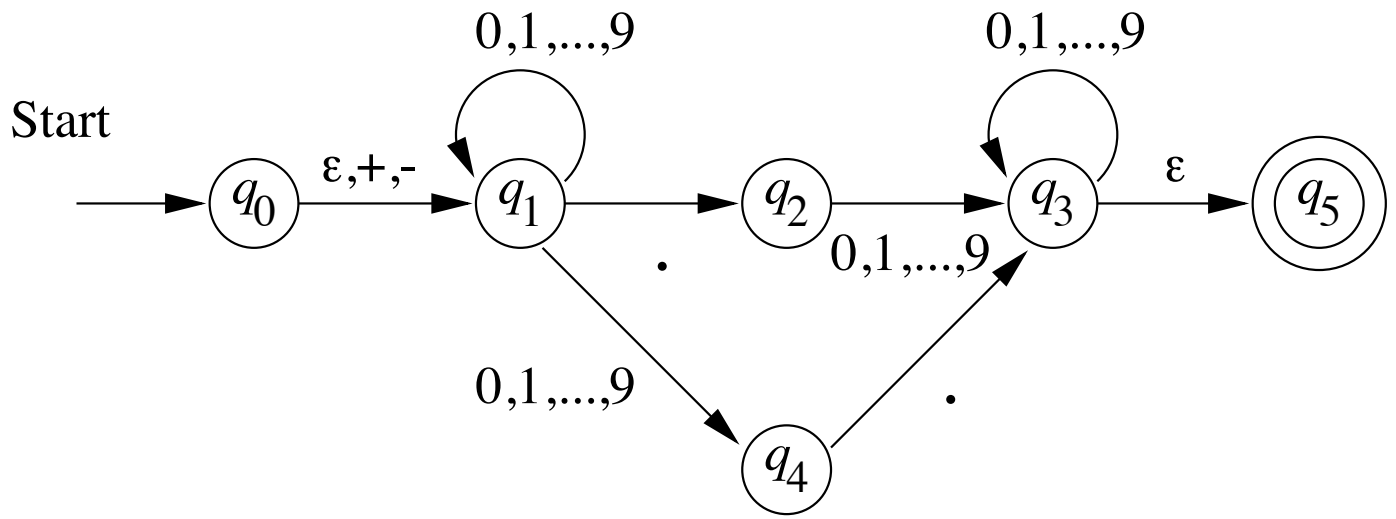
NFA per riconoscere numeri decimali

Vogliamo un NFA che accetta numeri decimali consiste di:

1. Un segno + o -, **opzionale**
2. Una stringa di cifre decimali
3. un punto decimale
4. un'altra stringa di cifre decimali

Una delle stringhe (2) e (4), ma non entrambi, può essere vuota.
Cosa succede se non compare il segno? Uso un ϵ-NFA.



Definizione ed esempio

Un ϵ-NFA è una quintupla $(Q, \Sigma, \delta, q_0, F)$ dove:

- $\delta : Q \times \Sigma \cup \{\epsilon\} \to 2^Q$ è una funzione da $Q \times \Sigma \cup \{\epsilon\}$ all'insieme dei sottoinsiemi di Q.

Esempio: L'ϵ-NFA della pagina precedente

$$E = (\{q_0, q_1, \ldots, q_5\}, \{., +, -, 0, 1, \ldots, 9\}\ \delta, q_0, \{q_5\})$$

dove la tabella delle transizioni (con una colonna per ϵ) per δ è

	ϵ	+,-	.	$0, \ldots, 9$
$\to q_0$	$\{q_1\}$	$\{q_1\}$	$\emptyset$	$\emptyset$
q_1	$\emptyset$	$\emptyset$	$\{q_2\}$	$\{q_1, q_4\}$
q_2	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_3\}$
q_3	$\{q_5\}$	$\emptyset$	$\emptyset$	$\{q_3\}$
q_4	$\emptyset$	$\emptyset$	$\{q_3\}$	$\emptyset$
$\star q_5$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Epsilon-chiusura

ECLOSE(q) = gli stati raggiungibili da q tramite una sequenza di ϵ-transizioni

Definizione induttiva di ECLOSE(q)

Base:

$q \in \text{ECLOSE}(q)$

Induzione:

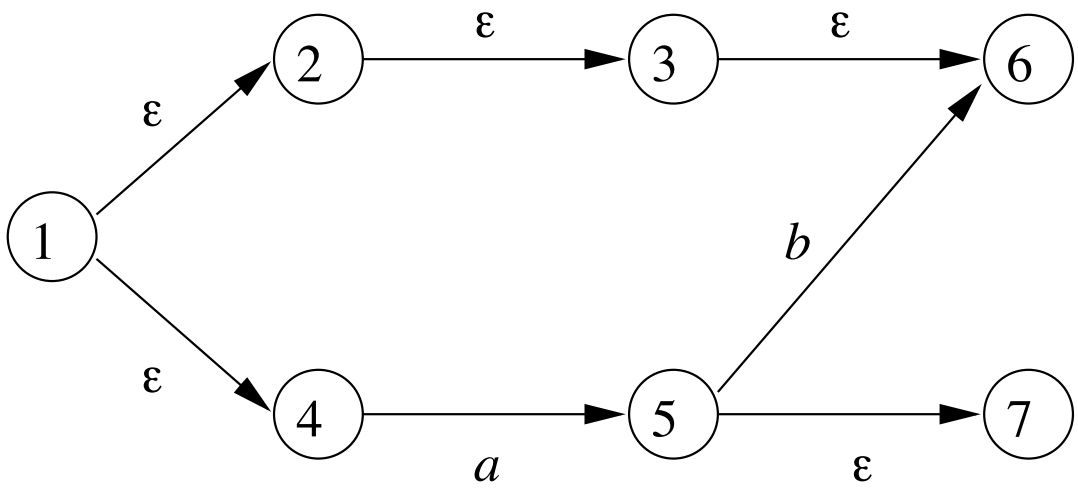
$p \in \text{ECLOSE}(q)$ and $r \in \delta(p, \epsilon) \quad \Rightarrow r \in \text{ECLOSE}(q)$

Nell'esempio di prima ogni stato coincide con la propria epsilon-chiusura, tranne $\text{ECLOSE}(q_0) = \{q_0, q_1\}$ e $\text{ECLOSE}(q_3) = \{q_3, q_5\}$ che corrispondono alle due ϵ-transizioni.

