Macchine a responsabilità limitata

Breve Introduzione alla Teoria della Calcolabilità: II Parte - la Tesi di Turing-Church e problemi indecidibili

Roberto Maieli

Università degli Studi "Roma Tre"
maieli@uniroma3.it
http://logica.uniroma3.it/∼maieli/teaching

Mini-Corso di II Livello per studenti di Informatica e Filosofia

Il decimo problema di Hilbert

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

> X PROBLEMA: *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, $x = 5, y = 3$ e $z = 0$ 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 us un sistema notazionale chiamato λ-**calculus**,
- Turing us un modello di **macchina computazionale**.

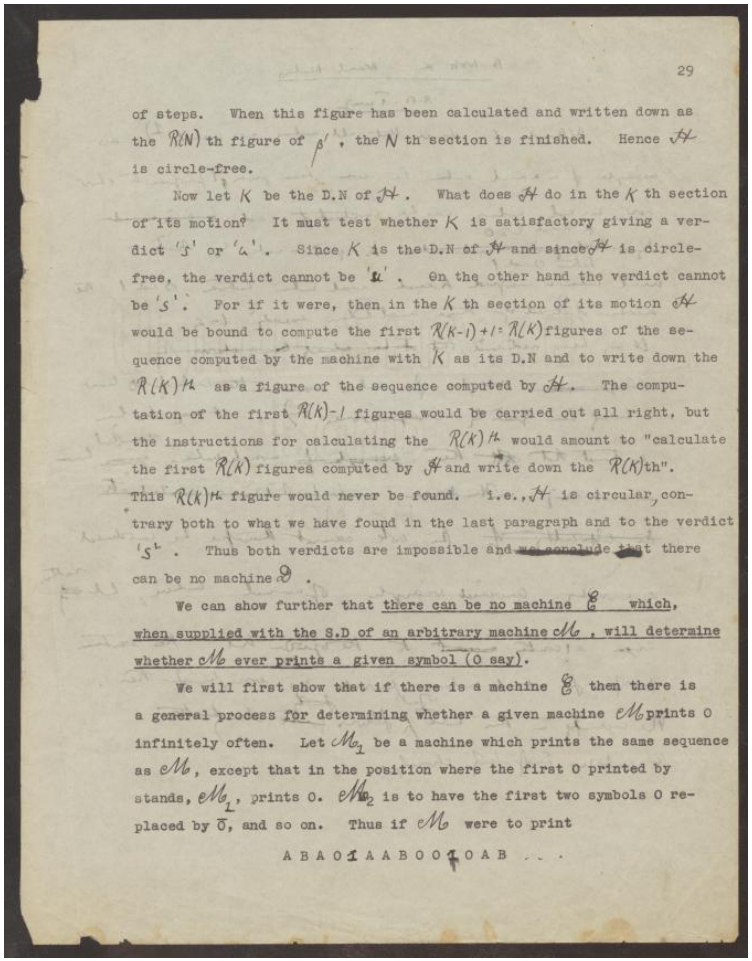
Queste due definizioni furono poi mostrate essere equivalenti.

Questa equivalenza diede origine alla seguente:

tesi di Church-Turing:

"nozione intuitiva di algoritmo" = "macchina di Turing"

Alan M. Turing



29

of steps. When this figure has been calculated and written down as the $R(N)$ th figure of β', the N th section is finished. Hence $\mathcal{A}$ is circle-free.

Now let K be the D.N of $\mathcal{A}$. What does $\mathcal{A}$ do in the K th section of its motion? It must test whether K is satisfactory giving a verdict 's' or 'a'. Since K is the D.N of $\mathcal{A}$ and since $\mathcal{A}$ is circle-free, the verdict cannot be 'a'. On the other hand the verdict cannot be 's'. For if it were, then in the K th section of its motion $\mathcal{A}$ would be bound to compute the first $R(K-1)+1 = R(K)$ figures of the sequence computed by the machine with K as its D.N and to write down the $R(K)$th as a figure of the sequence computed by $\mathcal{A}$. The computation of the first $R(K)-1$ figures would be carried out all right, but the instructions for calculating the $R(K)$th would amount to "calculate the first $R(K)$ figures computed by $\mathcal{A}$ and write down the $R(K)$th". This $R(k)$th figure would never be found. i.e., $\mathcal{A}$ is circular, contrary both to what we have found in the last paragraph and to the verdict 's'. Thus both verdicts are impossible and we conclude that there can be no machine $\mathcal{D}$.

We can show further that there can be no machine $\mathcal{E}$ which, when supplied with the S.D of an arbitrary machine $\mathcal{M}$, will determine whether $\mathcal{M}$ ever prints a given symbol (0 say).

We will first show that if there is a machine $\mathcal{E}$ then there is a general process for determining whether a given machine $\mathcal{M}$ prints 0 infinitely often. Let $\mathcal{M}_1$ be a machine which prints the same sequence as $\mathcal{M}$, except that in the position where the first 0 printed by stands, $\mathcal{M}_1$, prints 0. $\mathcal{M}_2$ is to have the first two symbols 0 replaced by $\overline{0}$, and so on. Thus if $\mathcal{M}$ were to print

A B A 0 1 A A B 0 0 1 0 A B . . .

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

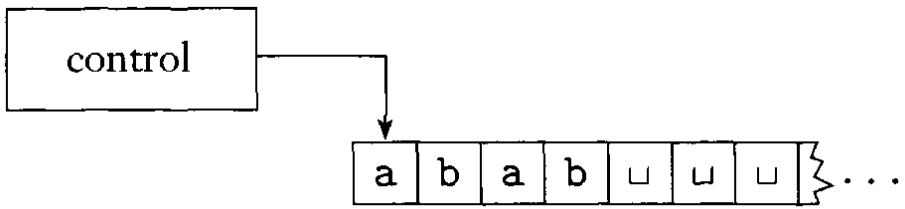
Un commento un pò acido

Turings "machines": These machines are humans who calculate.

[L. Wittgenstein,
Remarks on the Philosophy of Psychology, Vol. 1,
Blackwell, Oxford, 1980.]

Macchine di Turing (TM, 1936)

Simile agli automi a stati finiti, ma con un memoria illimitata.
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.



Ecco le principali differenze rispetto agli automi:

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,
4. gli stati finali di accettazione/rifiuto hanno effetto immediato.

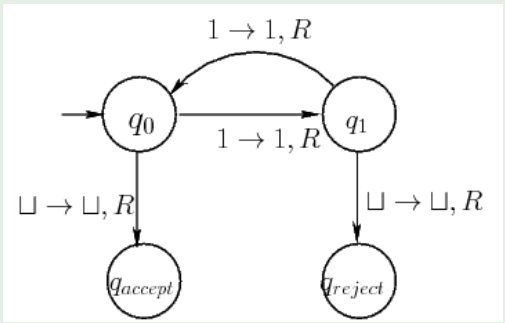
Definition (Macchina di Turing (MT))

Una TM è una 7-upla $\langle Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject} \rangle$ tale che:

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)

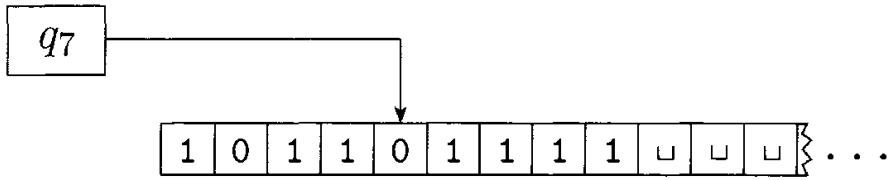
1. $Q = \{q_0, q_1, q_{accept}, q_{reject}\}$
2. $\Sigma = \{1\}$
3. $\Gamma = \Sigma \cup \{\sqcup\}$
4. δ è dato dal diagramma di stati:



linguaggio di una TM

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

$$\langle q_i,\ symbol \rangle$$



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

1. C_1 è la configurazione iniziale $(q_0.w)$
2. per ogni $1 \leq i \leq n$, C_{i+1} è ottenuto da C_i mediante a transizione di stato δ
3. C_n è una configurazione di accettazione $\langle q_0,\ symbol \rangle$

