Automi e Linguaggi Formali

Macchine di Turing, indecidibilità e problemi intrattabili

Lamberto Ballan
lamberto.ballan@unipd.it

Padova, 16 Maggio 2017

Università degli Studi di Padova

Comunicazioni / calendario delle lezioni

- Settimana prossima faremo lezione di martedì mattina + pomeriggio (11:30-12:15, 13:30-15:30) e venerdì (13:30-15:30)

- **Il compitino**: in data 16/6/2017, orario 13:00-15:30
  - possono partecipare tutti: sia chi ha sostenuto e passato il I compitino, sia chi non l'ha passato che chi non l'ha sostenuto
  - i primi due appelli (giugno e luglio) saranno divisi in due parti; chi ha superato una delle due parti - o vuol provare a migliorarne il voto - può farlo consegnandone soltanto una

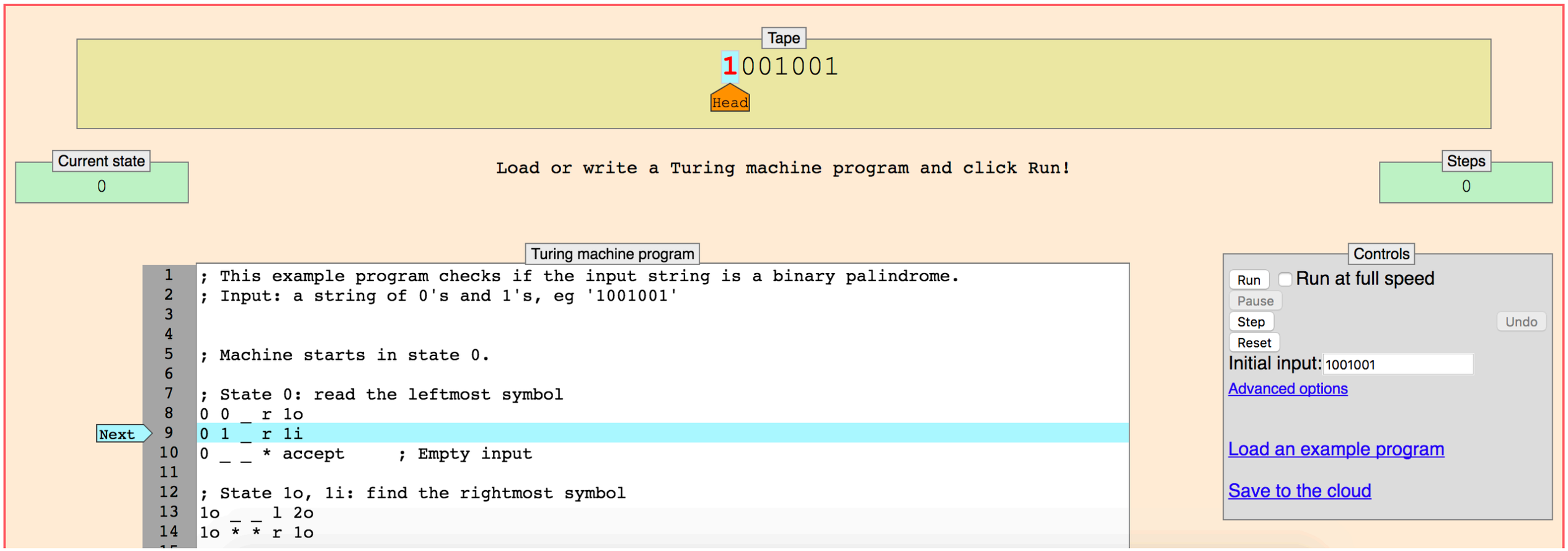
Simulatore di macchine di Turing su web

Turing machine simulator

[Back to home page]

This is a Turing machine simulator. To use it:

1. Load one of the example programs, or write your own in the Turing machine program area. See below for syntax.
2. Enter something in the 'Input' area - this will be written on the tape initially as input to the machine. Click 'Reset' to initialise the machine.
3. Click on 'Run' to start the Turing machine and run it until it halts (if ever). Click on 'Pause' to interrupt the Turing machine while it is running. Alternately, click 'Step' to run a single step of the Turing machine.
4. Click 'Reset' to restore the Turing machine to its initial state so it can be run again.



