Lezione 8

Macchine di Turing

agli inizi del ‘900 Hilbert si chiedeva se fosse possibile trovare un algoritmo 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 (36) 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

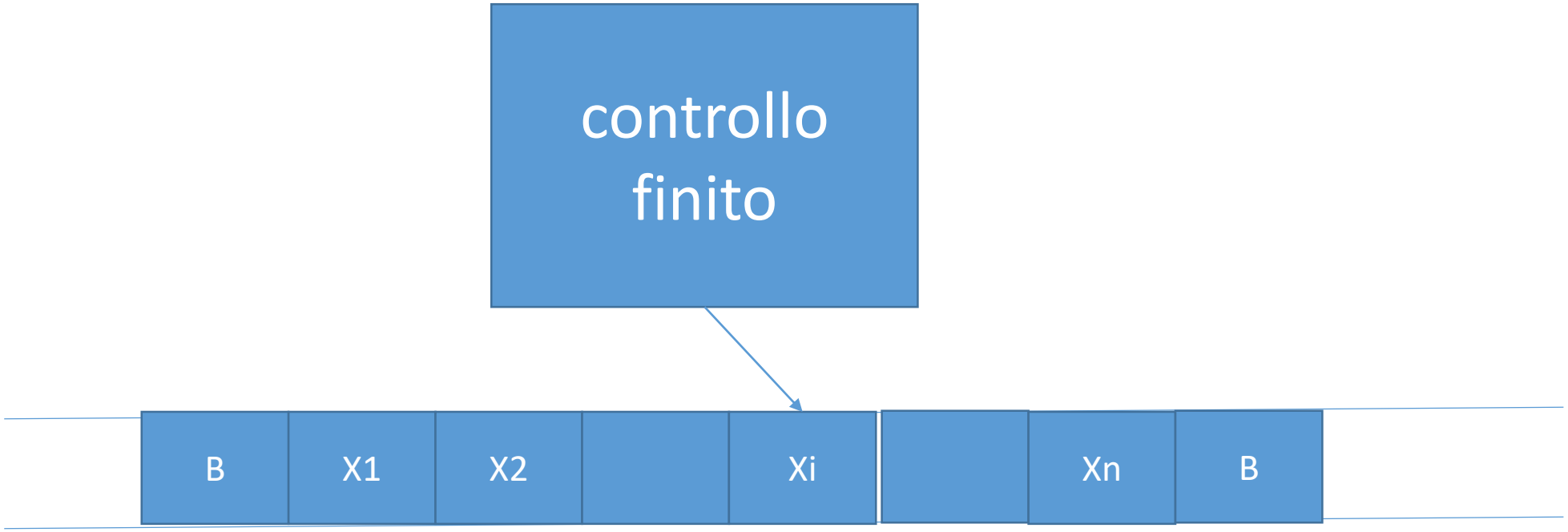
è importante saperlo per evitare di cercare di risolverli

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, ∑, Γ, δ, q0, B, F)

Q=stati, ∑=alfabeto input, Γ=alfabeto nastro, q0 in Q =iniziale, B=blank appartiene a Γ, F⊆ Q= stati finali, δ = 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

X1...Xi-1 p Xi...Xn

all'inizio q0 X1....Xn testina in 1

se $\delta(p,Xi)=(q,W,L)$
allora nuova ID è:
X1...Xi-1 p Xi...Xn |- X1....q Xi-1 W Xi+1...Xn

