5. LINGUAGGI DECIDIBILI

I **problemi di decisione** sono problemi che hanno come soluzione una risposta SI o NO.

Esempi:

- PRIMO: Dato un numero *x*, *x* è primo?
- CONNESSO: Dato un grafo *G*, *G* è connesso? Si ricordi che un grafo di dice *connesso* se per ogni coppia di nodi esiste un cammino che li collega all'interno del grafo.
- ACCETTAZIONE DI UN DFA: Dato un DFA *ß* e una stringa *w*, l'automa *ß* accetta *w*?

Se si vuole ***risolvere un problema di decisione*** utilizzando una MdT, in qualche modo ***occorre trasformare il problema in un linguaggio***: questo perché le MdT sono essenzialmente accettatori di linguaggi.

Ricorda: l'input per una MdT è sempre una stringa. Se vogliamo dare in input altri oggetti, questi devono essere codificati come stringhe: ciò rappresenta il primo step per effettuare questa trasformazione. Qualsiasi oggetto, infatti, può essere codificato come stringa.

Rappresenteremo, quindi, i ***problemi di decisione mediante linguaggi***.

Esempi:

- Il linguaggio che rappresenta il problema "PRIMO" è

$$P = \{\langle x\rangle \mid \text{x è un numero primo}\},$$

dove *⟨x⟩* denota una rappresentazione sotto forma di stringa su un alfabeto Σ dell'oggetto *x* (*codifica*).

Utilizzeremo sempre la notazione **⟨x⟩** per indicare una tale rappresentazione. Quindi, se *x* è un intero, ed usiamo come alfabeto Σ = {0, 1}, denoteremo con ⟨x⟩ una codifica binaria che rappresenta tale intero.

Nota. ⟨x⟩ ∈ *P* **se e solo se** PRIMO ha risposta SI su input *x*.

- Il linguaggio che rappresenta il problema "CONNESSO" è

$$A = \{\langle G\rangle \mid \text{G è un grafo connesso}\},$$

dove *⟨G⟩* denota una "ragionevole" codifica di *G* mediante una stringa su un alfabeto Σ.

Dunque, se *G* è connesso allora ⟨*G*⟩ ∈ *A* e la stringa viene accettata. Se *G'* non è connesso, allora ⟨*G*⟩ ∉ *A* e la stringa non viene accettata.

Per rappresentare un grafo, possiamo prendere Σ = {0, 1, …, 9, (,), ,} e ⟨G⟩ = ({1, 2, 3}, {(1, 2), (2, 3), (3, 1)}), cioè una sequenza in cui il primo elemento è l'insieme dei vertici ed il secondo elemento è un insieme di *edge*, che sono tutti gli edge del grafo. Questa è la rappresentazione che useremo.

Un modo alternativo per rappresentare un grafo è prendere Σ = {0, 1, (,), #} e codificare *G* mediante una stringa (1#10#11)#((1#10)#(10#11)#(11#1)).

Sia $A = \{\langle G\rangle \mid \text{G è un grafo connesso}\}$ un linguaggio di stringhe che rappresentano grafi connessi (non orientati). Si ha che ⟨G⟩ ∈ *A* se e solo se G è istanza SI per CONNESSO. Risolvere CONNESSO equivale a decidere il linguaggio *A*.

In questo modo esprimiamo un problema computazionale in termini di riconoscimento di un linguaggio (cioè, l'insieme delle codifiche di istanze SI per il problema).

Esempio:

Si ha in input un grafo G. Un modo per verificare se G è connesso è il seguente:

1. Seleziona un nodo di G e marcalo;
2. Ripeti finché si trovano nuovi nodi da marcare:
   2.1. Per ogni nodo *v* in G, se *v* è connesso ad un nodo marcato, allora marcalo;
3. Se tutti i nodi risultano marcati allora accetta, altrimenti reject.

Vogliamo convincerci che il risultato precedente è realizzabile mediante una MdT. Quindi, vogliamo una MdT che riconosce l'insieme $A = \{\langle G\rangle \mid \text{G è un grafo connesso}\}$.

Il grafo può essere rappresentato mediante due liste: la lista dei *nodi* (numeri naturali) e la lista degli *edges* (coppie di numeri). Ad esempio, abbiamo il grafo ⟨G⟩ = ({A, B, C, D}, {(A, B), (A, D), (B, C), (C, D)}).

Nota. Non specifichiamo l'alfabeto (binario, decimale, …). Ignoriamo i dettagli non importanti dell'implementazione.

