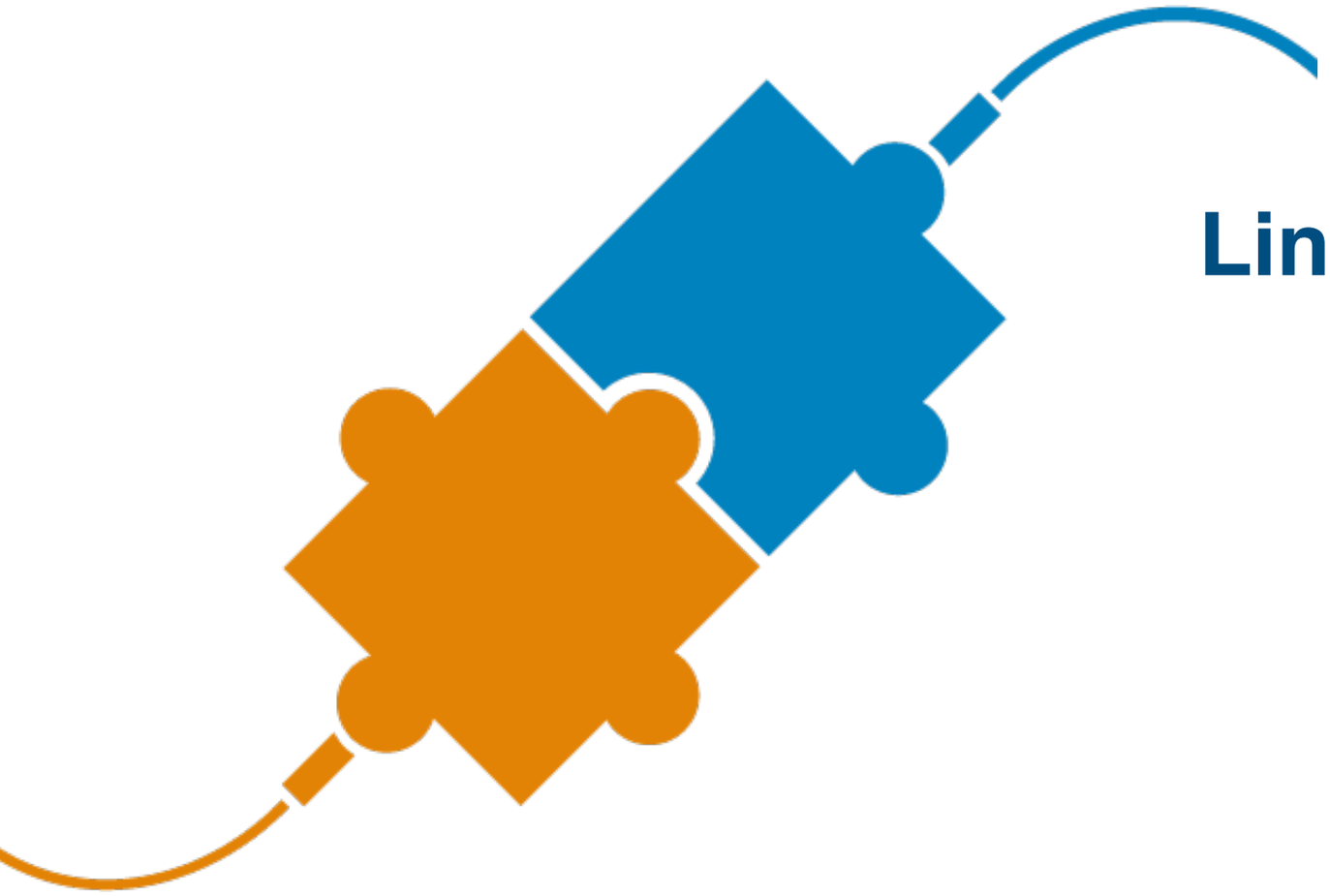


Linguaggi di Programmazione

Roberta Gori

L'indecidibilita' in breve

Quali problemi possiamo risolvere?

Quali problemi possiamo affrontare con i nostri programmi?

.

Quali problemi possiamo sperare di risolvere?

.

Non Tutti !!!!!

Problemi indecidibili

Un problema decisionale (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 che valga in tutti i casi.

Un po' di storia

Alan Turing attraverso la sua macchina dimostrò che non esiste un algoritmo generale in grado di decidere, per ogni possibile input, se un dato programma terminerà o meno.

