Sapienza
Università di Roma

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Dipartimento di Informatica

Automi Calcolabilità e Complessità

Autore:
Simone Lidonnici

9 ottobre 2024

Indice

1
Linguaggi regolari

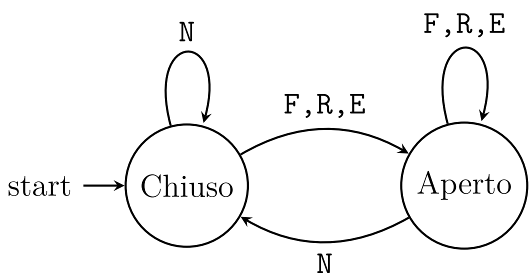
Definizione di linguaggio

Dato un **alfabeto** Σ, cioè un insieme di elementi, un **linguaggio** Σ^* è l'insieme di tutte le strighe ottenibili usando l'alfabeto Σ.

1.1 Automi Deterministici a Stati Finiti (DFA)

Il modello usato per definire i **linguaggi regolari** è l'**automa a stati finiti**, cioè una macchina che permette tramite l'input di passare da uno stato ad un altro, che ha memoria limitata e gestione dell'input limitata, ma è molto semplice.
Esempio:
Una porta che si apre tramite dei sensori può essere descritta tramite un automa con due stati (Aperta e Chiusa) e quattro input dati dai sensori (`N, F, R, E`):

- `N`: se non ci sono persone da nessun lato della porta
- `F`: se c'è una persona davanti alla porta
- `R`: se c'è una persona dietro la porta
- `E`: se ci sono persone da entrambi i lati della porta

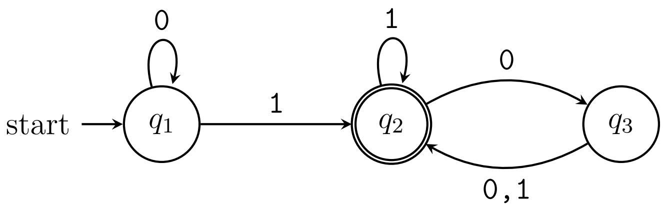


Automa Deterministico a Stati Finiti (DFA)

Un **DFA (Deterministic Finite Automaton)** è una tupla $(Q, \Sigma, \delta, q_o, F)$ in cui:

- Q è l'insieme degli stati dell'automa
- Σ è l'alfabeto dell'automa
- $\delta : Q \times \Sigma \to Q$ è la funzione di transizione degli stati
- $q_0 \in Q$ è lo stato iniziale dell'automa
- $F \subseteq Q$ è l'insieme di **stati accettanti** dell'automa

Esempio:
Preso il seguente DFA:



In questo caso:

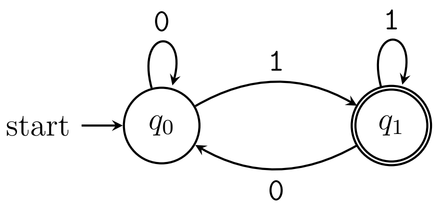
- $Q = \{q_1, q_2, q_3\}$
- $\Sigma = \{0, 1\}$
- $\delta = \begin{array}{c|ccc} \delta & q_1 & q_2 & q_3 \\ \hline 0 & q_1 & q_3 & q_2 \\ 1 & q_2 & q_2 & q_2 \end{array}$
- q_1 è lo stato iniziale dell'automa
- $F = \{q_2\}$ è l'insieme degli stati accettanti

Funzione di transizione estesa

Dato un DFA D, definiamo una **funzione di transizione estesa** $\delta^* : Q \times \Sigma^* \to Q$ in modo ricorsivo:

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

Esempio:
Preso il seguente DFA:



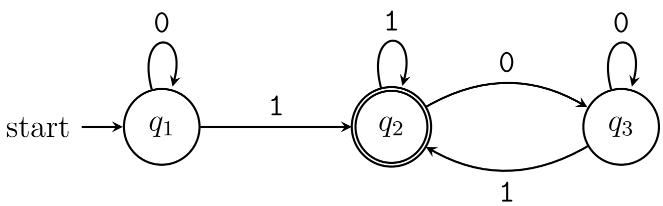
In questo DFA:

- $\delta^*(q_0, 011) = \delta^*(\delta(q_0, 0), 11) = \delta^*(q_0, 11) = \delta^*(\delta(q_0, 1), 1) = \delta^*(q_1, 1) = \delta^*(\delta(q_1, 1), \varepsilon) = \delta^*(q_1, \varepsilon) = q_1$

Linguaggio di un automa

Il **linguaggio di un DFA** D è l'insieme delle stringhe in input che l'automa accetta, cioè quelle per cui l'automa termina in uno stato accettante $q_0 \in F$.
Ogni automa ha un solo linguaggio che si scrive $L(D) = \{x \in \Sigma^* | D \text{ accetta } x\}$. Una stringa $x \in \Sigma^*$ sarà accettata da una DFA se $\delta^*(q_0, x) = q \in F$

Esempio:
Preso il seguente DFA:



Il linguaggio di questa DFA è l'insieme di tutte le stringhe che finiscono con 1:

$$L(D) = \{x \in \{0, 1\}^* | x = y1 \land y \in \{0, 1\}^*\}$$

1.1.1 Configurazione di un DFA

Configurazione di un DFA

Dato un DFA D, una **configurazione** di D è una coppia $Q \times \Sigma^*$ che indica:

- lo stato attuale dell'automa
- l'input ancora da leggere

La configurazione iniziale è sempre (q_0, x).

Passo di computazione di un DFA

Un **passo di computazione** è una relazione binaria con simbolo $\vdash_D$ per cui:

$$(q_1, ax) \vdash_D (q_2, x) \iff \delta(q_1, a) = q_2$$

La **chiusura per riflessione e transitività** di $\vdash_D$, scritta come $\vdash_D^*$, ha delle proprietà:

1. $(q_1, ax) \vdash_D (q_2, x) \implies (q_1, ax) \vdash_D^* (q_2, x)$
2. $\forall q, x \ (q, x) \vdash_D^* (q, x)$
3. $(q_1, abc) \vdash_D (q_2, bc) \vdash_D (q_3, c) \implies (q_1, abc) \vdash_D^* (q_3, c)$

