

# Automi e Linguaggi Formali

Sara Feltrin

26-02-18

# Capitolo 1

## Introduzione

Per iniziare, ci sono alcuni concetti di base da tenere a mente:

- **Alfabeto**: insieme finito e non vuoto di simboli, per esempio  $\Sigma = \{0,1\}$  oppure  $\Sigma = \{a,b,c,d,e,\dots,z\}$ ;
- **Stringa**: sequenza finita di simboli da un alfabeto  $\Sigma$ , per esempio: 011001 o abc;
- **Stringa vuota**: stringa con zero occorrenze di simboli dell'alfabeto  $\Sigma$ , denotata da  $\varepsilon$ ;
- **Lunghezza di una stringa**: numero di simboli nella stringa, per esempio  $|w|$  denota la lunghezza della stringa  $w$ , quindi  $|01001| = 5$ ;
- **Potenze di un alfabeto**:  $\Sigma^k$  insieme delle stringhe di lunghezza  $k$  con simboli da  $\Sigma$ , per esempio preso l'alfabeto  $\Sigma = \{0,1\}$ :  $\Sigma^0 = \{\varepsilon\}$ ,  $\Sigma^1 = \{0,1\}$ ,  $\Sigma^2 = \{00,01,10,11\}$ . Viene chiamata potenza di un alfabeto poichè può essere vista come una potenza dove la base è il numero di simboli dell'alfabeto e l'esponente il numero della potenza dell'alfabeto (quindi, nell'alfabeto dei numeri binari con  $\Sigma^3$ , avremo  $2^3 = 8$ );
- **Insieme di tutte le stringhe**: per ottenere l'insieme di tutte le stringhe, usiamo il simbolo  $*$  e scriviamo  $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots$ ;
- **Linguaggio**: dato un alfabeto  $\Sigma$ , chiamiamo linguaggio ogni sottoinsieme  $L \subseteq \Sigma^*$  (compreso anche il linguaggio vuoto che non contiene nessuna parola).

## Capitolo 2

# Automi a stati finiti deterministici

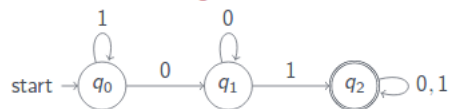
Un automa a stati finiti deterministico, chiamato anche DFA, è una quintupla  $A = (Q, \Sigma, \delta, q_0, F)$  dove

- $Q$  è un insieme finito di stati;
- $\Sigma$  è un alfabeto finito, si intende quindi l'insieme di input che può leggere l'automa;
- $\delta$  è una funzione di transizione  $(q, a) \mapsto q'$ , ovvero dallo stato in cui sono, quando leggo il simbolo  $a$ , passo allo stato  $q'$ ;
- $q_0 \subseteq Q$  è lo stato iniziale dell'automa;
- $F \subseteq Q$  è un insieme di stati finiti;

L'automa può essere rappresentato sia come diagramma di transizioni sia come tabella di transizioni:

**Esempio:** costruiamo un automa  $A$  che accetta il linguaggio delle stringhe con 01 come sottostringa

■ L'automa come **diagramma di transizione**:



■ L'automa come **tabella di transizione**:

	0	1
→ $q_0$	$q_1$	$q_0$
$q_1$	$q_1$	$q_2$
* $q_2$	$q_2$	$q_2$

## 2.1 Linguaggio accettato da un DFA

La funzione di transizione prende in input uno stato e una parola dando in output una nuova parola. Definizione:

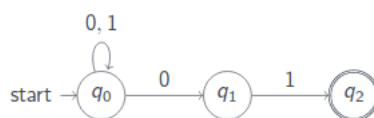
- **base:**  $\delta(q, \varepsilon) = q \rightarrow$  ritorna lo stadio in cui è;
- **induzione:**  $\widehat{\delta}(q, w) = \delta(\widehat{\delta}(q, x), a)$ , dove  $\widehat{\delta}$  rappresenta lo stato attuale e  $\delta$  lo stato in cui mi troverò,  $a$  indica l'ultima lettera della parola che voglio leggere. NB: in  $\widehat{\delta}$  faccio la ricorsione fino ad arrivare al caso base;

Detto ciò, possiamo definire il linguaggio accettato da  $A$  in questo modo:  $L(A) = \{w : \widehat{\delta}(q_0, w) \in F\}$ . Tutti i linguaggi accettati da DFA vengono chiamati **linguaggi regolari**.