casi particolari:
testina in 1) pX1...Xi-1Xi...Xn |- q B W X2....Xn

testina in n e $\delta(p,Xn)=(q,B,L)$)
X1...Xi-1Xi...pXn |- X1...q Xn-1

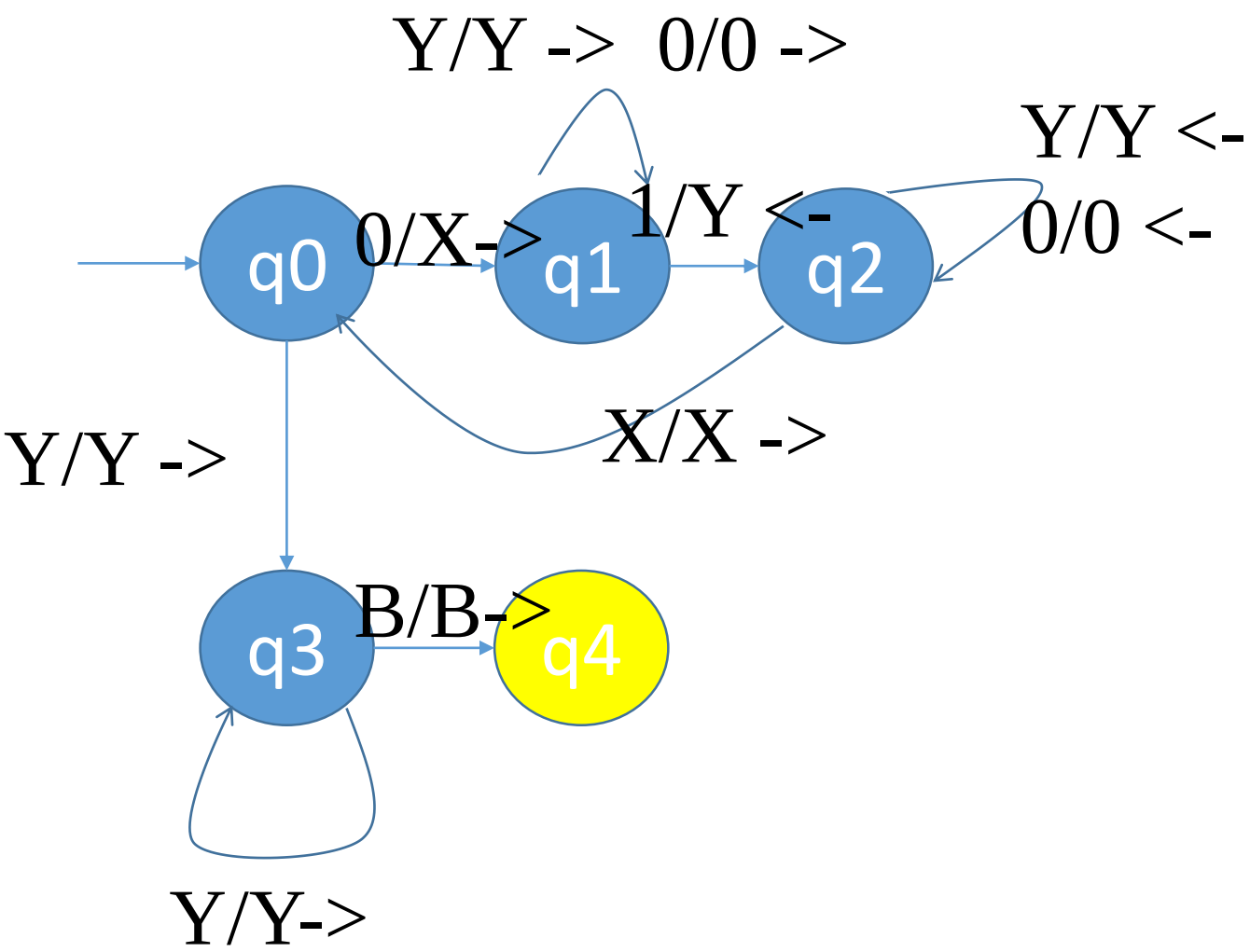
casi simili per i=1 e n quando la mossa va a destra

