

Macchine a responsabilità limitata

Breve Introduzione alla Teoria della Calcolabilità:
Tesi di Turing-Church e Problemi (in-)decidibili

Roberto Maieli

Università degli Studi "Roma Tre"
maieli@uniroma3.it

Architetture dell'Informazione e della Comunicazione

1. quanto sono potenti gli algoritmi?
2. ci sono problemi che neanche in linea di principio possono essere risolti (nè oggi nè in futuro) mediante un algoritmo?
3. in caso di risposta positiva alla precedente domanda, è possibile migliorare la nozione che abbiamo di algoritmo?

A queste ed altre domande cerca di rispondere la

Teoria della Calcolabilità.

Partiamo facendo qualche passo indietro nel tempo ...

Il decimo problema di Hilbert

Alla *Conferenza Internazionale dei Matematici* (Parigi, 1900), David Hilbert pone 23 problemi matematici, tra questi:

PROBLEMA N.10: *escogitare un processo secondo il quale si può determinare in un numero finito di operazioni se un polinomio ammette una radice intera.*

- un **polinomio** è una somma di *termini*
- un **termine** è un prodotto di variabili e costanti dette **coefficienti**;

$6x^3yz^2$ è un termine (*monomio*) con coefficiente “6”
 $6x^3yz^2 + 3xy^2 - x^3 - 10$ è un polinomio di 4 termini e variabili x , y e z .

Una **radice** di un polinomio è un assegnamento di valori alle variabili tale che il valore risultante del polinomio è 0.

Es.: per il polinomio di sopra, $\{x = 5, y = 3, z = 0\}$ è una **radice intera** (i valori assegnati alle variabili sono interi).

Alcuni polinomi hanno una radice intera altri no.

alcune osservazioni sul decimo problema di Hilbert

- ▶ Hilbert non usò il termine *algoritmo* per questo problema, ma quello di *processo* che, per definizione, è determinato da un numero finito di operazioni.
- ▶ Hilbert non chiedeva di controllare l'*esistenza* di un algoritmo ma solo di implementarlo. Questo perchè era convinzione comune che ogni problema dovesse ammettere una procedura risolutiva.

Oggi sappiamo che il **decimo problema di Hilbert è indecidibile!**

Questo risultato è stato formalmente provato nel 1970 da Yuri Matijasevic che estese una idea di Martin Davis, Hilary Putnam, and Julia Robinson.

Ai tempi di Hilbert, gli strumenti a disposizione dei matematici erano certamente adeguati per mostrare procedure per alcuni problemi, ma erano completamente inadeguati per provare l'inesistenza di una procedura per quelli che oggi sappiamo essere indecidibili.

La tesi di Church-Turing

I primi risultati sul *decimo problema* arrivarono \sim 1930, passando per una **definizione rigorosa del concetto di algoritmo.**

Tale definizione arrivò ad opera di Alonzo Church and Alan Turing:

- ▶ Church usò un sistema notazionale chiamato **λ -calculus**,
- ▶ Turing usò un modello di **macchina computazionale**.

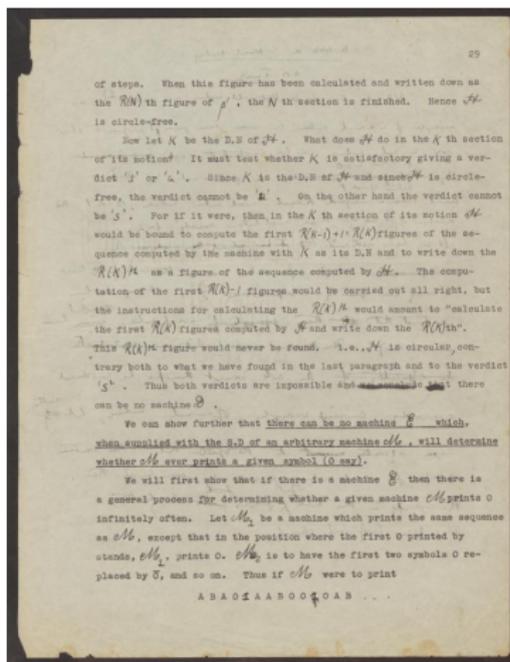
Queste due definizioni furono poi mostrate essere equivalenti.