Questa dimostrazione implica che ci sono limiti intrinseci alla capacità di risolvere certi problemi tramite algoritmi.
Esistono problemi indecidibili, ossia problemi per i quali non è possibile creare un algoritmo che fornisca sempre una risposta corretta in un tempo finito.

Il teorema di incompletezza di Gödel fornisce un ulteriore fondamento teorico alla nozione di limiti intrinseci alla risoluzione algoritmica di alcuni problemi, contribuendo alla comprensione dei problemi indecidibili in informatica.

Esistenza di problemi indecidibili

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.

Problemi come linguaggi

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

I problemi possono essere visti come dei linguaggi:

il linguaggio di un problema P è

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

Un problema e' identificato dall'insieme di stringhe su cui la soluzione e' si

Idea della Prova

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

Dal momento che possiamo passare dai problemi decisionali ai linguaggi

Mostriamo che la cardinalita' dei programmi e' inferiore a quella dei linguaggi

La cardinalita' dei programmi

Sia T_i l'insieme dei file di testo di lunghezza *i* caratteri 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,

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

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'

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

$$|L| \geq \mathcal{P}(\mathbb{N}) = 2^{|\mathbb{N}|} > |\mathbb{N}| = |T|$$

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

Il problema della fermata

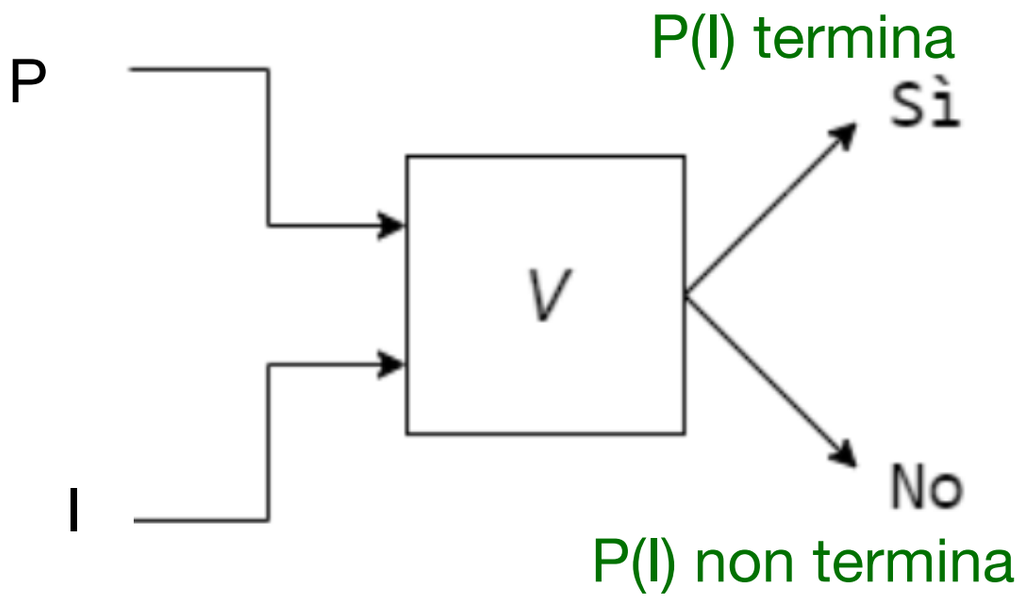
Vogliamo definire il seguente algoritmo: presi un algoritmo P e l'input I, si determini se l'esecuzione di P su l'input I termina in un numero finito di passi.

Questo problema e' indecidibile!

Dimostrazione

Procediamo per assurdo:

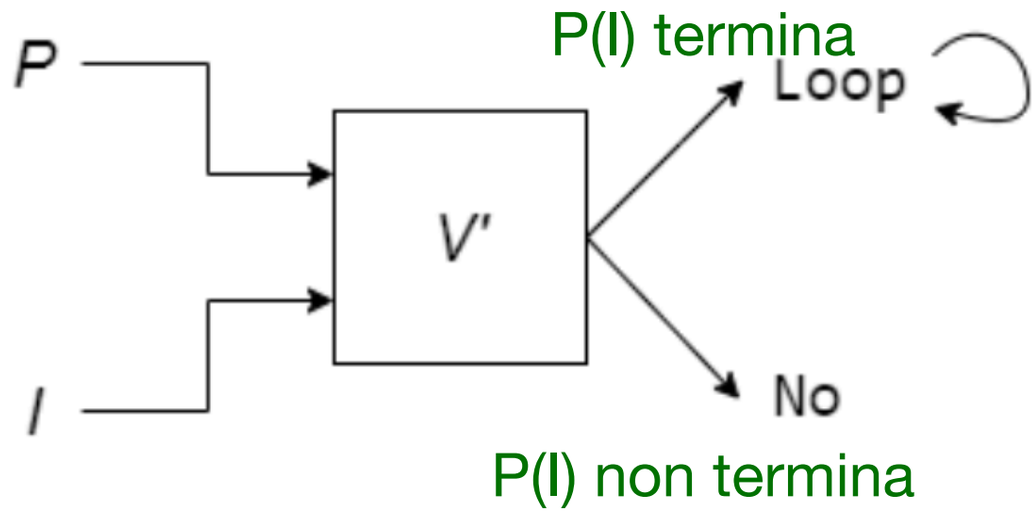
supponiamo che esista un algoritmo verificatore V che,
dati in input un algoritmo P e l'input I,
stampi sì e termini se P termina con l'input I,
altrimenti stampi no e termini.



