
# *Progetto e analisi di algoritmi*

Roberto Cordone

DTI - Università degli Studi di Milano

Polo Didattico e di Ricerca di Crema

Tel. 0373 / 898**089**

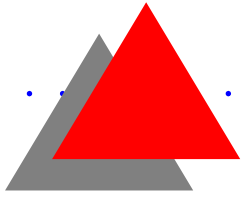
E-mail: **cordone@dti.unimi.it**

Ricevimento: **su appuntamento**

Web page: **http://www.dti.unimi.it/~cordone**

Lezioni: **Martedì dalle 11.00 alle 13.00**

**Giovedì dalle 11.00 alle 13.00**



# *Algoritmi e macchine*


Algoritmo è una sequenza di operazioni

1. elementari (di durata limitata)
2. determinate meccanicamente dall'istanza
3. in numero limitato

Per definire un algoritmo bisogna indicare

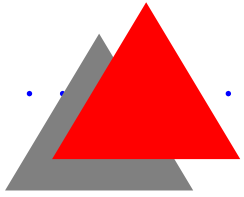
- quali operazioni sono lecite e
- come si determina l'operazione da eseguire a ogni passo

Queste definizioni specificano una macchina, cioè un meccanismo formale capace di manipolare stringhe di simboli (riconoscere parole, computare funzioni, risolvere problemi. . . )


# *Automa a stati finiti (1)*

Finite State Automaton: $FSA = (Q, q_s, A, \Sigma, \delta)$

- un insieme finito $Q$ di stati, fra cui
  - uno stato iniziale $q_s \in Q$
  - un sottoinsieme di stati accettanti $A \subseteq Q$
- un alfabeto di ingresso $\Sigma$ per codificare l'istanza
- un nastro di ingresso finito diviso in celle che contiene l'istanza (un simbolo per ogni cella)
- una posizione corrente di lettura sul nastro (testina)



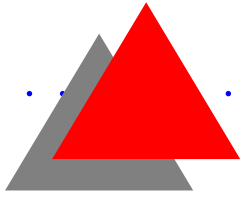
# *Automa a stati finiti (2)*


L'automa è guidato da una funzione di transizione totale

$$\delta : Q \times \Sigma \to Q$$

- l'automa parte dallo stato iniziale $q_s$
- ad ogni passo, in base allo stato e all'ingresso correnti
  - si sposta nello stato indicato da $\delta$
  - avanza la testina di una cella
- si arresta al termine del nastro
  - restituisce *Sì* se lo stato corrente è accettante
  - restituisce *No* se non lo è

Allora, forse possiamo risolvere problemi di decisione!

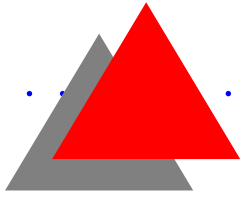


# *Un esempio: parità (1)*


Problema $P$: "Il numero descritto dalla stringa $I$ è pari?"

Linguaggio $\mathcal{L}(P) = ?$

Macchina: ?

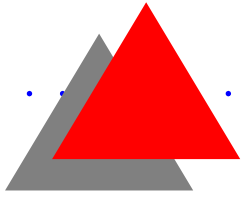


# *Un esempio: parità (1)*


Problema $P$: "Il numero descritto dalla stringa $I$ è pari?"

Linguaggio $\mathcal{L}(P) = \{I = (I', 0) : I' \in \{0,1\}^*\}$

Macchina: ?



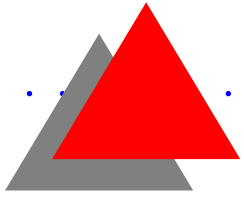
# *Un esempio: parità (1)*


Problema $P$: "Il numero descritto dalla stringa $I$ è pari?"

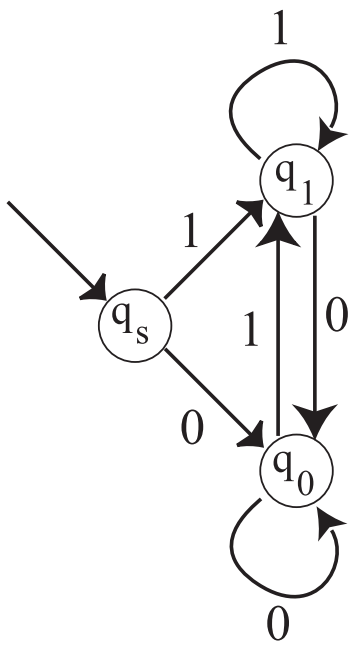
Linguaggio $\mathcal{L}(P) = \{I = (I', 0) : I' \in \{0,1\}^*\}$