Questa equivalenza diede origine alla seguente:

tesi di Church-Turing:

“nozione intuitiva di algoritmo” = “macchina di Turing”

Alan M. Turing



Alan M. Turing

*On Computable Numbers,
with an Application to the
Entscheidungsproblem,*
Proc. Lond. Math. Soc. (2)
42 pp. 230-265 (1936)

Turing intende dare una risposta negativa al problema della decisione

Ma centra il lavoro sulla nozione di *calcolabile*

On computable numbers (1/3)

Leggiamo assieme ...

- ▶ computable numbers [are] the real numbers whose expressions as a decimal are calculable by finite means;
- ▶ [that is,] its decimal can be written down by a machine
- ▶ Computing is [. . .] done by writing [. . .] symbols on paper
- ▶ We may suppose this paper is divided into squares
- ▶ The behaviour of the computer [. . .] is determined by the symbols which he is observing, and his state of mind
- ▶ operations [are] split up into simple operations
- ▶ in a simple operation not more than one symbol is altered

On computable numbers (2/3)

Leggiamo assieme ...

- ▶ The most general single operation must therefore be taken to be one of the following:
 1. a possible change of symbol together with a possible change of state of mind;
 2. a possible change of observed squares, together with a possible change of state of mind.
- ▶ The operation actually performed is determined [. . .] by the state of mind of the computer and the observed symbols. In particular, they determine the state of mind of the computer after the operation is carried out.

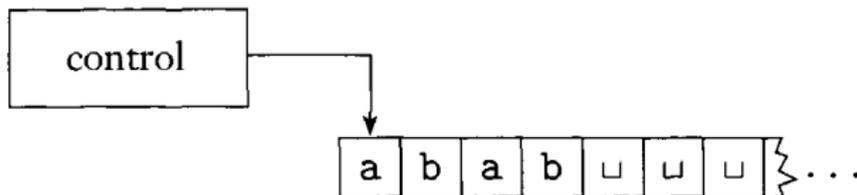
On computable numbers (3/3)

Leggiamo assieme ...

- ▶ we [may] avoid introducing the state of mind
- ▶ [At each step, the computer] must leave a note of instructions [. . .] explaining how the work is to be continued. This note is the counterpart of the state of mind.
- ▶ the state of progress of the computation at any stage is completely determined by the note of instructions and the symbols on the tape

Macchina di Turing (TM, 1936)

E' composta da un **nastro** e da una **testina** che scorre sul nastro. Inizialmente il nastro contiene solo la stringa input ed è vuoto altrove. Convenzionalmente all'**inizio del calcolo** la testina in uno stato interno iniziale e collocata in sul primo simbolo a sinistra dell'input.



Osserva:

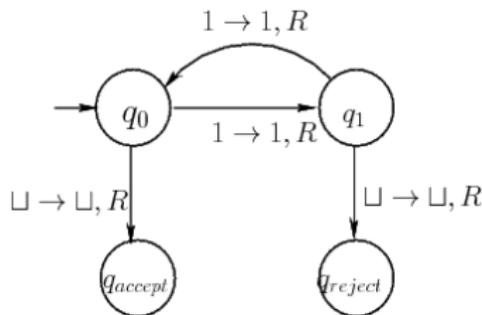
1. una TM può sia scrivere sul nastro che leggere dal nastro,
2. la testina di lettura/scrittura può muoversi a destra e a sinistra,
3. il nastro è infinito.

Una Macchina di Turing (TM) è una 7-upla $\langle Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject} \rangle$:

1. Q è un insieme (finito) di stati
2. Σ è l'alfabeto dell'input
3. $\Gamma = \Sigma \cup \{\sqcup\}$ è l'alfabeto del nastro (\sqcup denota *la cella vuota*)
4. $\delta : Q \times \Gamma \Rightarrow Q \times \Gamma \times \{L, R\}$ è la funzione di transizione
5. $q_0 \in Q$ è lo stato iniziale
6. $q_{accept} \neq q_{reject}$ sono gli stati finali.

