# AUTOMI, CALCOLABILITÀ E COMPLESSITÀ

PAGINA DEL CORSO:

dventura83.github.io

ESAME: Scritto. Tre parti, ogni parte 1 esercizio + 1 domanda aperta.

LOGISTICA: Mercoledì, 8-11 (AULA 3L)

Venerdì, 15-17 (AULA 3 - DE LOLLIS)

27/09: AULA A - PIETRO BENEDETTI

SYLLABUS: Tre macro-argomenti:

- AUTOMI. Semplice modello di calcolo. Usato in applicazioni reali, quali il riconoscimento di pattern nei dati. Anche le GRAMMATICHE, hanno applicazione nei linguaggi di programmazione (es. parser dei compilatori).
- COMPUTABILITÀ. La macchina di TURING. Quali problemi sono calcolabili e quali no, a prescindere dalle risorse. Vedremo, infatti, problemi non risolvibili da nessuna macchina di TURING.

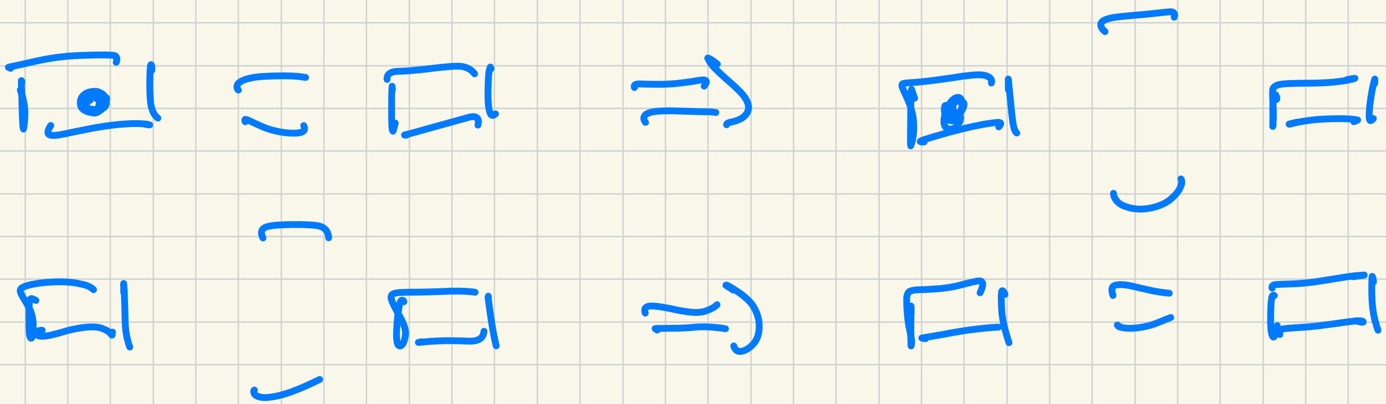
— COMPLESSITÀ. Entrano in gioco le risorse (spazio, tempo, ...). Quali problemi possono essere risolti in modo "efficiente" (con poche risorse).

Esistono problemi che non sappiamo risolvere in modo efficiente.

