La macchina di Turing Universale
Riduzioni fra Linguaggi

Alberto Marchetti Spaccamela

Definizione di Macchina di Turing

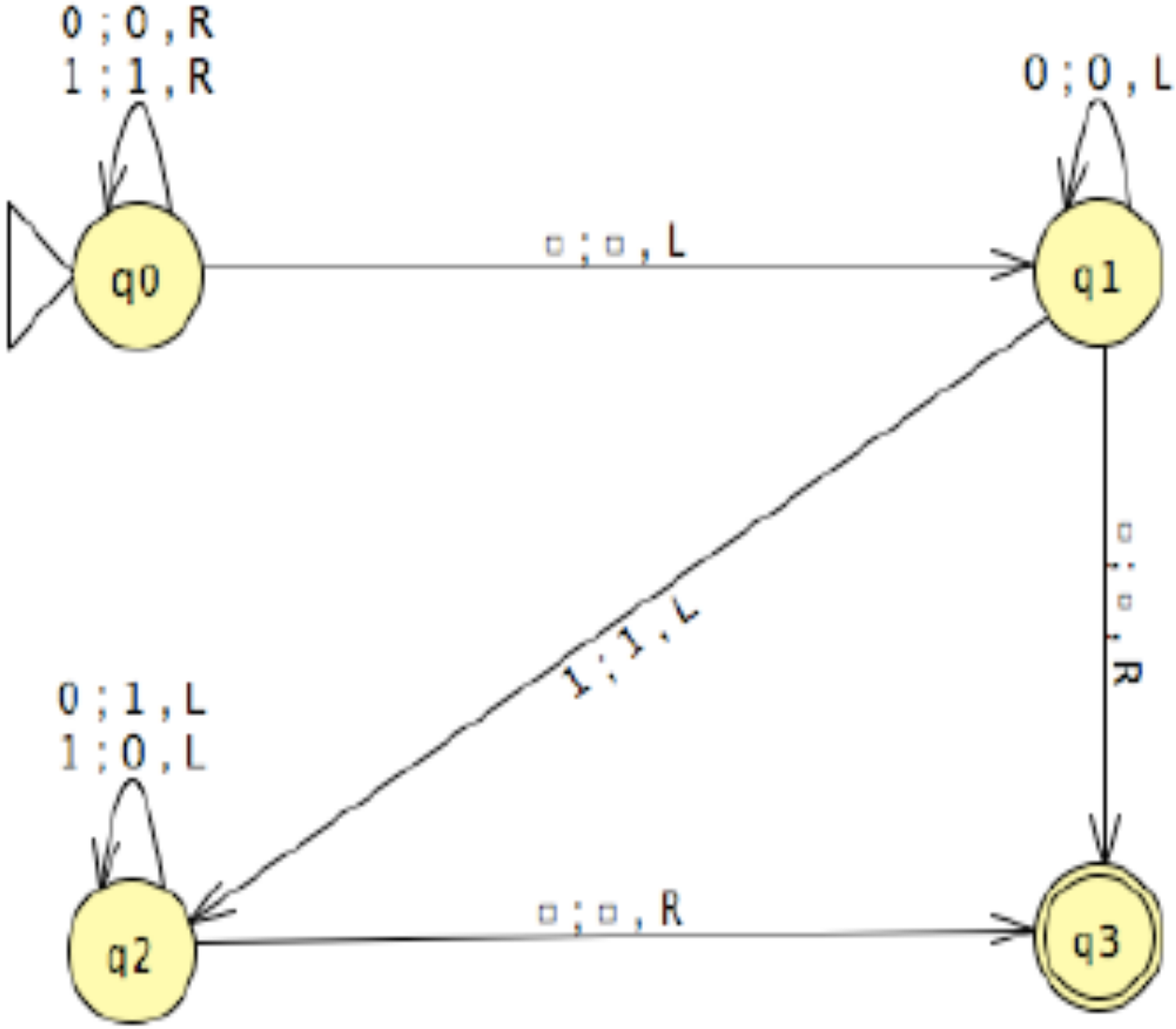
Definizione 1.1: macchina di Turing

Una *macchina di Turing* è costituita da un **alfabeto di lavoro** Σ contenente il simbolo $\square$ e da un **grafo delle transizioni**, ovvero un grafo etichettato $G = (V, E)$ tale che:

- $V = \{q_0\} \cup F \cup Q$ è l'insieme degli **stati** (q_0 è lo stato **iniziale**, F è l'insieme degli stati **finali** e Q è l'insieme degli stati che non sono iniziali né finali);
- E è l'insieme delle **transizioni** e, a ogni transizione, è associata un'etichetta formata da una lista l di triple (σ, τ, m), in cui σ e τ sono simboli appartenenti a Σ e $m \in \{R, L, S\}$, tale che non esistono due triple in l con lo stesso primo elemento.