Una stringa $x \in \Sigma^*$ sarà accettata da un DFA se:

$$\exists q \in F | (q_0, x) \vdash_D^* (q, \varepsilon)$$

1.2 Automi Non Deterministici a Stati Finiti (NFA)

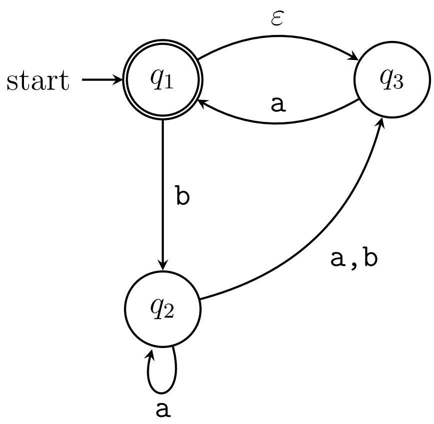
Automa Non Deterministico a Stati Finiti (NFA)

Un **NFA (Non-deterministic Finite Automaton)** è una tupla $(Q, \Sigma, \delta, q_o, F)$ in cui:

- Q è l'insieme degli stati dell'automa
- Σ è l'alfabeto dell'automa
- $\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ è la funzione di transizione degli stati
- $q_0 \in Q$ è lo stato iniziale dell'automa
- $F \subseteq Q$ è l'insieme di **stati accettanti** dell'automa

A differenza del DFA la funzione δ ha come uno dei parametri $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ e come ritorno un insieme di stati, contenuto nell'insieme delle parti di Q, cioè $\mathcal{P}(Q)$. Inoltre per considerare una stringa accettata da un NFA basta che in uno dei rami della computazione la stringa venga accettata.

Esempio:
Preso il seguente NFA:



In questo caso:

- $Q = \{q_1, q_2, q_3\}$
- $\Sigma = \{\texttt{a}, \texttt{b}\}$
- $\delta =$

δ	q_1	q_2	q_3
a	$\emptyset$	$\{q_2, q_3\}$	q_1
b	q_2	q_3	$\emptyset$
ε	q_3	$\emptyset$	$\emptyset$

- q_1 è lo stato iniziale dell'automa
- $F = \{q_1\}$ è l'insieme degli stati accettanti

1.2.1 Configurazione di un NFA

Configurazione di un NFA

Dato un NFA N, una **configurazione** di N è una coppia $Q \times \Sigma_\varepsilon^*$ che indica:

- lo stato attuale dell'automa
- l'input ancora da leggere

La configurazione iniziale è sempre (q_0, x).

Passo di computazione di un NFA

Un **passo di computazione** è una relazione binaria con simbolo $\vdash_N$ per cui:

$$(q_1, ax) \vdash_N (q_2, x) \iff q_2 \in \delta(q_1, a)$$

La **chiusura per simmetria e transitività** di $\vdash_N$, scritta come $\vdash_N^*$, ha delle proprietà:

1. $(q_1, ax) \vdash_N (q_2, x) \implies (q_1, ax) \vdash_N^* (q_2, x)$
2. $(q_1, abc) \vdash_N (q_2, bc) \vdash_N (q_3, c) \implies (q_1, abc) \vdash_N^* (q_3, c)$

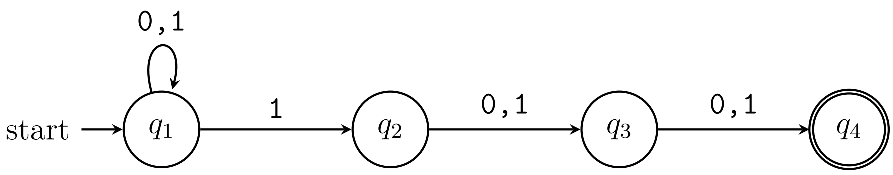
Una stringa $x \in \Sigma^*$ sarà accettata da un NFA se:

$$\exists q \in F | (q_0, x) \vdash_N^* (q, \varepsilon)$$

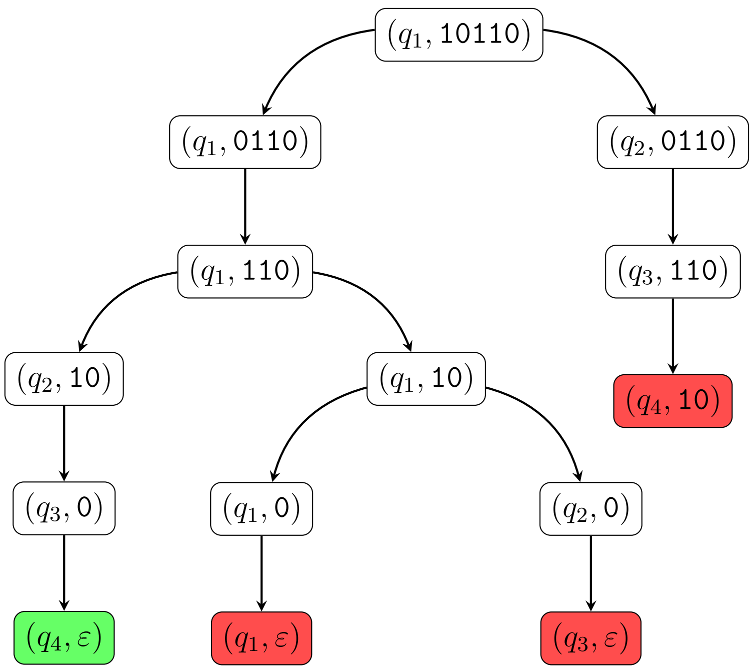
Oppure se:

$$x = y_1 \dots y_n \wedge \exists \underbrace{q_0 \dots q_n}_{\text{sequenza di stati}} | q_{i+1} = \delta(q_i, y_{i+1}) \wedge q_n \in F$$

Esempio:
Preso il seguente NFA:



Data la stringa `10110` la computazione sarà:



La stringa `10110` viene quindi accettata dal NFA.