Macchina:

- leggere il simbolo corrente
  - conservarlo nello stato
  - passare alla posizione successiva
- al termine
  - se il simbolo conservato è $0$, restituisce *Sì*
  - altrimenti, restituisce *No*



# *Un esempio: parità (2)*




- Insieme degli stati
  $Q = \{q_s, q_0, q_1\}$
- Stato iniziale $q_s$
- Stati accettanti
  $A = \{q_0\}$
- Alfabeto d'ingresso
  $\Sigma = \{0, 1\}$

Funzione di transizione $\delta$

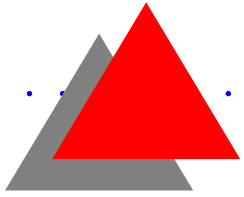
| $Q \times \Sigma$ | $Q$ |
|---|---|
| $(q_s, 1)$ | $q_1$ |
| $(q_s, 0)$ | $q_0$ |
| $(q_0, 1)$ | $q_1$ |
| $(q_0, 0)$ | $q_0$ |
| $(q_1, 1)$ | $q_1$ |
| $(q_1, 0)$ | $q_0$ |

Lo stato conserva le informazioni necessarie a completare il calcolo

# *Esercizi*

Progettare (se possibile) FSA per i seguenti problemi:

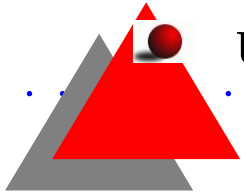
1. La stringa $I$ rappresenta un numero divisibile per $3$? ($A = \{0, \ldots, 9\}$)
2. La stringa $I$ rappresenta un numero divisibile per $6$? ($A = \{0, \ldots, 9\}$)
3. La stringa $I$ contiene consonanti doppie? ($A = \{a, b, c, d, e\}$)
4. La stringa $I$ contiene più $1$ che $0$? ($A = \{0, 1\}$)



# *Automa con pila (1)*


Push-Down Automaton: $PDA = (Q, q_s, A, \Gamma, \gamma_0, \Sigma, \delta)$

- un insieme finito $Q$ di stati, fra cui
  - uno stato iniziale $q_s \in Q$
  - un sottoinsieme di stati accettanti $A \subseteq Q$
- un alfabeto di lavoro $\Gamma$, fra cui un simbolo iniziale $\gamma_0$
- una pila in cima a cui si può leggere/scrivere simboli di $\Gamma$
- un alfabeto di ingresso $\Sigma$ per codificare le istanze
- un nastro di ingresso finito diviso in celle che contiene l'istanza (un simbolo per ogni cella)
- una posizione corrente di lettura sul nastro (testina)

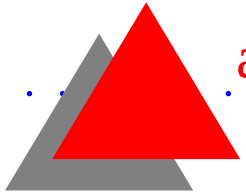


# *Automa con pila (2)*


L'automa è guidato da una funzione di transizione parziale

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$$

- l'automa parte dallo stato iniziale $q_s$, con $\gamma_0$ sulla pila
- ad ogni passo, se $\delta$ è definita per lo stato, l'ingresso e il simbolo in cima alla pila correnti
  - si sposta nello stato indicato da $\delta$
  - sostituisce al simbolo in cima alla pila la stringa indicata da $\delta$ (eventualmente vuota)
  - avanza la testina di una cella
- se $\delta$ non è definita per l'ingresso corrente, ma $\exists \delta\,(q, \epsilon, \gamma)$, si applica quest'ultima e la testina non avanza



# *Automa con pila (3)*


- L'automa si arresta al termine del nastro

  oppure quando la pila è vuota

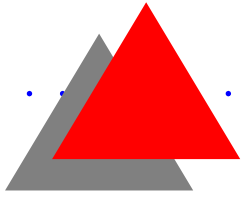
  - restituisce *Sì* se lo stato corrente è accettante
  - restituisce *No* se non lo è

La memoria è nello stato e nella pila

Un *PDA* risolve tutti i problemi risolubili da un *FSA*:

- si definisce $\Gamma = \{\gamma_0\}$
- ad ogni passo $\delta$ sostituisce $\gamma_0$ con $\gamma_0$ in cima alla pila

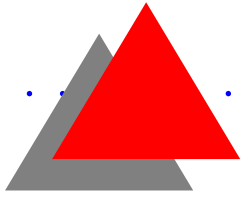
Non vale l'inverso! (ad es., correttezza di espressioni annidate)