I simboli σ e τ che appaiono all'interno di una tripla dell'etichetta di una transizione indicano, rispettivamente, il simbolo attualmente letto dalla testina e il simbolo da scrivere, mentre il valore m specifica il movimento della testina: in particolare, R corrisponde allo spostamento a destra, L allo spostamento a sinistra e S a nessun spostamento. Nel seguito di queste dispense, per evitare di dover specificare ogni volta

Rappresentazione grafica



Rappresentazione tabellare

.3: rappresentazione tabellare della macchina per il complemento bit a b

iferimento alla macchina di Turing introdotta nell'Esempio 1.1 e mostr
, la corrispondente rappresentazione tabellare di tale macchina è la segu

stato	simbolo	stato	simbolo	movimento
q0	0	q0	1	R
q0	1	q0	0	R
q0	□	q1	□	L
q1	0	q1	0	L
q1	1	q1	1	L
q1	□	q2	□	R

Un limite delle Macchine di Turing (MdT):

Le MdT sono "hardwired"

Le MdT eseguono un solo programma (codificato nella funzione di transizione)

I computer che usiamo sono programmabili (eseguono diversi programmi)

Soluzione: La macchina di Turing universale

Attributi:

- Riprogrammabile
- Simula ogni altra Macchina di Turing

Nota: Turing ha ideato la macchina universale nel 1936 (ben prima dell'introduzione dei computer) aveva all'epoca 24 anni!