La stessa operazione si può applicare ad un insieme di stati:

$$\text{ECLOSE}(S) = \bigcup_{q \in S} \text{ECLOSE}(q)$$

Esempio di epsilon-chiusura



Per esempio,

$$\text{ECLOSE}(1) = \{1, 2, 3, 4, 6\}$$

Ogni uno di questi stati può essere raggiunto a partire da 1 attraverso ϵ-transizioni.

Funzione di transizione estesa $\hat{\delta}$ e il linguaggio accettato

Intuitivamente $\hat{\delta}(q, w)$ è l'insieme degli stati raggiungibili da q seguendo il percorso w (con possibili ϵ-transizioni)

- Definizione induttiva di $\hat{\delta}$ per automi ϵ-NFA

Base:

$$\hat{\delta}(q, \epsilon) = \text{ECLOSE}(q)$$

Induzione: $a \neq \epsilon$ Se

- $\hat{\delta}(q, x) = \{p_1, ..., p_k\}$;
- $\bigcup_{p_i \in \hat{\delta}(q,x)} \delta(p_i, a) = \{r_1, ..., r_m\}$

$$\hat{\delta}(q, xa) = \text{ECLOSE}(\bigcup_{p_i \in \hat{\delta}(q,x)} \delta(p_i, a)) = \text{ECLOSE}\{r_1, ..., r_m\}$$

Il linguaggio di un ϵ-NFA E è dato da:

$$L(E) = \{w | \hat{\delta}(q_0, w) \cap F_E \neq \emptyset\}$$

Da ϵ-NFA a DFA

Procedimento simile alla costruzione per sottoinsiemi: vanno incorporate le ϵ-transizioni, usando le ϵ-chiusure.
Dato un ϵ-NFA

$$E = (Q_E, \Sigma, \delta_E, q_0, F_E)$$

costruiremo un DFA

$$D = (Q_D, \Sigma, \delta_D, q_D, F_D)$$

tale che

$$L(D) = L(E)$$

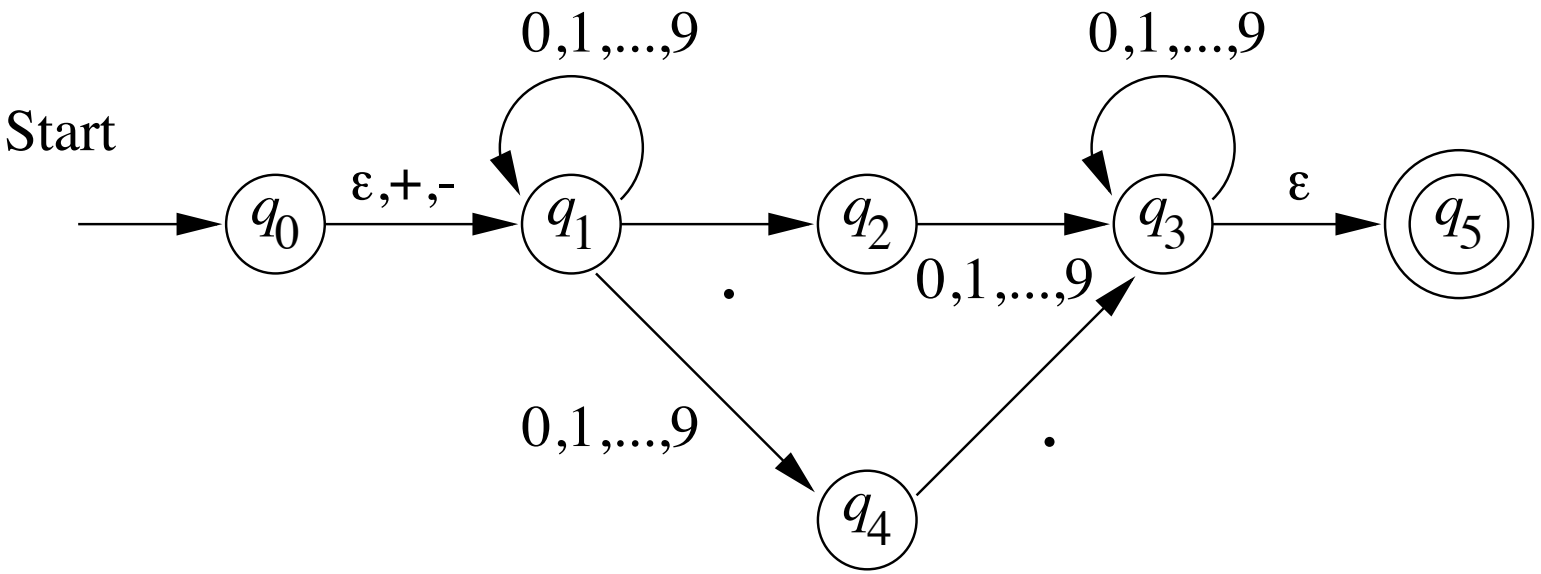
Dettagli della costruzione:

- $Q_D = \{S : S \subseteq Q_E \text{ e } S = \text{ECLOSE}(S)\}$
- $q_D = \text{ECLOSE}(q_0)$
- $F_D = \{S : S \in Q_D \text{ e } S \cap F_E \neq \emptyset\}$
- $\delta_D(S, a) =$

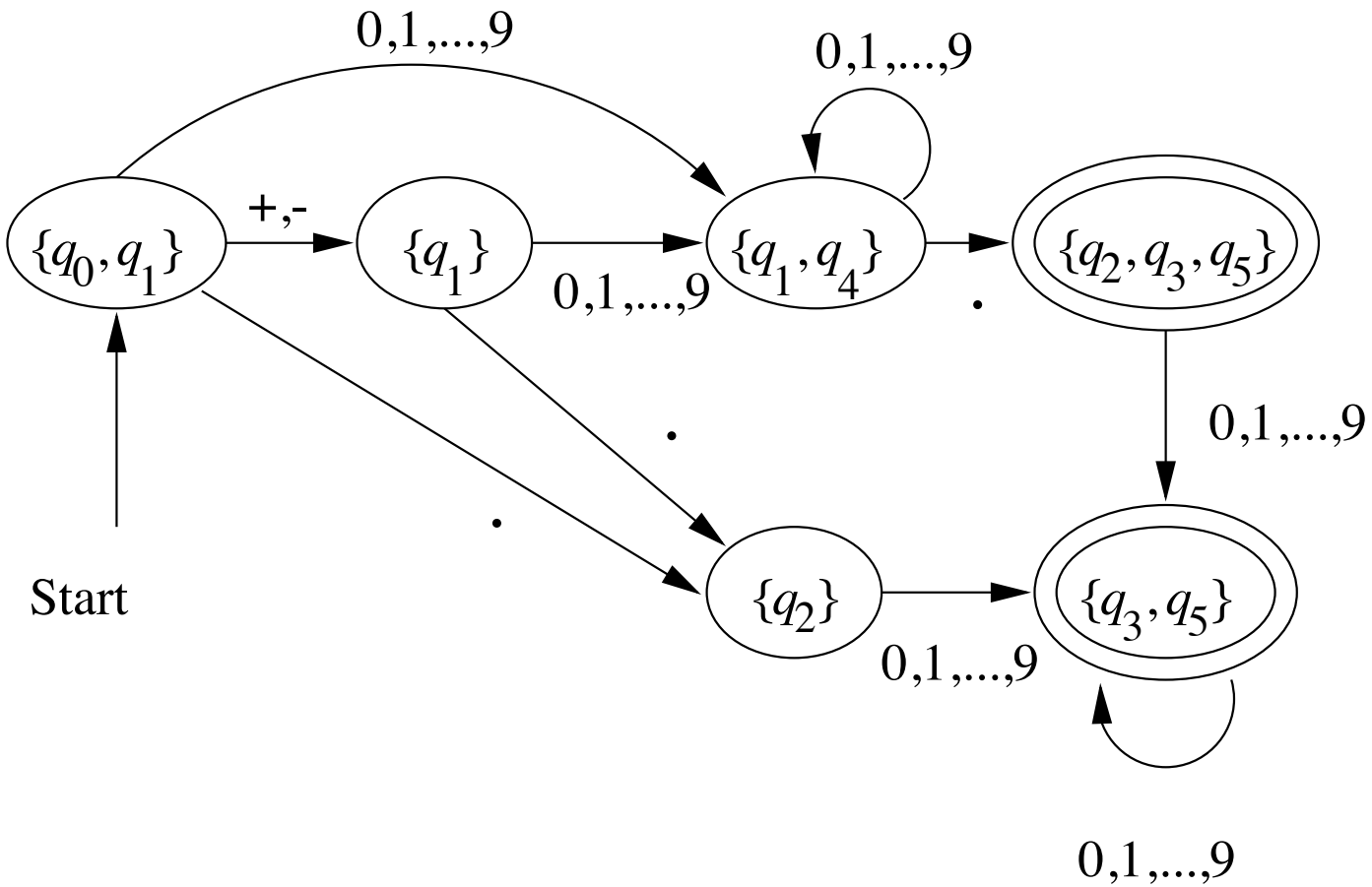
$$\bigcup \{\text{ECLOSE}(p) : p \in \delta(t, a) \text{ per alcuni } t \in S\}$$

Esempio

ϵ-NFA E per riconoscere numeri decimali in notazione anglo-sassone



DFA D corrispondente ad E



Teorema 2.22: Un linguaggio L è accettato da un ϵ-NFA E se e solo se L è accettato da un DFA.
Dimostrazione: Usiamo D costruito come sopra e mostriamo per induzione che $\hat{\delta}_E(q_0, w) = \hat{\delta}_D(q_D, w)$

Base: $\hat{\delta}_E(q_0, \epsilon) = \text{ECLOSE}(q_0) = q_D = \hat{\delta}(q_D, \epsilon)$

Induzione: $a \neq \epsilon$ Sia $w = xa$ con l'ipotesi valida per x, ovvero $\hat{\delta}_E(q_0, x) = \hat{\delta}_D(q_0, x) = \{p_1, ..., p_k\}$. Per la definizione di $\hat{\delta}_E$, $\hat{\delta}_E(q_0, w)$ si calcola come segue:

- sia $\bigcup_{p_i \in \hat{\delta}_E(q_0,x)} \delta(p_i, a) = \{r_1, ..., r_m\}$,
- allora

$$\hat{\delta}_E(q, xa) = \text{ECLOSE}\Big(\bigcup_{p_i \in \hat{\delta}(q,x)} \delta(p_i, a)\Big) = \text{ECLOSE}\,\{r_1, ..., r_m\}$$

Ricapitolando

- Tutti i tipi di automi visti, DFA, NFA e ϵ-NFA, accettano lo stesso insieme di linguaggi: i linguaggi regolari.
- Solo i DFA possono tuttavia essere implementati!

Espressioni regolari

- Un FA (NFA o DFA) è un metodo per costruire una macchina che riconosce linguaggi regolari.
- Le *espressioni regolari* una notazione algebrica per descrivere, in modo dichiarativo, i linguaggi regolari.
- Esempio: $\mathbf{01}^* + \mathbf{10}^*$
- Le espressioni regolari sono usate, ad esempio,
  - nella descrizione di particolari tipi di sequenze (pattern) nei testi
  - nei comandi UNIX (`grep`)
  - negli strumenti per l'analisi lessicale di UNIX (Lex (Lexical analyzer generator) e Flex (Fast Lex)).

Operazioni sui linguaggi

Le espressioni regolari usano le seguenti operazioni.

- **Unione:**
  $L \cup M = \{w : w \in L \text{ o } w \in M\}$
  Esempio: $\{0, 10\} \cup \{1, 10, 011\} = \{0, 10, 1, 10, 011\}$
- **Concatenazione:**
  $L.M = \{w : w = xy, x \in L, y \in M\}$
  Esempio:
  $\{0, 10\} \cup \{1, 10, 011\} = \{01, 010, 0011, 101, 1010, 10011\}$
- **Potenze:**
  $L^0 = \{\epsilon\},\ L^1 = L,\ L^{k+1} = L.L^k$
- **Chiusura di Kleene:**
  $L^* = \bigcup_{i=0}^{\infty} L^i$
  Esempio: $\{0, 10\}^* = \{\epsilon, 0, 10, 00, 010, 100, 1010, ...\}$

Definizione induttiva di espressioni regolari

Se E è un'espressione regolare, $L(E)$ denota il linguaggio che E definisce.

- **Base:**
  - ϵ e $\emptyset$ sono espressioni regolari.
    $L(\epsilon) = \{\epsilon\}$ e $L(\emptyset) = \emptyset$.
  - Se a è un simbolo in Σ, allora $\boldsymbol{a}$ è un'espressione regolare.
    $L(\boldsymbol{a}) = \{a\}$.
- **Induzione:**
  - Se E e F sono espressioni regolari, allora $E + F$ è un'espressione regolare. $L(E + F) = L(E) \cup L(F)$.
  - Se E e F sono espressioni regolari, allora $E.F$ è un'espressione regolare. $L(E.F) = L(E).L(F)$.
  - Se E è un'espressione regolare, allora (E) è un'espressione regolare. $L((E)) = L(E)$.
  - Se E è un'espressione regolare, allora E^* è un'espressione regolare. $L(E^*) = (L(E))^*$.

Ordine di precedenza per gli operatori

1. Chiusura (*)
2. Concatenazione (.)
3. Più (+)

Esempio: $\mathbf{01}^* + \mathbf{1}$ è raggruppato in $(\mathbf{0}(\mathbf{1})^*) + \mathbf{1}$

Esempi

- $L(\boldsymbol{ab}) = L(\boldsymbol{a}).L(\boldsymbol{b}) = \{ab\}$
- $L(\boldsymbol{ab} + \boldsymbol{b}) = L(\boldsymbol{ab}) \cup L(\boldsymbol{b}) = \{ab, b\}$
- $L(\boldsymbol{a}(\boldsymbol{ab} + \boldsymbol{b})) = L(\boldsymbol{a}).(L(\boldsymbol{ab}) \cup L(\boldsymbol{b})) =$
  $L(\boldsymbol{a}).L(\boldsymbol{ab}) \cup L(\boldsymbol{a}).L(\boldsymbol{b}) = \{aab, ab\}$
- $L(\boldsymbol{ab}^*) = (L(\boldsymbol{a})).(L(\boldsymbol{b}))^* = \{a, ab, abb, abbb, ...\}$
- $L((\boldsymbol{ab})^*) = (L(\boldsymbol{ab}))^* = \{\epsilon, ab, abab, ababab, ...\}$

Esempio

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

$$(\mathbf{01})^* + (\mathbf{10})^* + \mathbf{0}(\mathbf{10})^* + \mathbf{1}(\mathbf{01})^*$$

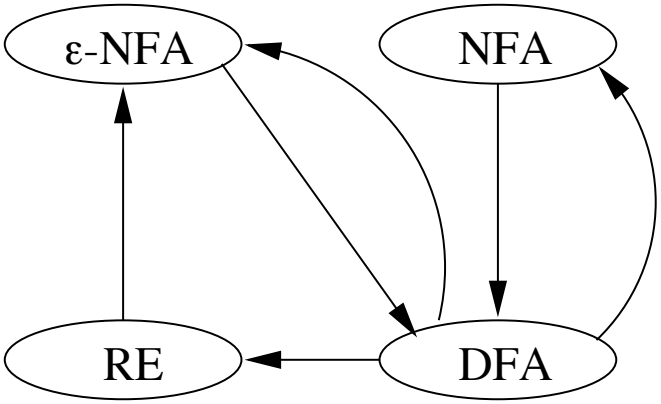
o, equivalentemente,

$$(\epsilon + \mathbf{1})(\mathbf{01})^*(\epsilon + \mathbf{0})$$

Notare che il linguaggio accetta anche le stringhe 0 e 1.