http://morphett.info/turing/turing.html

Simulatore di macchine di Turing su web

Esempio: definiamo una TM che accetta $L = \{0^n1^n : n \geq 1\}$

```
; This example program checks if the input string is in {0^n1^n: n>=1}
; Input: a string of 0's and 1's, eg '01','0011','000111',...

; Machine starts in state 0.

; State 0: read the leftmost symbol, change 0 in X, move right and search for 1
0 0 X r 1
0 Y Y r 3

; State q1: move right if 0 or Y until 1; once 1 is found change in Y and move
left
1 0 0 r 1
1 Y Y r 1
1 1 Y l 2

; State q2: move left if 0 or 1 until X and then iterate again
2 0 0 l 2
2 Y Y l 2
2 X X r 0

; State q3: count Y and move right
3 Y Y r 3
3 _ _ r 4

; State q4:
4 * * * halt-accept
```

TM che accetta L delle stringhe con stesso num 0,1

- **Esercizio:** Definire la TM che accetta il linguaggio L costituito dalle stringhe con stesso numero di 0 e 1 (es. 8.2.2a del libro)
  - {01,10,0011,0101,1001,1010,1100,0110,…}
- La TM usa i simboli *X* e *Y* per sostituire “coppie” di 0 e 1
  - *X* garantisce che non ci sono 0 o 1 alla sua sinistra (quindi la testina non può muoversi a sinistra di una *X*)
  - Y invece può avere 0 o 1 alla sua sinistra

TM che accetta L delle stringhe con stesso num 0,1

- Costruzione della TM:
  - partendo da q_0, i due stati q_1 e q_2 tengono memoria di quando abbiamo letto un 1 o uno 0 (cancelliamo poi il simbolo con *X*)
  - nello stato q_1 la TM muove verso destra cercando uno 0; se lo trova lo sostituisce con *Y*, entra in q_3, e cerca una *X* a sinistra
  - nello stato q_2 la TM muove verso destra cercando un 1; se lo trova lo sostituisce con *Y*, entra in q_3, e cerca una *X* a sinistra
  - nello stato q_3 la TM muove a sinistra in cerca di una *X*; se la trova entra in q_0 e il ciclo inizia nuovamente
  - la TM termina accettando l'input se nello stato q_0 trova B (cioè tutti gli 0 e 1 sono stati accoppiati)