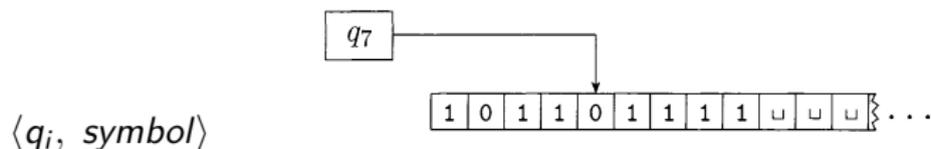
Example (M_1)

- 1 $Q = \{q_0, q_1, q_{accept}, q_{reject}\}$
- 2 $\Sigma = \{1\}$
- 3 $\Gamma = \Sigma \cup \{\sqcup\}$
- 4 δ è dato dal diagramma di stati:



linguaggio di una TM

Una **configurazione** di una TM rispetto ad un input w è una coppia



Una TM **accetta** (computa o riconosce) un input w se esiste una sequenza di configurazioni C_1, \dots, C_n (un cammino nel diagramma degli stati, detta **computazione**) tale che:

1. C_1 è la configurazione iniziale, $\langle q_0, symbol_i \rangle$
2. per ogni $1 \leq i \leq n$, C_{i+1} è ottenuto da C_i per transizione di stato δ
3. C_n è una configurazione di accettazione $\langle q_{accept}, symbol_j \rangle$

Il **linguaggio** L di una TM M è dato dall'insieme

$$L_M = \{w \mid M \text{ accetta } w\}$$

cioè l'insieme delle sequenze w che la macchina M accetta.

L_{M_1} : il linguaggio di M_1 è dato dall'insieme di tutte le stringhe 1-arie di lunghezza pari

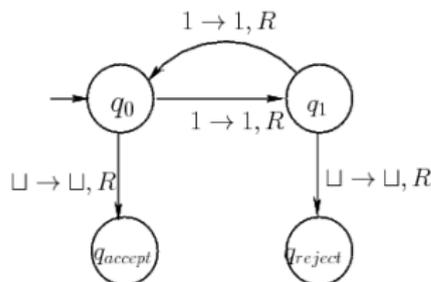
1. $Q = \{q_0, q_1, q_{accept}, q_{reject}\}$
2. $\Sigma = \{1\}$
3. $\Gamma = \Sigma \cup \{\sqcup\}$
4. la funzione di transizione δ é data dalle seguenti istruzioni:
(le quintuple a sinistra oppure il diagramma degli stati a destra)

$$\delta_1: (q_0, \sqcup) \mapsto (q_{accept}, \sqcup, R)$$

$$\delta_2: (q_0, 1) \mapsto (q_1, 1, R)$$

$$\delta_3: (q_1, 1) \mapsto (q_0, 1, R)$$

$$\delta_4: (q_1, \sqcup) \mapsto (q_{reject}, \sqcup, R)$$



Esempi di Computazioni di M_1 :

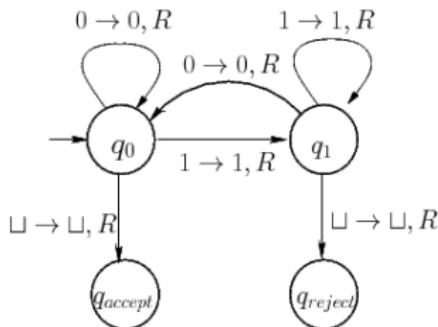
M_1 accetta 11: $(q_0, 1) \rightsquigarrow_{\delta_2} (q_1, 1) \rightsquigarrow_{\delta_3} (q_0, \sqcup) \rightsquigarrow_{\delta_1} (q_{accept}, \sqcup)$