Equivalenza di FA e espressioni regolari

Abbiamo già mostrato che DFA, NFA, e ϵ-NFA sono tutti equivalenti.

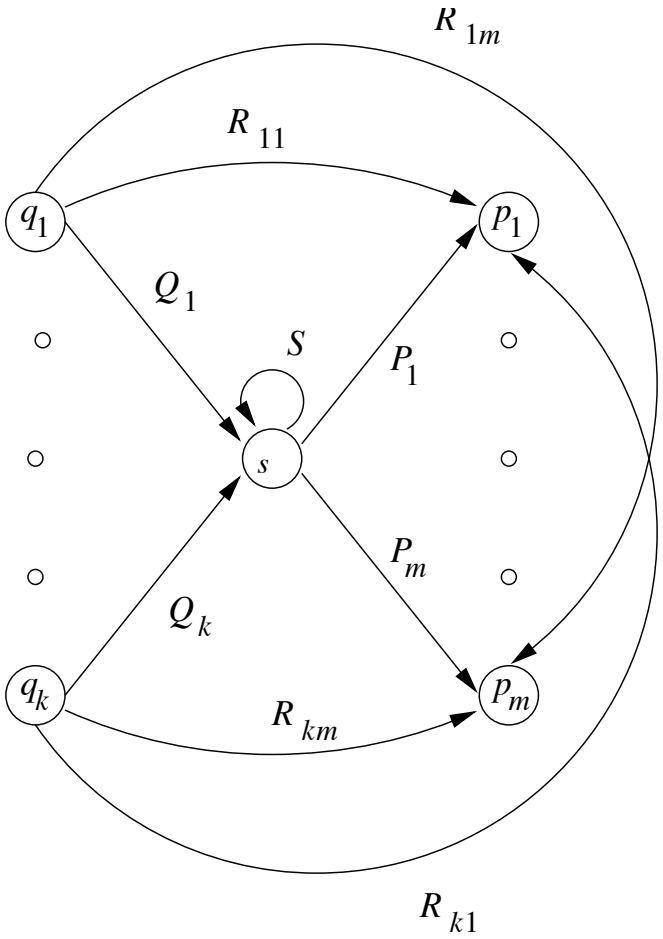


Per mostrare che gli FA sono equivalenti alle espressioni regolari, mostreremo che

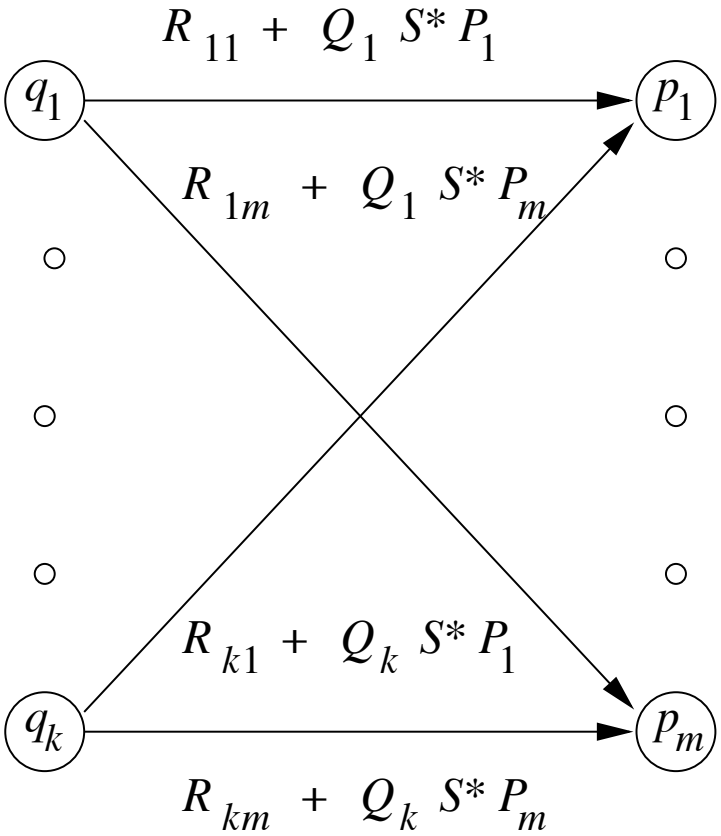
1. Per ogni DFA A possiamo trovare (costruire, in questo caso) un'espressione regolare R, tale che $L(R) = L(A)$.
2. Per ogni espressione regolare R esiste un ϵ-NFA A, tale che $L(A) = L(R)$.

La tecnica di eliminazione di stati

Etichettiamo gli archi con espressioni regolari di simboli

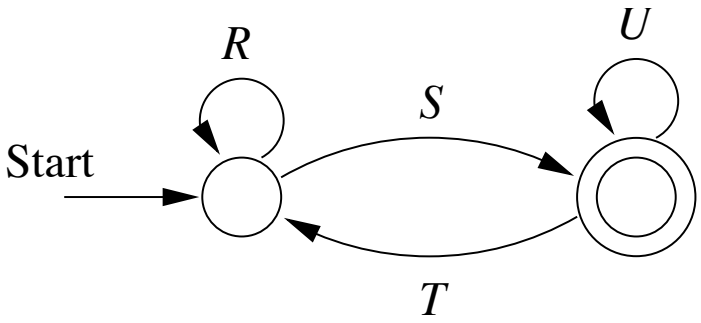


Ora eliminiamo lo stato s.