# *Esercizi*

Progettare (se possibile) PDA per i seguenti problemi:

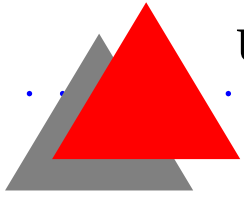
1. La stringa $I$ rappresenta un'espressione algebrica corretta (parentesi aperte e chiuse)?
2. La stringa $I$ contiene almeno il doppio di $1$ che $0$? ($A = \{0, 1\}$)
3. La stringa $I$ contiene più $2$ che $1$ e più $1$ che $0$? ($A = \{0, 1, 2\}$)



# *Rete di Petri (1)*

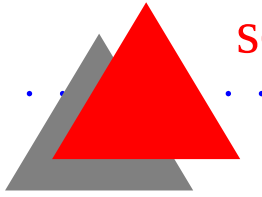

Petri Net: $PN = (P, T, F, M_0, M_f, \Sigma, l)$

- un insieme di posti $P$ e un insieme di transizioni $T$
- un insieme di archi $F \subseteq (P \times T) \cup (T \times P)$ posto-transizione o transizione-posto
- una marcatura iniziale $M_0 : P \to \mathbb{N}$ che assegna ad ogni posto un certo numero di gettoni (*token*)
- un insieme di marcature finali $\mathcal{M}_f$
- un alfabeto d'ingresso $\Sigma$ per codificare le istanze
- una mappatura $l : T \to \Sigma$ che assegna ad ogni transizione un simbolo



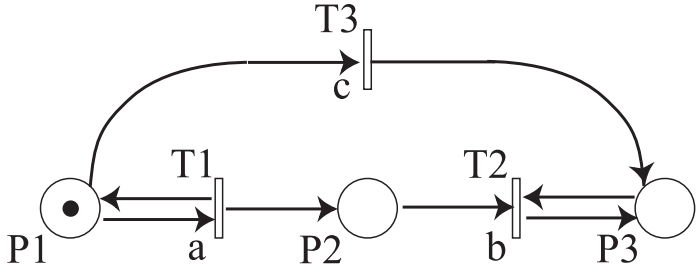
# *Rete di Petri (2)*


- transizione abilitata è una transizione i cui posti in ingresso (pre-set) sono tutti dotati di gettoni
- quando una transizione scatta
  - ogni posto in ingresso perde un gettone
  - ogni posto in uscita (post-set) acquista un gettone
- si parte dalla marcatura $M_0$
- ad ogni passo, si legge un simbolo dell'istanza e si fa scattare una transizione $t$ tale che $l(t)$ sia il simbolo letto
- se alla fine dell'istanza la marcatura è in $\mathcal{M}_f$, restituisce *Sì*
- se nessuna transizione può scattare oppure
  se la marcatura finale non è in $\mathcal{M}_f$, restituisce *No*



# *Reti di Petri: esempio 1*

$$\mathcal{L} = \{a^n c\, b^n : n \geq 0\}$$



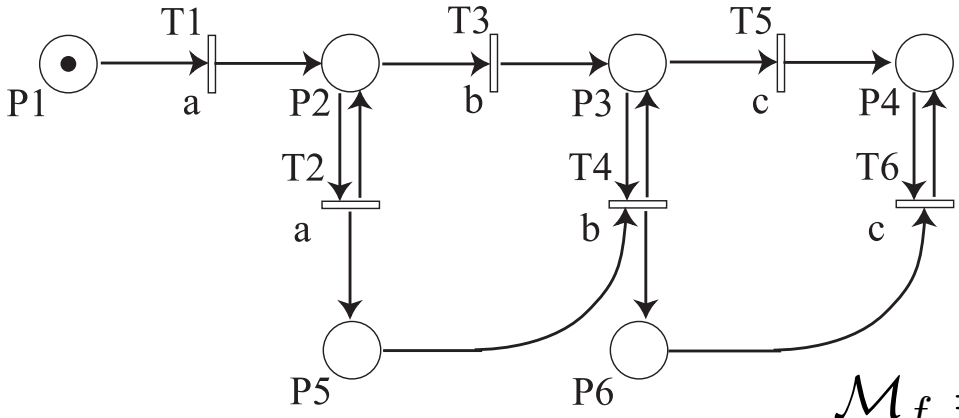

$$\mathcal{M}_f = \{(0,0,1)\}$$

- Ogni occorrenza di $a$ sposta un gettone da P1 a P2 e lo reintegra in P1
- Un'occorrenza di $c$ (anche subito) sposta il gettone da P1 a P3 e vieta ogni scatto futuro di T1 e T3
- T2 può scattare solo dopo che T3 ha caricato P3 di un gettone
- Ogni occorrenza di $b$ toglie un gettone da P2 e reintegra quello in P3