Se facciamo questo, bisogna controllare prima di tutto che l'input sia:

- una lista di nodi (digit) senza ripetizioni;
- una lista di coppie di nodi (digit presenti nella lista precedente).

Un'implementazione può essere la seguente:

1) Seleziona un nodo di G e marcalo: Marca il primo nodo sulla lista (ad esempio, con un · sul digit più a sinistra);

2) Ripeti finché i nuovi nodi sono marcati:

2.1) Per ogni nodo v in G, se v è connesso ad un nodo marcato, allora marcalo:

2.1.1) Sottolinea il primo nodo n_1 senza · sulla lista (sottolineando il digit più a inistra);

2.1.2) Sottolinea il primo nodo n_2 con · sulla lista;

2.1.3) Cerca se esiste edge (n_1, n_2): se SI, allora marca n_1 con · e vai a 2); se NO, sia n_2 il successivo nodo con · sulla lista, sottolinealo e vai a 2.1.3)

3) Se tutti i nodi risultano marcati allora accetta, altrimenti reject: Scorri la lista dei nodi: se tutti i nodi hanno · allora accetta, altrimenti reject.

5.1 DECIDIBILITÀ

L'obiettivo è analizzare i *limiti* della risoluzione di problemi mediante algoritmi: ne studieremo il potere computazionale nella soluzione dei problemi, e proveremo che esistono problemi che possono essere risolti mediante algoritmi ed altri no.

Nell'esempio "PRIMO", abbiamo espresso un problema computazionale come un problema di *riconoscimento di un linguaggio* (cioè, l'insieme delle codifiche di istanze SI per il problema): dunque, risolvere "PRIMO" equivale a decidere un linguaggio $P = \{\langle x\rangle \mid \text{x è un numero primo}\}$.

Sia data in input una stringa *w* ad una MdT *M*: vogliamo sapere rispondere alla domanda "*M* si arresta su input *w*?". Il linguaggio corrispondente a questo problema è $\boldsymbol{HALT_{TM} = \{\langle M, w\rangle \mid M \text{ è una MdT che si arresta su } w\}}$.

Per ora, introduciamo degli strumenti con cui lavoreremo: ***cardinalità di insiemi (infiniti)*** e ***diagonalizzazione (metodo introdotto da Cantor).***

La **cardinalità** di un insieme è banalmente la sua taglia. Due insiemi hanno la stessa cardinalità se è possibile stabilire una corrispondenza tra i loro elementi.

Es. A = {1, 2, 3}, B = {4, 3, 5} ⇒ 1 – 4, 2 – 3, 3 – 5.

Paradosso di Hilbert:

Nel paese senza confini esiste il più grande di tutti gli alberghi, cioè un albergo con infinite stanze. Tuttavia, anche gli ospiti sono infiniti, e il proprietario ha esposto un cartello con la scritta "Completo". Ad un tratto, si presenta un viaggiatore che ha assolutamente bisogno di una camera per la notte. Egli non fa questione di prezzo e infine convince l'albergatore, il quale trova il modo di alloggiarlo. *Come fa?*

Sposta tutti i clienti nella camera successiva (l'ospite della 1 alla 2, l'ospite della 2 alla 3, …); in questo modo, è possibile, essendo l'albergo infinito, sistemare (nella camera 1) il nuovo ospite anche se l'albergo è pieno.

[Passo completato]

Poco dopo, arriva una comitiva di *infiniti turisti*, anche in questo caso l'albergatore si lascia covincere (in fondo si tratta di un grosso affare), e trova posto ai *nuovi infiniti ospiti* con la stessa facilità con cui aveva alloggiato l'ospite in più. *Come fa* (senza ripetere infinite volte il passo visto prima)*?*

Sposta ogni ospite nella stanza con un numero doppio rispetto a quello attuale (dalla 1 alla 2, dalla 2 alla 4, …), lasciando ai nuovi ospiti tutte le camere con i numeri dispari, che sono essi stessi infiniti, risolvendo dunque il problema. Questa operazione, infatti, fa sì che le sole stanze pari siano occupate, il che implica che le stanze dispari siano libere.

Gli ospiti sono tutti, dunque, sistemati, benché l'albergo fosse pieno.

