

# Lezione 8

Macchine di Turing

agli inizi del '900 Hilbert si chiedeva se fosse possibile trovare un algoritmo (una sequenza di calcoli che termina sempre) in grado di decidere la verità o la falsità di qualsiasi proposizione matematica

La sua ipotesi era che un tale algoritmo esistesse

Kurt Gödel nel '31 mostrò che non era così

definì una formula nel calcolo dei predicati applicato agli interi che non poteva essere né dimostrata né confutata nello stesso calcolo

la logica dei predicati è un formalismo per esprimere funzioni che siano calcolabili

anche altri formalismi: lambda calcolo, regole di composizione di funzioni,....

Turing machines (1936) che sono un modello di computer

tutti i diversi formalismi definiscono la stessa classe di funzioni

Tesi di Church : questa è la classe delle funzioni calcolabili

i computer moderni definiscono questa classe di funzioni

esistono funzioni/problemi che non sono calcolabili

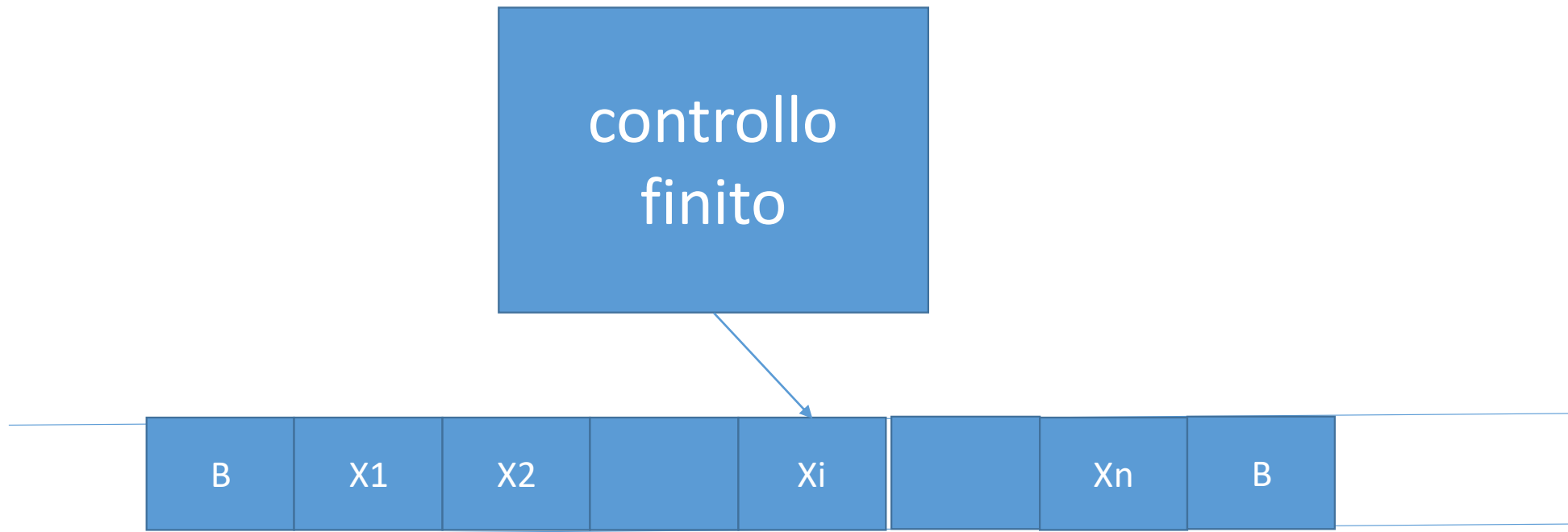
è importante saperlo per evitare di cercare di risolverli

da questa teoria sono nati strumenti e concetti utili anche per lo studio dell'intrattabilità delle funzioni/problemi

importante riconoscere problemi intrattabili: per affrontarli con euristiche

Le macchine di Turing sono importanti per una teoria generale perché sono semplici