Questa PN accetta tutte e sole le stringhe di $\mathcal{L}$

# *Reti di Petri: esempio 2*

$$\mathcal{L} = \{a^n b^p c^q : n \geq p \geq q \geq 1\}$$

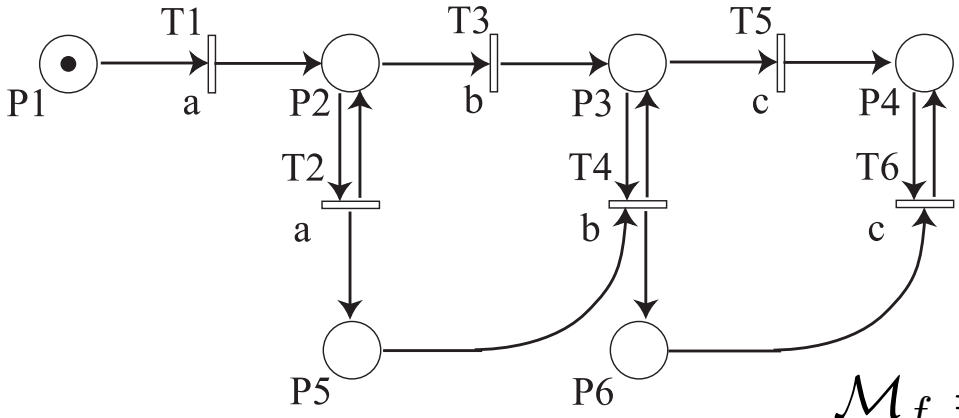



$$\mathcal{M}_f = \{(0, 0, 0, 1, h, k) : h, k \geq 0\}$$

- Può scattare solo T1: la prima occorrenza di $a$ svuota P1 e carica P2
- Se $n \geq 1$, legge $a$ e scatta T2: carica P5 con $n - 1$ gettoni e reintegra P2
- La prima occorrenza di $b$ fa scattare T3, svuotando P2 e caricando P3
- Ora può scattare T4 (al più $n - 1$ volte) o T5

# *Reti di Petri: esempio 2*

$$\mathcal{L} = \{a^n b^p c^q : n \geq p \geq q \geq 1\}$$




$$\mathcal{M}_f = \{(0, 0, 0, 1, h, k) : h, k \geq 0\}$$

- Se $p \geq 1$, legge $b$ e scatta T4: carica P6 con $p - 1$ gettoni e reintegra P3
- La prima occorrenza di $c$ fa scattare T5, svuotando P3 e caricando P4
- Ora può scattare T6 (al più $p - 1$ volte)

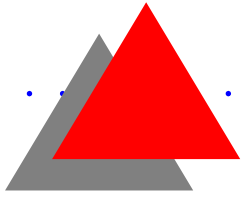
Questa PN accetta tutte e sole le stringhe di $\mathcal{L}$

# *Reti di Petri: esempio 3*

Ci sono però anche linguaggi che un opportuno automa con pila riconosce e nessuna rete di Petri riconosce

In altre parole, PDA e PN dominano i FSA, ma non si dominano a vicenda

Altre macchine li dominano entrambi



# *Macchina di Turing (1)*

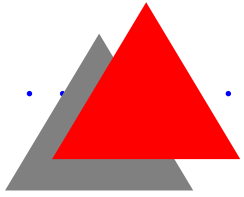

Turing Machine: $TM = (Q, q_s, A, \Gamma, \Sigma, -, \delta)$

- un insieme finito $Q$ di stati, fra cui
  - uno stato iniziale $q_s \in Q$
  - un sottoinsieme di stati accettanti $A \subseteq Q$
- un alfabeto di lavoro $\Gamma$, che include l'alfabeto di ingresso $\Sigma$ e un simbolo *blank* $-$ ($\Gamma \supseteq \Sigma \cup \{-\}$)
- un nastro infinito in entrambi i versi e diviso in celle che possono contenere simboli di $\Gamma$
- una posizione corrente di lettura/scrittura sul nastro (testina)