M_1 rifiuta 111: $(q_0, 1) \rightsquigarrow_{\delta_2} (q_1, 1) \rightsquigarrow_{\delta_3} (q_0, 1) \rightsquigarrow_{\delta_2} (q_1, \sqcup) \rightsquigarrow_{\delta_4} (q_{rej}, \sqcup)$

Example (M_2)

Si consideri la MT M_2 definita nel modo seguente:

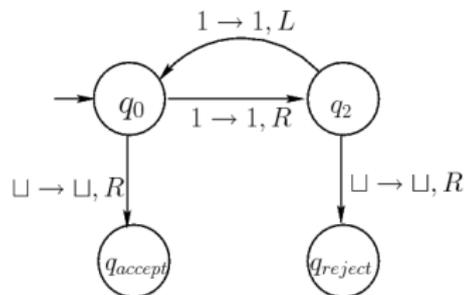
- 1 $Q = \{q_0, q_1, q_{accept}, q_{reject}\}$
- 2 $\Sigma = \{0, 1\}$
- 3 $\Gamma = \Sigma \cup \{\sqcup\}$
- 4 δ è dato dal diagramma di stati:



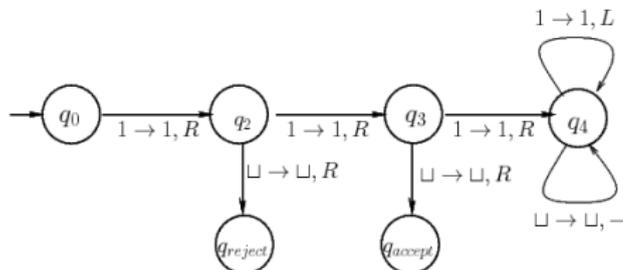
- ▶ è facile osservare che il linguaggio L_{M_2} della MT M_2 è dato dall'insieme delle sequenze binarie che terminano con 0 oppure vuote.
- ▶ Come esercizio si provi a scrivere le quintuple di cui si compone la funzione di transizione (il diagramma di stati) della MT M_2 .
- ▶ Si provi inoltre a scrivere la computazione di M_2 sull'input 101 e la computazione di M_2 sull'input 1010.

stati di accettazione, rifiuto e loop

dato input una MT può terminare la sua computazione, **accettando o rifiutando l'input**, oppure può non terminare la computazione, non accettando nè rifiutando l'input (*la macchina va in loop*)



- accepts $w = \sqcup$ (halts)
- rejects $w = 1$ (halts)
- loops on $w = 11\dots 1$ (does not halt)



- rejects $w = 1$ (halts)
- accepts $w = 11$ (halts)
- loops on $w = 11\dots 1$ (does not halt)

linguaggi riconoscibili e linguaggi decidibili

Data una MT M , conosciamo il suo linguaggio L_M , cioè l'insieme delle stringhe w accettate da M .

Viceversa, dato un qualsiasi linguaggio L , definito su un alfabeto Σ , vogliamo adesso capire in che rapporto L sta con le MT.

Un linguaggio L , definito su un alfabeto Σ , è detto:

1. **Turing-riconoscibile** (o *semi-decidibile* o *ricorsivamente enumerabile*) se esiste una MT M che lo *riconosce* o *accetta*, cioè:

$$\exists M \forall w \in \Sigma^*, \text{ se } w \in L, \text{ allora } M \text{ accetta } w.$$

2. **Turing-decidibile** (*ricorsivo*) se esiste una TM M che lo *decide*:

$$\exists M \forall w \in \Sigma^*, \text{ se } w \in L, \text{ allora } M \text{ accetta } w, \text{ altrimenti } M \text{ rigetta } w.$$

problemi e soluzioni

Senza perdere di generalità, tutti i problemi (matematici, linguistici, ecc.) possono essere riformulati in termini di linguaggi; avere una soluzione effettiva (praticabile in tempo finito) per un problema si traduce nell'avere un metodo (algoritmo) che verifichi se un elemento (stringa) appartiene ad un dato linguaggio (quello nel quale è formulato il problema).