[Passo completato]

Ancora più complesso: ci sono infiniti alberghi con infinite stanze, tutti al completo. Tutti gli alberghi chiudono, tranne uno. Tutti gli ospiti vogliono alloggiare nell'unico albergo rimasto aperto. *Come fa* (senza ripetere infinite volte il passo visto prima)*?*

Assegna ad ogni persona una coppia di numeri (*n*, *m*) in cui *n* indica l'albergo di provenienza, e *m* la relativa stanza. Gli ospiti sono quindi etichettati come segue:

$$\begin{array}{cccccc} (1,1) & (1,2) & (1,3) & (1,4) & \dots \\ (2,1) & (2,2) & (2,3) & (2,4) & \dots \\ (3,1) & (3,2) & (3,3) & (3,4) & \dots \\ (4,1) & (4,2) & (4,3) & (4,4) & \dots \\ \dots & \dots & \dots & \dots & \dots \end{array}$$

A questo punto, basta assegnare le nuove stanze agli ospiti secondo un criterio ordinato, ad esempio per le diagonali, come indicato nella pagina seguente.

$$(1,1) \to 1;\ (1,2) \to 2;\ (2,1) \to 3;\ (1,3) \to 4;(2,2) \to 5;\ (3,1) \to 6;\dots$$

Possiamo, quindi, indicare una corrispondenza tra cardinalità di insiemi infiniti? Ad esempio, l'insieme dei numeri naturali è INFINITO, così come l'insieme dei numeri naturali pari e l'insieme dei numeri naturali dispari. Anche l'insieme dei numeri reali è INFINITO.

La quantità di numeri reali è la stessa di quella dei numeri naturali? Come si misura la cardinalità di insiemi infiniti?

5.2 INDECIDIBILITÀ

Metodo della Diagonalizzazione:

introdotto da Cantor nel 1973 mentre cercava di determinare come stabilire se, dati due insiemi infiniti, uno è più grande dell'altro. Cantor osservò che due insiemi finiti hanno la stessa cardinalità se gli elementi dell'uno possono essere messi in corrispondenza uno a uno con quelli dell'altro. Successivamente, estese questo concetto agli insiemi infiniti.

Una *funzione* $f: X \to Y$ è una relazione input-output, dove *X* è l'insieme dei possibili input (*dominio*) e *Y* è l'insieme dei possibili output (*codominio*). Per ogni input $x \in X$ esiste un solo output $y = f(x) \in Y$.
Una funzione $f: X \to Y$ è *iniettiva* se $\forall x, x' \in X, x \neq x' \Rightarrow f(x) \neq f(x')$.
Una funzione $f: X \to Y$ è *suriettiva* se $\forall y \in Y, y = f(x)$ per qualche $x \in X$.
Una funzione $f: X \to Y$ è una funzione *biiettiva* di *X* su *Y* (o una *biezione* tra *X* e *Y*) se f è iniettiva e suriettiva.

Una funzione biiettiva è una corrispondenza uno a uno tra gli elementi del dominio e gli elementi del codominio.

Esempi:

1. $f: \{1,2,5\} \to \{2,4,7\}$, con $1 \to 2, 2 \to 5, 5 \to 4$ è una funzione, ma non è né iniettiva né suriettiva.
2. $f: \{1,2,5\} \to \{2,4,7,9\}$, con $1 \to 2, 2 \to 4, 5 \to 7$ è una funzione iniettiva ma non suriettiva.
3. $f: \{1,2,5\} \to \{2,4\}$, con $1 \to 2, 2 \to 4, 5 \to 2$ è una funzione suriettiva ma non iniettiva.
4. $f: \{1,2,5\} \to \{2,4,7\}$, con $1 \to 2, 2 \to 4, 5 \to 7$ è una funzione biiettiva.
5. $f: \mathbb{N} \to \{2n \mid n \in \mathbb{N}\}$, con $n \to 2n$ è una funzione biiettiva.

Due insiemi *X* e *Y* hanno *la stessa cardinalità* se esiste una funzione biiettiva $f: X \to Y$ di *X* su *Y*.
Un insieme è *enumerabile* (o *numerabile*) se ha la stessa cardinalità di un sottoinsieme di $\mathbb{N}$. Se A è numerabile, allora possiamo "numerare" gli elementi di *A* e scrivere una lista (a_1, a_2, …); cioè, per ogni numero naturale *i*, allora possiamo specificare l'elemento *i*-esimo della lista.

Esempio:

Per l'insieme dei numeri naturali pari, l'elemento *i*-esimo della lista corrisponde a 2*i*.

Per quest'ultima proprietà, quindi, possiamo associare ad un insieme *A* (*finito*) un sottoinsieme dell'insieme $\mathbb{N}$ (ovviamente, $\mathbb{N}$ è *infinito*) attraverso una funzione biettiva $f: A \to \mathbb{N}$. Ciò vuol dire che avremo associato un unico elemento di *A* ad un unico elemento di $\mathbb{N}$.

L'insieme dei numeri razionali è numerabile. Mostriamo che possiamo formare una lista di tutti i numeri razionali. Formiamo un rettangolo infinito, come mostrato nella figura a destra. Questa tabella, con infinite righe e colonne, è corretta, e si noti che se scorre per le righe s'incrementa man mano il numeratore, mentre se si scorre per le colonne s'incrementa man mano il denominatore.

$$\begin{array}{cccccc} 1/1 & 1/2 & 1/3 & 1/4 & \dots \\ 2/1 & 2/2 & 2/3 & 2/4 & \dots \\ 3/1 & 3/2 & 3/3 & 3/4 & \dots \\ 4/1 & 4/2 & 4/3 & 4/4 & \dots \\ \dots & \dots & \dots & \dots & \dots \end{array}$$

Per far vedere che questo insieme è numerabile, bisogna mostrare la biezione (o, equivalentemente, bisogna listare tutti i numeri razionali). Sicuramente non possiamo procedere per righe, dato che se prendiamo tutta la prima linea non arriviamo mai alla seconda; allo stesso modo, non

possiamo procedere per colonne. Un'idea può essere procedere *in diagonale* (usando la secondaria, in questo caso). Dunque, prendiamo tutti gli elementi della prima diagonale (1/1) per poi prendere tutti gli elementi della seconda diagonale (2/1, 1/2), e così via.

1/1	1/2	1/3	1/4	…
2/1	2/2	2/3	2/4	…
3/1	3/2	3/3	3/4	…
4/1	4/2	4/3	4/4	…
…	…	…	…	…

1/1, 2/1, 1/2, 3/1, 2/2, 1/3, …

Nota: Dovremmo eliminare i duplicati (ad esempio, 1/1 = 1 e 2/2 = 1), ma è solo una questione tecnica. Costruita la lista (a_1, a_2, …), possiamo quindi numerarla. Ciò vuol dire che esiste la biezione tra l'insieme dei numeri razionali e l'insieme dei numeri reali, il che significa che l'insieme dei numeri razionali è numerabile.

L'insieme Σ* è numerabile: per dimostrarlo, listiamo prima la stringa vuota, poi le stringhe (in ordine lessicografico) lunghe 1, poi 2, e così via. A questo punto, come nell'esempio precedente, possiamo numerare la lista (partendo da 1) e quindi *numerare* l'insieme Σ*.

Esempio:

Σ = {0, 1} ⇒ w_0 = ε, w_1 = 0, w_1 = 1, w_3 = 00, …

In questo caso, possiamo sapere anche la posizione in cui compare una determinata stringa nella lista. Presa 001, ad esempio, sappiamo che |001| = 3, e che essa compare in posizione $2^0 + 2^1 + 2^2 + 2$ (quest'ultimo in quanto è la *seconda* stringa in ordine lessicografico tra tutte le stringhe di lunghezza 3) = 9; quindi, la stringa 001 si trova in posizione 9 nella lista.

L'insieme delle descrizioni di MdT $\{\langle M\rangle \mid M$ è una MdT sull'alfabeto Σ$\}$ è numerabile: è possibile codificare una MdT *M* con una stringa su un alfabeto Σ.

In generale, *per determinare se un insieme è numerabile bisogna mostrare che esiste una biezione con* $\mathbb{N}$, cioè per ogni elemento possiamo far vedere che posizione occupa all'interno dell'insieme.

Teorema:

L'insieme dei numeri reali R non è numerabile

Dimostrazione:

Sia per assurdo $\mathbb{R}$ numerabile; allora possiamo costruire la lista f(1), f(2), f(3), …

Per ogni $i \geq 1$, scriviamo $f(i) = f_0(i),f_1(i)f_2(i)f_3(i)$… Cioè, sappiamo che ogni f(*i*) è un numero reale, ed essendo tale ha una parte intera ed una parte decimale. Nella rappresentazione precedente, $f_0(i)$ è la parte intera, separata da una virgola dalla parte decimale $f_1(i)f_2(i)f_3(i)$… Ad esempio, se f(1) = 4,256… allora $f_0(1) = 4$, $f_1(1) = 2$, $f_2(1) = 5$, $f_3(1) = 6$, …

Organizziamoli in una matrice, posta a lato, in cui le colonne sono indicizzate con gli interi 1, 2, 3, …, *i* e la riga *i*-esima è l'elemento f(*i*) che compare nella lista.

$i \backslash f(i)$	f_1	f_2	f_3	…	…	…
1	$f_1(1)$	$f_2(1)$	$f_3(1)$	…	$f_i(1)$	…
2	$f_1(2)$	$f_2(2)$	$f_3(2)$	…	$f_i(2)$	…
3	$f_1(3)$	$f_2(3)$	$f_3(3)$	…	$f_i(3)$	…
⋮	⋮	⋮	⋮	⋱	⋮	⋮
i	$f_1(i)$	$f_2(i)$	$f_3(i)$	…	$f_i(i)$	…
⋮	⋮	⋮	⋮	⋮	⋮	⋱

Quindi, ad esempio, nella riga 1 compaiono le cifre decimali del primo elemento; stiamo ignorando la parte intera di ogni numero.

A questo punto, consideriamo la diagonale di questa matrice. Sia $x \in (0, 1)$ il numero x=0,$x_1x_2x_3$…x_i… ottenuto scegliendo $x_i \neq f_i(i)$ per ogni $i \geq 1$. Chiaramente, $x \in \mathbb{R}$.

A questo punto, risulta *x* nella lista? Se $x = f(j)$, allora il suo *j*-esimo digit soddisfa $x_j = f_j(j)$: ma $x_j \neq f_j(j)$ per definizione di *x*. Questa è una contraddizione.

Quindi, $x \in \mathbb{R}$ non può comparire nella lista e $\mathbb{R}$ non è numerabile.

Abbiamo detto che Σ* = {w_1, w_2, …} e l'insieme delle descrizioni di MdT su Σ (cioè {M_1, M_2, …}) sono numerabili. Anche in questo caso possiamo usare il Metodo della diagonalizzazione; quindi, costruiamo la tabella seguente, dove $x_{i,j} = 1$ se $w_j \in L(M_i)$, $x_{i,j} = 0$ altrimenti.

	w_1	w_2	w_3	…	w_i	w_j
M_1	$x_{1,1}$	·	·	·	·	·
M_2	·	$x_{2,2}$	·	·	·	·
·	·	·	$x_{3,3}$	·	·	·
·	·	·	·	$x_{4,4}$	·	·
M_i	·	·	·	·	$x_{i,i}$	$x_{i,j}$
·	·	·	·	·	·	·
·	·	·	·	·	·	·

Possiamo sfruttare la diagonale principale ($x_{1,1}$, $x_{2,2}$, ..., $x_{i,j}$, ...) per costruire un nuovo linguaggio $L = \{w_i \in \Sigma^* \mid w_i \notin L(M_i)\}$. Questo linguaggio è il "complemento della diagonale":

- se l'elemento (M_i, w_i) della diagonale $x_{i,i} = 1$, allora $w_i \notin$ L;
- se l'elemento (M_i, w_i) della diagonale $x_{i,i} = 0$, allora $w_i \in$ L.

Vogliamo stabilire se L può comparire nella lista, cioè se L = L(M_h) per qualche *h*; supponiamo che L = L(M_h), allora:

- $w_h \in$ L ⇒ $x_{h,h} = 0$ ⇒ $w_h \notin$ L(M_h) = L, che è una contraddizione;
- $w_h \notin$ L ⇒ $x_{h,h} = 1$ ⇒ $w_h \in$ L(M_h) = L, che è una contraddizione.

Da quest'ultimo risultato, otteniamo che L ≠ L(M_h) per ogni h, il che significa che L non può comparire nella lista. Da ciò segue che "esistono più linguaggi che Macchine di Turing".

Corollario:

Esistono linguaggi che non sono Turing riconoscibili.

Macchina di Turing universale:

Una MdT **universale** *U* simula la computazione di una qualsiasi MdT *M*.

Allora *U* riceve in input una *rappresentazione* $\langle M, w\rangle$ di *M* e di un possibile input *w* di *M*.

N.B. Abbiamo visto che è possibile codificare una MdT *M* e una stringa *w* con una stringa su un alfabeto Σ.

Esempio:

$\langle M, w\rangle$ = "codifica di *M*"#"codifica di *w*", cioè la concatenazione (#) tra le due codifiche.

