Linguaggi di Programmazione
Corso di Laurea in "Informatica"

Fondamenti e Calcolabilità

Valeria Carofiglio

a.a. 2015-2016

(questo materiale è una rivisitazione del materiale prodotto da Nicola Fanizzi)

Teoria della calcolabilità

Studio dei formalismi nei quali esprimere algoritmi e loro limitazioni

- <u>Problema di semantica statica</u>:
  - si può stabilire, mediante un analizzatore della semantica statica
    - Se un programma può generare una divisione per zero?
    - Più in generale:
      se un programma può entrare in un ciclo infinito (divergere) ?

Un semplice programma C

```c
#include <stdio.h>

main()

{

  printf("Ciao, mondo\n");

}
```

Teorema di Fermat espresso come forma di saluto

```c
int exp(int i, n)
/*calcola i elevato ad n*/
{
  int ans, j;
  ans=1;
  for (j=1; j<=n; j++) ans*=i;
  return(ans);
}
```

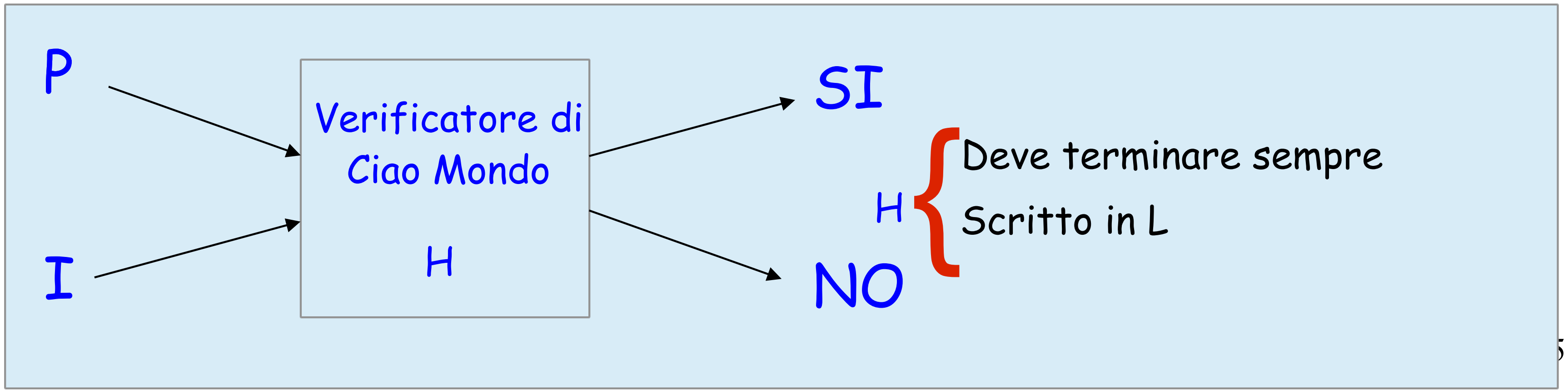
```c
main()
{ int n, total,x,y,z;
  scanf("%d", &n);
  Total=3;
  While(1) {
     for(x=1; x<=total-2; x++)
         for(y=1; x<=total-2; x++) {
            z= total-x-y;
            if (exp(x,n) + exp(y,n)== exp(z,n))
              printf("hello, world\n");
         }
      Total++;
  }
}
```

Per qualunque intero n>2 il programma non troverà alcun (x, y, z) che soddisfi la condizione

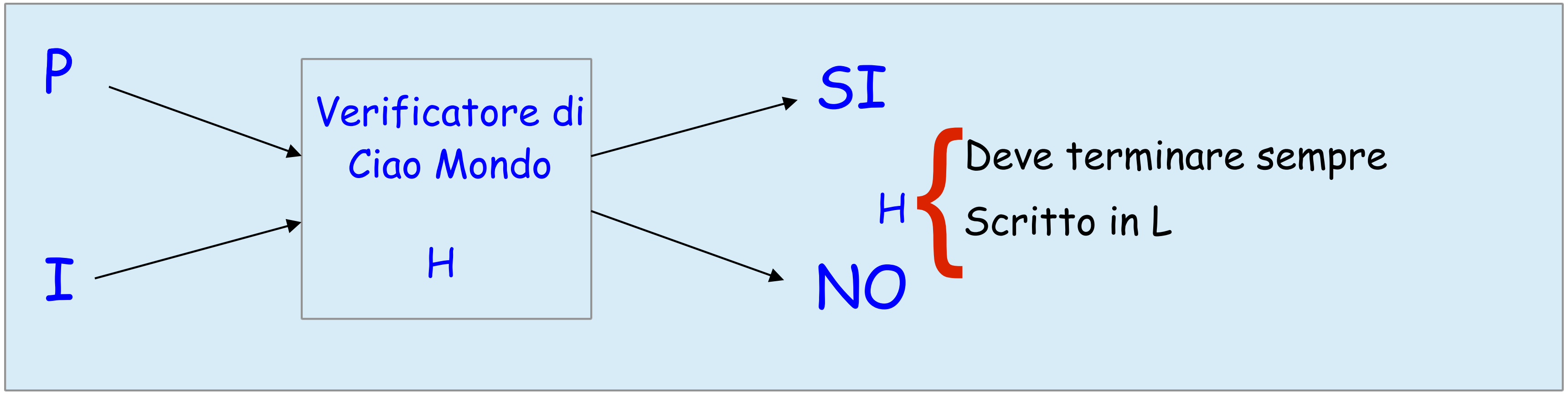
Sulla possibilità di un analizzatore statico H

- ***L*** è un linguaggio di programmazione
- ***P*** è un programma scritto in L (*p potrebbe non terminare*)
- ***I*** è un input per P
- H dice se P con input I stampa "Ciao mondo"

PUO' ESISTERE H????



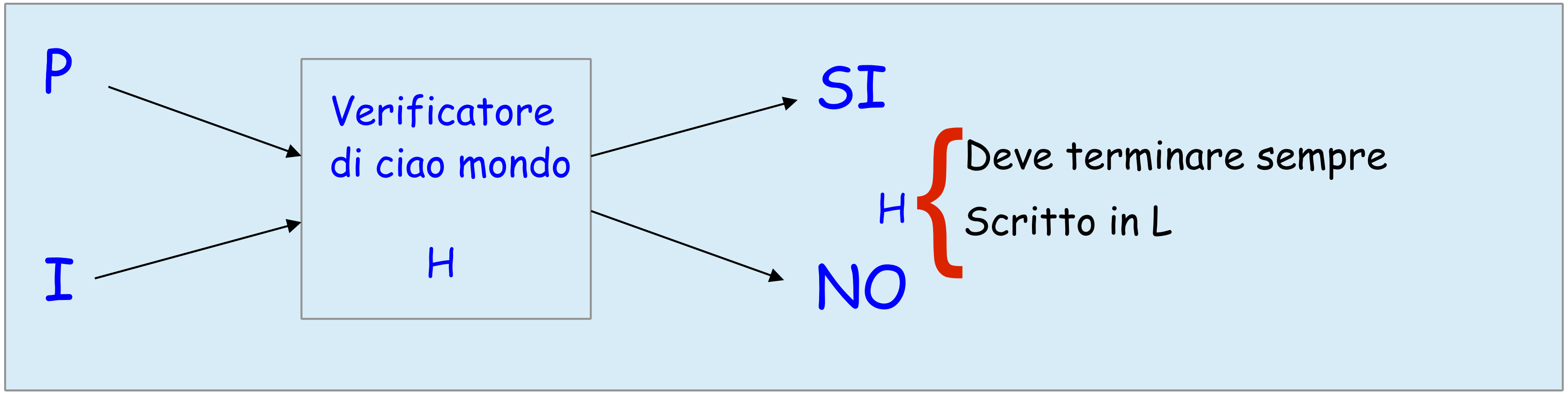
Problemi decidibili



Se un problema ha un algoritmo come H il problema è detto decidibile

Un ipotetico verificatore di "Ciao mondo"

Dimostrazione per assurdo



Assunzioni

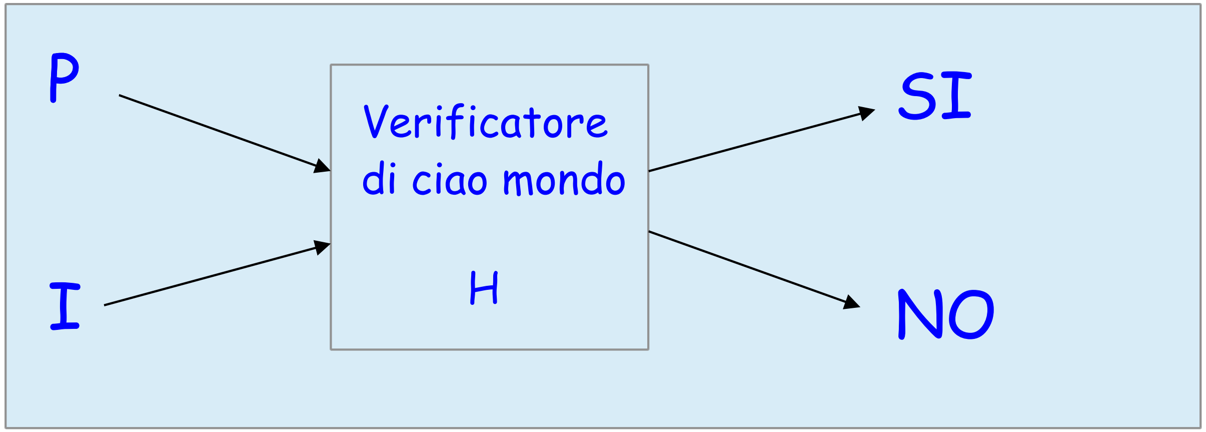
H esiste

H termina sempre

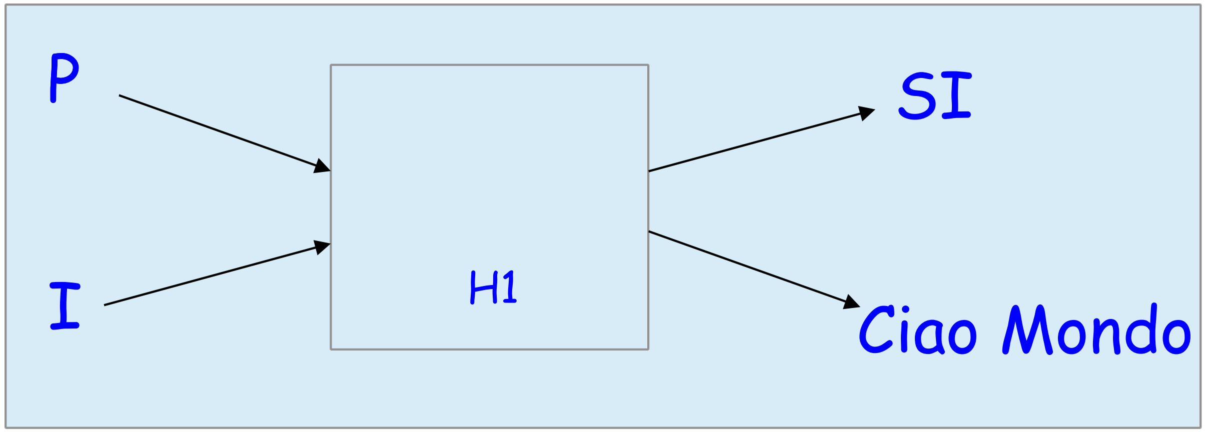
P richiede una stringa I in input

Un ipotetico verificatore di "Ciao mondo"

Dimostrazione per assurdo



Costruiamo H1

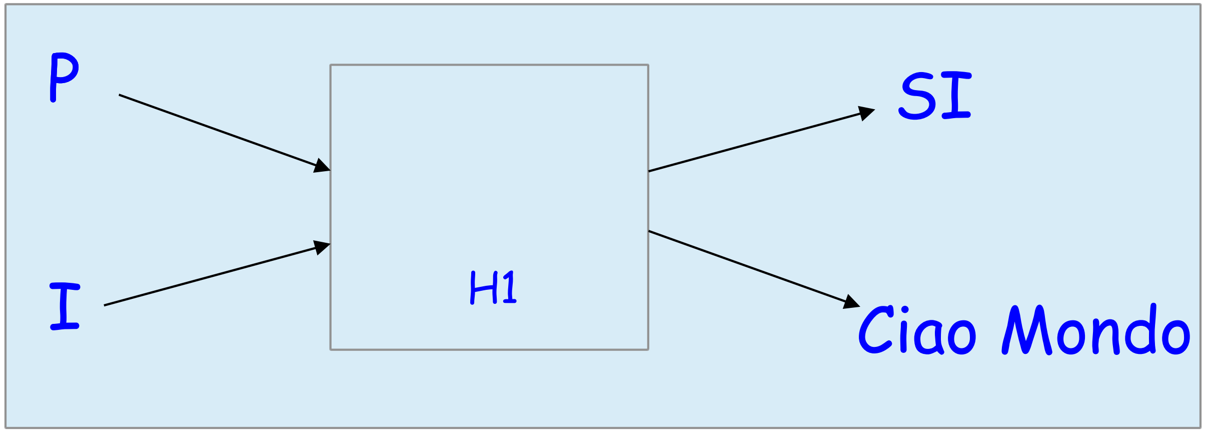


H1 stamperebbe "Ciao Mondo" ogni volta che H stamperebbe NO.

Ovvero ogni volta P non stamperebbe "Ciao mondo"

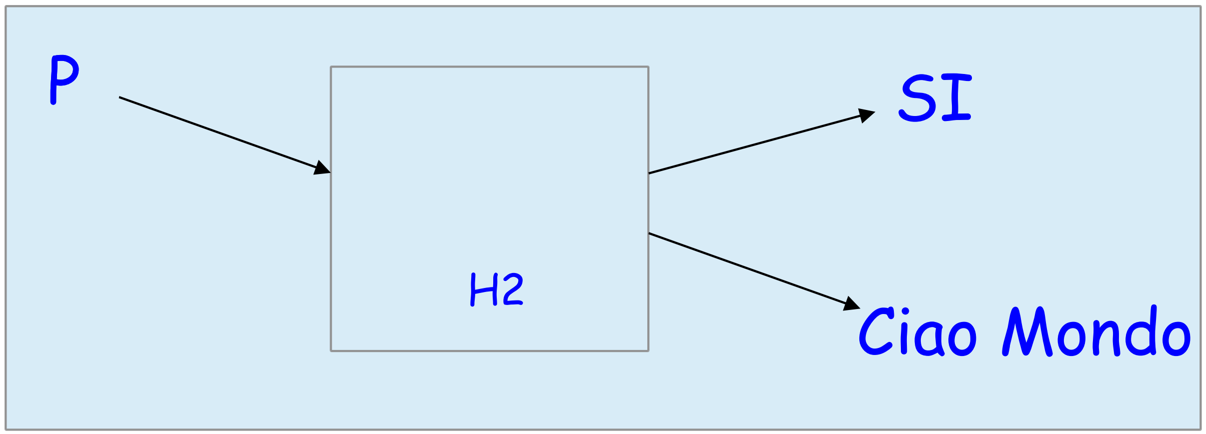
Un ipotetico verificatore di "Ciao mondo"

Dimostrazione per assurdo



H1 stamperebbe "Ciao Mondo" ogni volta che H stamperebbe NO

Limitiamo H1 (Interesse: programmi che ricevono in input programmi)



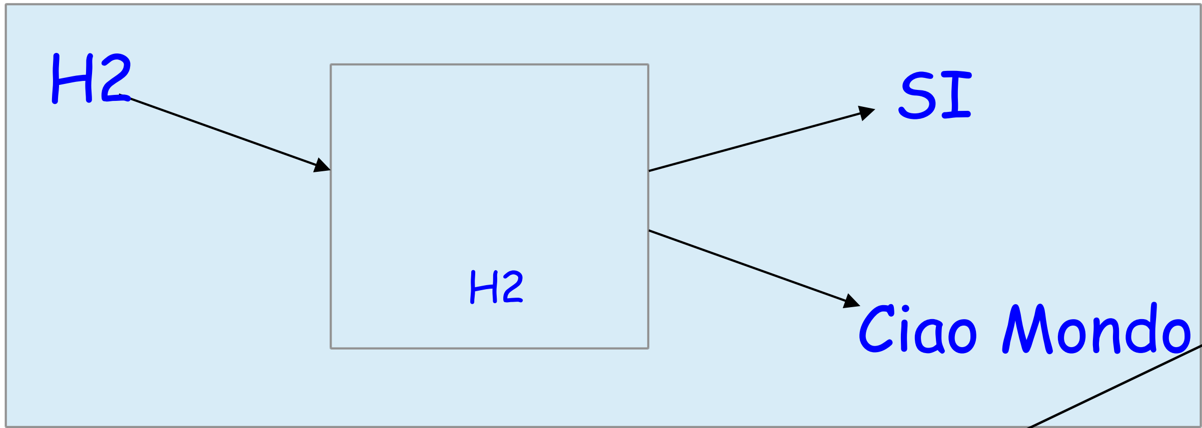
H2 determina che cosa farebbe P se l'input fosse il suo stesso codice

Ovvero cosa farebbe H1 avendo in input P sia come programma sia come dato

Un ipotetico verificatore di "Ciao mondo"

Dimostrazione per assurdo

Cosa fa H2 se riceve in input se stesso?



Ricordiamo che:

Per qualunque P come input,

H2 produce SI se P stampa "ciao mondo", quando gli viene dato se stesso (P) come input

H2 produce "Ciao mondo" se P, dato se stesso come input, non stampa "ciao mondo"

H2 ricevendo se stesso come input produce SI

<u>Supponiamo che H2 produca output SI:</u>

Allora H2, quando riceve il suo input H2 stampa "Ciao mondo"

MA AVREBBE DOVUTO STAMPARE SI

<u>Allora H2 deve produrre "Ciao mondo":</u>

MA AVREBBE DOVUTO STAMPARE SI

Un ipotetico verificatore di "Ciao mondo"

Dimostrazione per a…rdo

Co…a fa H2 se ri… in… e stesso?

H2

Qualunque sia l'output ipotizzato di H2
Possiamo dedurre che produca l'altro

H2 non esiste, H1 nemmeno,
==> H non esiste

… come input,

…ce SI se P stampa … quando gli viene … e stesso (P) come input

… "Ciao mondo" se P, dato se s… come input, non stampa "ciao mondo"

Allora H2, …

MA … ARE …

Allora H2 de… durre "Ci…o mondo":

MA AVREBBE DOVUTO STAMPARE SI

H2 ricevendo se stesso come input produce SI

Il problema della fermata

Non esiste alcuna *procedura di decisione* (nel linguaggio L) capace di verificare se un altro generico programma in L termina, su un generico input

procedura di decisione:
programma che funzioni per

1) argomenti arbitrari
2) termini sempre
3) discrimini gli argomenti che sono soluzione del problema da quelli che non lo sono

Il problema della fermata: siamo sicuri??
(Indecidibilità e linguaggi)

Se prendiamo un altro linguaggio L'?

Assunzioni su L decisamente generiche :

1) Capace di forma condizionale, per discernere i casi nella definizione di H1 e H2

2) Capace di iterazioni o ricorsioni (infinite)

Il problema della fermata non è un fatto contingente, legato al linguaggio L

Problemi di semantica statica: valori indefiniti

- Il valore di un'espressione potrebbe essere <u>indefinito</u>
  - Caso 1: operazione non definita, es., la divisione per zero
    - 4/0 non ha valore
    - L'implementazione si fermerebbe in condizione d'errore
  - Caso 2: non-terminazione
    - f(x) = if x=0 then 1 else f(x-2)
      - Questa è una funzione parziale: non è definita per tutti i possibili argomenti in input
      - Non può essere individuata come tale al momento della compilazione; è un problema di fermata (*halting problem*)
  - Questi due casi sono
    - "Matematicamente" equivalenti
    - Operativamente diversi

Problemi di semantica statica valori indefiniti

- Il valore di un'espressione potrebbe essere <u>indefinito</u>
  - Caso 1: operazione non definita, es., la divisione per zero
    - 4/0 non ha valore
    - L'implementazione si fermereb... ndi... ...ore
  - Caso 2: non-ter...
    -
      - Q... ...rgomenti in
      - ... ...: è un problema

PROBLEMA:

APPARTENENZA DI UNA STRINGA AD UN LINGUAGGIO

  - Questi due ca...
    - "Matematica...nte" equivalenti
    - Operativamente diversi

Perchè devono esistere problemi indecidibili

PROBLEMA:

APPARTENENZA DI UNA STRINGA AD UN LINGUAGGIO

- Il numero di linguaggi diversi su qualunque alfabeto di piu' di un simbolo non è numerabile
- I programmi sono numerabili
  - Ordine per lunghezza
    - Ordine lessicale
- Quindi: Esistono infinitamente piu' problemi (appartenenza di stringa a linguaggio) che programmi

Esistono più funzioni che algoritmi (o programmi)

- Dato il linguaggio L, l'insieme dei programmi che si possono scrivere con L è infinito numerabile:
  - programma: stringa di caratteri codificati con numeri = lungo numero naturale (binario)
  - esiste una funzione di ordinamento totale sui numeri naturali (es. ordinamento <u>lessicografico</u>)
    - quindi la cardinalità dell'insieme dei programmi su L coincide con quella di N
- Consideriamo F insieme delle funzioni N --> {0,1}
  - Teorema (Cantor): F non è numerabile
  - F ha una cardinalità maggiore di quella di N (quella dei numeri reali)
  - Non riuaciamo ad esprimere tutte le funizoni in F con programmi in L

Esistono più funzioni che algoritmi (o programmi)

- <u>Dim</u>: (per abs.) se F fosse numerabile

  allora F = $\{f_j\}$ con j in N

  ⇒ esiste una bigezione con l'insieme B delle sequenze infinite di cifre binarie:

  - ad **ogni** f_j corrisponde una sequenza di cifre binarie: $b_{j,1}, b_{j,2}, b_{j,3}, \ldots$ con $b_{j,i} = f_j(i)$ per i, j in N
  - essendo **B** numerabile si può costruire la matrice:

$b_{1,1}, b_{1,2}, b_{1,3}, \ldots$

$b_{2,1}, b_{2,2}, b_{2,3}, \ldots$ La riga j contiene la sequenza di cifre binarie relativa alla

$b_{3,1}, b_{3,2}, b_{3,3}, \ldots$ funzione j-esima

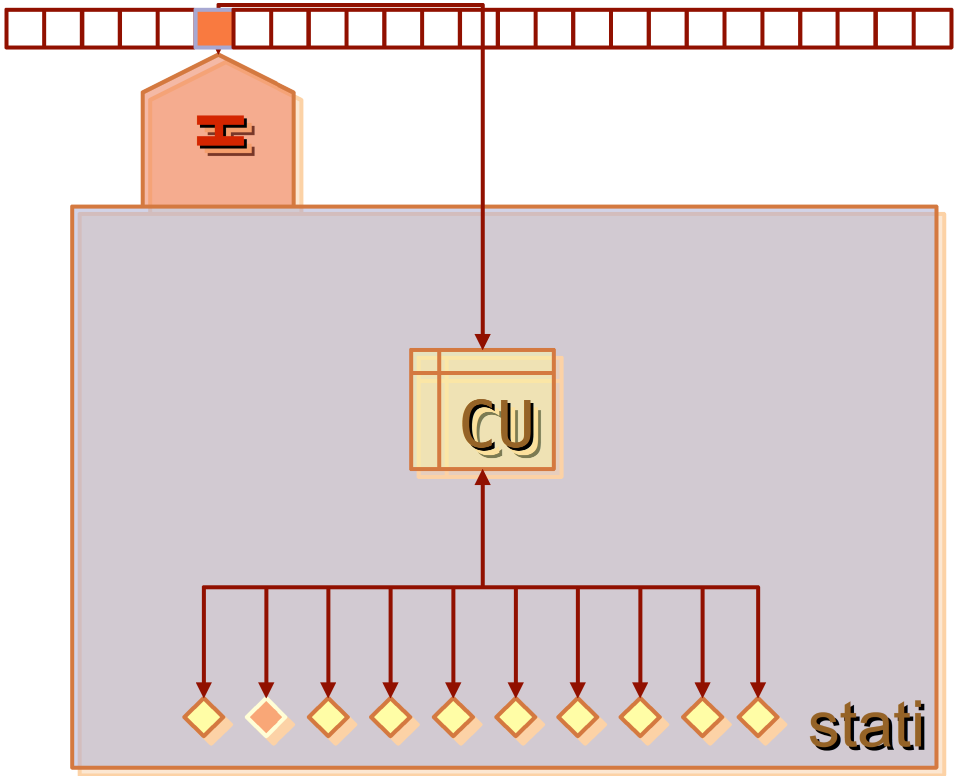
...

la stringa (nuova funzione) ottenuta complementando gli elementi della diagonale $b_{1,1}, b_{2,2}, b_{3,3}, \ldots$ è certo un elemento di B, ma non compare nella matrice, perchè si differenzia almeno in un punto da ogni riga (**assurdo**)

Problemi Indecidibili: alcuni esempi

1) Verificare se un programma calcola una funzione costante
2) Verificare se due programmi calcolano la stessa funzione
3) Verificare se un programma termina per ogni input
4) Verificare se un programma diverge per ogni input
5) Verificare se un programma, dato un input, genererà in fase d'esecuzione un errore /errore di tipo

Macchina di Turing
(il primo linguaggio col quale si è espressa l'indecidibilità del problema della fermata)



- nastro T infinito
- testina H insiste su un elemento di T (legge/scrive)
- unità di controllo CU
  - in base al simbolo letto dalla testina
  - e allo stato corrente (n° finito)
- decide
  - il simbolo da scrivere nella cella
  - lo spostamento del nastro (di max 1 cella)
  - il nuovo stato

Macchina di Turing

$$(Q,\Sigma,\Gamma,\delta,q_{o,}q_{accept,}q_{reject})$$

- Insieme Stati Q
- q_0: stato iniziale ($q_0 \in Q$)
- Alfabeto di lavoro Σ ($\_ \in / \Sigma$)
- Γ: Alfabeto del nastro($\_ \in \Gamma$, $\Sigma \subset \Gamma$)
- $\delta: Q\times\Gamma \rightarrow Q\times\Gamma\times\{L,R\}$: funzione di transizione
- $q_{accept,}$: stato di accept
- q_{reject} : stato di reject

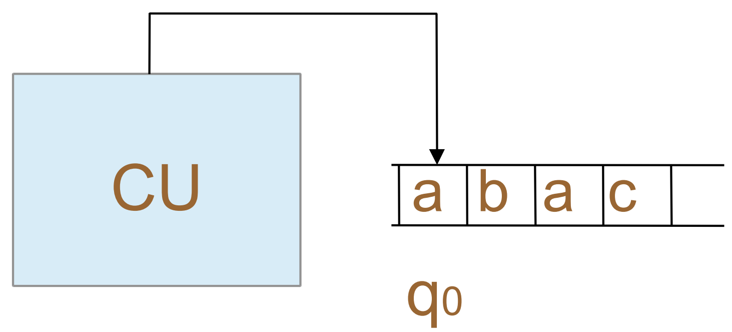
Macchina di Turing

$(Q,\Sigma,\Gamma,\delta,q_{o},q_{accept},q_{reject})$

- Ad ogni istante M occupa uno degli stati in Q

- La testina si trova in un quadrato del nastro contenente un qualche simbolo $\gamma \in \Gamma$:

- La funzione di transizione $\delta:Q\times\Gamma\rightarrow Q\times\Gamma\times\{L,R\}$ dipende dallo stato q e dal simbolo di nastro γ



Macchina di Turing

La funzione di transizione $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$ dipende dallo stato q e dal simbolo di nastro γ

- Il range della funzione di transizione sono triple (q', γ', d) con
  - $q' \in Q$
  - $\gamma' \in \Gamma$ è il simbolo scritto dalla testina sulla cella del nastro su cui la testina si trova ALL' INIZIO della transizione
  - $d \in \{L,R\}$ è la direzione in cui la testina muove un passo

Macchina di Turing

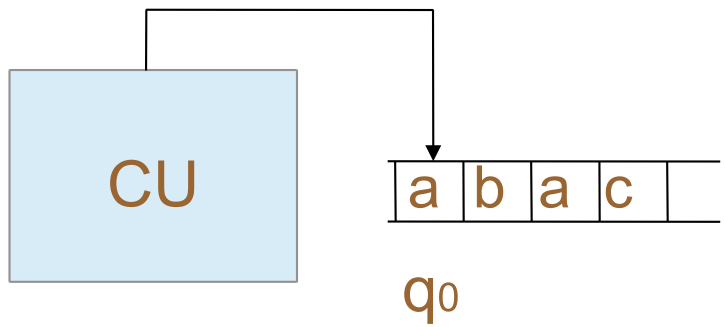
$$(Q,\Sigma,\Gamma,\delta,q_{o},q_{accept},q_{reject})$$

La computazione parte sempre

- da stato iniziale q_0

- con input posizionato sulla parte più a sinistra del nastro: le prime n celle a sinistra, se n è la lunghezza dell'input

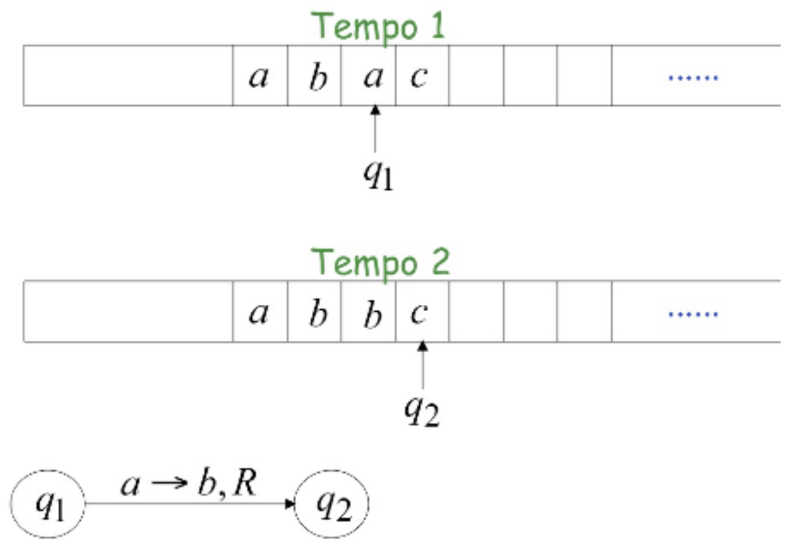
- la testina si trova nella prima cella a sinistra del nastro (cella 0)



Macchina di Turing

$(Q,\Sigma,\Gamma,\delta,q_{o},q_{accept},q_{reject})$

La funzione di transizione



Macchina di Turing

Una Macchina di Turing è una settupla

$$(Q,\Sigma,\Gamma,\delta,q_{o},q_{accept},q_{reject})$$

La computazione termina quando MdT raggiunge:

- Uno stato accettazione q_{accept}: Computazione Accept
- Uno stato rifiuto q_{reject}: Computazione Reject

Macchina di Turing

Stati di arresto

$q_1 \longrightarrow q_{accept}$ Permesso

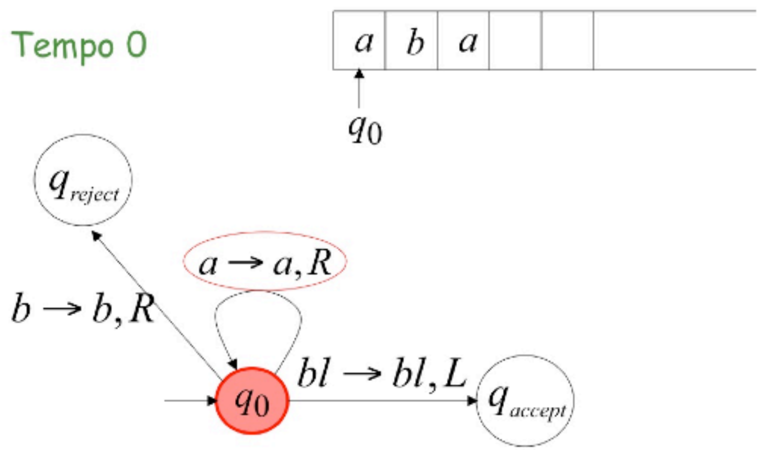
$q_{reject} \longrightarrow q_2$ **Non permesso**

- Gli stati di arresto non hanno archi uscenti
- In uno stato di arresto la computazione termina

Macchina di Turing

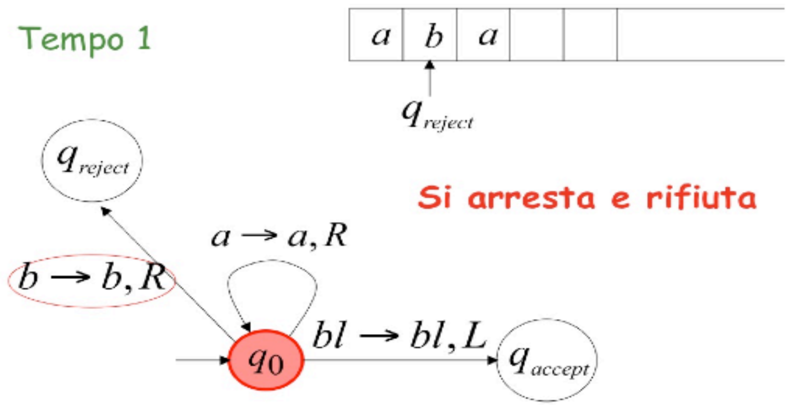
Esempio di stringa non accettata

Tempo 0



Esempio di stringa non accettata

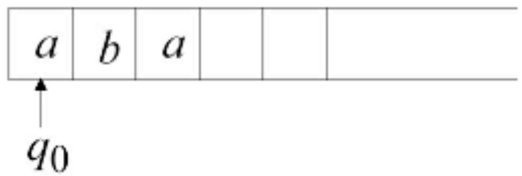
Tempo 1

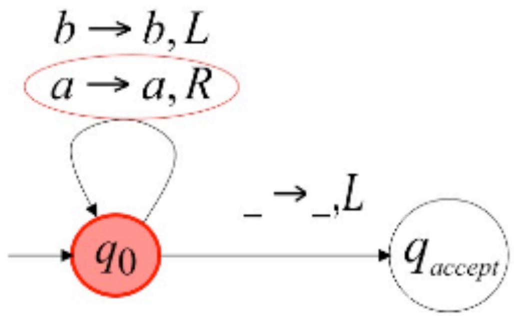


Si arresta e rifiuta

Macchina di Turing

Tempo 0





11

Lo stato q_{accept} non raggiungibile
La computazione non termina. L' input non viene accettato

Calcolabilità

Definizione

La funzione f è **calcolabile** da un linguaggio L sse esiste un programma P, scritto in L, che la calcoli:

per ogni input ***x***, il calcolo di ***P(x)*** termina con output ***f(x)***

(la computazione termina

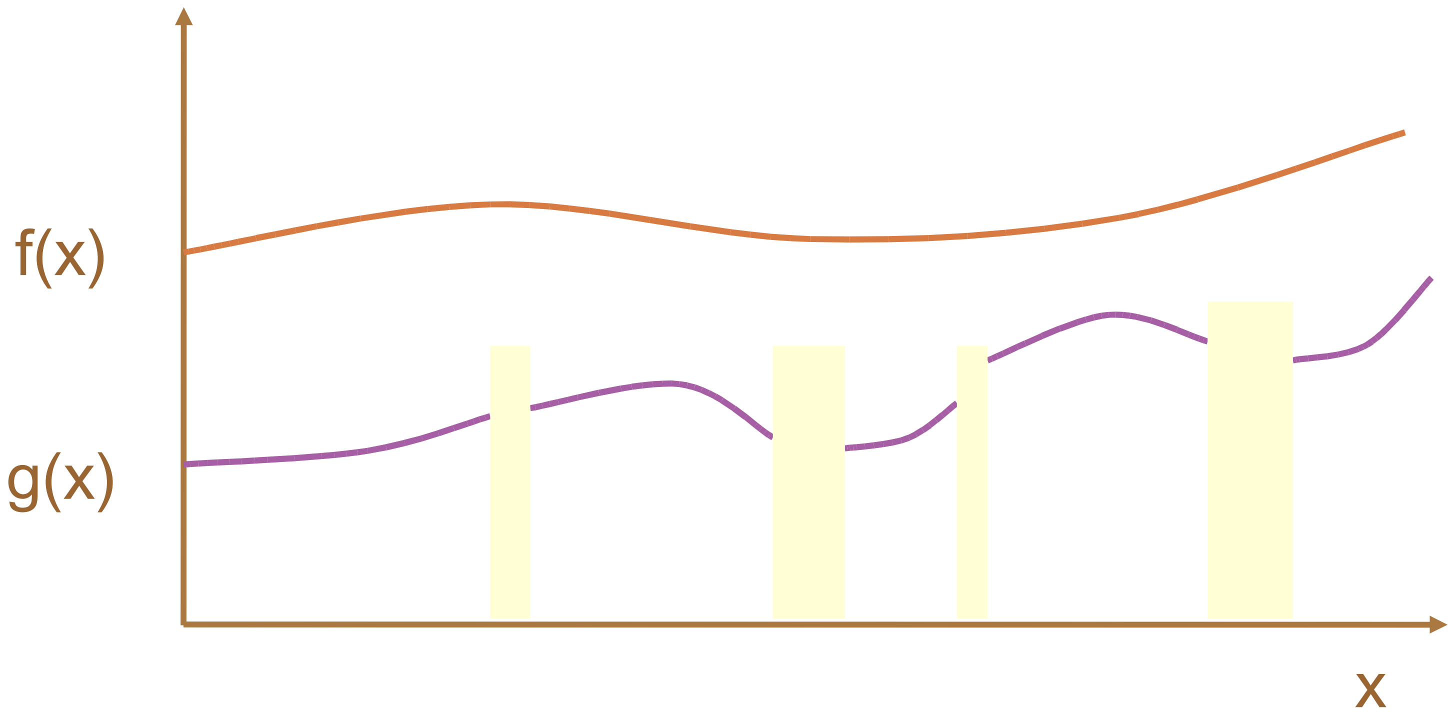
solo per i valori per cui la funzione f è definita)

Funzioni parziali e totali programmi e funzioni

- I programmi definiscono funzioni parziali per due motivi
  - Operazioni parziali (come la divisione)
  - Non terminazione
    - f(x) = if x=0 then 1 else f(x-2)

Funzioni parziali e totali grafici

- Funzione <u>totale</u>: f(x) ha un valore per ogni x
- Funzione <u>parziale</u>: g(x) non ha un valore per qualche x



- Grafico di f = { (x,y) | y = f(x) }
- Grafico di g = { (x,y) | y = g(x) }

Formalismi per la calcolabilità

- Indipendente dal linguaggio
- **Macchina di Turing** (MdT)
  - anni 30: inventata da Alan M. Turing
  - Semplice ma potente:
    - Puo' esprimere computazioni come quelle necessarie per scrivere H1 & H2 da H
    - Esiste una MdT che funge da interprete di tutte le altre (un semplice calcolatore)
  - **Turing-completezza**:
    MdT simula tutti gli altri formalismi (anche i più complessi):
    - **funzioni ricorsive** *generali* di Church-Gödel-Kleene
    - Lambda-calcolo
    - Linguaggi di programmazione esistenti (Turing-completi - Teoremi)

Calcolabilità: punti cardine
(risultato definitivo)

- Alcune funzioni sono calcolabili, alcune non lo sono, alcune lo sono parzialmente
  - Problema della fermata
  - ……
- Turing-completezza:
  - In ciascun formalismo per esprimere algoritmi si puo' scrivere un interprete per un altro formalismo
  - Tutti i formalismi per esprimere algoritmi possono calcolare le stesse funzioni delle MdT
  - I risultati di indecidibilità possono essere espressi dicendo che esistono funzioni non calcolabili con una MdT

Calcolabilità: punti cardine

- Implementazione dei linguaggi di programmazione
  - Può registrare un errore se il risultato del programma è indefinito a causa della divisione per zero,
    - operazione di base non definita
  - Non può riportare errori se il programma non termina

Espressività dei linguaggi

- Flessibilità d'uso
- Pragmatica
- Principi di astrazione
- .....

MANCANZA DI ACCORDO SU COME CONFRONTARE QUESTI ASPETTI