1.3 Linguaggi regolari

Insieme dei Linguaggi regolari

Dato un alfabeto Σ, l'insieme dei **linguaggi regolari** di Σ, scritto come REG, è l'insieme dei linguaggi per cui esiste una DFA che li accetta:

$$\text{REG} = \{L \subseteq \Sigma^* | \exists D \; L(D) = L\}$$

1.3.1 Proprietà dei linguaggi regolari

I linguaggi sono insiemi di stringhe di un alfabeto Σ, quindi dati due linguaggi $L_1, L_2 \subseteq \Sigma^*$ possiamo definire le operazioni:

- **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:**
$$\neg L = \{x \in \Sigma^* | x \notin L\}$$
- **Concatenazione:**
$$L_1 \circ L_2 = \{xy \in \Sigma^* | x \in L_1 \wedge y \in L_2\}$$
- **Potenza:**
$$L^n = \begin{cases} \{\varepsilon\} & n = 0 \\ L \circ L^{n-1} & n > 0 \end{cases}$$
- **Star di Kleene:**
$$L^* = \{x_1 \dots x_k \in \Sigma^* | x_i \in L\} = \bigcup_{n \geq 0} L^n$$

1.3.2 Chiusura dei linguaggi regolari

Chiusura dell'unione in REG

L'unione è chiusa in REG, cioè:

$$\forall L_1, L_2 \in \text{REG} \implies L_1 \cup L_2 \in \text{REG}$$

Dimostrazioni:

- **Tramite DFA:**
  Dati $L_1, L_2 \in \text{REG}$, allora esistono:
  - $D_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) | L(D_1) = L_1$
  - $D_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) | L(D_2) = L_2$

  Creo il DFA $D_0 = (Q_0, \Sigma, \delta_0, q_0, F_0)$ per cui:
  - $Q_0 = Q_1 \times Q_2 = \{(q_x, q_y) | q_x \in Q_1 \wedge q_y \in Q_2\}$
  - $q_0 = (q_1, q_2)$
  - $F_0 = (F_1 \times Q_2) \cup (Q_1 \times F_2) = \{(q_x, q_y) | q_x \in F_1 \vee q_y \in F_2\}$
  - $\forall (q_x, q_y) \in Q_0, a \in \Sigma$:

  $$\delta((q_x, q_y), a) = (\delta_1(q_x, a), \delta_2(q_y, a))$$

  Quindi $x \in L(D_0) \iff x \in L_1 \vee x \in L_2$, quindi $L_1 \cup L_2 \in \text{REG}$.

- **Tramite NFA:**
  Dati $L_1, L_2 \in \text{REG}$, allora esistono:
  - $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) | L(N_1) = L_1$
  - $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) | L(N_2) = L_2$

  Creo il NFA $N_0 = (Q_0, \Sigma, \delta_0, q_0, F_0)$per cui:
  - $Q_0 = Q_1 \cup Q_2 \cup \{q_0\}$
  - q_0 è un nuovo stato iniziale
  - $F_0 = F_1 \cup F_2$
  - $\forall q \in Q_0, a \in \Sigma$:

  $$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \\ \delta_2(q, a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \wedge a = \varepsilon \\ \emptyset & q = q_0 \wedge a \neq \varepsilon \end{cases}$$

  Quindi $x \in L(N_0) \iff x \in L_1 \vee x \in L_2$, quindi $L_1 \cup L_2 \in \text{REG}$.

Chiusura dell'intersezione in REG

L'intersezione è chiusa in REG, cioè:

$$\forall L_1, L_2 \in \text{REG} \implies L_1 \cap L_2 \in \text{REG}$$

Dimostrazione:
Dati $L_1, L_2 \in \text{REG}$, allora esistono:

- $D_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) | L(D_1) = L_1$
- $D_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) | L(D_2) = L_2$

Creo il DFA $D_0 = (Q_0, \Sigma, \delta_0, q_0, F_0)$ per cui:

- $Q_0 = Q_1 \times Q_2 = \{(q_x, q_y) | q_x \in Q_1 \land q_y \in Q_2\}$
- $q_0 = (q_1, q_2)$
- $F_0 = F_1 \times F_2$
- $\forall (q_x, q_y) \in Q_0, a \in \Sigma$:
$$\delta((q_x, q_y), a) = (\delta_1(q_x, a), \delta_2(q_y, a))$$

Quindi $x \in L(D_0) \iff x \in L_1 \land x \in L_2$, quindi $L_1 \cap L_2 \in \text{REG}$.

Chiusura del complemento in REG

Il complemento è chiuso in REG, cioè:

$$\forall L \in \text{REG} \implies \neg L \in \text{REG}$$

Dimostrazione:
Dato $L \in \text{REG}$, esiste $D = (Q, \Sigma, \delta, q_0, F) | L(D) = L$.
Creo il DFA $D^* = (Q, \Sigma, \delta, q_0, Q - F)$, uguale a D ma con gli stati accettanti invertiti, quindi $x \in L(D^*) \iff x \notin L(D)$, quindi $\neg L \in \text{REG}$.

Chiusura della concatenazione in REG

La concatenazione è chiusa in REG, cioè:

$$\forall L_1, L_2 \in \text{REG} \implies L_1 \circ L_2 \in \text{REG}$$

Dimostrazione:
Dati $L_1, L_2 \in \text{REG}$, allora esistono:

- $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) | L(N_1) = L_1$
- $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) | L(N_2) = L_2$

Creo il NFA $N_0 = (Q_0, \Sigma, \delta_0, q_0, F_0)$ per cui:

- $Q_0 = Q_1 \cup Q_2$
- $q_0 = q_1$

- $F_0 = F_2$
- $\forall q \in Q_0, a \in \Sigma$:

$$\delta_0(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 - F_1 \\ \delta_1(q,a) & q \in F_1 \wedge a \neq \varepsilon \\ \delta_1(q,a) \cup q_2 & q \in F_1 \wedge a = \varepsilon \\ \delta_2(q,a) & q \in Q_2 \end{cases}$$

Quindi $x \in L(D_0) \iff x \in L_1 \circ L_2$, quindi $L_1 \cap L_2 \in \text{REG}$.

Chiusura di star in REG

Star è chiusa in REG, cioè:

$$\forall L \in \text{REG} \implies L^* \in \text{REG}$$

Dato $L \in \text{REG}$, esiste $N = (Q, \Sigma, \delta, q_0, F) | L(D) = L$.
Creo il NFA $N^* = (Q^*, \Sigma, \delta^*, q_0^*, F^*)$ in cui:

- q_0^* è un nuovo stato iniziale
- $Q^* = Q \cup q_0^*$
- $F^* = F \cup q_0^*$
- $\forall q \in Q^*, a \in \Sigma$:

$$\delta^*(q,a) = \begin{cases} \delta(q,a) & q \in Q - F \\ \delta(q,a) & q \in F \wedge a \neq \varepsilon \\ \delta(q,a) \cup q_0 & q \in F \wedge a = \varepsilon \\ q_0 & q = q_0^* \wedge a = \varepsilon \\ \emptyset & q = q_0^* \wedge a \neq \varepsilon \end{cases}$$

Quindi $x \in L(N^*) \iff x \in L^*$, quindi $L^* \in \text{REG}$.