Linguaggi riconoscibili e decidibili da TM

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

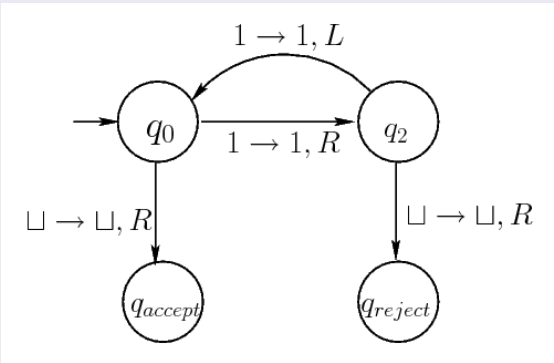
1. **Turing-riconoscibile** (o *semi-decidibile* o *ricorsivamente enumerabile*) se esiste una MT M che lo *riconosce* o *accetta*, cioè:

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

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

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

Remark (M su w può accettare, rigettare o non fermarsi)

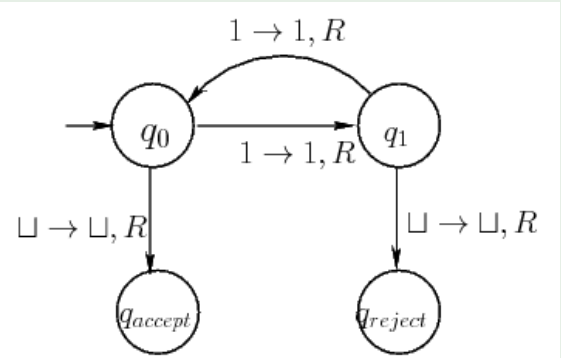


- accepts $w = \sqcup$ (halts)
- rejects $w = 1$ (halts)
- loops on $w = 11...1$ (does not halt)

M_1 ed M_2 "decidono" linguaggi "equivalenti" (modulo codifica input)

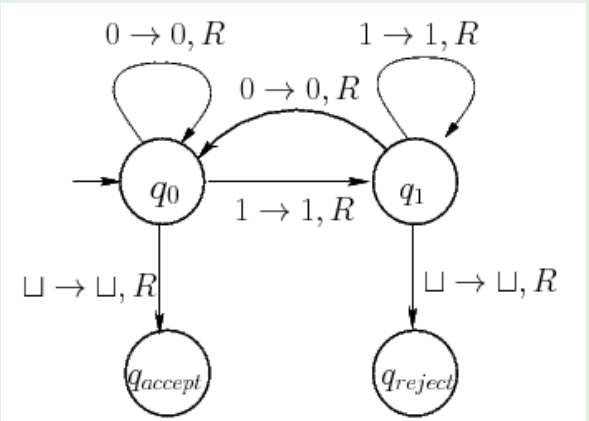
Example (M_1 accetta le sequenze unarie vuote o con un numero pari di 1)

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:



Example (M_2 accetta le sequenze binarie vuote o che terminano con 0)

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:



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) ad un problema si traduce nell'avere un metodo (algoritmo) che verifichi se un elemento (stringa) appartiene ad un dato linguaggio.

Domande: sia L **un linguaggio qualsiasi**:

1. L è **sempre riconoscibile** (semi-decidibile) da una TM?
   esiste una TM tale che se $w \in L$ allora M termina con accettazione di w?
2. se L è riconoscibile è anche **decidibile** (risolvibile) da una TM?
   esiste una TM M tale che se $w \in L$ allora M termina con l'accettazione di w altrimenti termina con il rifiuto di 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 (1/2)

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 $f : A \to B$ 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 ($\mathcal{E}$)

$$f : \mathcal{N} \to \mathcal{E}$$
$$f : n \mapsto 2n$$
$$f(n) = 2n$$

n	$f(n)$
1	2
2	4
3	6
⋮	⋮

Un insieme è **numerabile** se è finito o se ha la stessa taglia di $\mathcal{N}$
Osserva; $\mathcal{Q}$ (razionali) è numerabile, $\mathcal{R}$ (reali) no!

esistono più linguaggi che TM (2/2)

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

$$\begin{array}{rccccccccccc} \Sigma^* = \{ & \varepsilon & , & 0 & , & 1 & , & 00 & , & 01 & , & 10 & , & 11 & , & 000 & , & 001 & , & \cdots & \} \\ A = \{ & & & 0 & , & & & 00 & , & 01 & , & & & & & 000 & , & 001 & , & \cdots & \} \\ \chi_A = & 0 & & 1 & & 0 & & 1 & & 1 & & 0 & & 0 & & 1 & & 1 & & \cdots & \end{array}$$

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!

n	$f(n)$
1	0
2	1
3	00
4	01
⋮	⋮

n	$f(n)$
1	01100...
2	11010...
3	00100...
4	01000...
⋮	
x	1001...?

$\forall n, x \neq f(n)$

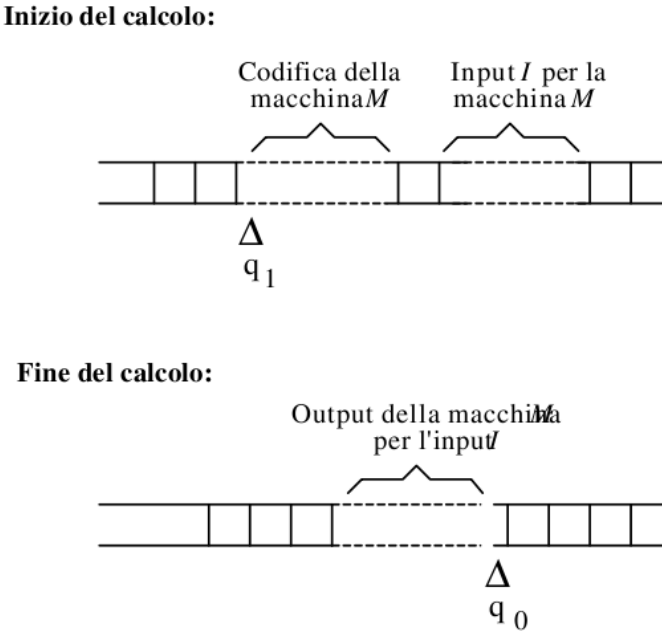
Universal Turing Machine & Halting Problem

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.



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?)

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

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

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

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.

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

se $M = D$, allora:

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

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:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*accept*		*accept*		
M_2	*accept*	*accept*	*accept*	*accept*	
M_3					$\cdots$
M_4	*accept*	*accept*			
$\vdots$			$\vdots$		

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*accept*	*reject*	*accept*	*reject*	
M_2	*accept*	*accept*	*accept*	*accept*	$\cdots$
M_3	*reject*	*reject*	*reject*	*reject*	
M_4	*accept*	*accept*	*reject*	*reject*	
$\vdots$			$\vdots$		