## Capitolo 3

# Automi stati finiti non deterministici

È un automa che può trovarsi contemporaneamente in più stati diversi e le transizioni non devono per forza essere complete, per esempio:



Infatti questo non può essere un DFA perchè da  $q_0$  se leggo 0 posso trovarmi contemporaneamente in  $q_0$  e  $q_1$ , in più da  $q_1$  posso muovermi solo in  $q_2$  e da  $q_2$  non posso proprio muovermi. Un automa a stati finiti non deterministici (NFA) è una quintupla  $A = (Q, \Sigma, \delta, q_0, F)$  dove

- $Q$  è un insieme finito di stati;
- $\Sigma$  è un alfabeto finito;
- $\delta$  è una funzione di transizione che prende in input  $(q, a)$  e restituisce un sottoinsieme di  $Q$ ;
- $q_0 \in Q$  è lo stato iniziale;
- $F \subseteq Q$  è un insieme di stati finali;

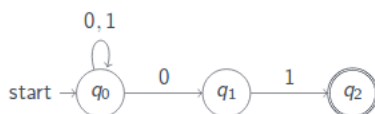
Anche per i NFA abbiamo una definizione rigorosa:

- **base:**  $\widehat{\delta}(q, \varepsilon) = q$ ;
- **induzione:**  $\widehat{\delta}(q, w) = \bigcup_{p \in \widehat{\delta}(q, x)} \delta(p, a)$ ;

Data una parola, il nostro automa potrà trovarsi in uno dei tanti stati che siamo andati a calcolare.

### 3.1 Equivalenza tra DFA e NFA

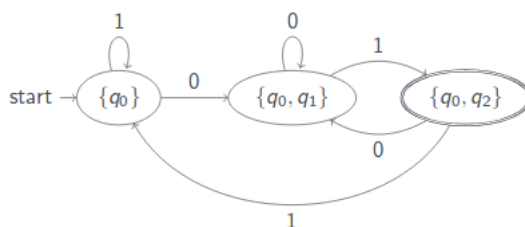
NFA e DFA sono in grado di riconoscere gli stessi linguaggi e l'equivalenza si dimostra mediante una **costruzione a sottoinsiemi**. Infatti, dato un NFA  $N = (Q_N, \Sigma, q_0, \delta_N, F_N)$  costruiremo un DFA  $D = (Q_D, \Sigma, q_0, \delta_D, F_D)$  tale che  $L(D)=L(N)$ . Ogni stato del DFA,  $Q_D$  corrisponde ad un insieme di stati dell'NFA. *Uno stato del DFA  $F_D$  è finale se c'è almeno uno stato finale corrispondente all'NFA*. La funzione di transizione  $\delta_D$  percorre tutte le possibili strade. Ad esempio:



Costruiamo  $\delta_D$  per l'NFA qui sopra:

	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
$*\{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$*\{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
$*\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

La tabella di transizione per D ci permette di ottenere il **diagramma di transizione**



Per semplificare il disegno, ho ommesso gli stati **non raggiungibili**

**Teorema 3.1.1.** *Sia  $D$  il DFA ottenuto da un NFA  $N$  con la costruzione a sottoinsiemi. Allora  $L(D)=L(N)$ .*

**Teorema 3.1.2.** *Un linguaggio  $L$  è accettato da un DFA se e solo se è accettato da un NFA.*

## Capitolo 4

# NFA con epsilon-transizioni

Le epsilon-transizioni vengono usate per muovere l'automa di stato anche se non viene dato nessun simbolo in input. Un automa a stati finiti non deterministico con  $\varepsilon$ -transizioni ( $\varepsilon$ -NFA) è una quintupla  $A = (Q, \Sigma, \delta, q_0, F)$  dove cambia solo

- $\delta$  che è una funzione di transizione che prende in input uno stato in  $Q$  oppure un simbolo nell'alfabeto  $\Sigma \cup \{\varepsilon\}$

L'eliminazione delle  $\varepsilon$ -transizioni procede per  $\varepsilon$ -chiusura degli stati, cioè prendendo tutti gli stati raggiungibili da  $q$  con una sequenza  $\varepsilon\varepsilon...\varepsilon$ . L' $\varepsilon$ -chiusura viene indicata con **ECLOSE**( $q$ ).

### 4.1 Equivalenza tra DFA e $\varepsilon$ -NFA

Per ogni  $\varepsilon$ -NFA  $E$  c'è una DFA  $D$  tale che  $L(E)=L(D)$ , e viceversa. Ogni stato del DFA corrisponde ad un insieme di stati chiuso per  $\varepsilon$ -chiusura. Uno stato del DFA è finale se c'è **almeno** uno stato finale corrispondente nell' $\varepsilon$ -NFA.