1.4 Equivalenza tra DFA e NFA

Linguaggi accettati da DFA e NFA

Sia $\mathcal{L}(\text{NFA})$ l'insieme dei linguaggi per cui esiste un NFA che li accetta e $\mathcal{L}(\text{DFA})$ l'insieme dei linguaggi per cui esiste un DFA che li accetta, allora:

$$\mathcal{L}(\text{NFA}) = \mathcal{L}(\text{DFA}) = \text{REG}$$

Dimostrazione per doppia inclusione:

1. $\mathcal{L}(\text{DFA}) \subseteq \mathcal{L}(\text{NFA})$:
   Dato un $L \in \mathcal{L}(DFA)$ allora esiste $D = (Q, \Sigma, \delta, q_0, F)$ tale che $L(D) = L$.
   Visto che NFA è una generalizzazione di DFA, allora è ovvio che:

$$\mathcal{L}(\text{DFA}) \subseteq \mathcal{L}(\text{NFA})$$

2. $\mathcal{L}(\text{NFA}) \subseteq \mathcal{L}(\text{DFA})$:
   Dato un $L \in \mathcal{L}(\text{NFA})$ allora esiste $N = (Q_N, \Sigma, \delta_N, q_{0_N}, F_N)$ tale che $L(N) = L$.
   Considero un DFA $D = (Q_D, \Sigma, \delta_D, q_{0_D}, F_D)$, costrito partendo da N in cui:

   - $Q_D = \mathcal{P}(Q_N)$
   - Dato $R \in Q_D$, definiamo l'**estensione** di R:

   $$E(R) = \left\{ q \in Q_N \middle| \begin{array}{c} q \text{ può essere raggiunto da uno stato } r \in R \\ \text{tramite solamente } \varepsilon\text{-archi} \end{array} \right\}$$

   - $q_{0_D} = E(\{q_{0_N}\})$
   - $F_D = \{R \in Q_D | R \cap F_N \neq \emptyset\}$
   - Dati $R \in Q_D$ e $a \in \Sigma$, δ_D è definita:

   $$\delta_D(R, a) = \bigcup_{r \in R} E(\delta_N(r, a))$$

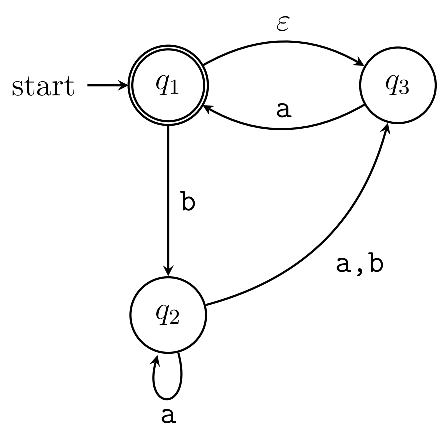
   Detto questo allora abbiamo che:

   $$w \in L(N) \iff w \in L(D)$$

   quindi:

   $$\mathcal{L}(\text{NFA}) \subseteq \mathcal{L}(\text{DFA})$$

Esempio:
Preso il seguente NFA:



Per costruire il DFA equivalente, che sarà definito:

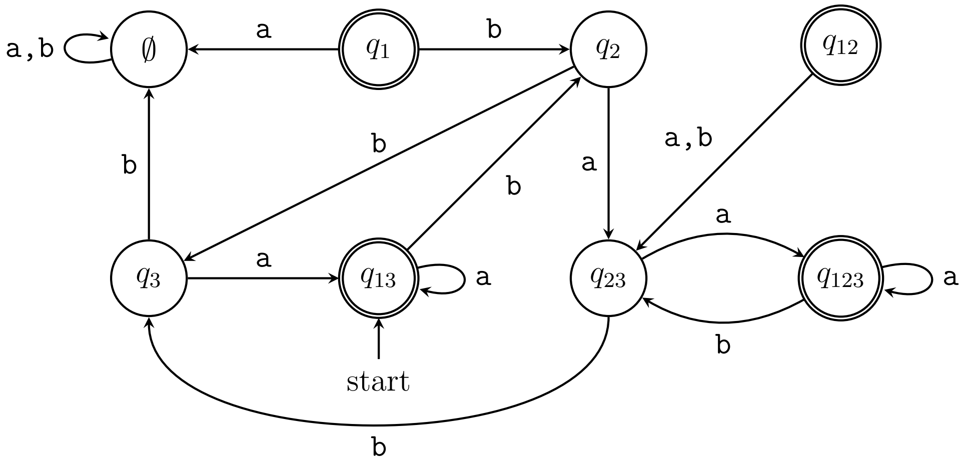
- $Q_D = \{\emptyset, q_1, q_2, q_3, \{q_1, q_2\}, \{q_1, q_3\}, \{q_2, q_3\}, \{q_1, q_2, q_3\}\}$
  che scriviamo in notazione semplificata:

  $$Q_D = \{\emptyset, q_1, q_2, q_3, q_{12}, q_{13}, q_{23}, q_{123}\}$$

- $q_{0_D} = E(\{q_{0_N}\}) = E(q_1) = \{q_1, q_3\} = q_{13}$
- $F_D = \{q_1, q_{12}, q_{13}, q_{123}\}$
- δ_D viene definita come scritto prima, per esempio:

- $\delta_D(q_1, a) = E(\delta_N(q_1, a)) = \emptyset$
- $\delta_D(q_1, b) = E(\delta_N(q_1, b)) = q_2$
- $\delta_D(q_2, a) = E(\delta_N(q_2, a)) = \{q_2, q_3\} = q_{23}$
- $\delta_D(q_2, b) = E(\delta_N(q_2, b)) = q_3$
- $\delta_D(q_{12}, b) = E(\delta_N(q_1, b)) \cup E(\delta_N(q_2, b)) = \{q_2, q_3\} = q_{23}$
- $\delta_D(q_{13}, a) = E(\delta_N(q_1, a)) \cup E(\delta_N(q_3, a)) = \{\emptyset, q_1, q_3\} = q_{13}$

Il DFA equivalente quindi è:



1.5 Espressioni regolari

Definizione di espressione regolare

Dato un alfabeto Σ, un'**espressione regolare** di Σ è una stringa r che rappresenta un linguaggio $L(r) \subseteq \Sigma^*$. L'insieme delle espressioni regolari di un alfabeto Σ, scritte come $\text{re}(\Sigma)$, è definito:

- $\emptyset \in \text{re}(\Sigma)$
- $\varepsilon \in \text{re}(\Sigma)$
- $a \in \text{re}(\Sigma)\ \forall a \in \Sigma$
- $r_1, r_2 \in \text{re}(\Sigma) \implies r_1 \cup r_2 \in \text{re}(\Sigma)$
- $r_1, r_2 \in \text{re}(\Sigma) \implies r_1 \circ r_2 \in \text{re}(\Sigma)$
- $r \in \text{re}(\Sigma) \implies r^* \in \text{re}(\Sigma)$