una TM M che riconosce $L=\{0^n1^n \mid n>=1\}$

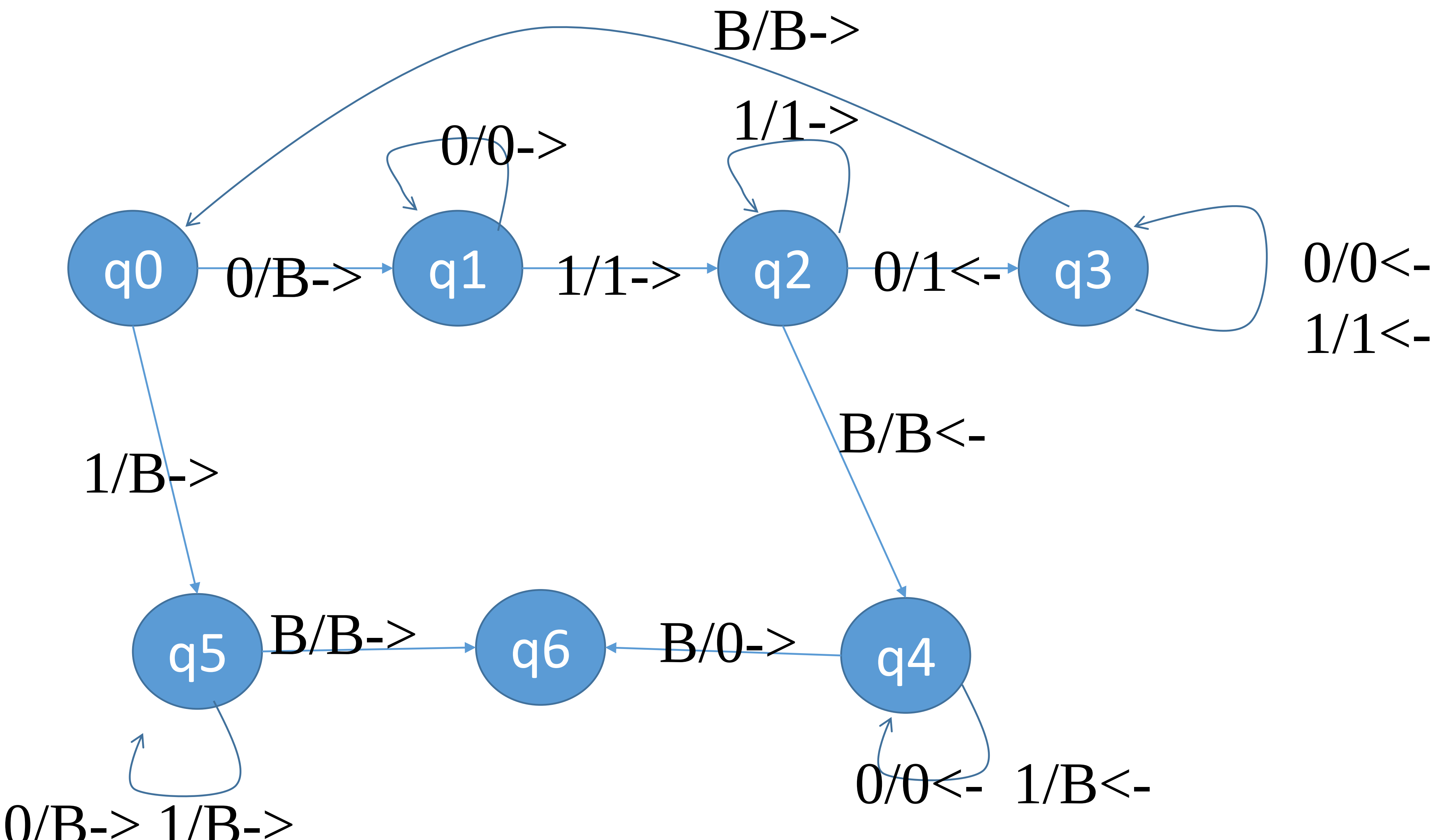
nastro iniziale contiene B* w B*
M accetta w se e solo se w in L
M= ({q0,q1,q2,q3,q4}, {0,1},{0,1,X,Y,B}, δ, q0, B, {q4}) dove δ è:

stato	0	1	X	Y	B
q0	(q1,X,R)	-	-	(q3,Y,R)	-
q1	(q1,0,R)	(q2,Y,L)	-	(q1,Y,R)	-
q2	(q2,0,L)	-	(q0,X,R)	(q2,Y,L)	-
q3	-	-	-	(q3,Y,R)	(q4,B,R)
q4	-	-	-	-	-