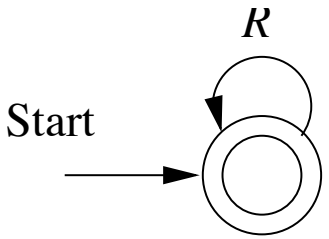


Per lo stato accettante q eliminiamo dall'automa originale tutti gli stati eccetto q_0 e q.

Per ogni $q \in F$ saremo rimasti con A_q della forma



che corrisponde all'espressione regolare $E_q = (R + SU^*T)^*SU^*$, se $q_0 \neq q$, o con A_q della forma



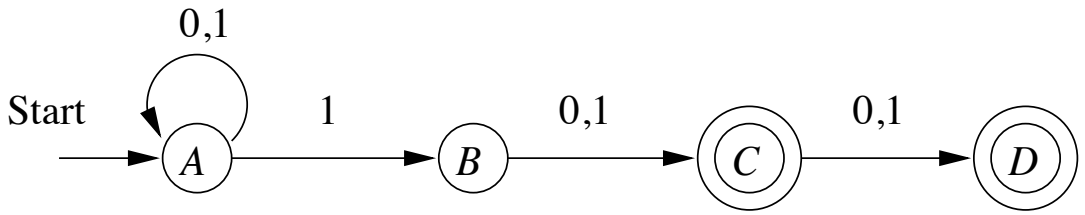
che corrisponde all'espressione regolare $E_q = R^*$, se $q_0 \in F$.

- L'espressione finale è

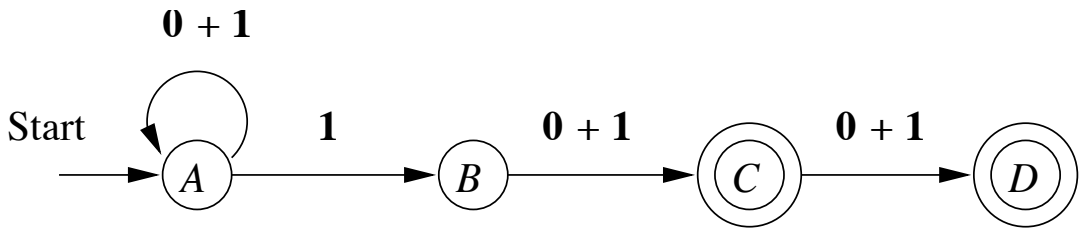
$$\bigoplus_{q \in F} E_q$$

Esempio

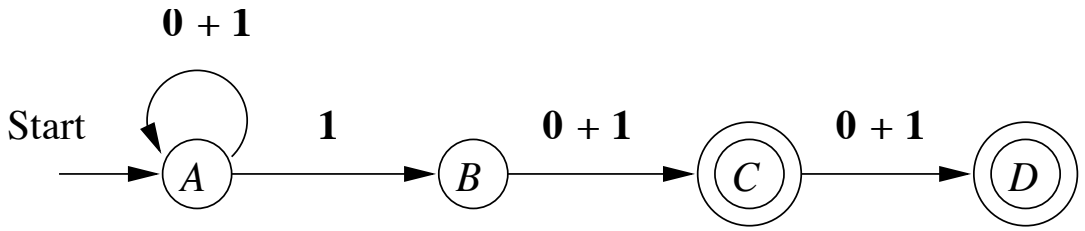
$\mathcal{A}$, dove $L(\mathcal{A}) = \{W : w = x1b,\ \text{o}\ w = x1bc,\ \ x \in \{0,1\}^*, \{b,c\} \subseteq \{0,1\}\}$



La trasformiamo in un automa con espressioni regolari come etichette

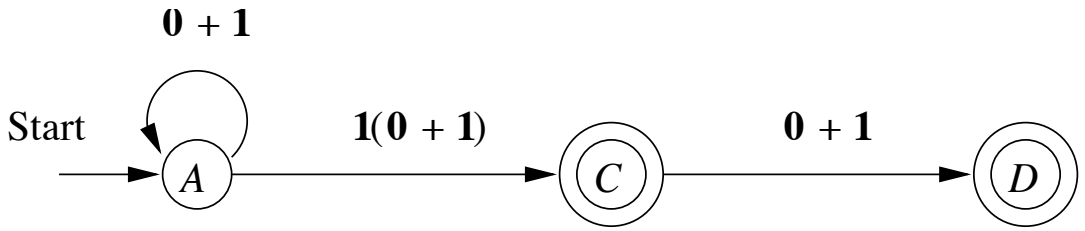


Esempio



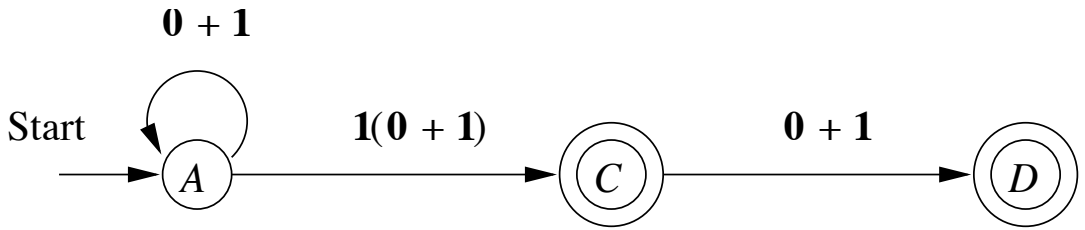
Eliminiamo lo stato B. L'espressione sul nuovo arco che va da A a C è $R_{11} + Q_1 S^* P_1$, dove $R_{11} = \emptyset$, $Q_1 = \mathbf{1}$, S $= \emptyset$, $P_1 = \mathbf{0} + \mathbf{1}$. Considerando che $\emptyset^* = \epsilon$, l'espressione è

$$\emptyset + \mathbf{1}\emptyset^*(\mathbf{0} + \mathbf{1}) = \mathbf{1}\emptyset^*(\mathbf{0} + \mathbf{1}) = \mathbf{1}(\mathbf{0} + \mathbf{1})$$