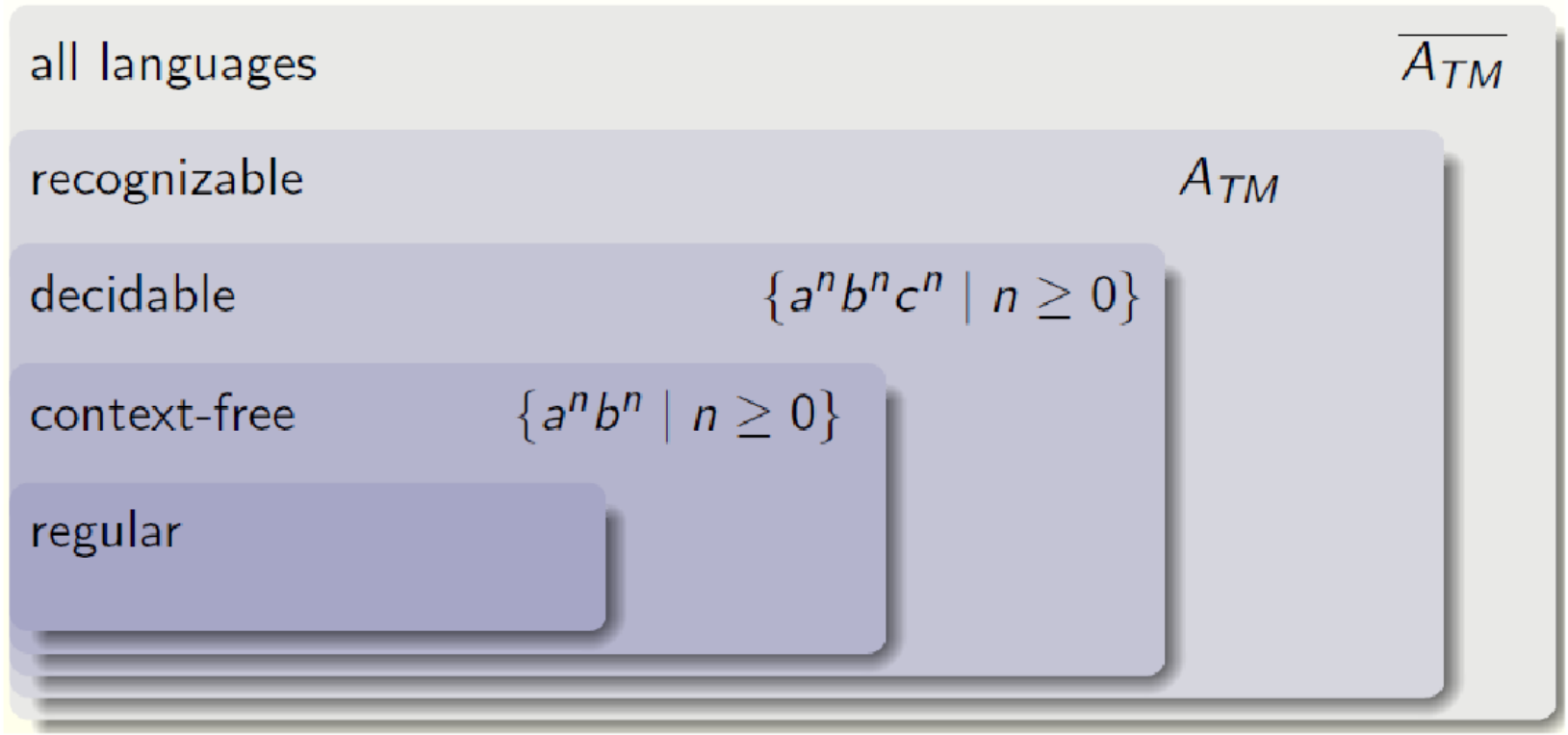
D è il duale di H: computa l'opposto dei valori della diagonale di H.
E' una TM e deve quindi apparire nella tabella del comportamento di H.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
M_1	*accept*	*reject*	*accept*	*reject*		*accept*	
M_2	*accept*	*accept*	*accept*	*accept*	$\cdots$	*accept*	$\cdots$
M_3	*reject*	*reject*	*reject*	*reject*		*reject*	
M_4	*accept*	*accept*	*reject*	*reject*		*accept*	
$\vdots$			$\vdots$		$\ddots$		
D	*reject*	*reject*	*accept*	*accept*		?	
$\vdots$			$\vdots$				$\ddots$

contraddizione!

gerarchia dei linguaggi/modelli computazionali

- Complemento $\overline{L}$ di un linguaggio L è l'insieme delle stringhe $\notin L$.
- Un linguaggio L è decidibile sse $\mathcal{L}$ e $\overline{\mathcal{L}}$ sono T-recognizable.
- $\overline{A_{TM}}$ non è T-recognizable (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 gli Automi e le TM 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 $0, 1, -1, 2, -2, 3, 3....$
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 varabile *esiste, allora*

$$-k\frac{c_{max}}{c_1} \leq r \leq +k\frac{c_{max}}{c_1} \tag{1}$$

– *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

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

Osserva infatti che $x = 1$.

Theorem (Matijasevic, 1970)

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

Bibliografia introduttiva essenziale

- D. Harel, *Computers Ltd.*
- C. H. Papadimitriou, *Elements of the theory of computation*.
- M. Sipser, *Introduction to 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]