Esempio:
Dato l'alfabeto $\Sigma = \{0, 1\}$

- $0 \cup 1 = \{0\} \cup \{1\} = \{0, 1\}$
- $0^*10^* = \{0\}^* \circ \{1\} \circ \{0\}^* = \{x1y | x, y \in \{0\}^*\}$

- $\Sigma^*1\Sigma^* = \Sigma^* \circ \{1\} \circ \Sigma^* = \{x1y | x, y \in \Sigma^*\}$
- $1^*\emptyset = \{1\}^* \circ \emptyset = \emptyset$
- $\emptyset^* = \varepsilon$

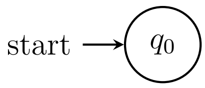
Conversione da espressione regolare a DFA

Date la classe dei linguaggi descritti da un'espressione regolare $\mathcal{L}(\text{re})$ e $\mathcal{L}(DFA)$:

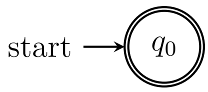
$$\mathcal{L}(\text{re}) \subseteq \mathcal{L}(DFA)$$

Dimostrazione per induzione:

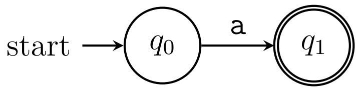
- Caso base:
  - $r = \emptyset \in \text{re}(\Sigma)$, definiamo il DFA $D_\emptyset$:



    in cui $x \in L(r) \iff x \in L(D_\emptyset)$ quindi $L(r) \in \mathcal{L}(DFA)$
  - $r = \varepsilon \in \text{re}(\Sigma)$, definiamo il DFA D_ε:



    in cui $x \in L(r) \iff x \in L(D_\varepsilon)$ quindi $L(r) \in \mathcal{L}(DFA)$
  - $r = a \in \text{re}(\Sigma)$, definiamo il DFA D_a:



    in cui $x \in L(r) \iff x \in L(D_a)$ quindi $L(r) \in \mathcal{L}(DFA)$

- Passo induttivo:
  - $r = r_1 \cup r_2$, allora abbiamo che:

$$L(r) = L(r_1) \cup L(r_2) = L(D_1) \cup L(D_2) \in \mathcal{L}(DFA)$$

  - $r = r_1 \circ r_2$, allora abbiamo che:

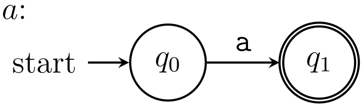
$$L(r) = L(r_1) \circ L(r_2) = L(D_1) \circ L(D_2) \in \mathcal{L}(DFA)$$

  - $r = r_1^*$, allora abbiamo che:

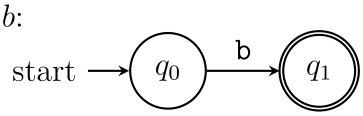
$$L(r) = L(r_1)^* = L(D_1)^* \in \mathcal{L}(DFA)$$

Esempio:
Data l'espressione regolare $(a \cup ab)^*$, il NFA corrispondente a tale espressione creata partendo dai sotto-componenti:

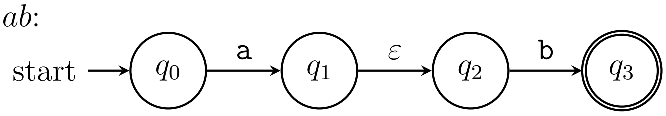
- a:



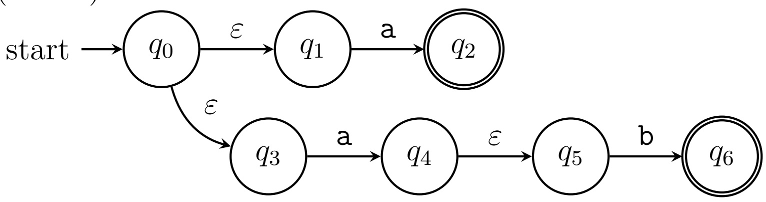
- b:



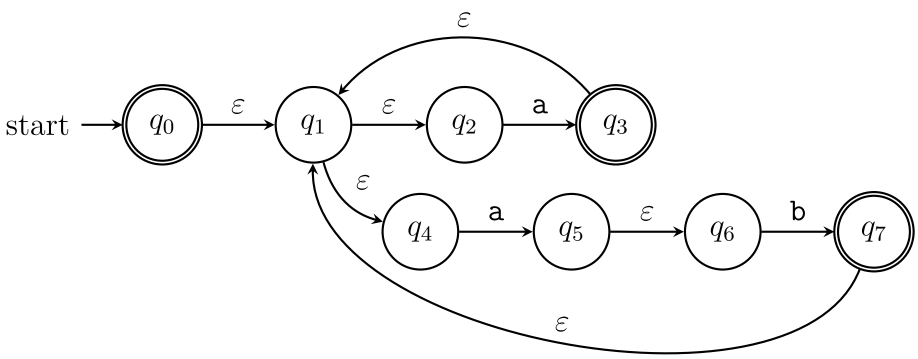
- ab:



- $(a \cup ab)$:



- $(a \cup ab)^*$:



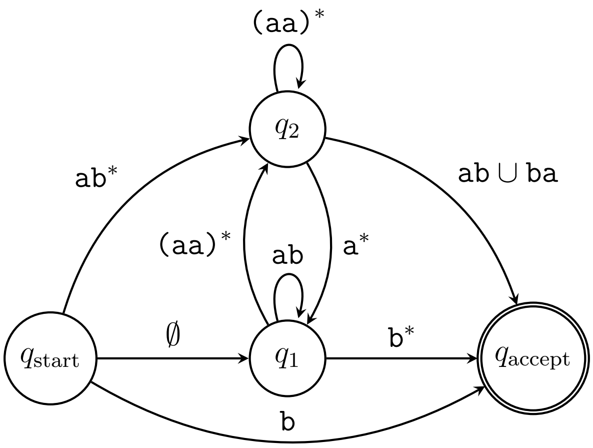
1.5.1 NFA generalizzati (GNFA)

NFA generalizzato (GNFA)

Un **GNFA (Generalized NFA)** è una tupla $(Q, \Sigma, \delta, q_{\text{start}}, q_{\text{accept}})$ in cui:

- $|Q| \geq 2$ è l'insieme degli stati dell'automa
- Σ è l'alfabeto dell'automa
- $q_{\text{start}} \in Q$ è lo stato iniziale dell'automa
- $q_{\text{accept}} \in Q$ è l'unico stato accettante dell'automa
- $\delta : (Q - q_{\text{accept}}) \times (Q - q_{\text{start}}) \to \text{re}(\Sigma)$ è la funzione di transizione degli stati in cui però:
  - q_{start} ha solo archi uscenti
  - q_{accept} ha solo archi entranti
  - Per ogni coppia di stati (q_1, q_2), c'è esattamente un arco $q_1 \to q_2$ e un arco $q_2 \to q_1$, incluse le coppie (q, q)
  - Le etichette degli archi sono espressioni regolari

Esempio:



Conversione da DFA a GNFA

Date $\mathcal{L}(DFA)$ e $\mathcal{L}(GNFA)$ si ha che:

$$\mathcal{L}(DFA) \subseteq \mathcal{L}(GNFA)$$

Dimostrazione:
Dato $L \in \mathcal{L}(DFA)$, allora esiste $D = (Q, \Sigma, \delta, q_0, F)$ tale che $L(D) = L$.
Costruisco un GNFA $G = (Q_G, \Sigma, \delta_G, q_{\text{start}}, q_{\text{accept}})$ costruito da D in cui:

- $Q_G = Q \cup \{q_{\text{start}}, q_{\text{accept}}\}$
- $\delta_G(q_{\text{start}}, q_0) = \varepsilon$

- $\forall q \in F\ \delta_G(q, q_{\text{accept}}) = \varepsilon$
- Ogni transizione con etichette multiple in D viene trasformata in un'etichetta con l'unione delle etichette multiple
- Per ogni coppia per cui non ci sia un arco in un verso o nell'altro in D viene aggiunto un arco in G con etichetta $\emptyset$

Quindi $x \in L(D) \implies x \in L(G)$, quindi $\mathcal{L}(DFA) \subseteq \mathcal{L}(GNFA)$.

1.5.2 Riduzione minimale di un GNFA

Dato un GNFA $G = (Q, \Sigma, \delta, q_{\text{start}}, q_{\text{accept}})$, possiamo usare un'algoritmo per ottenere un GNFA G' con solo due stati (equivalente quindi ad un'espressione regolare) e per cui $L(G) = L(G')$:

Algoritmo: Riduzione minimale di un GNFA

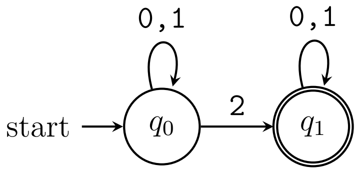
```
def ReduceGNFA(G):
    if |Q|==2 :
        return G
    q = q ∈ Q − {q_start, q_accept}// scelto in modo casuale
    Q′ = Q − q
    for q_i in Q′ − {q_accept} :
        for q_j in Q′ − {q_start} :
            δ′(q_i, q_j) = δ(q_i, q)δ(q, q) ∗ δ(q, q_j) ∪ δ(q_i, q_j)
    G′ = (Q′, Σ, δ′, q_start, q_accept)
    return ReduceGNFA(G’)
```

Dimostrazione per induzione:
Induzione in base al numero degli stati k

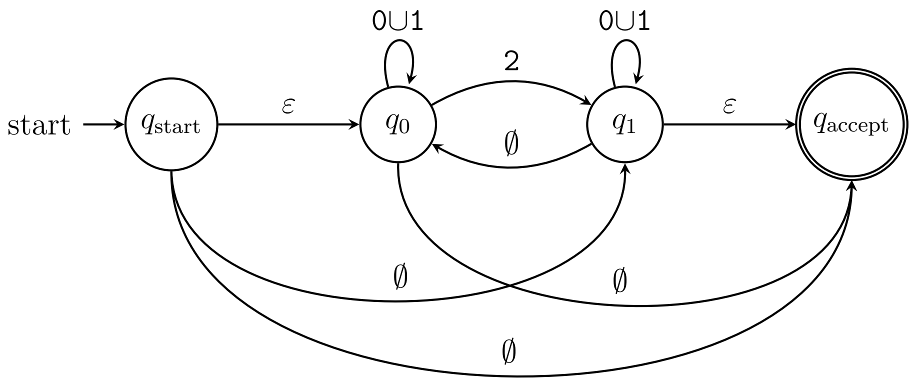
- Caso base:
  $k = 2 \implies G' = G \implies L(G') = L(G)$
- Caso induttivo:
  Assumo che per un G_k sia vero che $L(G_k) = L(G)$
- Passo induttivo:
  Dato $G_{k-1} = G_k - \{q_r\}$ si ha che se G_k accetta una stringa x, allora esiste una serie di stati $q_{\text{start}} q_1 \dots q_{\text{accept}}$ in cui abbiamo due casi:
  - $q_r \notin q_{\text{start}} q_1 \dots q_{\text{accept}}$ allora se $x \in L(G_k) \implies x \in L(G_{k-1})$
  - $q_r \in q_{\text{start}} q_1 \dots q_{\text{accept}}$ allora gli stati q_i e q_j vicini allo stato rimosso saranno collegati da una nuova espressione regolare che per cui $\delta'(q_i, q_j) = \delta(q_i, q_r)\delta(q_r, q_r) * \delta(q_r, q_j)$ per cui $x \in L(G_k) \implies x \in L(G_{k-1})$

  Allora se $x \in L(G_{k+1})$ allora per ogni coppia (q_i, q_j) l'etichetta rappresenterà tutti i cammini tra q_i e q_j anche in G_k quindi $x \in L(G_{k-1}) \implies x \in L(G_k)$. Quindi alla fine si ha che $L(G) = L(G_k) = L(G_{k-1})$

Esempio:
Dato il seguente DFA:

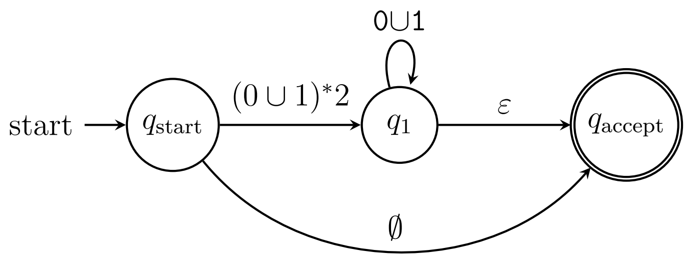


Il GNFA equivalente sarà:



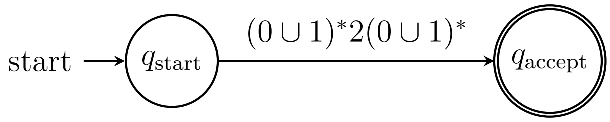
Rimuovendo lo stato q_0 e ricalcolando le transizioni:

- $\delta'(q_{\text{start}}, q_1) = \delta(q_{\text{start}}, q_0)\delta(q_0, q_0)^*\delta(q_0, q_1) \cup \delta(q_{\text{start}}, q_1) = \varepsilon(0 \cup 1) * 2 \cup \emptyset = (0 \cup 1)^*2$
- $\delta'(q_{\text{start}}, q_{\text{accept}}) = \delta(q_{\text{start}}, q_0)\delta(q_0, q_0)^*\delta(q_0, q_{\text{accept}}) \cup \delta(q_{\text{start}}, q_{\text{accept}}) = \varepsilon(0 \cup 1)^*\emptyset \cup \emptyset = \emptyset$
- $\delta'(q_1, q_1) = \delta(q_1, q_0)\delta(q_0, q_0)^*\delta(q_0, q_1) \cup \delta(q_1, q_1) = \emptyset(0 \cup 1)^*2 \cup (0 \cup 1) = 0 \cup 1$
- $\delta(q_1, q_{\text{accept}}) = \delta(q_1, q_0)\delta(q_0, q_0)^*\delta(q_0, q_{\text{accept}}) \cup \delta(q_1, q_{\text{accept}}) = \emptyset(0 \cup 1)^*\emptyset \cup \varepsilon = \varepsilon$



Rimuovendo anche lo stato q_1 e ricalcolando l'ultima transizione:

$$\delta'(q_{\text{start}}, q_{\text{accept}}) = \delta(q_{\text{start}}, q_1)\delta(q_1, q_1)^*\delta(q_1, q_{\text{accept}}) \cup \delta(q_{\text{start}}, q_{\text{accept}}) =$$
$$(0 \cup 1)^*2(0 \cup 1)^*\varepsilon \cup \emptyset = (0 \cup 1)^*2(0 \cup 1)^*$$



Conversione da GNFA a espressione regolare

Dati $\mathcal{L}$(GNFA) e $\mathcal{L}$(re) si ha che:

$$\mathcal{L}(\text{GNFA}) \subseteq \mathcal{L}(\text{re})$$

Equivalenza nella classe dei linguaggi regolari

Per un alfabeto Σ:

$$\text{REG} = \mathcal{L}(\text{DFA}) = \mathcal{L}(\text{NFA}) = \mathcal{L}(\text{GNFA}) = \mathcal{L}(\text{re})$$

1.6 Pumping Lemma

Preso un linguaggio L, ad esempio:

$$L = \{0^n 1^n | n \geq 0\}$$

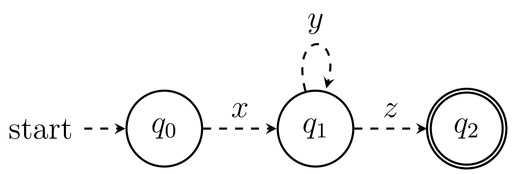
Questo linguaggio avrebbe bisogno di un automa ad infiniti stati per poter sapere il numero di 0 e 1, quindi impossibile con un DFA, NFA o GNFA.

Pumping Lemma

Dato un linguaggio $L \in \text{REG}$, allora esiste un DFA D con p stati tale che $L(D) = L$. $\forall w \in L$, con $|w| \geq p$, è ovvio che alcuni stati devono ripetersi e dividendo $w = xyz$ con x, y, z sottostringhe di w si ha che:

- $xy^k z \in L \ \forall k \geq 0$
- $|y| > 0$
- $|xy| < p$

Graficamente, con le freccie tratteggiate che indicano che in mezzo possono esserci altri stati:



Dimostrazione:
Dato $L \in \text{REG}$ e il DFA D per cui $L(D) = L$, consideriamo $p = |Q|$ e $w = w_1 \dots w_n$, con $n \geq p$, allora esiste una sequenza di stati $q_1 \dots q_{n+1}$ in cui:

$$\delta(q_k, w_k) = q_{k+1} \quad \forall k$$

La sequenza è lunga $n+1 \geq p+1$ perciò tra i primi $p+1$ stati ci deve essere almeno uno stato ripetuto. Siano q_i e q_j le due occorrenze di questo stato ripetuto, con $i < j \leq p+1$. Dividiamo la stringa in:

- $x = w_1 \dots w_{i-1} \implies \delta^*(q_1, x) = q_i$
- $y = w_i \dots w_{j-1} \implies \delta^*(q_i, y) = q_j = q_i$
- $z = w_j \dots w_n \implies \delta^*(q_j, z) = q_{n+1} \in F$

Quindi $\delta(q_1, xy^k z) = q_{n+1} \in F \implies xy^k z \in L$.
Inoltre sapendo che $i \neq j \implies |y| > 0$ e $j \leq p + 1 \implies |xy| \leq p$.

1.6.1 Dimostrazione di non regolarità tramite pumping lemma

Il pumping lemma viene usato per dimostrare che una linguaggio non è regolare, per farlo basta trovare una stringa che non possa essere scomposta secondo il pumping lemma.
Esempio:
Dato il linguaggio:

$$L = \{0^n 1^n | n \geq 0\}$$

Preso un valore p del pumping lemma, scelgo la stringa $w = 0^p 1^p$ con quindi $|w| = 2p > p$, visto che $|xy| < p \implies y$ è composta da solo 0, questo vuol dire che con indice $i = 2$:

$$xy^2 z = 0^q 1^p \; q > p \implies xy^2 z \notin L$$

Quindi il linguaggio non è regolare.

2
Grammatiche acontestuali

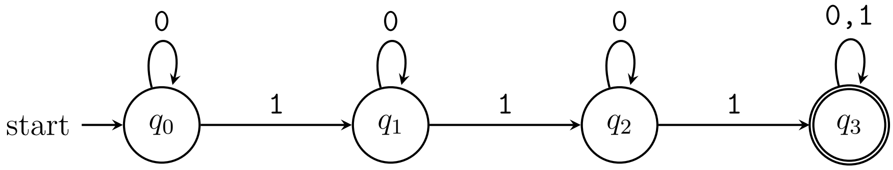
E

Esercizi

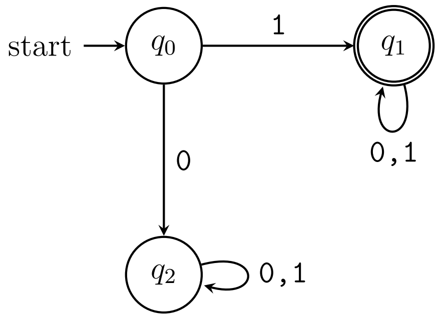
E.1 Esercizi sui linguaggi regolari

E.1.1 Costruire un automa da un linguaggio

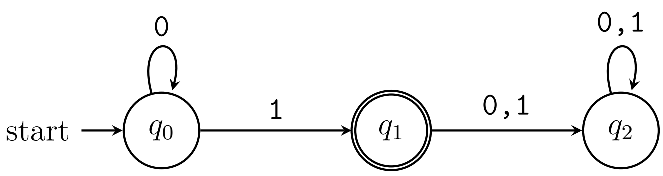
1. Dato un linguaggio $L(D) = \{x \in \{0,1\}^* | w_H(x) \geq 3\}$, per cui $w_H(x) = \{\text{numero di 1 in } x\}$, costruire un DFA che accetta questo linguaggio:



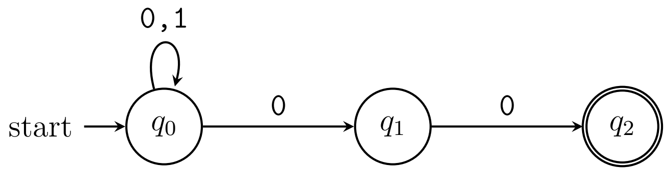
2. Dato un linguaggio $L(D) = \{x \in \{0,1\}^* | x = 1y \wedge y \in 0,1^*\}$, costruire un DFA che accetta questo linguaggio:



3. Dato un linguaggio $L(D) = \{x \in \{0,1\}^* | x = 0^n 1\}$, costruire un DFA che accetta questo linguaggio:



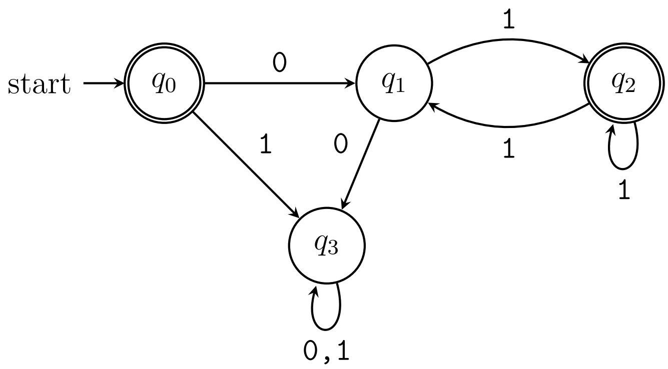
4. Dato un linguaggio $L = \{w \in \{0,1\}^* | w = x00 \ x \in \{0,1\}^*\}$, costruire un NFA che accetta questo linguaggio:



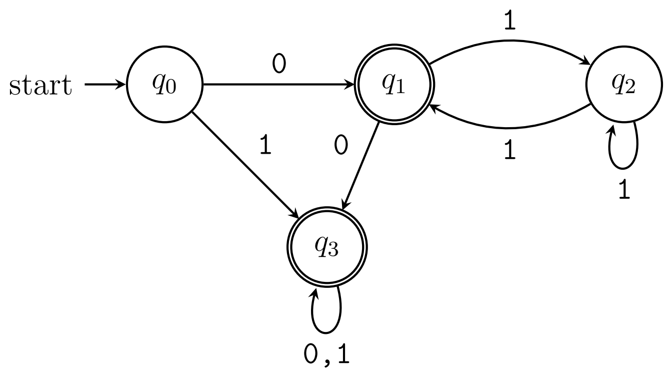
Dimostrazione per induzione:

- Caso base: $w = 00 \implies w \in L$
- Passo induttivo: $w = w'00$ $w = \{0,1\}^*$
  $w \in L$ perchè $\forall w'$ c'è sempre un ramo di computazione che si trova nello stato q_0. Quindi leggendo 00 alla fine arriva nello stato q_2, quindi $w \in L$.

5. Dato un linguaggio $L = \{w \in \{0,1\}^* | w \notin (01^+)^*\}$, costruire un DFA che accetta questo linguaggio:
   Posso creare un DFA che accetta il linguaggio complementare $\neg L = \{w \in (01^+)^*\}$:

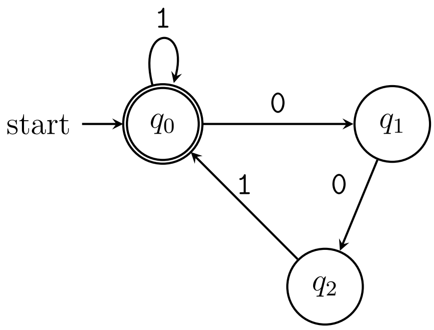


Quindi il DFA che accetta il linguaggio iniziale è quello con gli stati accettanti invertiti rispetto a quello sopra:



E.1.2 Costruire un automa da un'espressione regolare

1. Data l'espressione regolare $r = 1^*(001^+)^*$ costruire il DFA equivalente a r:



E.1.3 Dimostrare che un linguaggio non è regolare

1. Dimostrare che il linguaggio

$$L = \{ww^R | w \in \{0,1\}^*\}$$

dove w^R è w rovesciata ($w = 100 \implies w^R = 001$) non è regolare:
Presa la stringa $w = 0^p 1 1 0^p \in L$ con $|w| \geq p$ e $|xy| < p$ allora y contiene solo 0.
Scrivo $w = 0^k 0^l 0^m 1 1 0^p$ con:

- $x = 0^k \; k \geq 0$
- $y = 0^l \; l > 0$
- $z = 0^m 1 1 0^p$
- $k + l + m = p$

Con $i = 2$ la stringa $xy^2z = 0^k 0^{2l} 0^m 1 1 0^p \implies k + 2l + m > p \implies xy^2z \notin L \implies L$ non è regolare.

2. Dimostrare che il linguaggio

$$L = \{1^{n^2} | n \geq 0\}$$

non è regolare:
Presa la stringa $w = 1^{p^2}$ con $|w| > p$ e $|xy| < p$.
Scrivo $w = 1^k 1^l 1^{p^2-k-l}$ con:

- $x = 1^k \; k \geq 0$
- $y = 1^l \; l > 0$
- $z = 1^{p^2-l-k}$
- $k + l \leq p$

Con $i = 2$ la stringa $xy^2z = 1^k 1^{2l} 1^{p^2-l-k} = 1^{p^2+l} \implies p^2 < |xy^2z| < (p+1)^2 \implies xy^2z \notin L \implies L$ non è regolare.