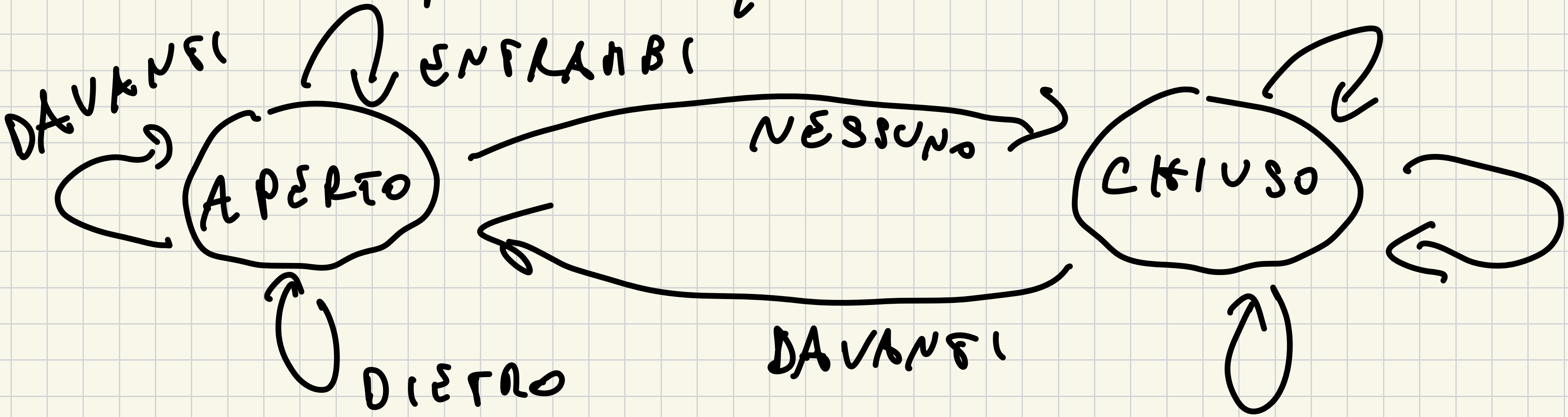
$$P \neq NP \; ??$$

# LINGUAGGI REGOLARI

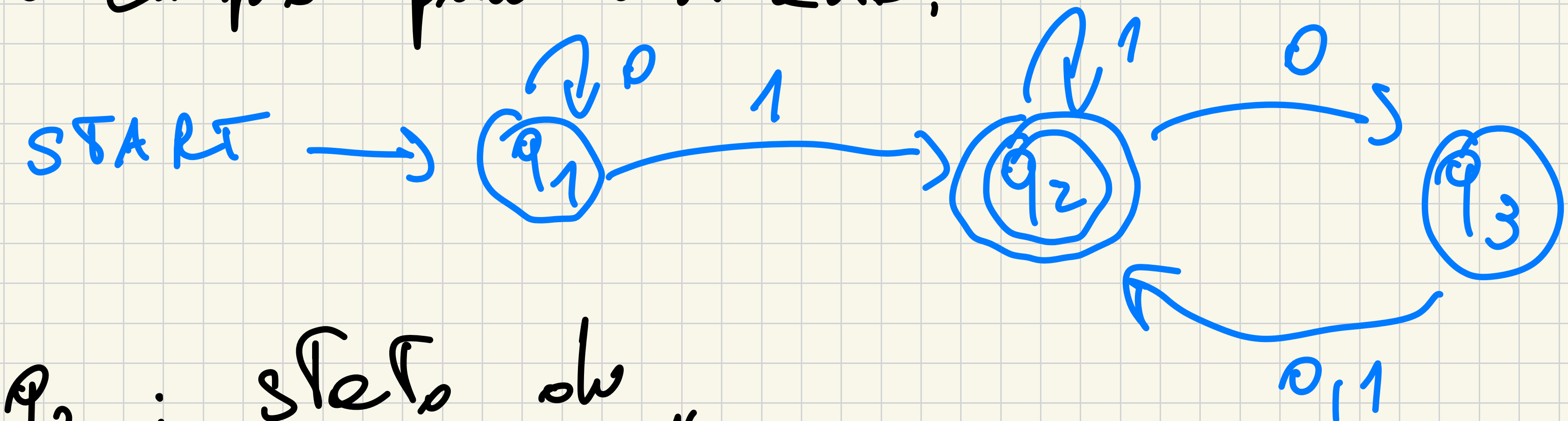
Il modello di computazione di partenza è AUTOMA A STATI FINITI.
Memoria limitata, gestione dell'input limitata. In contro, semplicità estrema.
Esempio: sensore della porta automatica.



| | NESSUNO | DAVANTI | DIETRO | ENTRAMBI |
|---|---|---|---|---|
| CHIUSA | CHIUSA | APERTO | CHIUSO | CHIUSO |
| APERTA | CHIUSA | APERTO | APERTO | APERTO |




Esempio più astratto.




Su input:

$w = 11101$

ACCETTA

$q_2$ : stato di accettazione.

**DEF (DFA)** Un DFA è una tupla $(Q, \Sigma, \delta, q_0, F)$ dove:

- $Q$ è insieme finito stati.
- $\Sigma$ è insieme finito alfabeto input.
- $\delta : Q \times \Sigma \to Q$ è la funzione di transizione
- $q_0$ è lo stato iniziale.
- $F \subseteq Q$ è l'insieme degli stati finali.

Esempio di sopra:

$Q = \{q_1, q_2, q_3\}$; $\Sigma = \{0, 1\}$

$q_0 = q_1$ ; $F = \{q_2\}$

| $\delta$ | 0 | 1 |
|---|---|---|
| $q_1$ | $q_1$ | $q_2$ |
| $q_2$ | $q_3$ | $q_2$ |
| $q_3$ | $q_2$ | $q_2$ |

Che lingua parla l'automa? quale insieme di input l'automa accetta.

