Automi e Linguaggi Formali

Parte 3 – Espressioni Regolari

Davide Bresolin
Ultimo aggiornamento: 13 marzo 2024

Università degli Studi di Padova

Sommario

Operazioni sui linguaggi

- **Unione:**

$$L \cup M = \{w : w \in L \text{ oppure } w \in M\}$$

- **Intersezione:**

$$L \cap M = \{w : w \in L \text{ e } w \in M\}$$

- **Complemento:**

$$\overline{L} = \{w : w \notin L\}$$

- **Concatenazione:**

$$L.M = \{uv : u \in L \text{ e } v \in M\}$$

- **Chiusura (o Star) di Kleene:**

$$L^* = \{w_1 w_2 \dots w_k : k \geq 0 \text{ e ogni } w_i \in L\}$$

Proprietà di chiusura

Se L e M sono linguaggi regolari, allora anche i seguenti linguaggi sono regolari:

- Unione: $L \cup M$
- Intersezione: $L \cap M$
- Complemento: $\overline{L}$
- Concatenazione: $L.M$
- Chiusura di Kleene: L^*

Espressioni Regolari

- Un FA (NFA o DFA) è un metodo per costruire una macchina che riconosce linguaggi regolari
- Una espressione regolare è un modo dichiarativo per descrivere un linguaggio regolare.
- Esempio: $01^* + 10^*$
- Le espressioni regolari sono usate, ad esempio, in:
  - comandi UNIX (`grep`)
  - strumenti per l'analisi lessicale di UNIX (`lex` (Lexical analyzer generator) e `flex` (Fast Lex))
  - editor di testo

Espressione regolari: sintassi

Le Espressioni Regolari sono costruite utilizzando

Espressione regolari: sintassi

Le Espressioni Regolari sono costruite utilizzando

- un insieme di costanti di base:
  - ε per la stringa vuota
  - $\emptyset$ per il linguaggio vuoto
  - $\mathrm{a}, \mathrm{b}, \ldots$ per i simboli $a, b, \dots \in \Sigma$

Espressione regolari: sintassi

Le Espressioni Regolari sono costruite utilizzando

- un insieme di costanti di base:
  - ε per la stringa vuota
  - $\emptyset$ per il linguaggio vuoto
  - $\mathrm{a}, \mathrm{b}, \ldots$ per i simboli $a, b, \dots \in \Sigma$
- collegati da operatori:
  - $+$ per l'unione
  - $\cdot$ per la concatenazione
  - $*$ per la chiusura di Kleene
- raggruppati usando le parentesi:
  - $(\quad)$

Espressioni regolari: semantica

Se E è un espressione regolare, allora $L(\mathrm{E})$ è il linguaggio rappresentato da E. La definizione di $L(\mathrm{E})$ è induttiva:

- **Caso Base:**
  - $L(\boldsymbol{\varepsilon}) = \{\varepsilon\}$
  - $L(\boldsymbol{\emptyset}) = \emptyset$
  - $L(\mathrm{a}) = \{a\}$

Se E è un espressione regolare, allora $L(\mathrm{E})$ è il linguaggio rappresentato da E. La definizione di $L(\mathrm{E})$ è induttiva:

- **Caso Base:**
  - $L(\varepsilon) = \{\varepsilon\}$
  - $L(\emptyset) = \emptyset$
  - $L(\mathrm{a}) = \{a\}$
- **Caso induttivo:**
  - $L(\mathrm{E} + \mathrm{F}) = L(\mathrm{E}) \cup L(\mathrm{F})$
  - $L(\mathrm{EF}) = L(\mathrm{E}).L(\mathrm{F})$
  - $L(\mathrm{E}^*) = L(\mathrm{E})^*$
  - $L((\mathrm{E})) = L(\mathrm{E})$

Espressioni regolari: esempio

- Scriviamo l'espressione regolare per

$$L = \{w \in \{0,1\}^* : 0 \text{ e } 1 \text{ alternati in } w\}$$

Espressioni regolari: esempio

- Scriviamo l'espressione regolare per

$$L = \{w \in \{0,1\}^* : 0 \text{ e } 1 \text{ alternati in } w\}$$

$$(01)^* + (10)^* + 1(01)^* + 0(10)^*$$

- Scriviamo l'espressione regolare per

$$L = \{w \in \{0,1\}^* : 0 \text{ e } 1 \text{ alternati in } w\}$$

$$(01)^* + (10)^* + 1(01)^* + 0(10)^*$$

oppure

$$(\varepsilon + 1)(01)^*(\varepsilon + 0)$$

Espressioni regolari: precedenza

Come per le espressioni aritmetiche, anche per le espressioni regolari ci sono delle regole di precedenza degli operatori:

1. Chiusura di Kleene
2. Concatenazione (punto)
3. Unione ($+$)

Esempio:

$$01^* + 1 \text{ è raggruppato in } (0(1)^*) + 1$$

e denota un linguaggio diverso da

$$(01)^* + 1$$

Esercizi (1)

Per ognuno dei seguenti linguaggi, costruire una ER sull'alfabeto $\{a, b, c\}$ che li rappresenti:

1. Tutte le stringhe w che contengono un numero pari di a;
2. Tutte le stringhe w che contengono $4k + 1$ occorrenze di b, per ogni $k \geq 0$;
3. Tutte le stringhe la cui lunghezza è un multiplo di 3;

Esercizi (2)

Per ognuno dei seguenti linguaggi, costruire una ER sull'alfabeto $\{0, 1\}$ che li rappresenti:

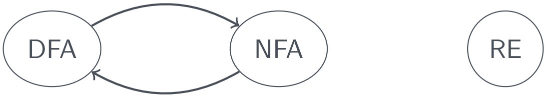
4. Tutte le stringhe w che contengono la sottostringa 101
5. Tutte le stringhe w che **non** contengono la sottostringa 101

Sfida!

Costruire una ER sull'alfabeto $\{0, 1\}$ per il linguaggio di tutti i numeri binari multipli di 3.

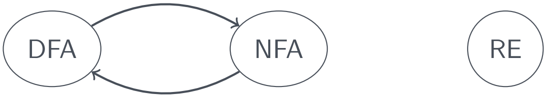
Sommario

Sappiamo già che DFA e NFA sono equivalenti.



Equivalenza tra FA e RE (1)

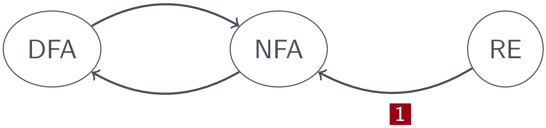
Sappiamo già che DFA e NFA sono equivalenti.



Gli FA sono equivalenti alle espressioni regolari:

Equivalenza tra FA e RE (1)

Sappiamo già che DFA e NFA sono equivalenti.

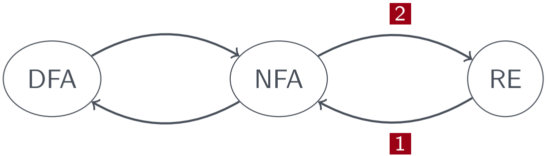


Gli FA sono equivalenti alle espressioni regolari:

1. Per ogni espressione regolare R esiste un *NFA A*, tale che $L(A) = L(\mathrm{R})$

Equivalenza tra FA e RE (1)

Sappiamo già che DFA e NFA sono equivalenti.



Gli FA sono equivalenti alle espressioni regolari:

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

Da RE a ε-NFA

Theorem

Per ogni espressione regolare R *possiamo costruire un NFA A tale che* $L(A) = L(\mathrm{R})$

Da RE a ε-NFA

Theorem

Per ogni espressione regolare R *possiamo costruire un NFA A tale che $L(A) = L(\mathrm{R})$*

Dimostrazione:

La dimostrazione è per induzione strutturale su R. Costruiremo un NFA A per:

- i casi base a, ε, $\emptyset$
- le operazioni regolari $+, \cdot, *$

Caso Base:

Caso Base:

- automa per ε

Da RE a ε-NFA

Caso Base:

- automa per ε

- automa per $\emptyset$

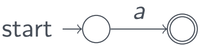
Da RE a ε-NFA

Caso Base:

- automa per ε

- automa per $\emptyset$

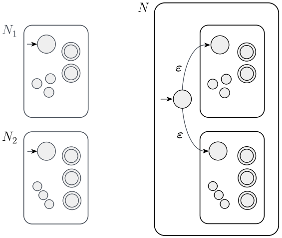
- automa per a



Caso Induttivo:

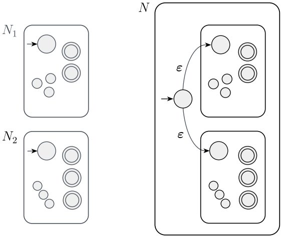
Caso Induttivo:

- automa per $R + S$

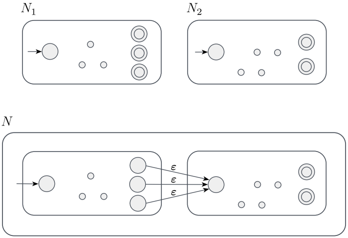


Caso Induttivo:

- automa per $R + S$

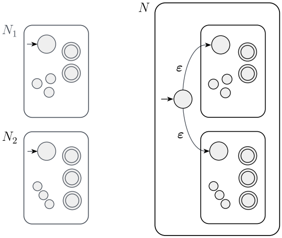


- automa per RS

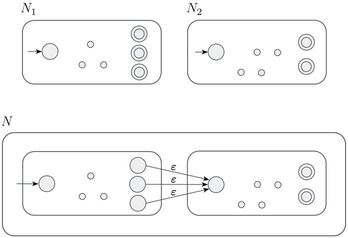


Caso Induttivo:

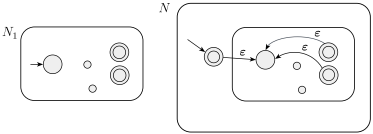
- automa per $R + S$



- automa per RS



- automa per R^*



Espressioni regolari: esercizi (1)

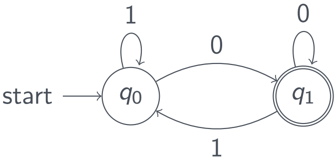
1. Trasformiamo $(0+1)^*1(0+1)$ in NFA
2. Scrivere un'espressione regolare per rappresentare il linguaggio sull'alfabeto $\{a, b, c\}$ che contiene
   - tutte le stringhe che iniziano con a e sono composte solo di a oppure b;
   - la stringa c
3. Trasformare l'espressione regolare dell'esercizio 2 in NFA

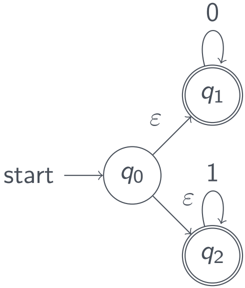
Espressioni regolari: esercizi (2)

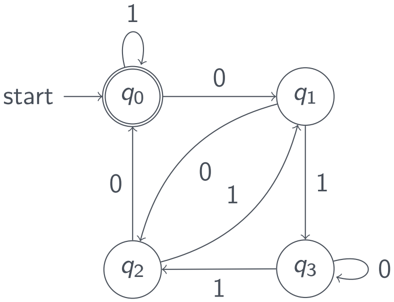
4. Scrivere una espressione regolare per tutte stringhe binarie che cominciano e finiscono per 1
5. Scrivere una espressione regolare per le stringhe binarie che contengono almeno tre 1 consecutivi
6. Scrivere una espressione regolare per le stringhe binarie che contengono almeno tre 1 (anche non consecutivi)
7. Scrivere una espressione regolare per stringhe di testo che descriva le date in formato `GG/MM/AAAA`

Espressioni regolari: esercizi (3)

Costruite una Espressione Regolare equivalente ai seguenti automi:

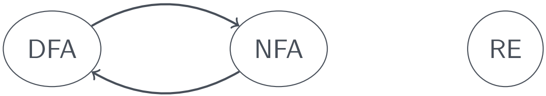






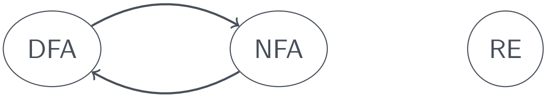
Equivalenza tra FA e RE (2)

Sappiamo già che DFA e NFA sono equivalenti.



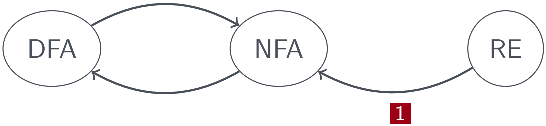
Equivalenza tra FA e RE (2)

Sappiamo già che DFA e NFA sono equivalenti.



Gli FA sono equivalenti alle espressioni regolari:

Sappiamo già che DFA e NFA sono equivalenti.

