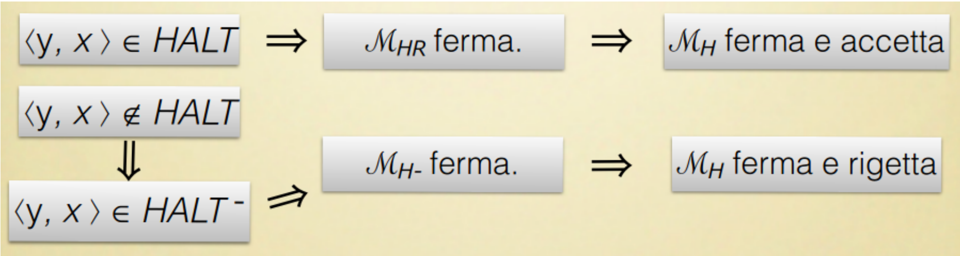
$$HALT = \{\langle y,x\rangle \in \sum^{*} x \sum^{*} \mid y{=}code(M) \text{ } e \text{ } M \text{ } ferma \text{ } su \text{ } x.\}$$

$$HALT^{-} = \{\langle y,x\rangle \in \sum^{*} x \sum^{*} \mid y \neq code(M) \; \forall \; M \;\; o \;\; y{=}code(M) \;\; e \;\; M \;\; non \;\; ferma \;\; su \;\; x\}$$

C'è una dimostrazione diretta, per contaddizione, ma è più interessante mostrare una dimostrazione più astratta. Deriva dal seguente teorema.

Theorem 5.5. *Se $HALT^{-}$ fosse riconoscibile, allora $HALT$ sarebbe decidibile.*

Proof. Abbiamo già visto che HALT è riconoscibile, diciamo da una TM M_{HR}. Supponiamo per assurdo che anche $HALT^{-}$ sia riconoscibile, e chiamiamo $M_{H^{-}}$ la TM che lo riconosce. Possimao ora costruire una TM M_H che decide HALT come segue. Su input $\langle y, x\rangle$, simula M_{HR} e $M_{H^{-}}$ in parallelo su input $\langle y, x\rangle$. Se M_{HR} ferma, accetta. Se $M_{H^{-}}$ ferma, rigetta.



Perciò M_H decide HALT. □

Dal momento che HALT è indecidibile, allora $HALT^{-}$ non può essere riconoscibile.

5.5 Osservazione 1. Complemento linguaggio riconoscibile

La dimostrazione data non sfruttaa in alcun modo il fatto che HALT sia definito nel modo in cui è definito" potremmo sostituire HALT con qualsiasi problema riconoscibile, e funzionerebbe lo stesso. Abbiamo dunque il seguente teorema.

Theorem 5.6. *Se L e L^{-} sono riconoscibili, allora L è decidibile.*

Proof. La stessa data per $L = HALT$ □

Dai teoremi 5.3 e 5.6 otteniamo il seguente corollario.

Corollary 5.6.1. *I linguaggi riconoscibili non sono chiusi sotto complemento.*

Proof. HALT è riconoscibile ma il suo complemento non è riconoscibile. □

5.6 Osservazione 2. Ridurre un problema ad un altro

La nostra dimostrazione del fatto che $HALT^{-}$ non sia riconoscibile ha la seguente struttura:

Se potessimo riconoscere L, allora potremmo decidere L'. Poichè non è decidibile, allora non possiamo riconoscere L.

Come ridurre L a L'? Vedremo come questa intuizione possa essere formalizzata in una tecnica di dimostrazione, che ci permette di ridurre problemi tra di loro al fine di dimostrarne la non calcolabilità.

6 esercitazione 1 Marzo 2023

6.1 Quesito 1

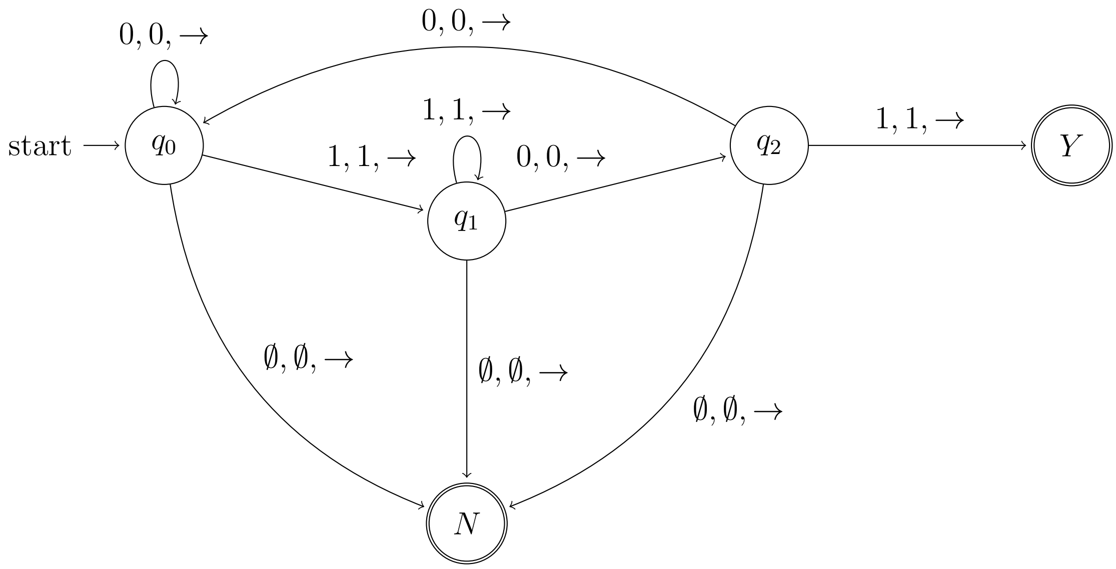
Un problema che vogliamo risolvere usando uno strumento di calcolo puo' essere espresso come un problema di decisione. Questi problemi hanno la componente dei dati e la risposta SI/NO. Per poter risolvere questi problemi con un calcolatore e' necessario codificare un problema di decisinoe come la funzione caratteristica di un linguaggio formale.

Dato a e b dati le cui codifiche sono code(a) e code(b) $\in \sum^*$ Le proprieta' che la codifica deve rispettare sono:

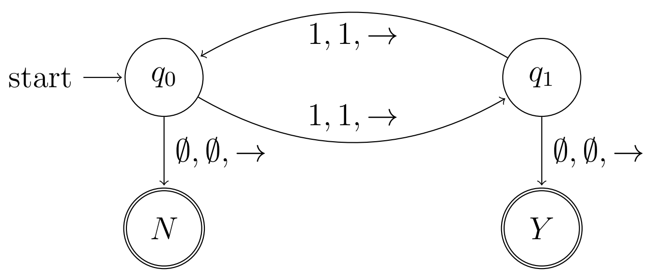
1. $a \neq b$ $code(a) \neq code(b)$
2. deve essere verificabile che se x $\in \sum^*$ è $code(a)$ per qualche a
3. deve essere calcolabile a partire da $code(a)$.

6.2 Problema 2.1

Alfabeto $\sum = \{0, 1\}$ La TM deve accettare l'input quando esso contiene 101 e rifiutare altrimenti.



6.3 Problema 2.2



7 Linguaggi sono non numerabili

Sia $S_{\sum}$ l'insieme di tutti i linguaggi sull'alfabeto finito $\sum$.

Theorem 7.1. *L;insieme $S_{\sum}$ non è numerabile.*

Proof. Ricorda che un linguaggio L è un sottoinsieme di $\sum^*$. Abbiamo già visto che $\sum^*$ è infinito numerabile, quindi possiamo scriverlo come $\sum^* = \{\sigma_1, \sigma_2, \sigma_3, ...\}$. Allora un linguaggio, diciamo $L_1 = \{\sigma_1, \sigma_4\}$, può essere rappresentato come una riga in una tabella:

	σ_1	σ_2	σ_3	σ_4	σ_5	....
L_1	1	0	0	1	0	....

□

Ciascun linguaggio su $\sum$ può essere rappresentato in questo modo. Per contaddizione, assumi che $S_{\sum}$ sia un insieme numerabile. Allora possiamo assegnare un numero naturale ai suoi elementi, così che ogni $L_i \in S_{\sum}$ appare come riga a destra.

	σ_1	σ_2	σ_3	σ_4	σ_5	....
L_1	1	0	0	1	0	....
L_2	0	1	1	0	1	....
L_3	0	0	0	0	0	....
L_4	1	1	1	0	1	....
L_5	1	1	1	1	1	....
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Davvero ogni elemento L di $S_{\sum}$ compare su una riga?
Definisci L come 00110..., allora $\sigma_i \in L \iff \sigma_i \notin L_i$. Quindi L è diverso da ogni linguaggio L_i sulla riga. Dunque L non può essere in una riga! **Contraddizione**.
Quindi, $S_{\sum}$ non è numerabile.

	σ_1	σ_2	σ_3	σ_4	σ_5	....
L_1	1	0	0	1	0	....
L_2	0	1	1	0	1	....
L_3	0	0	0	0	0	....
L_4	1	1	1	0	1	....
L_5	1	1	1	1	1	....
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

7.1 Riassumendo

Dato un alfabeto finito $\sum$, abbiamo visto che:

1. l'insieme di linguaggi riconoscibili da una TM è infinito numerabile.

2. l'insieme di tutti i linguaggi è non-numerabile.

Quindi, esistono limguaggi che non sono riconoscibili da alcuna TM, ad esempio $HALT^{-}, EQ, EQ^{-}$.

7.1.1 Quanti linguaggi non riconoscibili ci sono?

La risposta deriva da un un risultato generale.

Theorem 7.2. *Se S è un insieme infinito, non-numerabile e S' è un sottoinsieme infinito numerabile di S, allora $S \backslash S'$ non è un inifinito numerabile.*

Proof. Assumi $S \backslash S'$ sia infinito numerabile. Allora, poichè i linguaggi infiniti numerabili sono chiusi per unione, $(S \backslash S') \cup S' = S$ è numerabile. Contraddizione! □

Dal 7.2 otteniamo il seguente corollario.

Corollary 7.2.1. *L'insieme di linguaggi non riconoscibili non è infinito numerabile, allora ci sono più linguaggi non riconoscibili che riconoscibili.*

8 Teorema di Rice

Abbiamo visto che non possiamo decidere se una TM:

1. ferma su un dato input
2. ferma su input vuoto (stringa vuota)
3. è equivalente a unàltra TM.

Allora, quali problemi riguardanti le TM sono decidibili? Per esempio:

1. possimo verificare quanti stati ha una macchina, e desumere quanti stati ha
2. se va mai a dx o a sx

Cos'altro?

8.1 ripasso: linguaggi

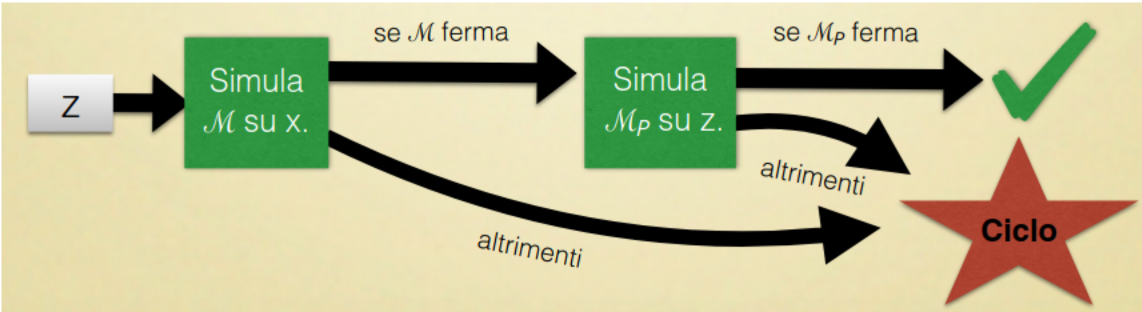
Una **proprietà di linguaggio** P è una funzione da un insieme di TM a $0, 1$ (falso/vero), tale che $L_M = L_{M'}$ implica P(M) = P(M'). Questo assicura che P dipenda solo dal linguaggio descritto dalla macchina. Per esempio: "ferma in 42 step" non è proprietà del linguaggio. Questa proprietà è **non-triviale** se esiste una TM M tale che P(M) - 1 e una TM M' tale che P(M')=0. Formalmente, identificheremo le TM che soddisfano la proprietà P con lìnsieme:

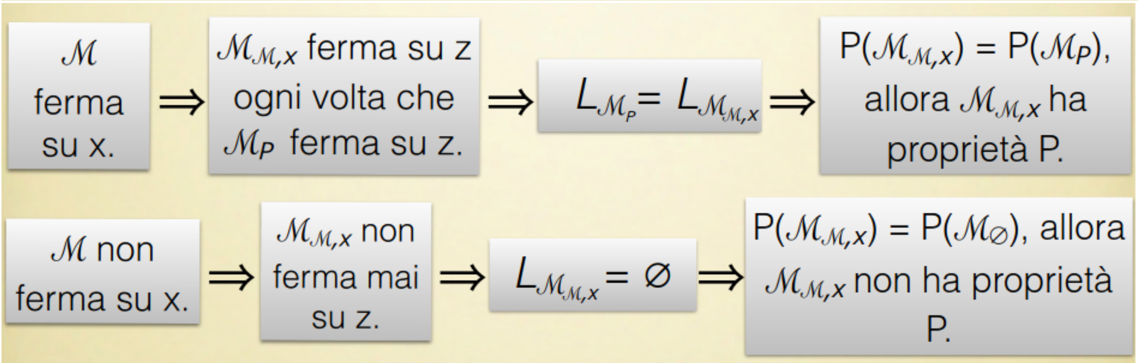
$$\{y \in \sum^* | y = code(M) \ e \ P(M) = 1\}$$

8.2 Teorema di Rice

Theorem 8.1. *Se P è una proprietà di lunguaggio non triviale, allora il problema "M ha proprietà P" è indecidibile.*

Proof. Per contraddizione dimostraiamo che se "M ha proprietà P" fosse decidibile, allora il problema della fermata sarebbe decidibile.
Considera una prprietà P. Assumiamo P($M_\emptyset$) = 0. ($M_\emptyset$ è una TM che riconosce il linuaggio vuoto.) Poichè P è non-triviale, possiamo considerare una TM M_P tale che P(M_P) = 1. Fissiamo M e x come parametri e costruiamo la seguente TM $M_{M,x}$:





Se potessimo decidere se $M_{M,x}$ ha la proprietà P, potremmo decidere il problema della fermata. Allora, $\{y|y = code(M)eP(M) = 1\}$ è indecidibile.
Abbiamo assunto P($M_\emptyset$)=0. Se P($M_\emptyset$)=1? In questo caso, ripetiamo lo stesso argomento, ma per la proprietà $\neg P$ ("M non ha la proprietà P"). Osserva che questo funziona perchè:

1. dato che P è non-triviale, anche $\neg P$ è non-triviale.

2. dato che P($M_\emptyset$) = 1, allora $\neg P(M_\emptyset) = 1\}$ è indecidibile.

Concludiamo che $\{y|y = code(M) \wedge \neg P(M) = 1\}$ è indecidibile. Questo implica che anche $\{y|y = code(M) \wedge P(M) = 1\}$ sia indecidibile. □

NB: lo schema di dimostrazione è:

- Devo dimostrare $A \iff B$
- Dimostriamo $A \Rightarrow B \wedge \neg A \Rightarrow \neg B$
- Che equivale a $A \Rightarrow B \wedge B \Rightarrow A$