**Teorema 4.1.1.** *Sia  $D = (Q_D, \Sigma, S_0, F_D)$  il DFA ottenuto da un  $\varepsilon$ -transizioni  $E$  con la costruzione a sottoinsieme modificata. Allora  $L(D)=L(E)$ .*

In altre parole, se riusciamo a dimostrare che hanno lo stesso risultato, allora i due automi sono equivalenti.

**Teorema 4.1.2.** *Un linguaggio  $L$  è accettato da un DFA se e solo se è accettato da un  $\varepsilon$ -NFA.*

## Capitolo 5

# Espressioni regolari

Diciamo che una parola è accettata oppure no in base allo stato in cui giungiamo alla fine di questa parola (se sono in uno stato finale, allora si tratta di una parola accettata dall'automa altrimenti no). Un'espressione regolare è un modo dichiarativo per descrivere un linguaggio regolare. Abbiamo diversi tipi di operazioni sui linguaggi:

- **Unione:**  $L \cup M = \{w : w \in L \text{ oppure } w \in M\}$
- **Concatenazione:**  $L.M = \{uv : u \in L \text{ e } v \in M\}$
- **Potenze:**
  - $L^0 = \{\varepsilon\}$
  - $L^1 = L$
  - $L^k = L.L...L(k \text{ volte})$
- **Chiusura di Kleene:**  $L^* = \bigcup_{i=0}^{\infty} L^i$

Inoltre le espressioni regolari sono costruite utilizzando

- un insieme di costanti di base:
  - $\varepsilon$  per la stringa vuota
  - $\emptyset$  per il linguaggio vuoto
  - $a, b, ..$  per i simboli  $a, b, .. \in \Sigma$
- collegati da operatori:
  - $+$  per l'unione;
  - $\cdot$  per la concatenazione;
  - $*$  per la chiusura di Kleene;

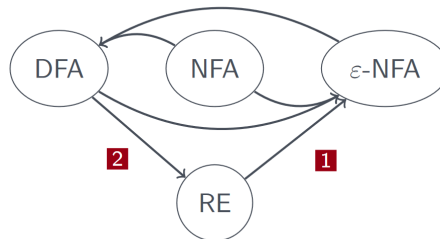


Esistono anche delle **regole di precedenza** degli operatori:

- 1 Chiusura di Kleene;
- 2 Concatenazione;
- 3 Unione.

## 5.1 Equivalenza tra FA e RE

DFA, NFA e  $\varepsilon$ -NFA sono tutti equivalenti.



Infatti

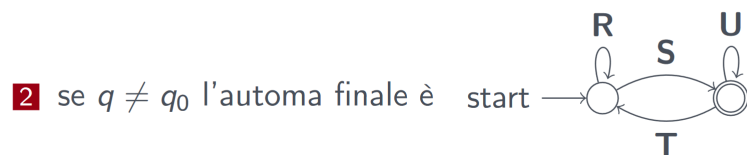
- 1 Per ogni espressione regolare **R** esiste un  $\varepsilon$ -NFA, tale che  $L(A)=L(\mathbf{R})$ ;
- 2 Per ogni FA A possiamo costruire un'espressione regolare **R**, tale che  $L(\mathbf{R})=L(A)$ .

## 5.2 Conversione per eliminazione di stati

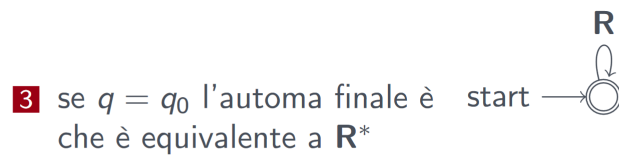
Quando uno stato **q** viene eliminato, i cammini che passano per q scompaiono. Quindi si aggiungono nuove transizioni etichettate con espressioni regolari che rappresentano questi cammini eliminati. Alla fine otteniamo un'espressione regolare che rappresenta tutti i cammini dallo stato iniziale ad uno stato finale, ovvero **il linguaggio riconosciuto dall'automa**. Il procedimento è spiegato nel seguente algoritmo:

- Per ogni stato finale  $q \in F$ :

1 elimina tutti gli stati **tranne lo stato iniziale  $q_0$  e  $q$**



che è equivalente a  $(R + SU^*T)^*SU^*$



- L'espressione regolare desiderata è l'**unione** (+) di tutte le espressioni ottenute dalle regole 2 e 3