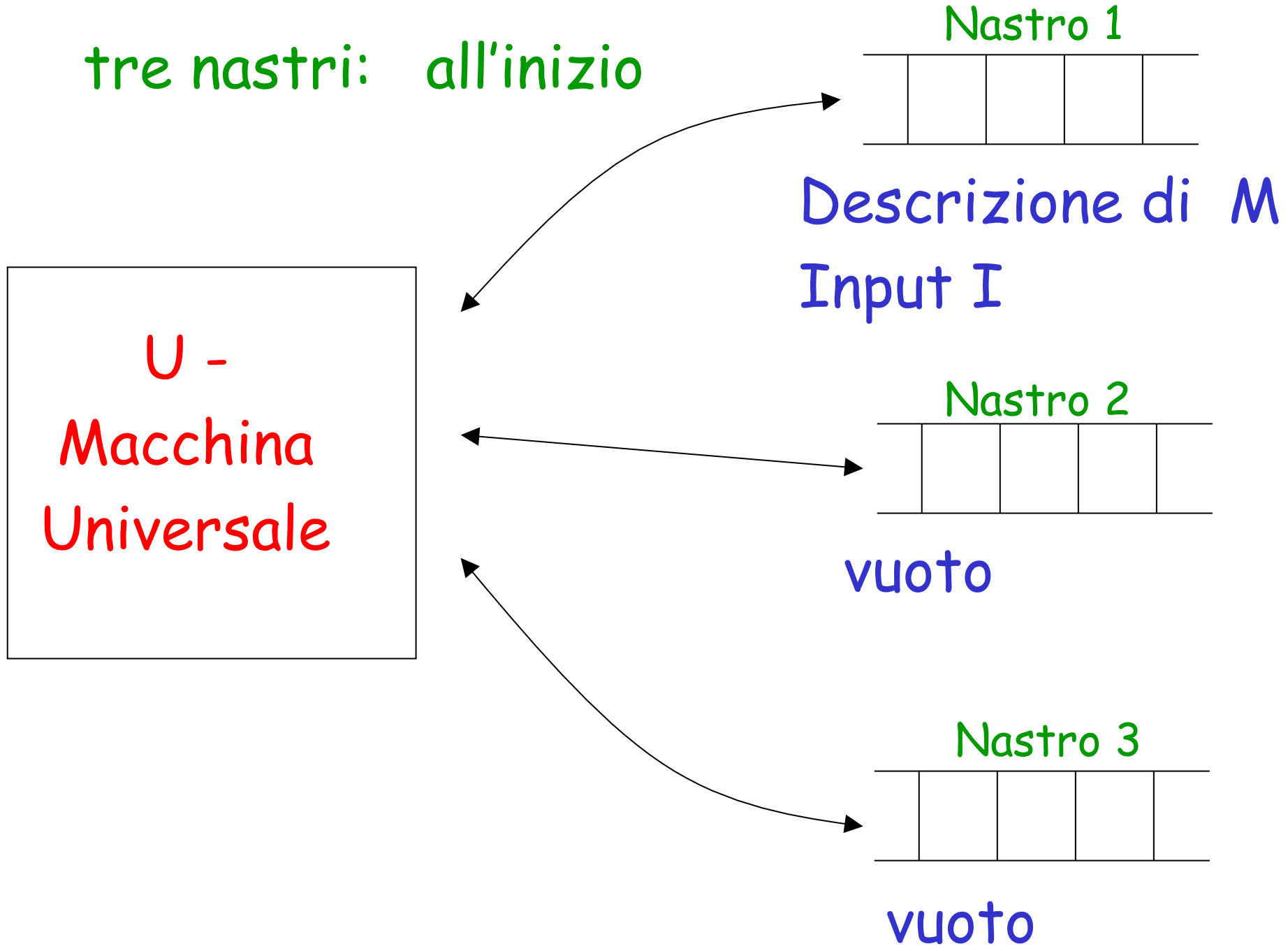
La macchina di Turing universale U

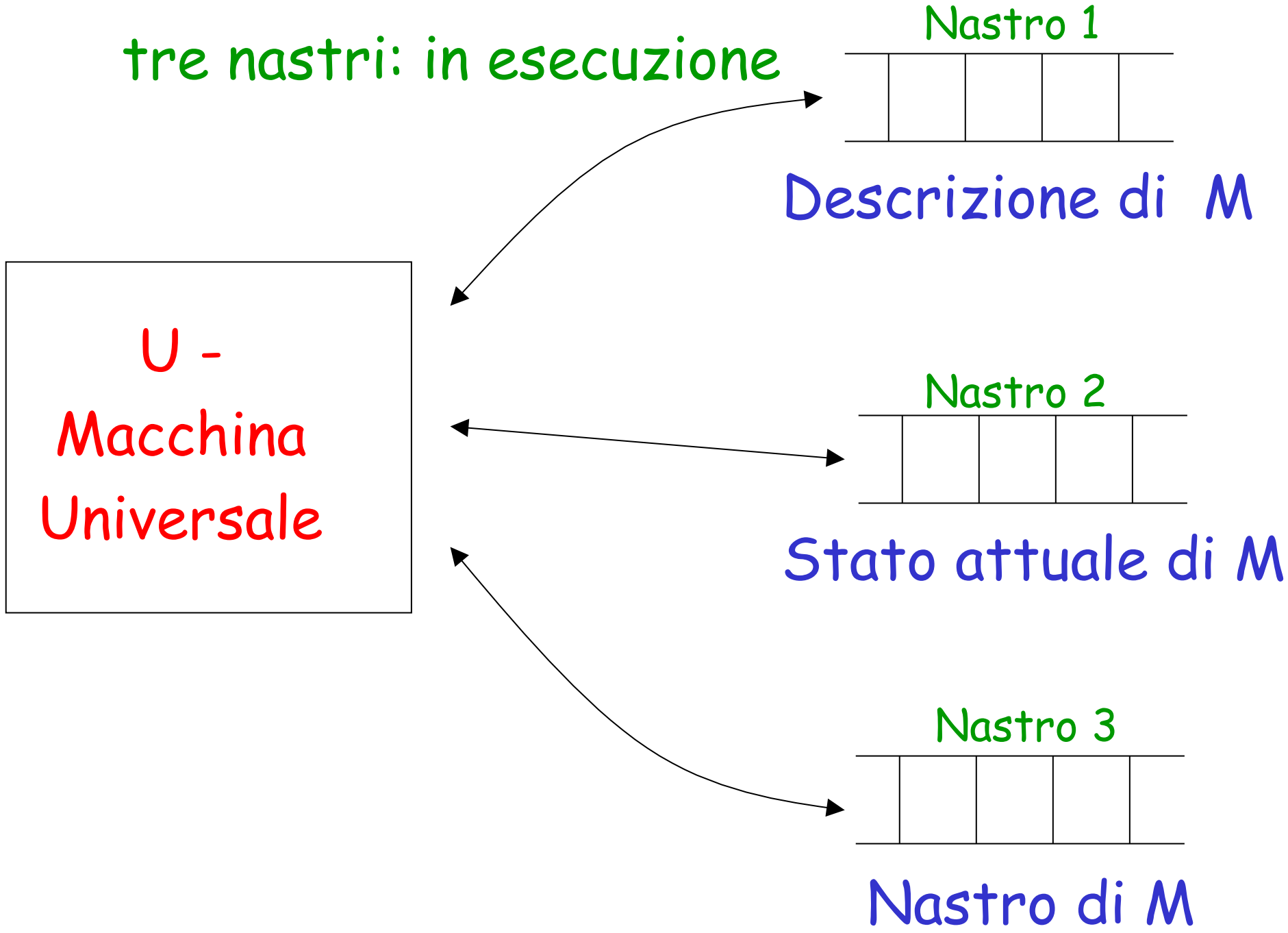
Simula ogni altra Macchina di Turing

Input della macchina di Turing universale

DM - descrizione di M -
una macchina di Turing
I - stringa di input di M

Output calcolo di M con input I


tre nastri: all'inizio
Nastro 1
Descrizione di M
Input I
U -
Macchina
Universale
Nastro 2
vuoto
Nastro 3
vuoto


tre nastri: in esecuzione
Nastro 1
Descrizione di M
U -
Macchina
Universale
Nastro 2
Stato attuale di M
Nastro 3
Nastro di M

Descrizione di M

Nastro di M

Descriviamo (codificchiamo) M
come una stringa di simboli :

Assumiamo che M abbia alfabeto {0,1, □ }

Codifichiamo

0 con Z 1 con U □ con B

Quindi se nastro M contiene □01100□

Terzo nastro di macchina universale è BZUUZZB

Descrizione di M

Descriviamo (codifichiamo) M

come una stringa di simboli :

Assumiamo che M abbia alfabeto {0,1, □ }

Codifichiamo

0 con Z 1 con U □ con B

Descrizione di M

Descriviamo (codificchiamo) M
come una stringa di simboli :

Assumiamo M abbia stati $q_0, q_1, q_2, \ldots q_n$

Stato iniziale q_0, stato finale q_1

Codifichiamo gli stati $q_0, q_1, q_2, \ldots q_n$
con rappresentazione binaria

q_0 con 0, q_1 con 1, q_2 con 10, q_3 con 11, ...

Descrizione di M

Descriviamo (codifichiamo) M
come una stringa di simboli :

I movimenti della testina sono così codficati

Sinistra L

Destra R

Nessun movimento S

Descrizione di M

Descriviamo (codifichiamo) M
come una stringa di simboli : riassunto

Alfabeto di macchina universale

Z, U, B (per codificare alfabeto di M)

0,1 (per codificare gli stati di M)

R, L, S (per codificare gli spostamenti)