Una tale macchina è chiamata "universale" perché la computazione di una qualsiasi MdT può essere simulata da *U*.

$$\langle M, w\rangle \rightarrow \text{ MdT universale } U \rightarrow \begin{cases} accetta & \text{se } M \text{ accetta } w \\ rifiuta & \text{se } M \text{ rifiuta } w \end{cases}.$$

Nota. *U* può anche andare in loop.

Dunque, abbiamo costruito una macchina universale *U* il cui linguaggio è $A_{TM} = \{\langle M, w\rangle \mid M \text{ accetta } w\}$.

Teorema:

Il linguaggio $A_{TM} = \{\langle M, w\rangle \mid M$ è una MdT che accetta la parola $w\}$ è Turing riconoscibile

Dimostrazione:

Definiamo una MdT *U* che accetta A_{TM}: sull'input $\langle M, w\rangle$, dove *M* è una MdT e *w* è una stringa:

1. Simula *M* sull'input *w*;
2. Se *M* accetta, allora accetta; se *M* rifiuta, allora rifiuta.

Abbiamo visto una MdT che simula un automa. Simulare una MdT *M* con un'altra MdT risulta molto simile:

1. Marca lo stato iniziale di *M* (stato corrente) e il primo simbolo sul nastro (posizione corrente della testina);
2. Cerca la prossima transizione (nella parte che descrive la funzione di transizione). Sia (q, *x*) → (q', *x'*, D);
3. Esegui la transizione;
4. Aggiorna lo stato corrente (marca q') e la posizione corrente della testina (marca simbolo a D);
5. Se lo stato corrente risulta q_{accept}/q_{reject} decidi di conseguenza, altrimenti ritorna al passo 2.

Nota: *U* è detta MdT universale.
Nota: *U* riconosce A_{TM}: accetta ogni coppia $\langle M, w\rangle \in A_{TM}$.
Nota: *U* cicla su $\langle M, w\rangle$ se (e solo se) *M* cicla su *w*. Quindi *U* **non decide** A_{TM}.

Teorema (INDECIDIBILITÀ DEL PROBLEMA DELLA FERMATA):

Il linguaggio $A_{TM} = \{\langle M, w\rangle \mid M$ è una MdT che accetta la parola $w\}$ non è decidibile.

Notiamo che un tale linguaggio può essere trasformato in un problema di decidibilità/indecidibilità su input *M*, *w* e che risponde alla domanda "*M* accetta *w*?". Non è possibile, però, decidere se una data macchina si ferma su un dato input.

Paradosso di Bertrand Russel:

In un paese vive un solo barbiere, un uomo ben sbarbato, che rade tutti e soli gli uomini del villaggio che non si radono da soli. *Chi sbarba il barbiere?*
- Se il barbiere rade sé stesso, allora per definizione il barbiere non rade se stesso;
- se il barbiere non rade se stesso, allora, dato che il barbiere rade tutti quelli che non si radono da soli, il barbiere rade se stesso.

Si tratta di un'**antinomia**: compresenza di due affermazioni contraddittorie che possono essere entrambe dimostrate o giustificate.
In generale, Russel pose il problema dell'insieme di tutti gli insiemi che non contengono se stessi. L'*autoreferenza* può causare problemi.
Possiamo usare un'autoreferenzialità del genere per dimostrare l'indecidibilità di A_{TM}.

Dimostrazione:

Supponiamo per assurdo che A_{TM} sia decidibile, cioè che esiste una macchina di Turing *H* con due possibili risultati di computazione (accettazione, rifiuto) e tale che

$$H = \begin{cases} accetta & \text{se } M \text{ accetta } w \\ rifiuta & \text{se } M \text{ non accetta } w \end{cases}.$$

Si noti che, a differenza di *U*, la macchina *H* è un decisore (o accetta o rifiuta, ma non va mai in loop).
Costruiamo una nuova MdT *D* che usa *H* come sottoprogramma *D* sull'input $\langle M\rangle$, dove *M* è una MdT:

1. Simula *H* sull'input $\langle M, \langle M\rangle\rangle$;
2. Fornisce come output l'opposto di *H*, cioè se *H* accetta, allora *rifiuta* e se *H* rifiuta, allora *accetta*.

$$\langle M\rangle \to D \to \langle M, \langle M\rangle\rangle \to H \to \begin{cases} accetta \\ rifiuta \end{cases} \to I \to \begin{cases} rifiuta & \text{se } M \text{ accetta } \langle M\rangle \\ accetta & \text{se } M \text{ non accetta } \langle M\rangle \end{cases}.$$