diagrammi di transizione



macchine di Turing per calcolare funzioni : m -· n (sottrazione propria)



linguaggio riconosciuto da una macchina di Turing:
è l'insieme delle stringhe w in $\sum^*$ t.c. q0w |- α p β con p stato finale

i linguaggi accettati dalle TM sono detti recursivamente 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 calcolava la sottrazione propria si ferma sempre

possiamo fare in modo che negli stati finali la δ 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 P <-----> linguaggio

istanza X di P=> stringa w_X, L_P={w_X | per ogni istanza X di P 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
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)

es. 8.2.1 (a)* e (b)
es. 8.2.2 (a)* e (b), pensare anche a (c)

es. 8.2.5 (a)*

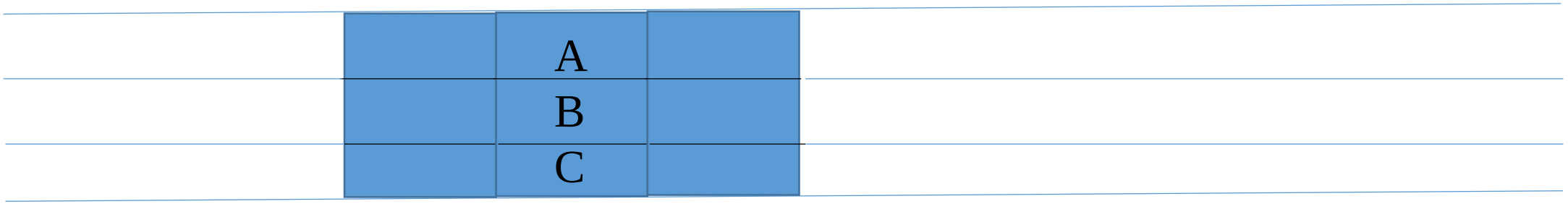
Tecniche di programmazione per le TM

1) memoria (finita nello stato) [q, α] con α in Γ+
esempio: riconoscere 01* + 10*

([q0,B],0) =([q0,0],R)
([q0,0],1)=([q0,0],R)
([q0,0],B)=(qf,B)

in modo simile si tratta il caso in cui il primo simbolo dell'input è 1

2) tracce multiple



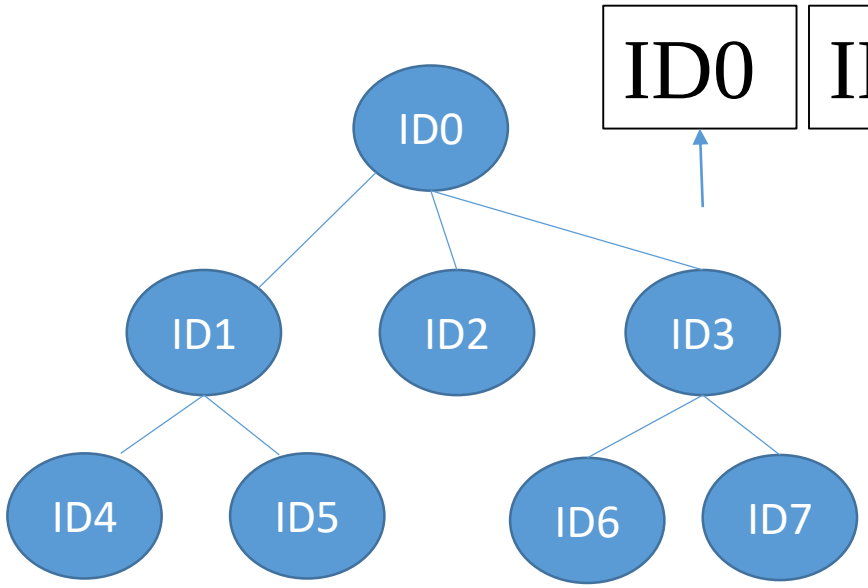
è una sola cella con 3 simboli che possiamo trattare
in modo indipendente
si calcola su una traccia lasciando le altre costanti