Dimostrazione

Ora, apportiamo delle modifiche successive a V, 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.

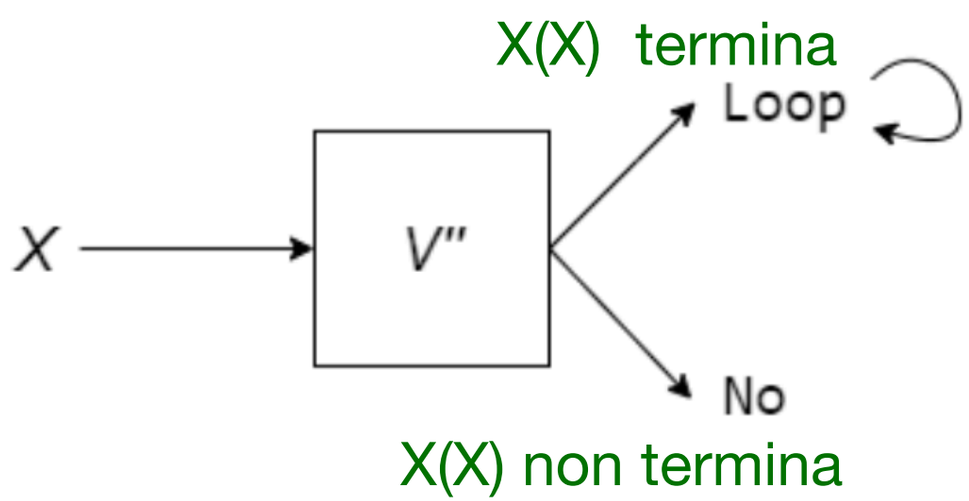


while (true);

Dimostrazione

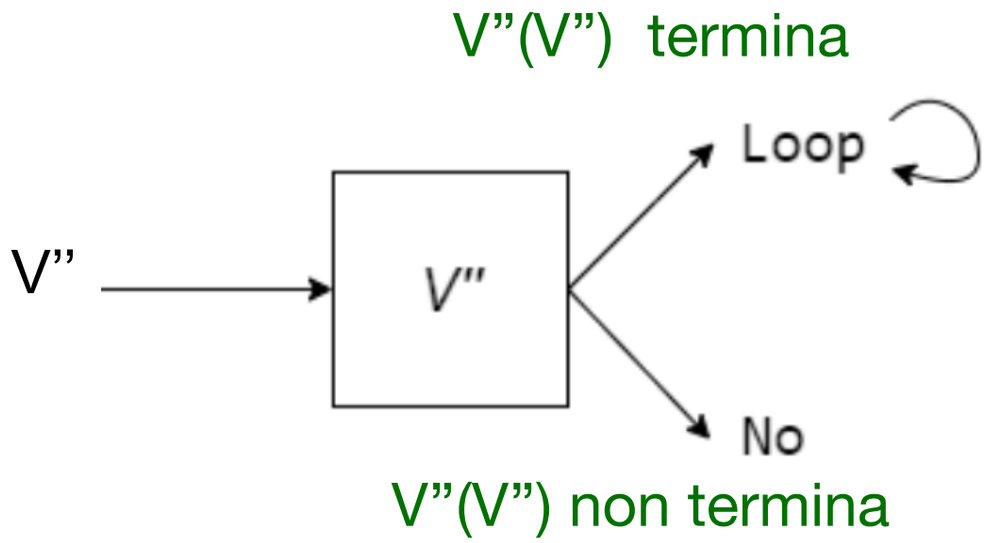
Ora, apportiamo delle modifiche successive a V ,
che possono essere fatte su ogni algoritmo:

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 ″?

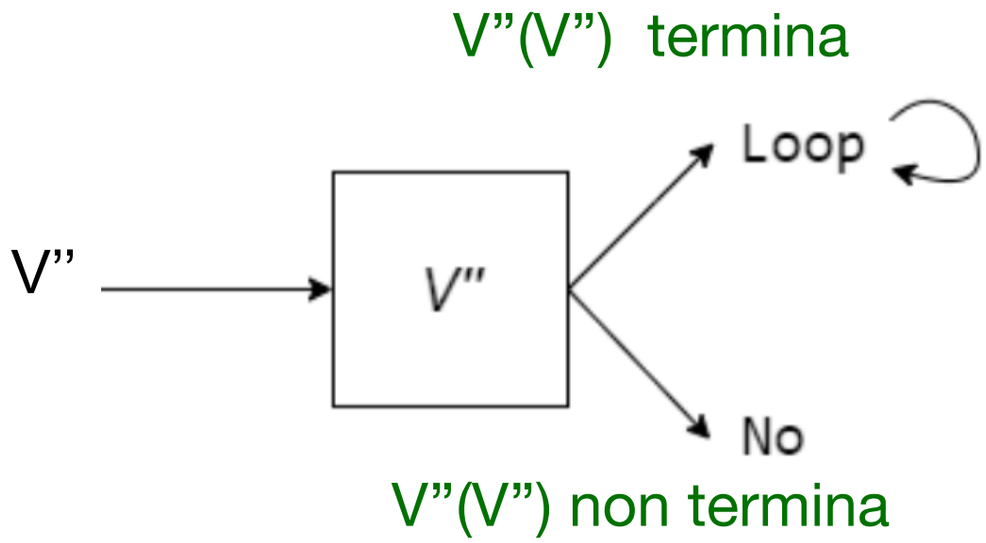


Ci sono due possibilità:

- 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 di questi passaggi:

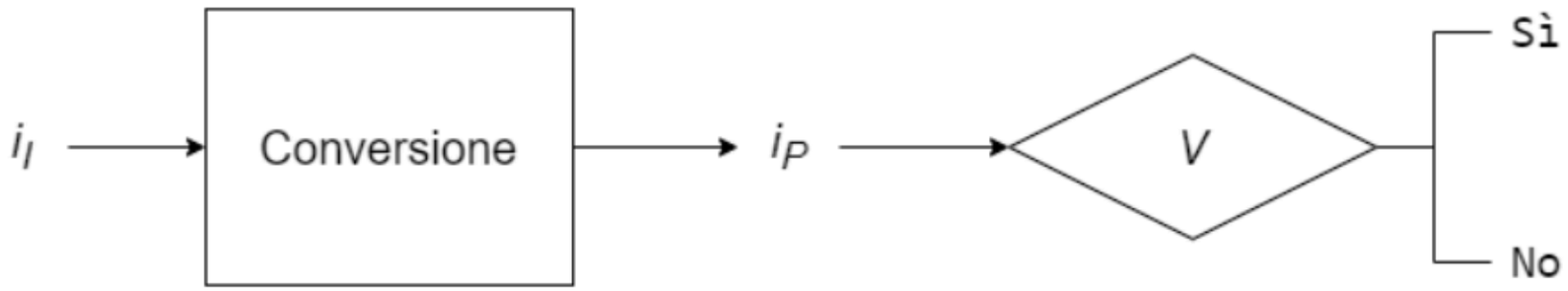
1. Supponiamo che esista un algoritmo V che verifichi il problema P, ovvero tale che, dato un input I_P $\text{stabilisca se } I_P \in P$

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 V è 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:

Determinare se un determinato programma di input, scritto in linguaggio C, stampa 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 di I 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!

Per alcuni programmi…

```
print(“Hello World”);
```

Per altri programmi…

```
while (n>1) {
    n = n+1;
}
print(“Hello World”);
```

Ma per questo programma?

```
while (n>1) {
    if (even(n)) { n = n/2; }
    else { n = 3n+1; }
}
print(“Hello World”);
```

A partire dal 2020, la congettura di Collatz è stata verificata al computer per tutti i valori di partenza fino a $268 \approx 2.95 \times 1020$.

Teorema di Rice

Ogni proprietà non banale (per cui esiste almeno un programma che la soddisfa e uno che non la soddisfa) di programmi scritti in linguaggi Turing equivalenti è indecidibile

- Per esempio e' indecidibile risolvere il problema di determinare per due programmi qls producano lo stesso risultato in corrispondenza degli stessi dati di ingresso.

- Per esempio e' indecidibile risolvere il problema di determinare se un programma non produce errori