Quindi:

$$D(\langle M\rangle) = \begin{cases} rifiuta & \text{se } M \text{ accetta } \langle M\rangle \\ accetta & \text{se } M \text{ non accetta } \langle M\rangle \end{cases}.$$

Se ora diamo in input a *D* la sua stessa codifica $\langle D\rangle$ abbiamo:

$$\langle D\rangle \to D \to \langle D, \langle D\rangle\rangle \to H \to \begin{cases} accetta \\ rifiuta \end{cases} \to I \to \begin{cases} rifiuta & \text{se } D \text{ accetta } \langle D\rangle \\ accetta & \text{se } D \text{ non accetta } \langle D\rangle \end{cases}.$$

Cioè, *D* accetta $\langle D\rangle$ se e solo se *D* non accetta $\langle D\rangle$.
Questo è assurdo. *Tutto è causato dall'assunzione che esiste H*. Quindi, *H* non esiste. □

Riepilogando la dimostrazione:

1. Definiamo $A_{TM} = \{\langle M, w\rangle \mid M$ è una MdT che accetta la parola$w\}$;
2. *Assumiamo che A_{TM} sia decibilie; sia H una MdT che lo decide*;
3. Usiamo *H* per costruire una MdT *D* che inverte le decisioni: $D(\langle M\rangle)$ accetta se *M* non accetta $\langle M\rangle$ e rifiuta se *M* accetta $\langle M\rangle$;
4. Diamo in input a *D* la sua codifica $\langle D\rangle$: $D(\langle D\rangle)$ accetta se e solo se D rifiuta $\langle D\rangle$. Contraddizione.

Anche se "nascosta", la dimostrazione precedente utilizza la diagonalizzazione. Consideriamo la tabella a lato, dove le M_t sono tutte le MdT numerate, e $\langle M_t\rangle$ sono le loro descrizioni (stringhe): se nella cella (i, j) c'è "acc", allora $\langle M_j\rangle \in L(M_i)$; se non c'è nulla, allora $\langle M_j\rangle \notin L(M_j)$.

	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle M_3\rangle$	$\langle M_4\rangle$	...
M_1	acc		acc		...
M_2	acc	acc	acc	acc	...
M_3					
M_4	acc	acc			...
⋮	⋮	⋮	⋮	⋮	⋮

Consideriamo *H*: la MdT *H* rifiuta anche se M_i va in loop (oltre a se M_i rifiuta). La sua tabella è posta a destra.

	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle M_3\rangle$	$\langle M_4\rangle$	...
M_1	acc	rej	acc	rej	...
M_2	acc	acc	acc	acc	...
M_3	rej	rej	rej	rej	...
M_4	acc	acc	rej	rej	...
⋮	⋮	⋮	⋮	⋮	⋮

Consideriamo, ora, *D* e *D*(⟨*D*⟩). Dobbiamo considerare la diagonale.
Dove sono posti i tre punti interrogativi (???) non eravamo in grado di mettere né accetta né rifiuta; infatti *D*, in corrispondenza della sua descrizione, non può né accettare né rifiutare.

Nella prova precedente, quindi, è stato usato il metodo della diagonalizzazione. In conclusione, A_{TM} è Turing riconoscibile ma è ***indecidibile***.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$	...
M_1	acc	rej	acc	rej	...	...	...
M_2	acc	acc	acc	acc	...	...	...
M_3	rej	rej	rej	rej	...	...	...
M_4	acc	acc	rej	rej	...	...	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
D	acc	acc	rej	rej	...	???	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Che differenza c'è tra le due dimostrazioni? Cioè, che differenza c'è tra *U* e *D*? Chiaramente, la differenza tra le due macchine è proprio nell'esistenza del loop.

Sappiamo che esistono linguaggi che non sono Turing riconoscibili. Vogliamo individuare uno specifico linguaggio non Turing riconoscibile ($\overline{A_{TM}}$, cioè il complemento di A_{TM}).

Diciamo che un linguaggio *L* è *co-Turing riconoscibile* se il suo complemento $\overline{L}$ è Turing riconoscibile.

Teorema:

Un linguaggio L è decidibile se e solo se L è Turing riconoscibile e co-Turing riconoscibile.