# *Macchina di Turing (2)*


All'inizio

- la macchina parte dallo stato iniziale $q_s$
- l'istanza $I$ occupa un insieme contiguo di celle sul nastro
- le altre celle del nastro contengono il simbolo *blank* –
- la testina è posta sulla prima cella dell'istanza

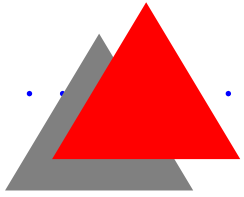


# *Macchina di Turing (3)*


La *TM* evolve guidata da una funzione di transizione parziale

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{S, D\}$$

- ad ogni passo, se $\delta$ è definita per lo stato e l'ingresso correnti
  - si sposta nello stato indicato da $\delta$
  - scrive un simbolo nella cella su cui è posta la testina
  - sposta la testina di una cella a sinistra ($S$) o a destra ($D$)
- se $\delta$ non è definita, la *TM* si arresta
  - risponde Sì se lo stato corrente è in $A$
  - risponde No altrimenti

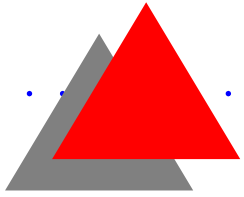


# *Un esempio: stringhe palindrome*


Problema $P$: "La stringa $I$ è palindroma?"

Linguaggio $L(P) = \{I \in (0,1)^* : I = \text{Reverse}(I)\}$

Macchina: ?



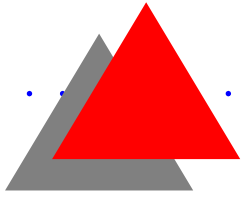
# *Un esempio: stringhe palindrome*


Problema $P$: "La stringa $I$ è palindroma?"

Linguaggio $L(P) = \{I \in (0,1)^* : I = \text{Reverse}(I)\}$

Macchina:

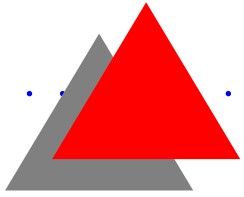
1. leggere il primo simbolo; se è $-$, *Sì*
2. conservarlo nello stato, scrivere $-$ sul nastro e avanzare
3. avanzare sino a leggere $-$, conservando l'ultimo simbolo letto
4. quando si legge $-$, confrontare primo e ultimo simbolo
5. se son diversi, *No*; altrimenti, retrocedere un passo e scrivere $-$
6. rieseguire i punti 1-5 retrocedendo; poi riavanzare




# *Esercizi*

Progettare (se possibile) TM per i seguenti problemi:

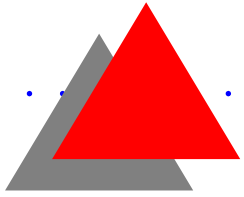
1. La parte della stringa $I$ che precede il separatore $\#$ e quella che lo segue sono anagrammi?
   ($A = \{a, \ldots, z\}$)




# *La Macchina RAM (1)*

È un altro modello computazionale, più sofisticato

- infiniti registri $R_i$: ciascuno può contenere un numero intero ($\leq M$)
- sul registro $R_0$ si possono eseguire operazioni aritmetiche
- un nastro di ingresso e un nastro di uscita, suddivisi in celle: ogni cella può contenere un solo numero intero ($\leq M$)
- una testina di lettura sul nastro di ingresso e una di scrittura sul nastro di uscita, capaci solo di avanzare

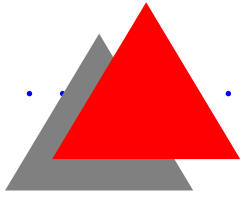


# *La Macchina RAM (2)*


- un programma o sequenza finita di istruzioni, non modificabili, composte da
  - operatore (lettura e scrittura su nastro, caricamento dati nei registri, salto condizionato, operazione aritmetica)
  - operando (intero, indice diretto o indiretto di registro, indirizzo dell'istruzione successiva al salto)
- un contatore $lc$ che indica l'istruzione corrente

N.B.: Celle e registri contengono interi limitati!

Per conservare un intero $> M$, occorrono più celle o registri



# *La Macchina RAM (3)*


Inizialmente

- le due testine sono poste all'inizio di ciascun nastro
- le prime celle del nastro di ingresso contengono l'istanza $I$
- le seguenti e quelle del nastro di uscita contengono $0$
- il contatore $lc$ punta la prima istruzione del programma

Ad ogni passo, la macchina

- modifica nastri e registri in base all'istruzione corrente
- passa all'istruzione successiva (o indicata dal salto)

L'istruzione *halt* fa arrestare la macchina