TM che accetta L delle stringhe con stesso num 0,1

- La TM che calcola tale operazione é specificata da

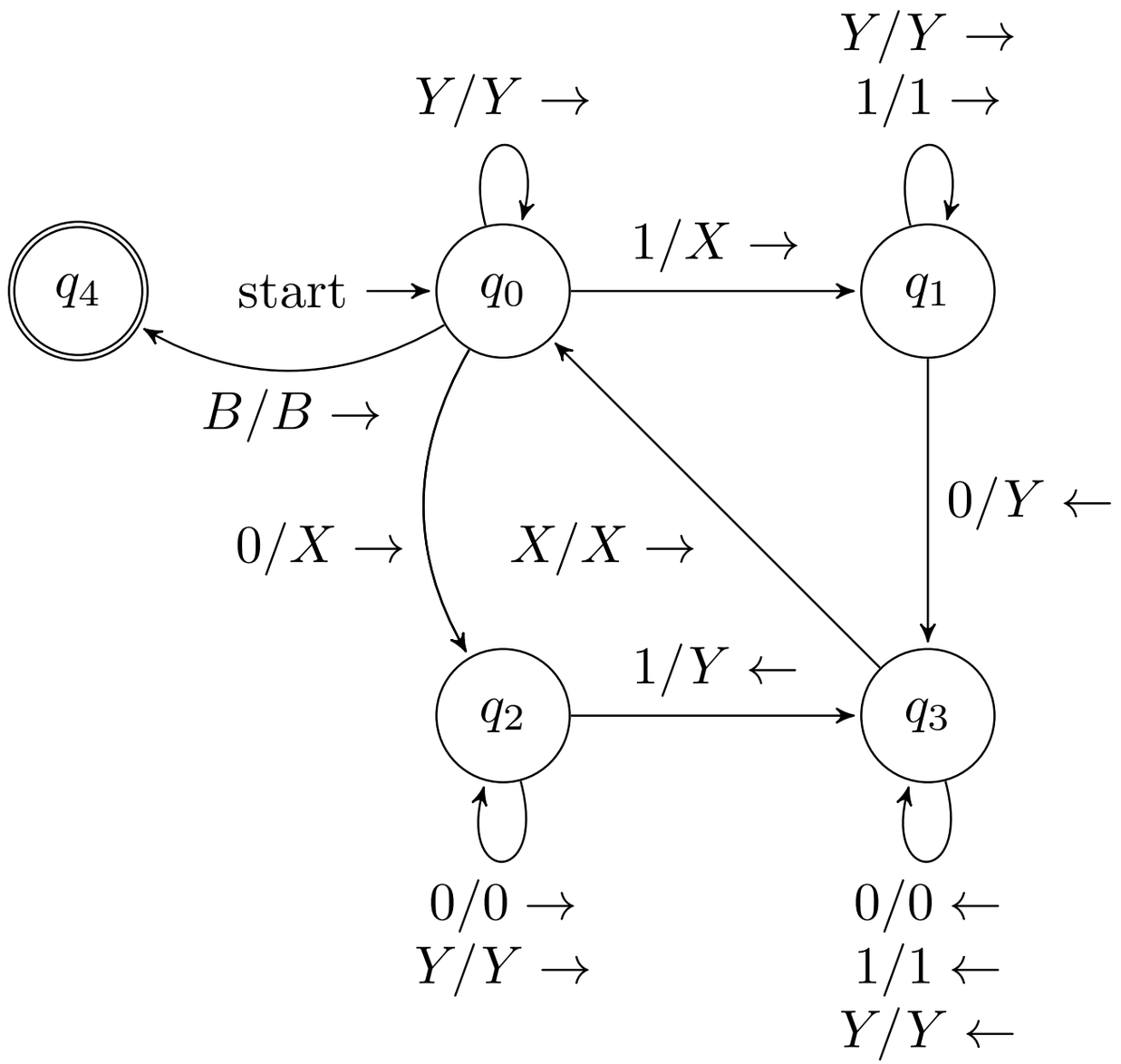
$$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$$

- La TM é definita dalla tabella di transizione:

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

TM che accetta L delle stringhe con stesso num 0,1

- Il diagramma di transizione è:



Esercizi

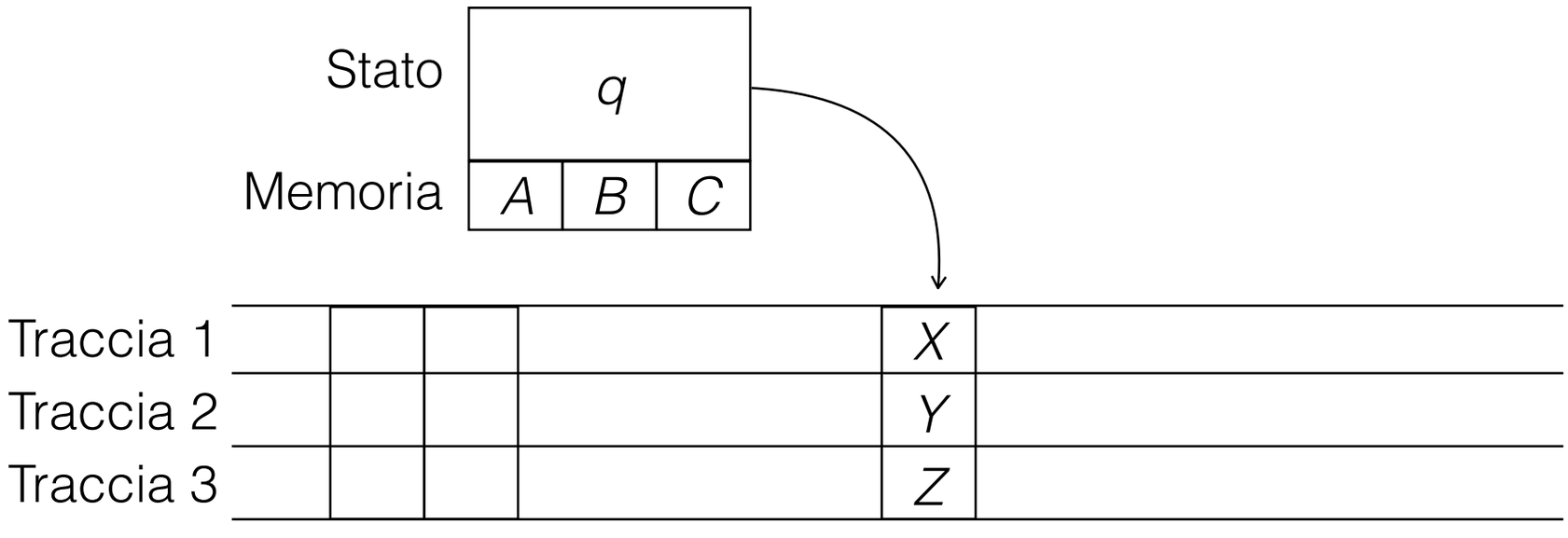
- Scrivere le ID della TM appena definita quando il nastro di input contiene la sequenza di simboli (in codifica unaria):
  - 1010
  - 0110
  - 10011

Tecniche di programmazione per le TM

- Una TM può essere utilizzata per calcolare in modo simile ad un computer tradizionale
- In particolare la TM ha la capacità “introspettiva” di svolgere elaborazioni su altre macchine di Turing
- Questa particolare capacità ci permetterà di dimostrare che certi problemi sono indecidibili
- Innanzitutto vediamo due semplici generalizzazioni del modello fondamentale

Tecniche di programmazione per le TM

- Memoria nello stato e tracce multiple
  - il controllo può essere utilizzato non solo per indicare la posizione nel programma, ma anche a memorizzare dati
  - occorre considerare lo stato come una ennupla, es. [q,A,B,C]
  - analogamente possiamo considerare una TM con più “tracce”, ognuna con un simbolo distinto, es. [X,Y,Z]

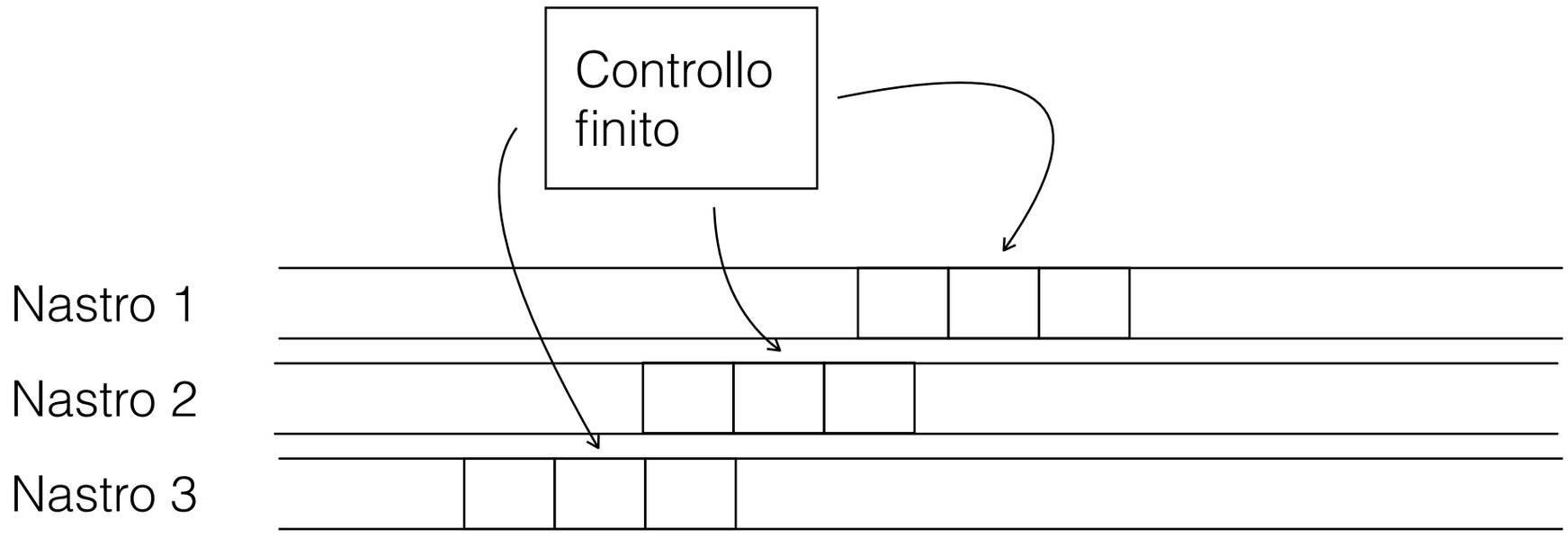


Tecniche di programmazione per le TM

- Come nel caso dei linguaggi di programmazione anche con le macchine di Turing si possono definire *subroutine*
  - una subroutine di una TM é un insieme di stati che eseguono una particolare procedura
  - in questo insieme di stati definiamo uno stato iniziale e uno stato temporaneamente senza mosse che serve da “ritorno”
  - parliamo di “chiamata” ad una subroutine quando c’è una transizione al suo stato iniziale
  - nota: la TM non può ricordare lo stato di ritorno; se vogliamo chiamare la subroutine da più stati dobbiamo farne più copie

(vedi esempio 8.8 sul libro - TM che calcola la moltiplicazione)

La macchina di Turing multinastro



Una **TM multinastro** ha un controllo finito e un numero finito di nastri. Inizialmente l'input si trova sul primo nastro, e gli altri nastri sono tutti vuoti. La testina del I nastro é all'estremità sinistra dell'input, le altre testine si trovano in posizioni arbitrarie.

In una mossa, la TM multinastro compie tre azioni:

- Cambia stato
- Su ogni cella visitata da una testina scrive un simbolo di nastro
- Ogni testina si muove indipendentemente a sinistra, destra o sta ferma

La macchina di Turing multinastro

- Una TM mononastro è un caso particolare di TM multinastro
  - domanda: esistono linguaggi non ricorsivamente enumerabili ma accettati da una TM multinastro?
  - risposta: no
  - Teorema (8.9): ogni linguaggio accettato da una TM multinastro è ricorsivamente enumerabile

    D: lo si dimostra simulando la TM multinastro con una TM mononastro a tracce multiple (vedi dim. nel paragrafo 8.4.2)

Tempo di esecuzione di una TM, complessità

- Definiamo tempo di esecuzione di una macchina di Turing *M*, dato l'input *w*, il num di passi che *M* compie prima di arrestarsi
  - se *M* non si arresta su *w* il tempo di esecuzione é infinito
- La complessità in tempo di *M* su input *w* é la funzione *T*(*n*) definita come il max, su tutti gli input *w* di lunghezza *n*, del tempo di esecuzione di *M* su *w*

- Nota: la complessità in tempo polinomiale è lo "spartiacque" fra problemi risolvibili e problemi irrisolvibili da un computer

Macchine di Turing non deterministiche

- Una macchina di Turing non deterministica (NTM) si distingue dalla funzione di transizione δ, che associa ad ogni stato q e ogni simbolo X un insieme (finito) di triple:

$$\delta(q,X) = \{(q_1,Y_1,D_1), (q_2,Y_2,D_2), \ldots, (q_k,Y_k,D_k)\}$$

- Il linguaggio accettato da una NTM *M* é così definito:
  - *M* accetta un input *w* se c'é una sequenza di scelte che porta dalla ID iniziale con input *w*, a una ID con stato accettante
- Le NTM non accettano linguaggi che non siano accettati anche da TM deterministiche