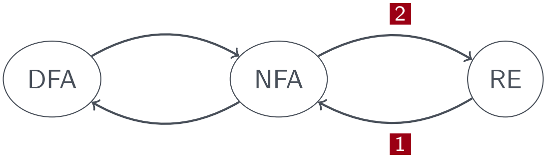


Gli FA sono equivalenti alle espressioni regolari:

1. Per ogni espressione regolare R esiste un *NFA A*, tale che $L(A) = L(\mathrm{R})$

Equivalenza tra FA e RE (2)

Sappiamo già che DFA e NFA sono equivalenti.



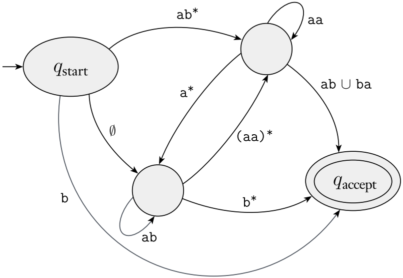
Gli FA sono equivalenti alle espressioni regolari:

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

Conversione per eliminazione di stati

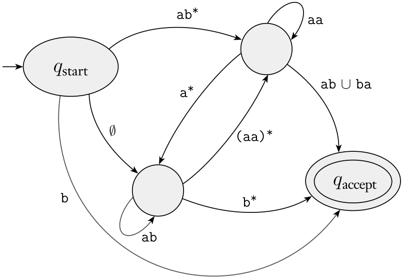
- La procedura che vedremo è in grado di convertire un qualsiasi automa (DFA o NFA) in una espressione regolare equivalente
- Si procede per eliminazione di stati
- Quando uno stato q viene eliminato, i cammini che passano per q scompaiono
- si aggiungono nuove transizioni etichettate con espressioni regolari che rappresentano i cammini eliminati
- alla fine otteniamo un'espressione regolare che rappresenta tutti i cammini dallo stato iniziale ad uno stato finale
  - $\Rightarrow$ cioè il linguaggio riconosciuto dall'automa

- Sono NFA dove le transizioni sono etichettate con espressioni regolari



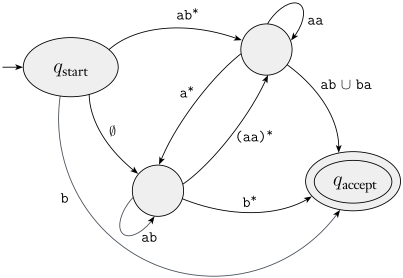
Automi nondeterministici generalizzati (GNFA)

- Sono NFA dove le transizioni sono etichettate con espressioni regolari
- Ogni transizione consuma un blocco di simboli dall'input che appartiene al linguaggio dell'espressione regolare



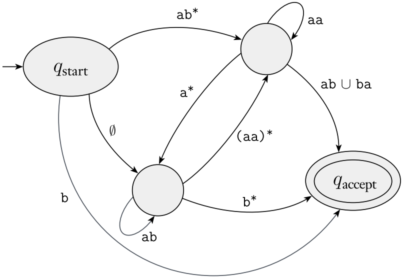
GNFA in forma speciale

1. C'è una transizione dallo stato iniziale verso ogni altro stato, e nessuna transizione entrante
2. Un unico stato finale, senza transizioni uscenti e con una transizione proveniente da ogni altro stato
3. C'è sempre una transizione per ogni coppia di stati, ed un self loop dallo stato verso se stesso (eccetto stati iniziale e finale)



Primo passo: da NFA a GNFA

1. Nuovo stato iniziale q_{start} con transizione ε verso il vecchio q_0
2. Nuovo stato finale q_{accept} con transizione ε da tutti i vecchi stati finali $q \in F$
3. Rimpiazzo transizioni multiple tra due stati con l'unione delle etichette
4. Aggiungo transizioni etichettate con $\emptyset$ tra stati non collegati da transizioni



Definizione formale di GNFA

Un Automa a Stati Finiti Non Deterministico Generalizzato (GNFA) è una quintupla

$$A = (Q, \Sigma, \delta, q_{start}, q_{accept})$$

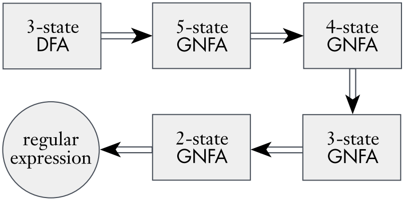
- Q è un insieme finito di stati
- Σ è un alfabeto finito.
- $\delta : Q \setminus \{q_{accept}\} \times Q \setminus \mapsto \mathcal{R}$ è una funzione di transizione che prende in input due stati e restituisce una espressione regolare su Σ
- $q_{start} \in Q$ è lo stato iniziale
- q_{accept} è lo stato finale