; (utilizzato per separare sul primo nastro la descrizione di M dall'input)

Descrizione di M

Alfabeto di macchina universale

Z, U, B (per codificare alfabeto di M)

0,1 (per codificare gli stati di M)

R, L, S (per codificare gli spostamenti)

stato	simbolo	stato	simbolo	movimento	codifica
q0	0	q0	1	R	0Z0UR
q0	1	q0	0	R	0U0ZR
q0	□	q2	□	R	0B10BR
q2	0	q2	0	L	10Z10ZL
q2	1	q2	1	L	10U10UL
q2	□	q1	□	R	10B1BR

Codifica COMPLETA macchina è

0Z0UR0U0ZR0B10BR10Z10ZL10U10UL10B1BR

stato	simbolo	stato	simbolo	movimento	codifica
q0	0	q0	1	R	0Z0UR
q0	1	q0	0	R	0U0ZR
q0	□	q2	□	R	0B10BR
q2	0	q2	0	L	10Z10ZL
q2	1	q2	1	L	10U10UL
q2	□	q1	□	R	10B1BR

Codifica: 0Z0UR0U0ZR0B10BR10Z10ZL10U10UL10B1BR

Stessa macchina codifica diversa

0B11BL0U0ZR0Z0….

stato	simbolo	stato	simbolo	movimento	codifica
q0	□	q2	□	R	0B10BR
q0	1	q0	0	R	0U0ZR
q0	0	q0	1	R	0Z0UR
q2	□	q1	□	R	10B1BR
q2	1	q2	1	L	10U10UL
q2	0	q2	0	L	10Z10ZL

Passi Macchina Universale

1. Copia sul terzo nastro l'input x codificato mediante i simboli U e Z.
2. Inizializza il contenuto del secondo nastro con la codifica dello stato iniziale di M.
3. In base allo stato (contenuto nel secondo nastro) e al simbolo letto (terzo nastro), cerca sul primo nastro una transizione che possa essere applicata. Se tale transizione non viene trovata, termina in una configurazione di rigetto.
4. Altrimenti, applica la transizione modificando il contenuto del terzo nastro e aggiornando il secondo nastro in base al nuovo stato di M.
5. Se il nuovo stato è uno stato finale di M, termina nell'unico stato finale di U. Altrimenti, torna al Passo 3.

passo 1: Copia sul terzo nastro l'input x codificato mediante i simboli U e Z.

1. Posiziona la testina sul primo simbolo alla destra del simbolo " ;" il quale viene cancellato.

2. Scorre il primo nastro verso destra e, per ogni simbolo diverso da , lo copia sul terzo nastro (spostando la testina di questo nastro a destra) e lo cancella.

3. Posiziona la testina del primo nastro e quella del terzo nastro sul simbolo diverso da più a sinistra.

Passo 2: Inizializza il contenuto del secondo nastro con la codifica dello stato iniziale di T.

il secondo passo nizializza il contenuto del secondo nastro con la codifica dello stato iniziale, deve semplicemente scrivere su tale nastro il simbolo 0 e posizionare la testina su di esso

Ricordiamo che in base alle assunzioni fatte nel paragrafo precedente, lo stato iniziale di una qualunque macchina di Turing è quello di indice 0.

Passo 3: In base allo stato (nastro 2) e al simbolo letto (nastro 3), cerca sul primo nastro una transizione da applicare

Cerca sul primo nastro la prima occorrenza del contenuto del secondo nastro (stato).

Una volta trovata tale occorrenza la ricerca prosegue verificando se il simbolo immediatamente successivo è uguale al simbolo attualmente scandito sul terzo nastro: in tal caso, posiziona la testina sulla seconda parte della transizione appena identificata.

Altrimenti, prosegui cercando sul primo nastro un'altra occorrenza del contenuto del secondo nastro.

Se si scorre tutto il primo nastro e non si trova la transizione ERRORE

Passo 4: Copia sul terzo nastro l'input x codificato mediante i simboli U e Z.

1. Cancella l'intero contenuto del secondo nastro e copia su di esso il nuovo stato (sequenza di simboli 0 e 1 contenuta sul primo nastro e il cui primo simbolo è attualmente scandito)
2. Sostituisci il simbolo attualmente scandito sul terzo nastro con quello presente sul primo nastro e sposta la testina di quest'ultimo nastro di una posizione a destra.
3. In base al simbolo letto sul primo nastro, sposta la testina del terzo nastro (se il nuovo simbolo letto su tale nastro è □ sostituiscilo con il simbolo B)
4. Posiziona la testina del primo nastro sul primo simbolo a sinistra diverso da □ (preparandosi per prossimo passo)

Passo 5: Se il nuovo stato è uno stato finale di M, termina nell'unico stato finale di U. Altrimenti, torna al Passo 3.

Ricordiamo che siamo in stato finale di M se il contenuto del secondo nastro è uguale alla stringa 1.

1. Se siamo in stato finale cancella il contenuto del primo nastro e copia l'output della macchina simulata, decodificato facendo uso di simboli 0 e 1.

2. Se non è così la macchina di Turing universale torna a eseguire il terzo passo

Sulle dispense la macchina U è data nei dettagli

Osservazioni

1. U è una Macchina di Turing in grado di simulare ogni altra macchina di Turing
2. U ha 28 stati (!) ed è in grado di simulare una qualunque macchina di Turing (anche una con un milione di stati)

Assurdo?

1. Sappiamo che esistono interpreti C scritti in C
2. Un interprete C ha dimensione finita ed è in grado di eseguire un programma C di lunghezza qualunque

Sulle dispense la macchina U è data nei dettagli

Osservazioni

1. U è una Macchina di Turing in grado di simulare ogni altra macchina di Turing
2. U ha 28 stati (!) ed è in grado di simulare una qualunque macchina di Turing (anche una con un milione di stati)

Assurdo?

1. Sappiamo che esistono interpreti C scritti in C
2. Un interprete C ha dimensione finita ed è in grado di eseguire un programma C di lunghezza qualunque

Il problema della fermata vale per ogni linguaggio di progrmmazione o modello di calcolo noto. Quindi il risultato di indecidiblità ha valore oltre l'informatica: si riferisce alla nostra capacità di calcolare

Domanda: il problema è importante in pratica?