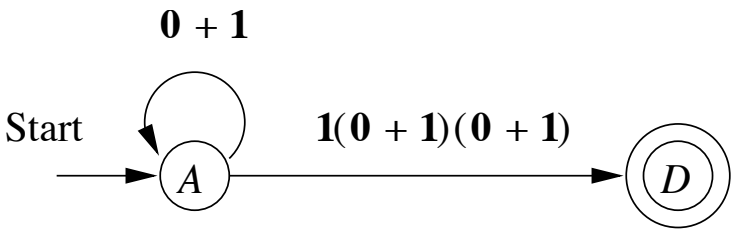


Esempio

Da



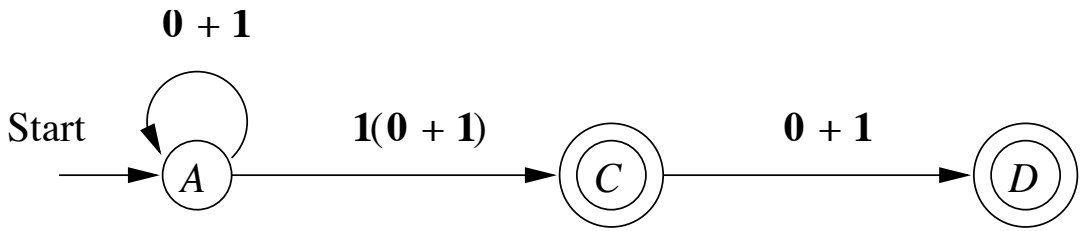
possiamo eliminare lo stato C e ottenere $\mathcal{A}_D$



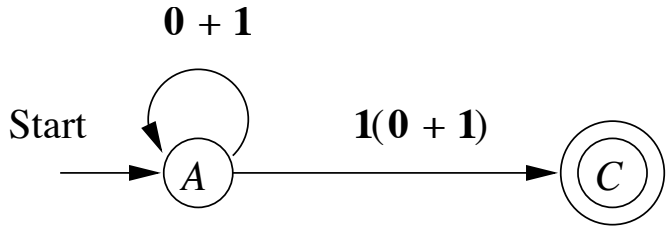
con espressione regolare $(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})(\mathbf{0}+\mathbf{1})$, dato che $R = \mathbf{0}+\mathbf{1}$, $S = \mathbf{1}(\mathbf{0}+\mathbf{1})(\mathbf{0}+\mathbf{1})$ e $U = T = \emptyset$ e che quindi $(R + SU^*T)^*SU^* = R^*S$.

Esempio

Da



possiamo eliminare D (e con esso l'arco che va da C a D) e ottenere $\mathcal{A}_C$ con espressione regolare $(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})$



- L'espressione finale è l'unione delle due precedenti:

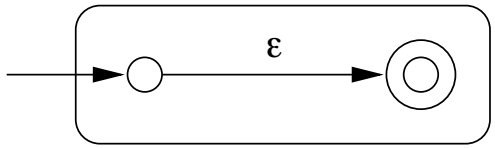
$$(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})(\mathbf{0}+\mathbf{1}) + (\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})$$

Da espressioni regolari a ϵ-NFA

Teorema 3.7: Per ogni espressione regolare R possiamo costruire un ϵ-NFA A, tale che $L(A) = L(R)$.

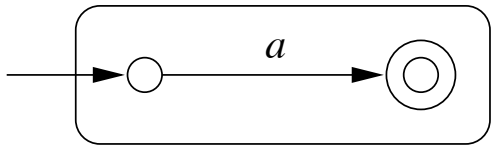
Dimostrazione: Per induzione strutturale:

Base: Automa per ϵ, $\emptyset$, e $\boldsymbol{a}$.



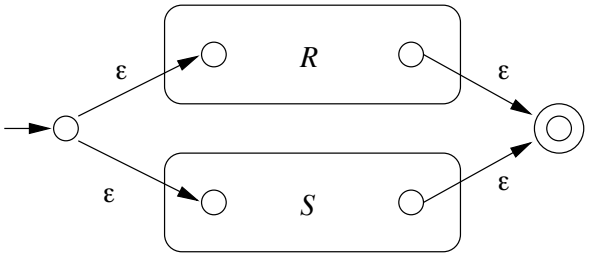
(a)

(b)

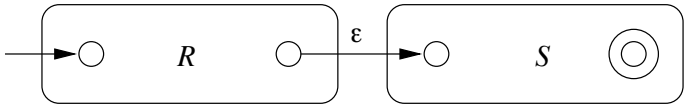


(c)

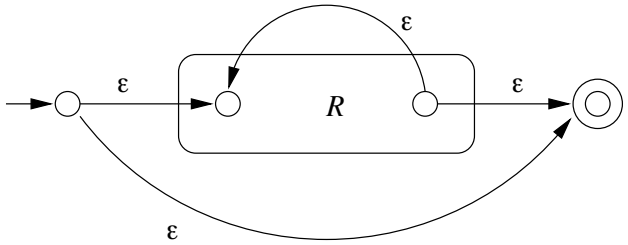
Induzione: Automa per $R + S$, RS, e R^* (quello per (R) coincide con quello per R)



(a)