Le domande che ci poniamo: dato **un qualsiasi linguaggio** L

1. L è **sempre riconoscibile** (semi-decidibile) da una TM?
cioè, esiste una TM tale che se $w \in L$, allora M accetta w ?
2. e, se L è riconoscibile, è anche **sempre decidibile** da una TM?
cioè, esiste sempre una TM M tale che se $w \in L$, allora M accetta w , altrimenti M rifiuta w ?
3. in caso di risposta negativa a(d una de)lle domande 1-2:
 - è possibile fare di meglio?
 - è possibile trovare un modello computazione più potente?

esistono più linguaggi che TM (1/3)

Per provare l'esistenza di linguaggi non riconoscibili (e non decidibili), si può ricorrere al **metodo della diagonalizzazione**, introdotto da Georg Cantor nel 1873, per la misurazione della **taglia di un insieme**.

Due insiemi A e B hanno la **stessa taglia** se gli elementi di un insieme possono essere accoppiati con gli elementi dell'altro, cioè se esiste una

corrispondenza $f : A \rightarrow B$ così definita

1. ogni elemento di A corrisponde ad un solo elemento di B e
2. ogni elemento di B ha un unico elemento di A corrispondente.

Esempio di corrispondenza tra Naturali (\mathcal{N}) e Pari (\mathcal{E})

$$f : \mathcal{N} \rightarrow \mathcal{E}$$

$$f : n \mapsto 2n$$

$$f(n) = 2n$$

n	$f(n)$
1	2
2	4
3	6
\vdots	\vdots

Un insieme è **numerabile** se è finito o se ha la stessa taglia dell'insieme dei numeri naturali \mathcal{N} . Osserva: \mathcal{Q} (razionali) è numerabile, \mathcal{R} (reali) no!

esistono più linguaggi che TM (2/3)

1. l'insieme Σ^* di tutte le **stringhe** su $\Sigma = \{0, 1\}$: Σ^* è numerabile.
Ogni TM M può essere *condificata* in una stringa $\langle M \rangle \in \Sigma^*$.
Quindi l'insieme di tutte le Macchine di Turing è numerabile!
2. sia $A \subseteq \Sigma^*$ un linguaggio (es. tutte le stringhe che iniziano con 0)
3. χ_A è la **sequenza binaria caratteristica** (infinita!) di A

$$\Sigma^* = \{ \epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \dots \}$$

$$A = \{ 0, 00, 01, 000, 001, \dots \}$$

$$\chi_A = 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad \dots$$

esistono più linguaggi che TM (3/3)

4. l'insieme di tutti i linguaggi \mathcal{L} può essere visto come l'insieme di tutte le **sequenze caratteristiche infinite**: \mathcal{L} non è numerabile!

altrimenti, seguendo l'argomento diagonale posso immaginare una sequenza caratteristica che abbia il seguente comportamento: dove nella diagonale c'è 1 la sequenza ha 0 e viceversa, dove lungo la diagonale c'è 0 la sequenza avrà 1. Questa sequenza rappresenterà un linguaggio che come tale dovrà essere enumerato tra gli altri linguaggi. Ma questo linguaggio non può essere enumerato perchè altrimenti la diagonale lo attraverserebbe in un punto (bit) che resterebbe pertanto indeterminato: per l'argomento diagonale infatti tale punto non potrebbe essere nè 0 nè 1, contraddicendo così la definizione di sequenza caratteristica.

n	$f(n)$
1	0
2	1
3	00
4	01
\vdots	\vdots

n	$f(n)$
1	01100...
2	11010...
3	00100...
4	01000...
\vdots	\vdots
x	1001...? $\forall n, x \neq f(n)$
\vdots	\vdots

Universal Turing Machine & Halting Problem

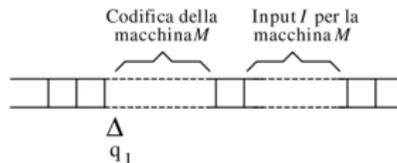
la UTM è una TM U il cui **input** u è composto da $\langle M, w \rangle$:

- la *codifica effettiva* di una TM, M
- un input w per M .

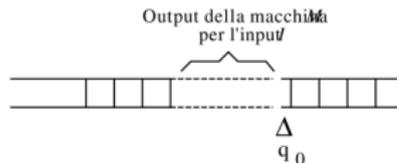
U sull'input $\langle M, w \rangle$ simula M su W

- 1 **accetta** se M accetta w ,
- 2 **rigetta** se M rigetta w ,
- 3 **loop** se M loop su w .

Inizio del calcolo:



Fine del calcolo:



HALTING PROBLEM: $A_{TM} = \{\langle M, w \rangle \mid M \text{ è una TM che accetta } w\}$

A_{TM} è riconoscibile (semi-decidibile)! (...dalla U)

Problema: A_{TM} è decidibile?

(cioè, la UTM è modificabile in modo che si arresti su ogni input?)

Osserva: un sistema operativo (SO) può essere visto come una UTM che prende in input altri programmi assieme ai loro input. Esempio Windows è un SO che prende in input il programma Word assieme ad un file ".doc".

L'HALTING PROBLEM A_{TM} è indecidibile (1/2)

1. Assumiamo per assurdo che esista un **decisore** H per A_{TM} :

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

2. Costruiamo una TM **duale** D che sull'input $\langle M \rangle$, se M è una TM:
 - 1 – lancia H sull'input $\langle M, \langle M \rangle \rangle$
 - 2 – restituisce in output l'opposto dell'output di H .

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

se $M = D$, allora:

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

contraddizione!

L'HALTING PROBLEM A_{TM} è indecidibile (2/2)

H decide A_{TM} : accetta $\langle M, \langle M \rangle \rangle$ se M accetta $\langle M \rangle$, altrimenti rifiuta:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$...
M_1	accept		accept		
M_2	accept	accept	accept	accept	
M_3					...
M_4	accept	accept			
\vdots		\vdots			

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$...
M_1	accept	reject	accept	reject	
M_2	accept	accept	accept	accept	...
M_3	reject	reject	reject	reject	
M_4	accept	accept	reject	reject	
\vdots		\vdots			

D è il duale di H : computa l'opposto dei valori della diagonale di H .

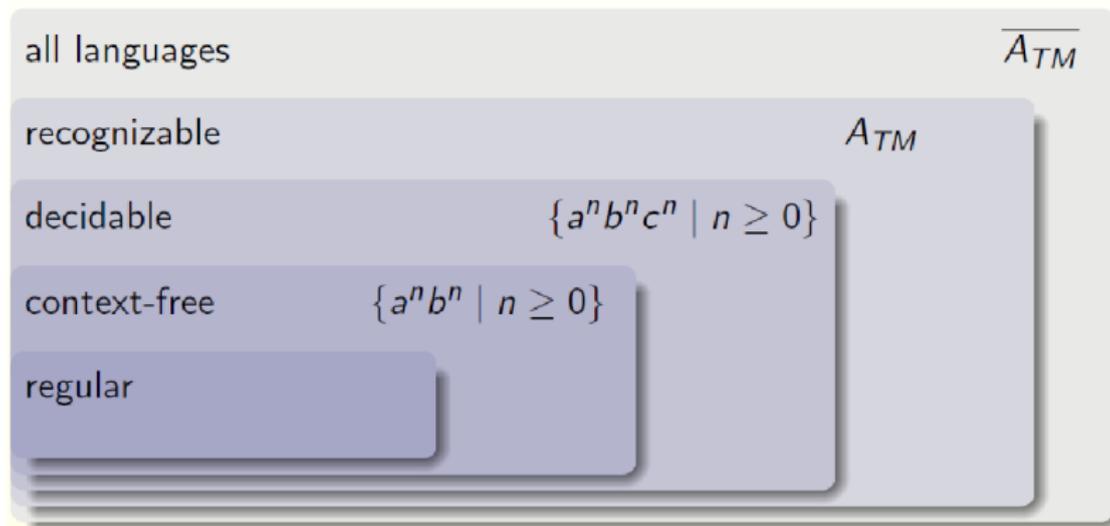
È una TM e deve quindi apparire nella tabella del comportamento di H .