# La macchina di Turing M



$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

$Q$ =stati,  $\Sigma$ =alfabeto input,  $\Gamma$ =alfabeto nastro,  $q_0$  in  $Q$  =iniziale,  
 $B$ =blank appartiene a  $\Gamma$ ,  $F \subseteq Q$ = stati finali,  $\delta$  = funzione di  
transizione

$$\delta(q, X) = (p, Y, D)$$

--q=stato corrente

--X simbolo del nastro sotto la testina

--p=nuovo stato

--Y = messo al posto di X

--D = left/right, L/R

descrizione istantanea

$X_1 \dots X_{i-1} \ p \ X_i \dots X_n$

all'inizio  $q_0 \ X_1 \dots X_n$  testina in 1

se  $\delta(p, X_i) = (q, W, L)$

allora nuova ID è:

$X_1 \dots X_{i-1} p X_i \dots X_n \vdash X_1 \dots q X_{i-1} W X_{i+1} \dots X_n$

casi particolari:

testina in 1)  $p X_1 \dots X_{i-1} X_i \dots X_n \vdash q B W X_2 \dots X_n$

testina in n e  $\delta(p, X_n) = (q, B, L)$

$X_1 \dots X_{i-1} X_i \dots p X_n \vdash X_1 \dots q X_{n-1}$

casi simili per  $i=1$  e  $n$  quando la mossa va a destra

una TM  $M$  che riconosce  $L = \{0^n 1^n \mid n \geq 1\}$

nastro iniziale contiene  $B^*$  w  $B^*$

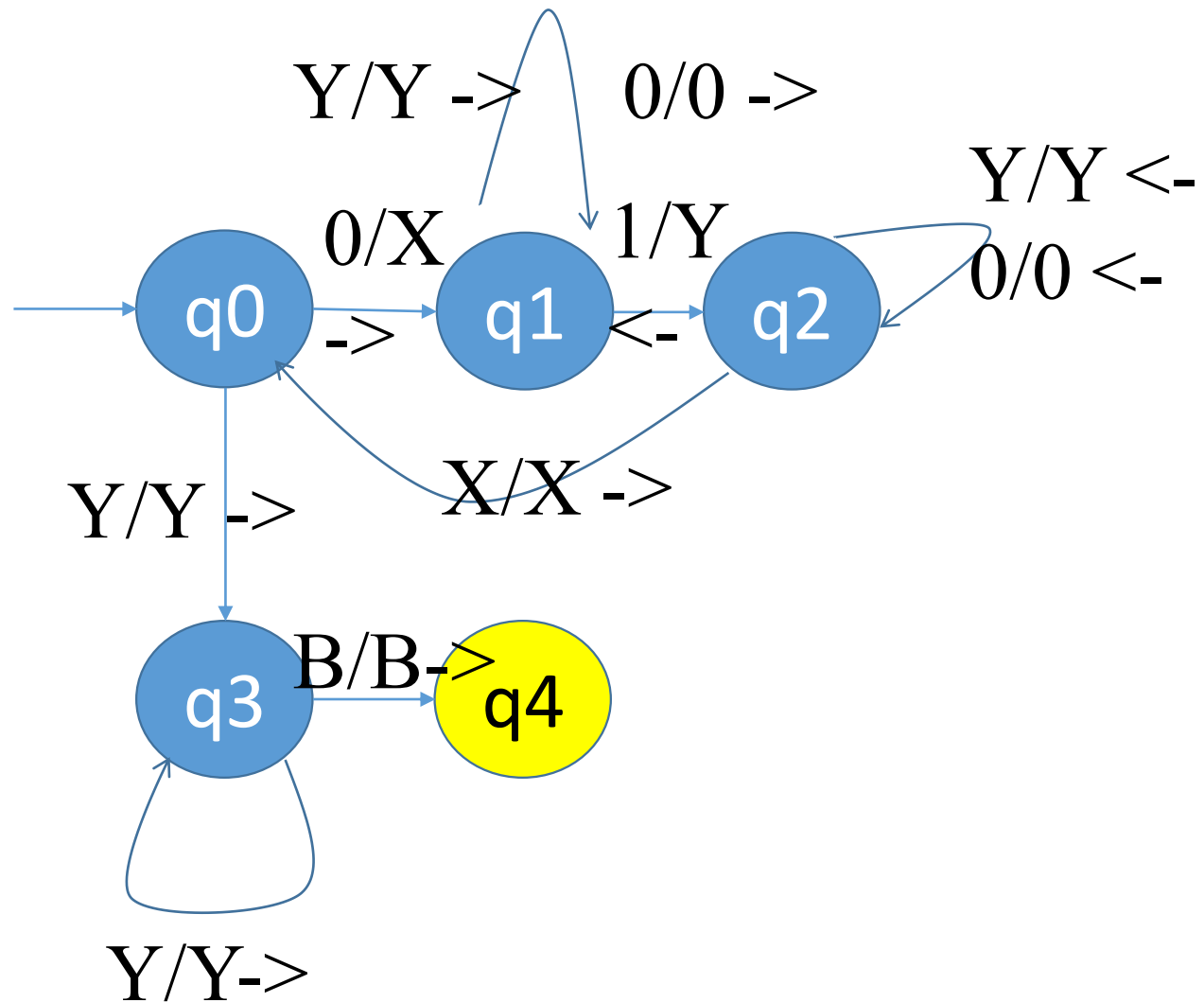
$M$  accetta  $w$  se e solo se  $w \in L$

$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$  dove  $\delta$  è:

stato	0	1	X	Y	B
$q_0$	$(q_1, X, R)$	-	-	$(q_3, Y, R)$	-
$q_1$	$(q_1, 0, R)$	$(q_2, Y, L)$	-	$(q_1, Y, R)$	-
$q_2$	$(q_2, 0, L)$	-	$(q_0, X, R)$	$(q_2, Y, L)$	-
$q_3$	-	-	-	$(q_3, Y, R)$	$(q_4, B, R)$
$q_4$	-	-	-	-	-



diagrammi di transizione



macchine di Turing che calcolano funzioni  $f$

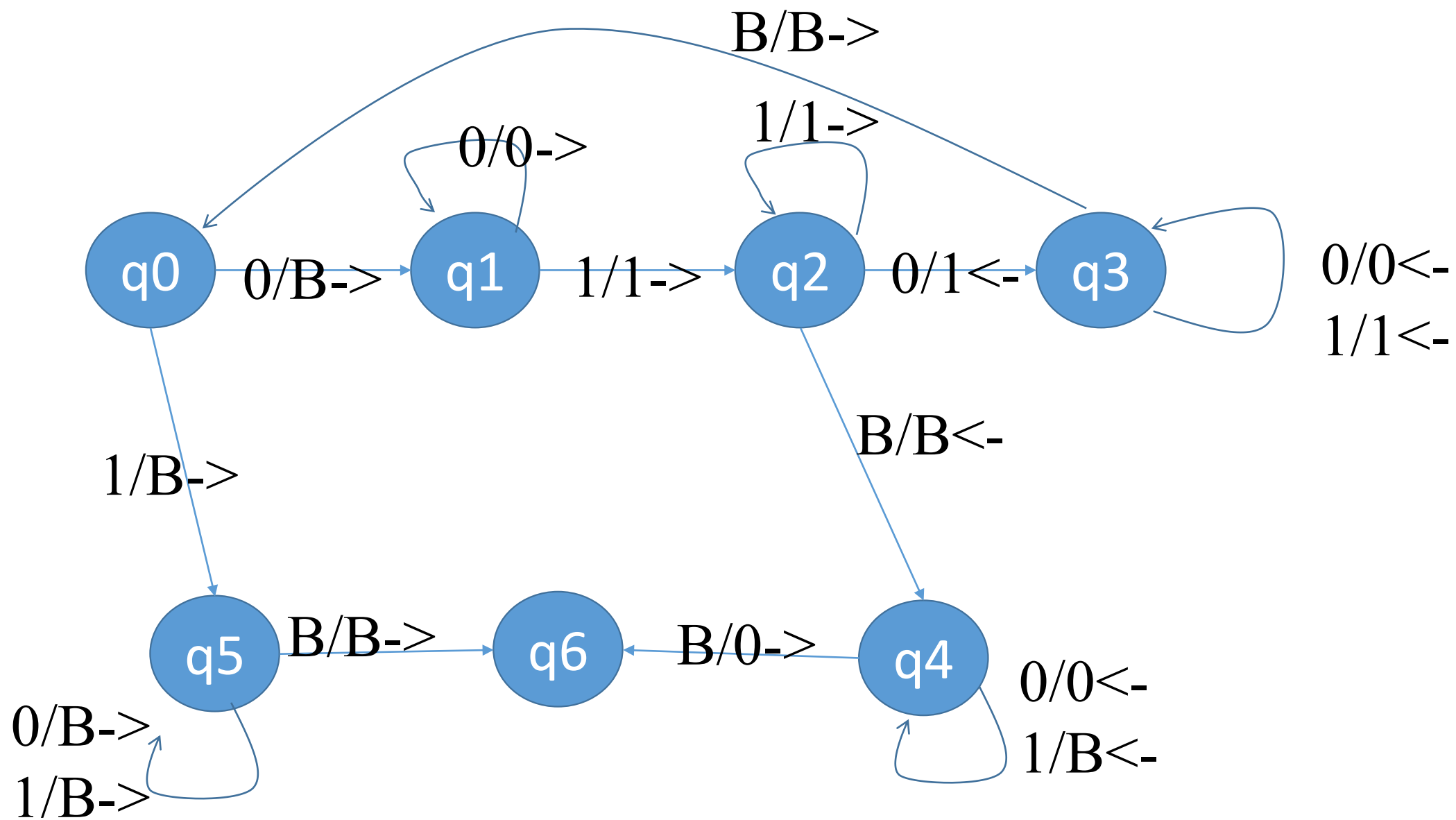
input iniziale  $x$

alla fine sul nastro scrivono  $f(x)$

esempio: sottrazione propria  $-.-$

$m.-.n =$  se  $m \geq n$  allora  $(m-n)$  altrimenti  $0$

macchine di Turing per calcolare funzioni :  $m \dot{-} n$  (sottrazione propria)



linguaggio riconosciuto da una macchina di Turing:

è l'insieme delle stringhe  $w$  in  $\Sigma^*$  t.c.  $q_0 w \vdash \alpha p \beta$  con  $p$  stato finale

i linguaggi accettati dalle TM sono detti ricorsivamente enumerabili (RE)

le TM accettano tutti i CFL e altro

facile col nastro simulare una pila

.....e il nondeterminismo dei PDA? Nelle TM non c'è differenza tra macchine deterministiche o no

macchine di Turing e l'arresto (terminazione del calcolo)

la TM che calcola la sottrazione propria si ferma sempre

possiamo fare in modo che negli stati finali la  $\delta$  sia indefinita,  
quindi le nostre TM, se accettano si fermano