Dimostrazione:

(⇒) Se *L* è decibilie, allora esiste una macchina di Turing *M* con due possibili risultati di una computazione (accettazione, rifiuto) e tale che *M* accetta *w* se e solo se $w \in L$. Allora *L* è Turing riconoscibile. Inoltre, è facile costruire una MdT $\overline{M}$ che accetta *w* se e solo se $w \notin L$:

$$w \to \boxed{M} \to \begin{cases} accetta & \text{se w} \in \text{L} \\ rifiuta & \text{se w} \notin \text{L} \end{cases} \to \begin{cases} rifiuta & \text{se M accetta (cioè, w} \notin \overline{\text{L}}) \\ accetta & \text{se M rifiuta (cioè, w} \in \overline{\text{L}}) \end{cases}.$$

(⇐) Supponiamo che *L* e il suo complemento siano entrambi Turing riconoscibili. Sia M_1 una MdT che riconosce *L* e sia M_2 una MdT che riconosce $\overline{L}$. Definiamo una MdT *N* a due nastri, la quale si limiterà a simulare in parallelo le macchine M_1 e M_2.

La macchina N, su input *x*, funzionerà come segue:

1. Copia *x* sui nastri di M_1 e M_2. Quindi, un nastro lo associamo alla macchina M_1 ed uno alla macchina M_2, dopodiché ogni mossa sarà data dalla coppia di mosse che farebbero M_1 e M_2;
2. Simula M_1 e M_2 in parallelo (usa un nastro per M_1, l'altro per M_2);
3. Se M_1 accetta, allora *N* accetta. Se M_2 accetta, allora *N* rifiuta.

$$x \to \begin{matrix} \boxed{M_1} \to accetta \\ \boxed{M_2} \to accetta \end{matrix} \to \begin{cases} accetta & \text{se } M_1 \text{ accetta} \\ rifiuta & \text{se } M_2 \text{ accetta} \end{cases}.$$

Segue che *N* decide *L*. Infatti, per ogni stringa *x* abbiamo due casi:

1. $x \in L$. Ma $x \in L$ se e solo se M_1 si arresta e accetta *x*. Quindi *N* accetta *x*;
2. $x \notin L$. Ma $x \notin L$ se e solo se M_2 si arresta e accetta *x*. Quindi *N* rifiuta *x*.

Poiché una e solo una delle due MdT accetta *x*, allora *N* è una MdT con soli due possibili risultati di una computazione (accettazione, rifiuto) e tale che *N* accetta *x* se e solo se $x \in L$.

Teorema:

$\overline{A_{TM}}$ *non è Turing riconoscibile*

Dimostrazione:

Supponiamo per assurdo che $\overline{A_{TM}}$ sia Turing riconoscibile. Sappiamo che A_{TM} è Turing riconoscibile. Quindi, A_{TM} è Turing riconoscibile e co-Turing riconoscibile. Per il precedente teorema, A_{TM} è decidibile. Questo è assurdo, in quanto abbiamo dimostrato che A_{TM} non è decidibile.

□

È importante riconoscere che un problema *P* è indecidibile. Si può estendere la classe dei problemi indecidibili in due modi:

1) Supporre l'esistenza di una MdT che decide *P* e provare che questo conduce ad una contraddizione (come fatto con A_{TM});
2) Considerare un problema *P'* di cui sia nota l'indecidibilità (ad esempio, A_{TM}) e dimostrare che *P' "non è più difficile"* del problema in questione *P*.

Esempio:

Sia Σ = {0, 1}. Consideriamo i problemi:

- $EVEN = \{w \in \Sigma^* \mid w \text{ è la rappresentazione binaria di } n \in \mathbb{N} \text{ pari}\}$;
- $ODD = \{w \in \Sigma^* \mid w \text{ è la rappresentazione binaria di } n \in \mathbb{N} \text{ dispari}\}$.

Sia $w \in L$ e sia *n* il corrispondente decimale di *w*. È facile costruire la MdT *INCR* che incrementa il valore di *n*:

$$w \to \boxed{INCR} \to w' \text{ (= rappresentazione binaria di } n + 1).$$

Possiamo dire che *EVEN* "non è più difficile" di *ODD*: se esiste una MdT *R* che decide *ODD*, allora la MdT *S* decide *EVEN*.

$$S : w \to \boxed{INCR} \to w' \to \boxed{R} \ (\to accetta\ w' \ \text{ se n} + 1 \text{ è dispari} \leftrightarrow \text{n è pari}).$$

Viceversa, se *EVEN* è indecidibile proviamo così che anche *ODD* lo è: se per assurdo esistesse una MdT *R* che decide *ODD*, allora la MdT *S* deciderebbe *EVEN*.