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$...	$\langle D \rangle$...
M_1	accept	reject	accept	reject		accept	
M_2	accept	accept	accept	accept		accept	...
M_3	reject	reject	reject	reject	...	reject	...
M_4	accept	accept	reject	reject		accept	
\vdots		\vdots			\ddots		
D	reject	reject	accept	accept		<u>?</u>	
\vdots		\vdots					\ddots

contraddizione!

linguaggi non riconoscibili

- ▶ Complemento \bar{L} di un linguaggio L è l'insieme delle stringhe $\notin L$.
- ▶ Un linguaggio L è decidibile sse L e \bar{L} sono T-riconoscibili.
- ▶ $\overline{A_{TM}}$ non è T-riconoscibile (altrimenti A_{TM} sarebbe decidibile!)



oltre le macchine di Turing?

abbiamo quindi risposto a 2 delle 3 domande poste:

1. esistono linguaggi/problemi non riconoscibili (neppure semi-decidibili) da una TM? **SI!**
2. esistono linguaggi/problemi riconoscibili (semi-decidibili) da una TM ma che non possono essere decisi (risolti) da alcuna TM? **SI!**
3. possiamo andare oltre le Macchine di Turing per catturare e rendere più potente la nozione di soluzione effettiva (algoritmica) di un problema (allargando la classe dei problemi decidibili)?
Se assumiamo

tesi di Church-Turing:

“nozione intuitiva di algoritmo” \equiv “macchina di Turing”

allora la risposta è **NO!**

il decimo problema di Hilbert è T-semi-decidibile

Consideriamo il caso semplificato di polinomi p ad una sola variabile

$$4x^3 - 2x^2 + x + 7.$$

Ecco una TM M in grado di riconoscere il seguente linguaggio:

$$D = \{p \mid p \text{ è un polinomio su } x \text{ con radice intera su } x\}$$

1. M prende input un polinomio p sulla variabile x .
2. M valuta p assegnando a x successivamente i valori $0, 1, -1, 2, -2, 3, 3, \dots$
3. se ad un certo punto il polinomio p si valuta 0 allora M accetta.

Se p ha una radice intera, allora M prima o poi la troverà e dunque accetterà p , altrimenti, M continuerà a computare all'infinito.

OSSERVA: se p ha n variabili, M' fa il test su *insiemi ordinati di valori interi differenti*.

il decimo problema di Hilbert non è T-decidibile

Convertiamo M (con una sola variabile) in un **decisore** via il risultato:

Lemma

se la radice intera r di un polinomio a 1 variabile esiste, allora

$$-k \frac{c_{max}}{c_1} \leq r \leq +k \frac{c_{max}}{c_1} \quad (1)$$

- k è il numero dei termini,
- c_{max} è il coefficiente con valore assoluto più grande e
- c_1 è il coefficiente del termine di grado massimo.

ESEMPIO: considerando $4x^3 - 2x^2 + x + 7 = 0$, abbiamo

$$-4\frac{7}{4} \leq x \leq +4\frac{7}{4} \quad \text{con } k = 4, c_{max} = 7 \text{ e } c_1 = 4.$$

Osserva infatti che $x = 1$.

Theorem (Matijasevic, 1970)

è impossibile calcolare il limite (1) per polinomi con $n > 1$ variabili.

Lecture consigliate ... per chi vuole approfondire

- ▶ D. Harel, *Computers Ltd* (... di facile lettura).
- ▶ M. Sipser, *Introduction to the theory of computation*.
- ▶ C. H. Papadimitriou, *Elements of the theory of computation*.

questioni di cardinalità

*There are more things in heaven and earth, Horatio,
Than are dreamt of in your philosophy.*

[W. Shakespeare, Hamlet, Act 1, Scene 5]