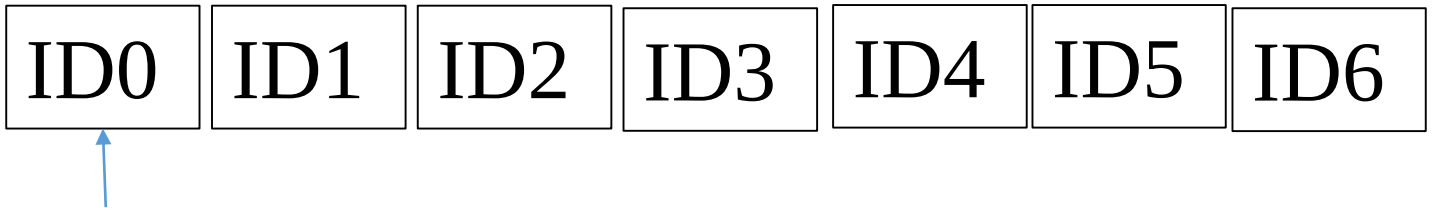
3) subroutine

estensioni del modello base

1) multi nastro => simulabile con multi track (aumento quadratico del tempo di esecuzione)

2) nondeterminismo => simulabile da TM deterministica





ID corrente si muove verso destra si seguono tutte le strade contemporaneamente

nonterminismo -> determinismo
la simulazione costa tempo esponenziale

es. 8.2.4*

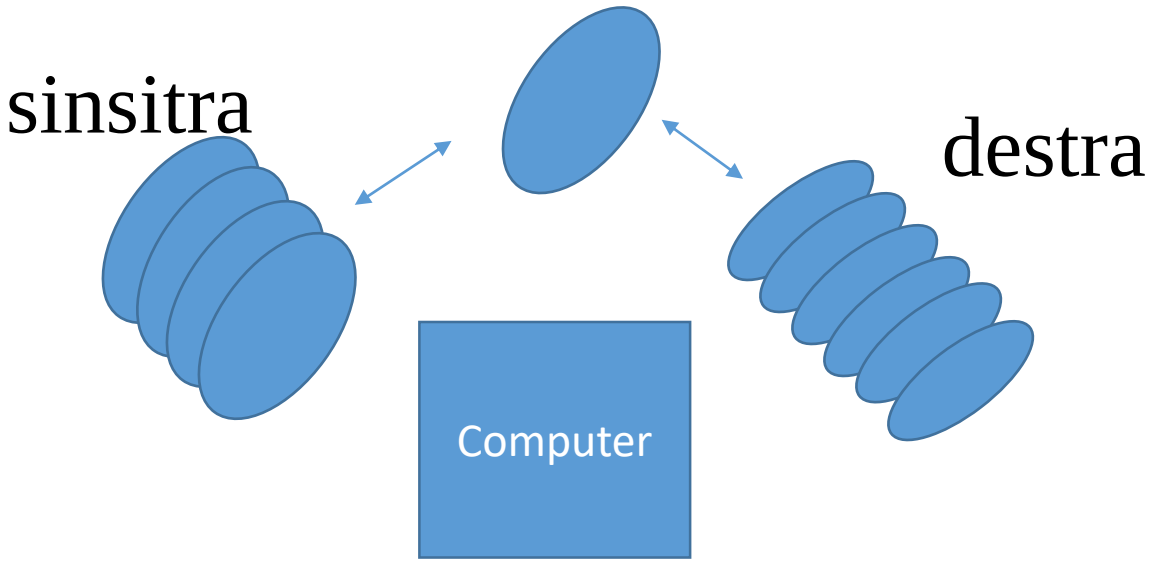
calcolo di funzioni <--> linguaggi

funzione f (interi positivi) = grafo =[x,f(x)] che è un linguaggio

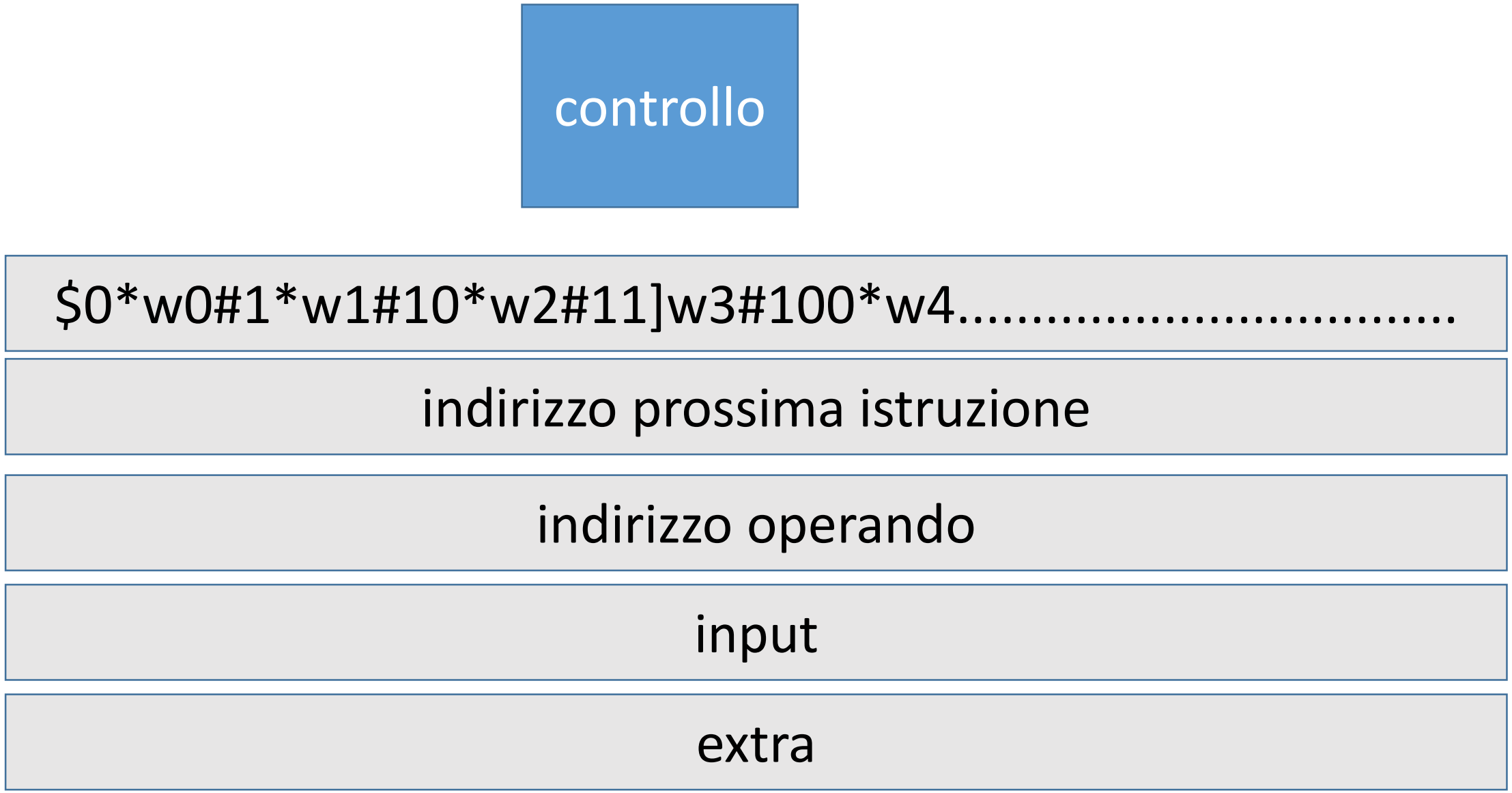
una TM calcola f se, dato x sul nastro, si ferma con f(x) sul nastro

un computer reale può simulare una TM, unico problema è il nastro potenzialmente illimitato

si può simulare con dispositivi di memoria di massa che si possono aggiungere al computer quando serve



una TM può simulare un Computer



e il computer ci mette n passi la TM ce ne mette n^c

relazione polinomiale