Computazione di un GNFA

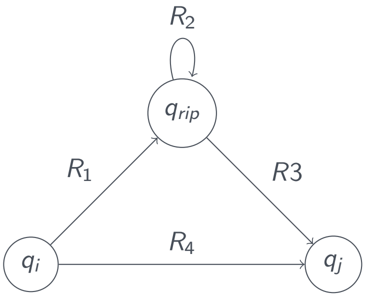
- Data una parola $w = w_1 w_2 \dots w_m$, dove $w_i \in \Sigma^*$
- una computazione di un GNFA A con input w è una sequenza di stati $r_0 r_1 \dots r_m$ che rispetta due condizioni:
  1. $r_0 = q_{start}$ (inizia dallo stato iniziale)
  2. Per ogni i, $w_i \in L(R_i)$, dove $R_i = \delta(r_{i-1}, r_i)$ (rispetta la funzione di transizione)
- Una computazione accetta la parola w se termina nello stato finale ($r_m = q_{accept}$)

Da GNFA a Espressioni Regolari

- Partiamo da un GNFA con k stati, dove $k \geq 2$
- Se $k > 2$, eliminiamo uno stato q_{rip} per ottenere un GNFA con $k - 1$ stati
- Quando $k = 2$, l'etichetta della transizione da q_{start} a q_{accept} è l'espressione regolare equivalente



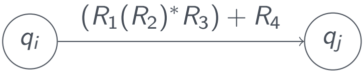
Eliminazione dello stato q_{rip}



Se, nel GNFA:

1. q_i va in q_{rip} con etichetta R_1
2. q_{rip} ha un self loop con etichetta R_2
3. q_{rip} va in q_j con etichetta R_3
4. q_i va in q_j con etichetta R_4

Eliminazione dello stato q_{rip}



Se, nel GNFA:

1. q_i va in q_{rip} con etichetta R_1
2. q_{rip} ha un self loop con etichetta R_2
3. q_{rip} va in q_j con etichetta R_3
4. q_i va in q_j con etichetta R_4

dopo l'eliminazione di q_{rip}, q_i va in q_j con etichetta

$$(R_1(R_2)^*R_3) + R_4$$

Algoritmo di conversione

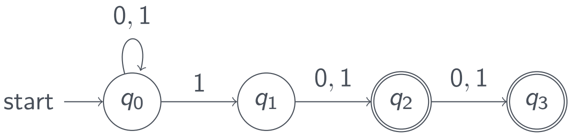
Convert(A):

1. Sia k il numero di stati di A
2. Se $k = 2$, ritorna l'espressione R che collega q_{start} con q_{accept}
3. Se $k > 2$, scegli $q_{rip} \in Q \setminus \{q_{start}, q_{accept}\}$ e costruisci un GNFA $A' = (Q', \Sigma, \delta', q_{start}, q_{accept})$ come segue:
   - $Q' = Q \setminus \{q_{rip}\}$
   - per ogni $q_i \in Q' \setminus \{q_{accept}\}, q_j \in Q' \setminus \{q_{start}\}$, sia

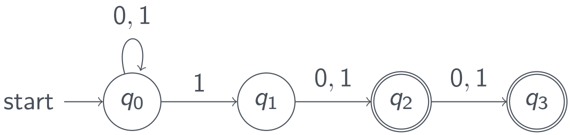
     $$\delta'(q_i, q_j) = (R_1(R_2)^* R_3) + R_4$$

     dove $R_1 = \delta(q_i, q_{rip}), R_2 = \delta(q_{rip}, q_{rip}), R_3 = \delta(q_{rip}, q_j)$ e $R_4 = \delta(q_i, q_j)$.
4. Ritorna il risultato calcolato da Convert(A')

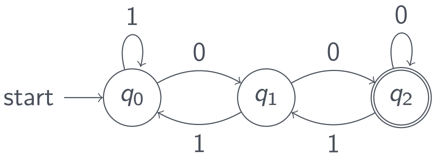
1. Costruiamo l'espressione regolare equivalente al seguente NFA:



1. Costruiamo l'espressione regolare equivalente al seguente NFA:



2. Costruiamo l'espressione regolare equivalente al seguente NFA:



Da FA a RE: esercizi

3. Costruiamo l'espressione regolare equivalente al seguente NFA: