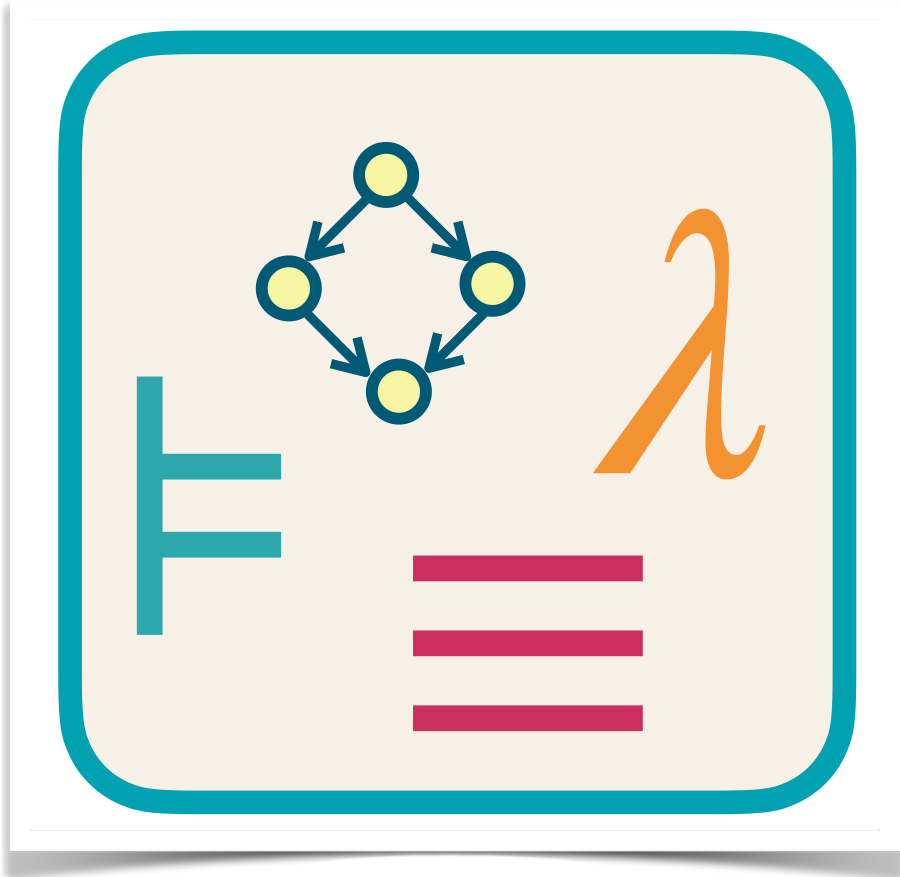




Linguaggi di Programmazione

Roberta Gori

L'indcidibilità in breve

Molte proprietà dei programmi sono indecidibili

Problemi come:

Equivalenza tra programmi,

Dato un programma e un input il codice stampa “1”.

Cosa vuol dire indecidibile?

Per qualche programma e' facile decidere

Per ogni input alla fine $x = 1$?

$C \triangleq$

$x := 1;$



Cosa vuol dire indecidibile?

Per qualche programma e' facile decidere

Per ogni input alla fine $x = 1$?

$c \triangleq$

```
while (n>1) {
```

```
    n := n+1;
```

```
    x := 0;
```

```
}
```

```
x := 1;
```



Cosa vuol dire indecidibile?

Per qualche altro programma non e' facile...

Per ogni input alla fine $x = 1$?

$c \triangleq$

```
while (n>1) {  
    if (even(n)) { n := n/2; }  
    else { n:= 3n+1; }  
}  
x := 1;
```

Definizione di problema decisionale

Un problema decisionale (ovvero che richiede una risposta sì/no) si dice **indecidibile** se non esiste alcun algoritmo che riesca a risolverlo.

N.B: Potrebbero anche esistere algoritmi in grado di risolvere alcuni casi specifici di un problema indecidibile: l'indecidibilità sta a significare che non è possibile trovare un algoritmo generale.

Problemi come linguaggi

Gli input dei problemi decisionali sono **stringhe** costruite su un certo alfabeto Σ ,

Ogni problema decisionale puo' essere visto come

Il sotto insieme degli input (**Linguaggio**) sui quali la risposta al problema e' positiva

$L = \{w \in \Sigma^* \mid \text{la risposta al problema su input } w \text{ è sì}\}$

Esistenza di problemi indecidibili

I problemi che non si possono risolvere esistono, e costituiscono la maggior parte dei problemi.

Mostriamo che, anche limitandoci alla classe dei problemi decisionali, non tutti sono risolvibili

Per farlo, mostriamo che i problemi sono "più numerosi" degli algoritmi che possono risolverli.

Idea della Prova

Mostriamo che **non esiste una funzione iniettiva** dall'insieme dei possibili algoritmi all'insieme dei problemi decisionali

Dal momento che possiamo passare dai **problemi decisionali** ai **linguaggi**

Facciamo vedere che la cardinalita' dei **programmi** e' inferiore a quella dei **linguaggi**

La cardinalita' dei programmi

Sia T l'insieme dei programmi.

Sia T_i l'insieme dei file di testo che hanno lunghezza i per ogni $i \in \mathbb{N}$.

Dato che l'insieme dei caratteri possibili è finito, T_i è finito per ogni i .

Quindi, dato che un'unione numerabile di insiemi finiti è numerabile,

$$|T| = \left| \bigcup_{n=0}^{\infty} T_n \right| = |\mathbb{N}|$$

La cardinalita' dei linguaggi

- La cardinalita' di tutte le possibili stringhe su un alfabeto e' $|\mathbb{N}|$
- Un linguaggio e' un qualsiasi sottoinsieme di stringhe quindi la cardinalita' dei possibili linguaggi L e'

$$\mathcal{P}(\mathbb{N}) = 2^{|\mathbb{N}|}$$

$$|\textit{Possibili linguaggi}| = \mathcal{P}(\mathbb{N}) = 2^{|\mathbb{N}|} > |\mathbb{N}| = |T|$$

Non puo' esistere una funzione
iniettiva da problemi decisionali ad algoritmi

Problemi indecidibili

- Tra tutti i problemi indecidibili ne esiste in particolare uno:
il problema della fermata
- Andiamo a dimostrare la sua indecidibilità direttamente

Il problema della fermata

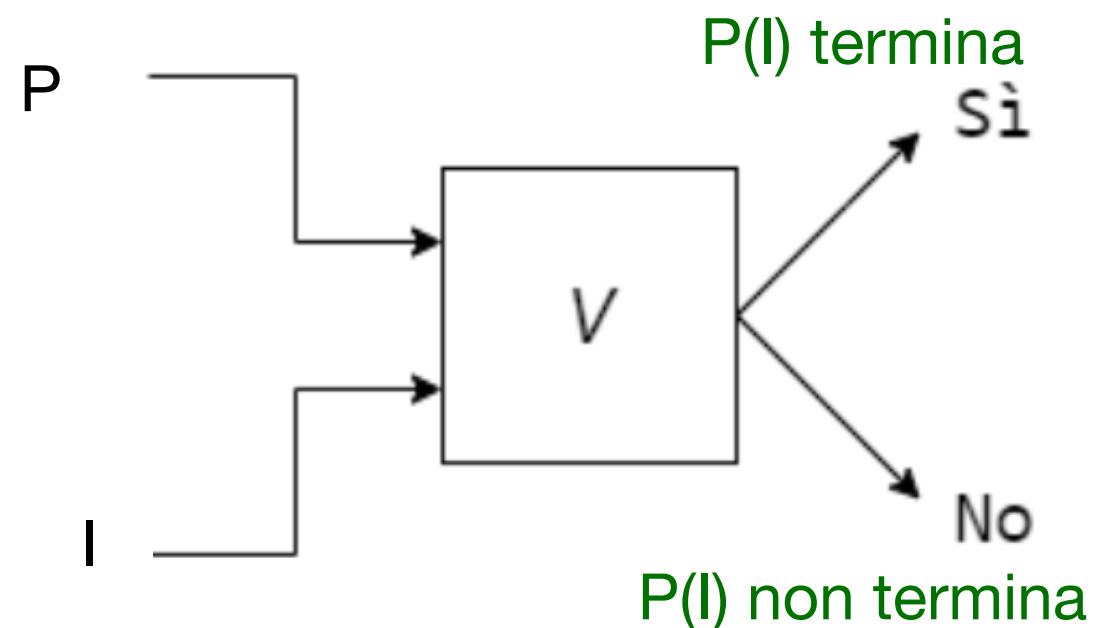
Vogliamo definire il seguente algoritmo: presi un algoritmo P e l'input I , l'algoritmo deve determinare se l'esecuzione di P su l'input I termina (si ferma) in un numero finito di passi.

Questo problema e' indecidibile!

Dimostrazione

Procediamo per assurdo:

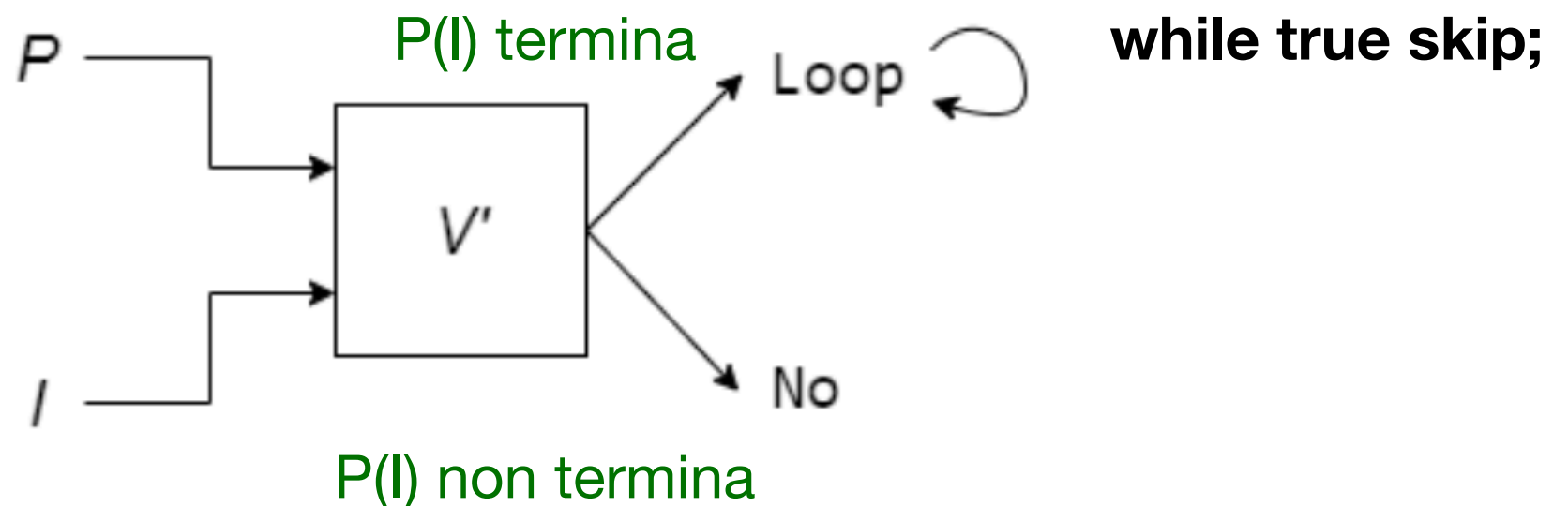
supponiamo che esista un algoritmo verificatore V che, dati in input un algoritmo P e l'input I , stampi sì e termini se P termina con l'input I , altrimenti stampi no e termini.



Dimostrazione

Ora, apportiamo delle modifiche successive all'algoritmo V , modifiche semplici che possono essere fatte su ogni algoritmo:

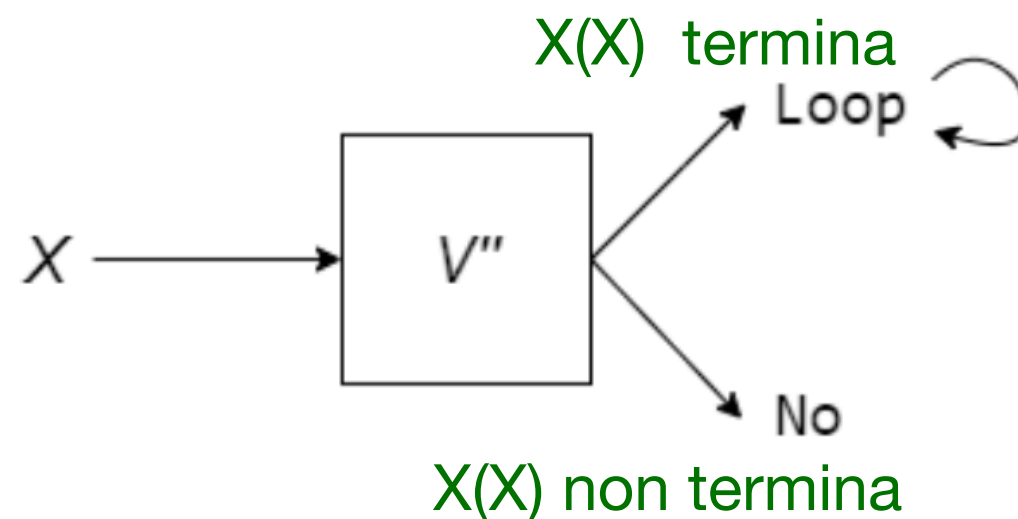
Modifichiamo V in V' in modo che, se $P(I)$ termina, invece di stampare sì entri in un loop infinito.



Dimostrazione

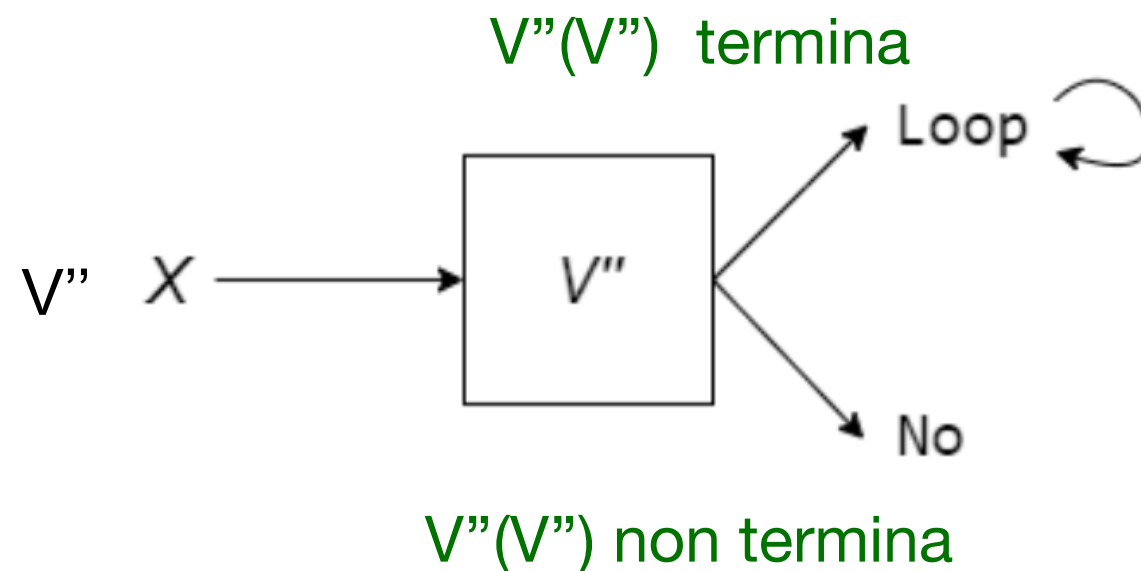
Ora, apportiamo delle modifiche successive a V ,
modifiche semplici che possono essere
fatte su ogni algoritmo:

Modifichiamo V' in V'' in modo che accetti un solo input X
che funge sia da programma che da input del programma.



Dimostrazione

Ma cosa succede se diamo a V'' come input V'' ?

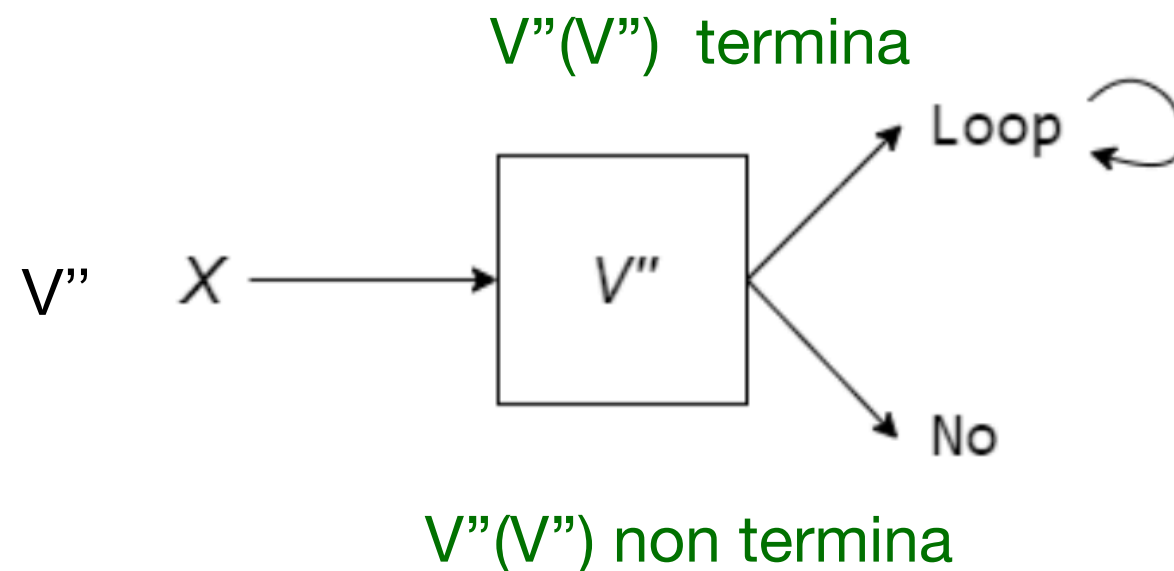


Ci sono due possibilità:

- Se $V''(V'')$ termina allora V'' entra in un loop infinito, quindi $V''(V'')$ non termina.

Dimostrazione

Ma cosa succede se diamo a V'' come input V'' ?



Ci sono due possibilità:

- Se $V''(V'')$ non termina allora V'' stampa "no" e termina quindi $V''(V'')$ termina.

Contraddizione: V'' non esiste e di conseguenza neanche V !

La riduzione

Abbiamo appena dimostrato che il problema della fermata è indecidibile

Come facciamo a dimostrare che altri problemi sono indecidibili?

Abbiamo due opzioni:

- Come abbiamo fatto prima: procedere per assurdo o
- Possiamo mostrare che se il nuovo problema fosse risolvibile, allora lo sarebbe anche il problema della fermata

Riduzione!

Riduzioni

Dato un problema P , di cui vogliamo dimostrare l'indecidibilità, e un problema I , che sappiamo essere indecidibile, la riduzione consiste in questi passaggi:

1. Supponiamo che esista un algoritmo V che verifichi il problema P , ovvero tale che, dato un input

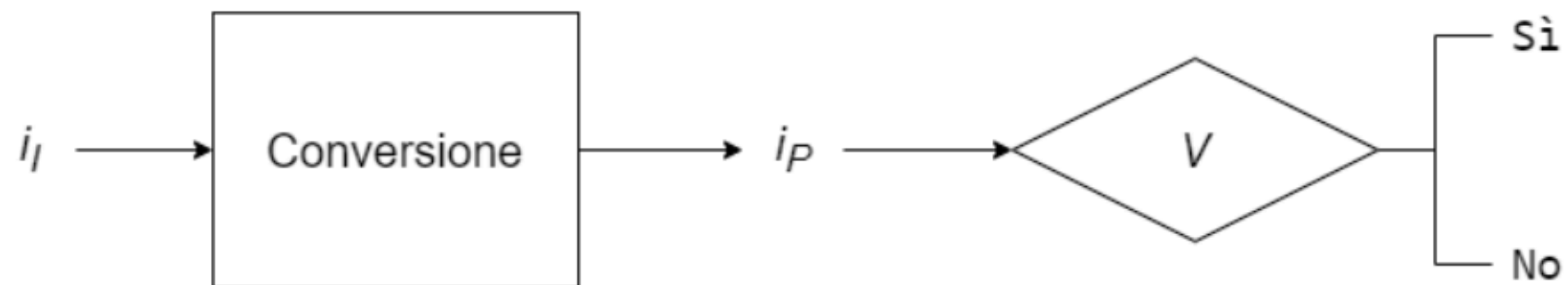
I_P stabilisca se $I_P \in P$

2. Convertire gli input I_I del problema I in input I_P del problema P in modo che

$$I_I \in I \Leftrightarrow I_P \in P$$

3. Dato che abbiamo un algoritmo in grado di decidere il problema P e il problema I è stato ridotto al problema P , concludiamo che I è decidibile e otteniamo una contraddizione

La riduzione



Schema della dimostrazione di indecidibilità per riduzione.

Esempio

Mostriamo che il seguente problema P è indecidibile:

Dare un algoritmo che presi un programma P e un input, determini se quando eseguo P sull'input ottengo in output come primi caratteri la stringa "Hello, world!"

è sufficiente trovare un modo di trasformare tutti gli input di un problema indecidibile I in input del problema P in modo che tale che

$$I_I \in I \Leftrightarrow I_P \in P$$

Riduzione al problema della fermata

Dato un programma Q input dell'algoritmo eseguiamo queste modifiche a Q :

Rimuoviamo tutte le istruzioni di output da Q e aggiungiamo l'istruzione `printf("Hello, world!")` alla fine di Q , ottenendo Q' .

Dato che le istruzioni di output non influiscono sull'elaborazione,
 Q' termina iff Q termina

dato che l'istruzione `printf("Hello, world!")` è l'unica istruzione di output alla fine di Q' ,

Q' stampa "Hello, world!" sse Q' termina.

Quindi

Q' stampa "Hello, world!" sse Q termina

Assurdo!

Teorema di Rice

Ogni proprietà **non banale** dei linguaggi ricorsivamente enumerabili è indecidibile

- Per esempio e' indecidibile dire se due programmi C producano lo stesso risultato in corrispondenza degli stessi dati di ingresso.