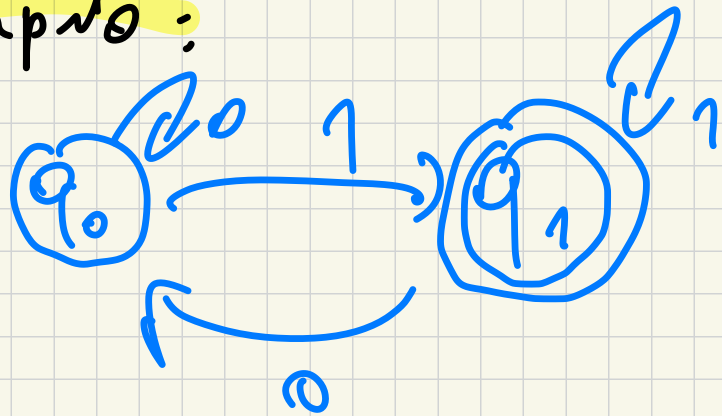
Se l'automa è $D$ allora $L(D) = A$ dove $A$ è l'insieme di stringhe che $D$ accetta.

Nota: $D$ accetta tante stringhe ma riconosce un solo linguaggio.

Esempio: L'esempio precedente ha

$$A = \{ w : w \text{ contiene almeno un '1' ed un numero pari di '0' segue l'ultimo '1'} \}$$

## Esempio:




$L(D) = A = \{ w : w$ termina con '1' $\}$.

Per definire il linguaggio associato ad un automa $D$ si usa la FUNZIONE DI TRANSIZIONE ESTESA $\delta^* : Q \times \Sigma^* \to Q$

STRINGA VUOTA

$$a \in \Sigma,\ x \in \Sigma^* \quad \begin{cases} \delta^*(q, \varepsilon) = \delta(q, \varepsilon) \\ \delta^*(q, ax) = \delta^*(\delta(q, a), x) \end{cases}$$

$y = ax$

Configurazione di un DFA: Coppia in $Q \times \Sigma^*$
che indica: (i) lo stato attuale
(ii) la parte di input ancora
da leggere

Configurazione iniziale: $(q_0, x)$
Un passo di computazione: Relazione binaria
$\vdash_D$ t.c. $(p, ax) \vdash_D (q, x)$ sse

$\delta(p, a) = q$ dove $p, q \in Q$
$a \in \Sigma$
$x \in \Sigma^*$

Posso estendere $\vdash_D$ e $\vdash_D^*$ ottenute aggiungendo tutte le coppie in $Q \times \Sigma^*$ che rendono $\vdash_D$ chiusa per riflessivo e transitività.

In generale, $R$ su dominio $D$ è:

**RIFLESSIVA:** se $\forall x \in D$, $R(x, x)$ è vera

**TRANSITIVA:** Se $\forall x, y, z \in D$, se $R(x, y) \wedge R(y, z)$ allora $R(x, z)$

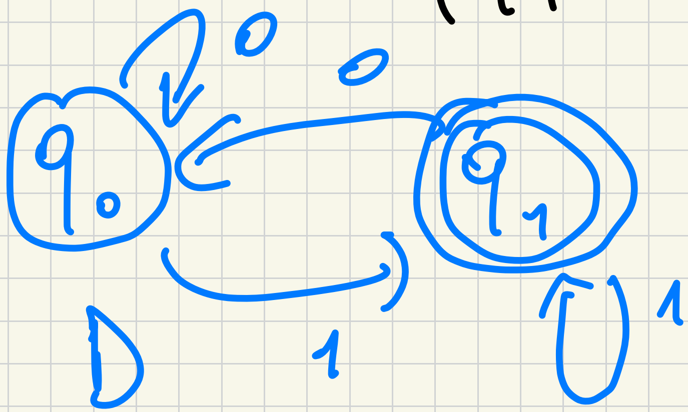
In particolare:

- la riflessività mi dice $(q, x) \vdash_D^* (q, x)$
- la transitività mi dice

se $(q, aby) \vdash_D (p, by)$

$(p, by) \vdash_D (r, y)$

allora $(q, aby) \vdash_D^* (r, y)$




$(q_0, 011) \vdash_D (q_0, 11) \vdash_D$

$\vdash_D (q_1, 1) \vdash_D (q_1, \varepsilon)$

$\Rightarrow (q_0, 011) \vdash_D^* (q_1, \varepsilon)$

Inoltre $\delta^*(q_0, 011) = \delta^*(q_0, 11)$
$= \delta^*(q_1, 1)$
$= \delta^*(q_1, \varepsilon) = q_1$

**DEF.** Una stringa $x \in \Sigma^*$ é ACCETTATA da DFA $D = (Q, \Sigma, \delta, q_0, F)$ se

$$\delta^*(q_0, x) \in F$$

$$\left( (q_0, x) \vdash_D^* (q, \varepsilon), \ q \in F \right)$$

Il linguaggio riconosciuto da $D$ é:

$$L(D) = \{ x \in \Sigma^* : \delta^*(q_0, x) \in F \}.$$

Made with Goodnotes

## DEF (LINGUAGGI REGOLARI).

$$REG = \{ L \subseteq \Sigma^* : \exists \text{ DFA } D \text{ t.c. } L(D) = L \}$$

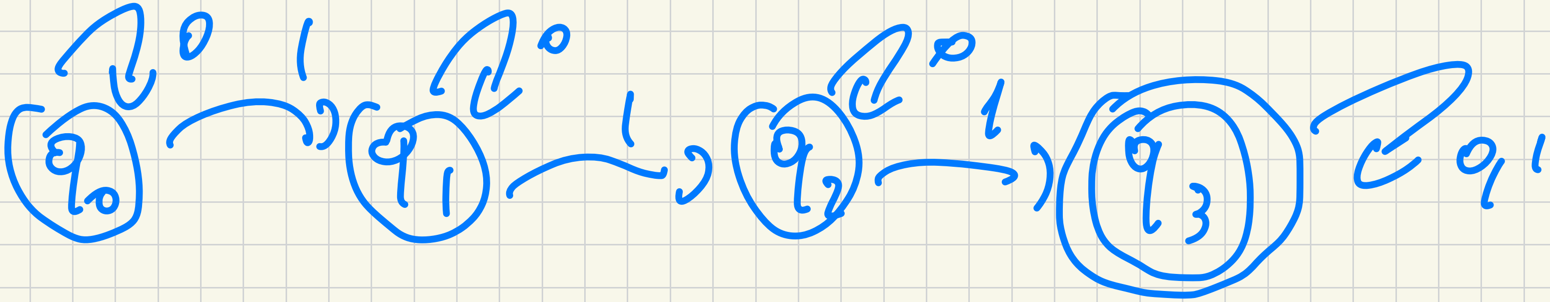
Scopo: Capire come progettare automi.
Tutti i linguaggi sono regolari?

Alcuni esempi semplici.

1) Costruire un DFA per il linguaggio

$$L(D) = \{ x \in \{0,1\}^* : W_H(x) \geq 3 \}$$

$W_H(x) = \#$ '1' in $x$

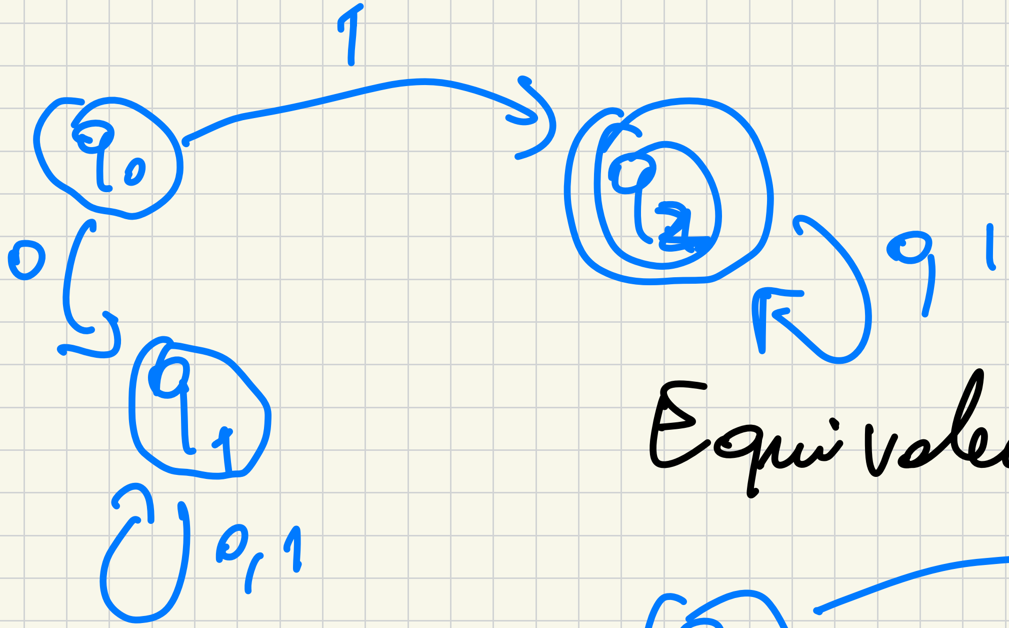



Correttezza: (i) Se $x \in L$ allora $D$ accetta $x$

(ii) Se $D$ accetta $x$ allora $x$ ha $w_H(x) \geq 3$.

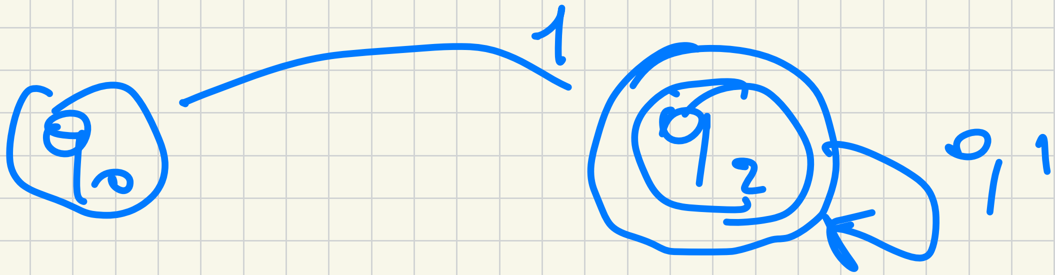
2) DFA $D$ con $L(D) = \{ x \in \{0,1\}^* :$

$x = 1y, \ y \in \{0,1\}^* \}$




Correttezza: Banale.

Equivalente:




3) Per casa: Progettare D t.c.

$$L(D) = \{ x \in \{0,1\}^* : x = 0^n 1, \ n \in \mathbb{N} \}$$

Studiare le proprietà dei linguaggi regolari.
Siccome i linguaggi sono insiemi di stringhe su alfabeto $\Sigma$, dati $L_1, L_2 \subseteq \Sigma^*$ posso considerare:

- UNIONE: $L_1 \cup L_2 = \{ x \in \Sigma^* : x \in L_1 \vee x \in L_2 \}$

- INTERSEZIONE: $L_1 \cap L_2 = \{ x \in \Sigma^* : x \in L_1 \wedge x \in L_2 \}$

- COMPLEMENTO: $\neg L = \{ x \in \Sigma^* : x \notin L \}$.

La concatenazione e la potenza.

Se $x = a_1 \dots a_n$ e $y = b_1 \dots b_m$

$$x, y \in \Sigma^* \quad , \; n, m > 0$$

$$xy = a_1 \dots a_n b_1 \dots b_m$$

$$x\varepsilon = \varepsilon x = x$$

Ricorsivamente:

$$\begin{cases} x\varepsilon = x \\ x(ya) = (xy)a \end{cases}$$

$$x, y \in \Sigma^* \text{ e } a \in \Sigma$$

Per i linguaggi:

$$L_1 \circ L_2 = \{ xy : x \in L_1, y \in L_2 \}$$

Esempio: $\Sigma = \{a, b\}$ ; $L_1 = \{a, ab, ba\}$

$$L_2 = \{ab, b\}$$

$$L_1 \circ L_2 = \{aab, ab, abab, abb, baab, bab\}$$

Neutralità: $\emptyset \circ L = L \circ \emptyset = L$

$$\{\varepsilon\} \circ L = L \circ \{\varepsilon\} = L$$

Made with Goodnotes

Concatenazione con se stesso: potenza.

Se $x \in \Sigma^*$, $n \in \mathbb{N}$ : $x^n = \underbrace{x x \cdots x}_{n \text{ volte}}$

Ricorsivamente:

$$\begin{cases} x^0 = \varepsilon \\ x^{n+1} = x^n x \; ; \; n \geq 0 \end{cases} \qquad \begin{cases} L^0 = \{\varepsilon\} \\ L^{n+1} = L^n \circ L \end{cases}$$

Esempio: Se $L = \{a, ab, ba\}$

$L^2 = \{aa, aab, aba, abab, abba, baa, baab, baba\}$

La star di Kleene:

$$L^* = \{x_1 \dots x_k : k \geq 0, x_i \in L\}$$

$$L^* = \bigcup_{n \geq 0} L^n = \{\varepsilon\} \cup L^1 \cup L^2 \cup \dots$$

Esempio: $L = \{a, b\}$

$$L^* = \{\varepsilon, a, b, aa, ab, ba, bb, \dots\}$$

Cosa dimostreremo: REG è chiuso per tutte queste operazioni.

Esempio: $L_1 \in REG$, $L_2 \in REG$

$$\Longrightarrow L_1 \cup L_2 \in REG$$