e se non accettano? Possono fermarsi (in uno stato non finale)  
oppure continuare per sempre

se richiedessimo TM che si fermano sempre? Riconoscerebbero  
meno linguaggi

con TM che si fermano sempre riconosceremmo i linguaggi ricorsivi (decidibili)

si tratta di un sottoinsieme stretto di RE

TM che si ferma sempre = algoritmo

Se esiste un algoritmo per un linguaggio, allora il linguaggio è decidibile: la TM ci dice si/no per ogni istanza del problema

problema si/no  $\longleftrightarrow$  linguaggio  $\longleftrightarrow$  funzione  
P LP fP

istanza  $X$  di  $P \Rightarrow$  stringa  $w_X$ ,  $L_P = \{w_X \mid \text{per ogni istanza } X \text{ di } P \text{ per cui vale la risposta SI}\}$

una Turing machine che risolve  $P$  è  $M_P$  a cui diamo in input stringhe qualsiasi  $w$  ed accetta solo quelle di  $L_P$  (fermandosi)  
e se non accetta  $w$  allora ....

---si ferma comunque ( $L_P$  è decidibile)

---a volte si ferma e a volta continua per sempre (o  $L_P$  non è decidibile o abbiamo sbagliato TM)

linguaggio  $\longleftrightarrow$  funzione  
 $L_f$   $f$

$$L_f = \{(x, y) \mid f(x) = y\}$$

$L_f$  è il grafo di  $f$

macchina di Turing  $M_f$

riceve  $x$  in input e lascia  $f(x)$  sul nastro alla fine



es. 8.2.1 (a)\* e (b)

es. 8.2.2 (a)\* e (b), pensare anche a (c)

es. 8.2.5 (a)\*