In fin dei conti i programmatori hanno risolto il problema per i programmi che usiamo in pratica. Possiamo dire che il problema della fermata è risolubile in tutti i casi pratici? (e allora chi se ne frega)

Due osservazioni

- Un programma che risolve il problema della fermata farebbe risparmiare tempo quando i nostri programmi ciclano (magari indicando il punto in cui cicla) COMODO
- Più in generale esistono altri problemi indecidibili che pongono limit i a quello che possiamo programmare

Congettura di Goldbach

uno dei più vecchi problemi irrisolti nella teoria dei numeri (dal 1742). Essa afferma che ogni numero pari maggiore di 2 può essere scritto come somma di due numeri primi (anche uguali).

Python any (all)

any(*iterable*)

Ritorna True se un qualunque elemento di *iterable* è true. Se iterable è vuoto ritorna False

all(*iterable*)

Ritorna True se ciascun elemento di *iterable* è True. Se iterable è vuoto ritorna True

Programma seguente si ferma se e solo se la congettura di Goldbach è falsa.

```
def isprime(p):
        return all(p % i for i in range(2,p-1))
def Goldbach(n):
        return any( (isprime(p) and isprime(n-p))
                for p in range(2,n-1))
n = 4
while True:
        if not Goldbach(n): break
        n+= 2
```

se sapessi risolvere problema fermata avrei risolto la congettura di Goldbch

Linguaggi decidibili e non decidibili

Un linguaggio L è decidibile se la sua funzione caratteristica X_L è calcolabile.

X_L (y)= 1 se y appartiene a L

X_L (y)= 0 altrimenti

Se X_L non è calcolabile L è indecidibile

Esempio Linguaggio derivato da problema fermata MdT (M si ferma con input x)?

X_{Stop} (c,x)= 1 se MdT codficata da c si ferma con input x

X_{Stop} (c,x)= 0 altrimenti

Linguaggi decidibili e non decidibili

Un linguaggio L è decidibile se la sua funzione caratteristica X_L è calcolabile.

X_L (y)= 1 se y appartiene a L

X_L (y)= 0 altrimenti

Esistono linguaggi non decidibili

X_{Stop}(linguaggio derivato dal problema della fermata) è indecidibile

Linguaggi decidibili e non decidibili

Un linguaggio L è semi-decidibile se esiste una macchina di Turing T tale che, per ogni stringa binaria y, se y appartiene L, allora T(y) termina in una configurazione finale, altrimenti T(y) non termina.

Teorema L_{Stop} è semidecidibile

Prova: esiste Macchina di Turing universale

Riduzione fra da problema Fermata a problema Fermata_input_0

Problema Fermata_input_0

Input: una stringa che descrive una MdT M

Output: 1 se M si ferma con input 0

Apparentemente Fermata_input_0
sembra più semplice di Fermata
(dobbiamo risolvere solo per un input)

Riduzione da problema Fermata a problema Fermata_input_0

Algoritmo: (obiettivo risolvere: Fermata assumendo che esista alg. A(M) che risolve Fermata_input_0 per ogni M)

Input: M,x ; Output Fermata(M,x)

1. Sia $N_{M,x}$ la MdT che risolve il seguente: "su input $z \in \{0,1\}^*$ valuta M su input x e ritorna il risultato" (Nota si ignora z e x è nella codifica della macchina di Turing)
2. Ritorna y= $A(N_{M,x})$ su input 0.

Riduzione da problema Fermata a problema Fermata_input_0

Algoritmo: (obiettivo risolvere: Fermata assumendo che esista alg. A(M) che risolve Fermata_input_0 per ogni M)

La prova è per assurdo e controintuitiva:

Ragionamento del tipo se "I maiali fischiano (probl. fermata decidibile) allora i cavalli volano (probl. fermata su input zero decidibile)"

Poichè so che i maiali non fischiano (problema fermata non decidibile) allora deduco che i cavalli non volano (fermata con input zero non decidibile)

Riduzione da problema Fermata a problema Fermata_input_0

Prova formale: idea (difficoltà principale) è pensare che in $N_{M,x}$ x non è input di M ma è una nuova costante inserita nel programma che definisce una macchina di Turing diversa da M.

L'algoritmo NON esegue $N_{M,x}$ ma semplicemente scrive la descrizione di $N_{M,x}$ come una stringa e fornisce questa stringa in input a A (che risolve il prob. Fermata_input_0)

Lemma: data la stringa M,x,z la macchina $N_{M,x}$ si ferma su input z se e solo se M si ferma con input x

Riduzione da problema Fermata a problema Fermata_input_0

Lemma: data la stringa M,x,z la macchina $N_{M,x}$ si ferma su input z se e solo se M si ferma con input x

Prova: $N_{M,x}$ ignora input z (fa la stessa cosa qualunque sia z); $N_{M,x}$ si ferma se e solo se M si ferma su input x

In altre parole

$$\text{Fermata}(M,x) = \text{Fermata\_input\_0}(N_{M,x})$$

NB: la descrizione di M e del suo input sono stringhe di bit; x in questo caso NON e' input di una MdT ma parte della descrizione di $N_{M,x}$

IMPORTANTE: per dimostrare indecidibilità la riduzione va da problema *A* che so indecidbile a problema B che voglio dimostrare indecidibile

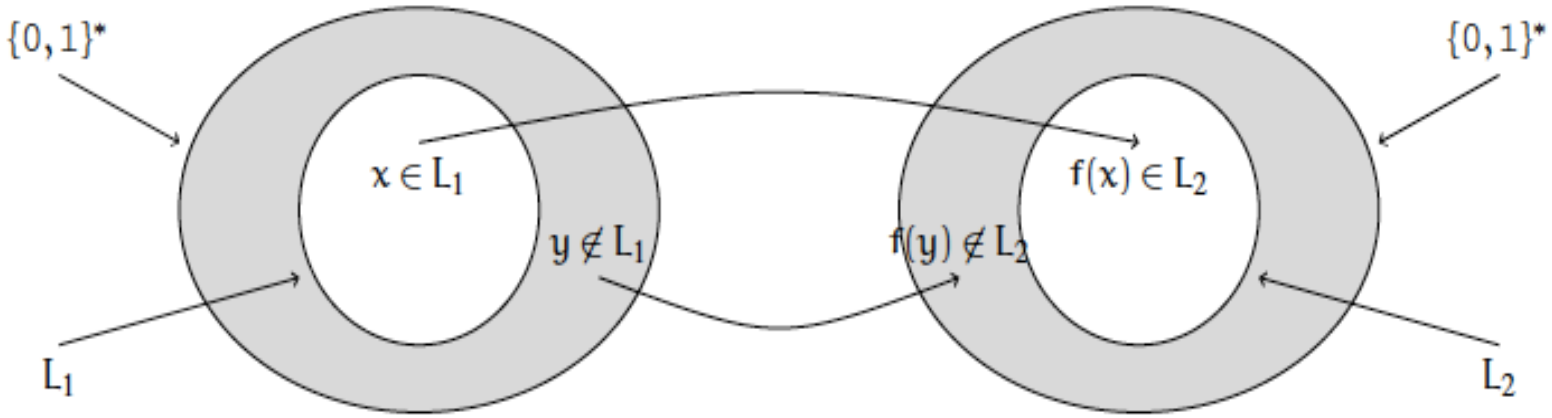
La riduzione mostra che se sapessi risolvere B allora risolvo anche problema *A*

Quindi B non è più facile di *A*

La riduzione nella direzione opposta non fornisce nulla (mi dice che *A* non è più facile di B ma non dice nulla su quanto sia difficile B)

Un linguaggio L1 è riducibile a un linguaggio L2 se esiste una funzione totale calcolabile $f : \{0,1\}^* \rightarrow \{0,1\}^*$, detta riduzione, tale che, per ogni stringa binaria x, $x \in L1$ se e solo se $f(x) \in L2$ (x in L1 se e solo se f(x) in L2)

Figura 3.4: riducibilità tra linguaggi



Siano L1 e L2 due linguaggi tali che L1 è riducibile a L2.

1- Se L2 è decidibile, allora L1 è decidibile.

2- Se L1 non è decidibile, allora L2 non è decidibile.

Intuizione se L1 riducibile a L2 allora

- L1 non è più difficile di L2 (vedi 1 sopra)
- L2 è almeno tanto difficile quanto L1 (2 sopra)

Risultato_sempre_0: data MdT M output 1 se e solo se M da in output 0 per ogni input

Linguaggio associato:

$L_{sempre\ 0}$={codifica di M tale che output di M è 0 per ogni input }

Teorema: Risultato_sempre_0 è indecidibile

Prova: Riduzione da Fermata_input_0 a problema Risultato_sempre_0

NB: la riduzione va da problema che so indecidbile a problema che voglio dimostrare indecidibile

Esempi di riduzione

Esempio 3.6: problema della terminazione con input fissato

Consideriamo il seguente linguaggio: $L_{stop-0} = \{c_T : c_T \in \mathcal{C} \wedge T(0) \text{ termina}\}$. Dimostriamo ora che L_{stop} è riducibile a L_{stop-0}: dal Teorema 3.5 e dal Corollario 3.2, segue che L_{stop-0} non è decidibile. Data una stringa binaria y, la riduzione f per prima cosa verifica se $y = \langle c_T, x \rangle$ con $c_T \in \mathcal{C}$: in caso contrario (per cui $y \notin L_{stop}$), $f(y)$ produce la codifica di una qualunque macchina di Turing che con input 0 non termina (per cui $f(y) \notin L_{stop-0}$). Altrimenti, $f(\langle c_T, x \rangle)$ produce la codifica $c_{T'}$ di una macchina di Turing T' che, con input una stringa binaria z, per prima cosa cancella z e lo sostituisce con x e, quindi, esegue T con input x. Chiaramente, $\langle c_T, x \rangle \in L_{stop}$ se e solo se $c_{T'} = f(\langle c_T, x \rangle) \in L_{stop-0}$.

Esempi di riduzione

Esempio 3.7: problema dell'accettazione

Consideriamo il seguente linguaggio: $L_{acc} = \{\langle c_T, x\rangle : c_T \in \mathcal{C} \wedge T(x)$ termina in una configurazione finale$\}$. Dimostriamo ora che L_{stop} è riducibile a L_{acc}: dal Teorema 3.5 e dal Corollario 3.2, segue che L_{acc} non è decidibile. Date due stringhe binarie $c_T \in \mathcal{C}$ e x, $f(\langle c_T, x\rangle)$ produce la coppia $\langle c_{T'}, x\rangle$ dove $c_{T'}$ è la codifica di una macchina di Turing T' che, con input una stringa binaria z, esegue T con input z: se $T(z)$ termina, allora T' termina in una configurazione finale. Chiaramente, $\langle c_T, x\rangle \in L_{stop}$ se e solo se $T'(x)$ termina in una configurazione finale se e solo se $\langle c_{T'}, x\rangle = f(\langle c_T, x\rangle) \in L_{acc}$.

Esempi di riduzione

Esempio 3.8: problema del linguaggio vuoto

Consideriamo il seguente linguaggio: $L_{empty} = \{c_T : c_T \in \mathcal{C} \wedge \forall x \in \{0,1\}^* [T(x) \text{ non termina in una configurazione finale}]\}$. Dimostriamo ora che L_{acc} è riducibile a L^c_{empty}: dall'esempio precedente, dal Corollario 3.2 e dal Teorema 3.1, segue che L^c_{empty} e L_{empty} non sono decidibili. Date due stringhe binarie $c_T \in \mathcal{C}$ e x, $f(\langle c_T, x\rangle)$ produce la codifica $c_{T'}$ di una macchina di Turing T' che, con input una stringa binaria y, per prima cosa cancella y e lo sostituisce con x e, quindi, esegue T con input x. Abbiamo che $\langle c_T, x\rangle \in L_{acc}$ se e solo se $T(x)$ termina in una configurazione finale se e solo se, per ogni stringa binaria y, $T'(y)$ termina in una configurazione finale se e solo se $c_{T'} \in L^c_{empty}$ (in quanto T' si comporta allo stesso modo indipendentemente dalla stringa di input y).

Esempi di riduzione

Esempio 3.9: problema dell'equivalenza tra linguaggi

Consideriamo il seguente linguaggio.

$$L_{eq} = \{\langle c_{T_1}, c_{T_2}\rangle \quad : \quad c_{T_1} \in \mathcal{C} \wedge c_{T_2} \in \mathcal{C} \wedge \forall x \in \{0,1\}^* [T_1(x) \text{ termina in una configurazione finale se e solo se } T_2(x) \text{ termina in una configurazione finale}]\}$$

(in altre parole, L_{eq} include tutte le coppie di codifiche di macchine di Turing che decidono lo stesso linguaggio). Riduciamo ora L_{empty} a L_{eq}: dall'esempio precedente e dal Corollario 3.2, segue che L_{eq} non è decidibile. Data una stringa binaria $c_T \in \mathcal{C}$, $f(c_T)$ produce la coppia $\langle c_T, c_R\rangle$ dove c_R è la codifica di una macchina di Turing R che non accetta alcuna stringa. Chiaramente, $c_T \in L_{empty}$ se e solo $\langle c_T, c_R\rangle \in L_{eq}$.

Esercizi

Dimostrare che se L oppure L^c è un linguaggio finito, allora L e L^c sono due linguaggi decidibili.

Dimostrare che se L1 e L2 sono due linguaggi decidibili, allora anche L1 unione L2 e L1 intersezione L2 sono decidibili.

Disegnare una MdT con due nastri che verifichi se una sequenza di parentesi (e) sono ben bilanciate (es. (()()) SI; ()))(NO)