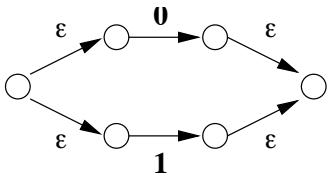
(b)



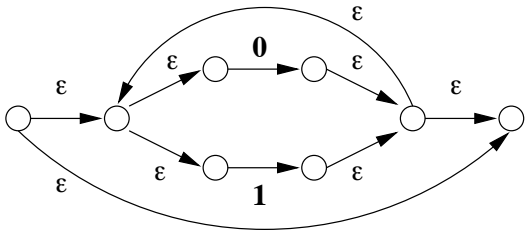
(c)

Esempio

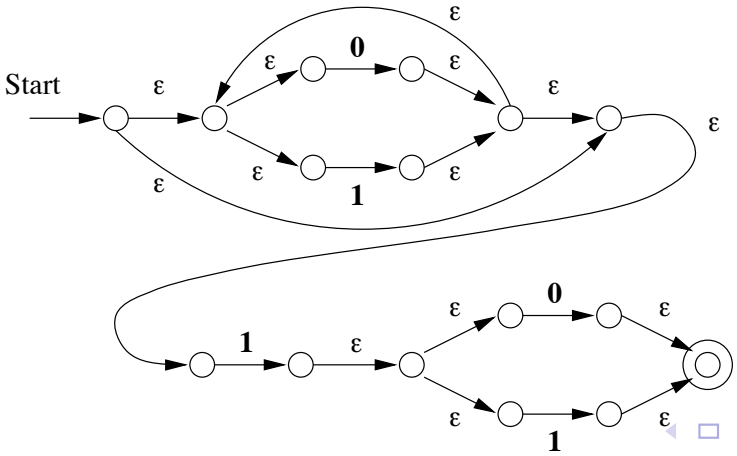
Trasformiamo $(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})$



(a)

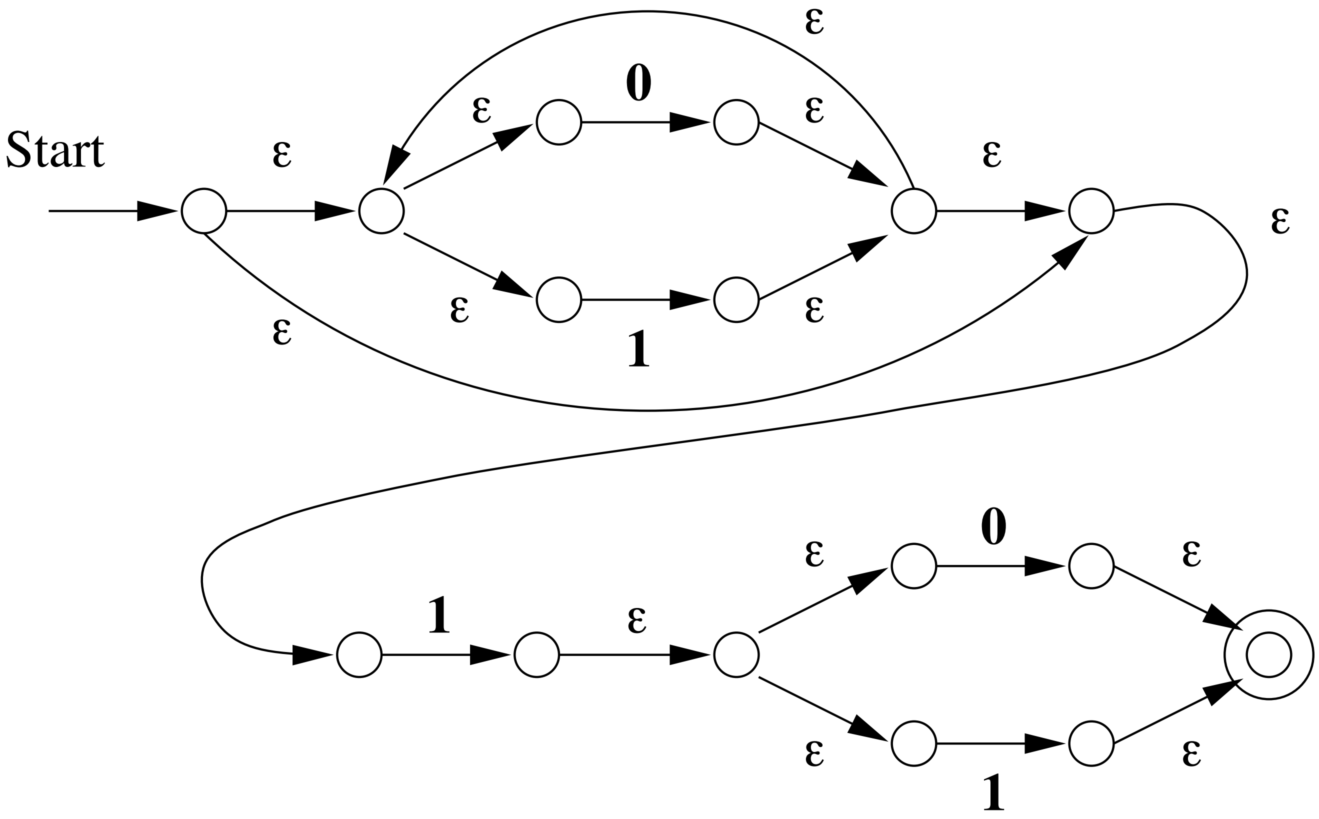


(b)



(b)

Esempio



(c)

Leggi algebriche per i linguaggi

- $L \cup M = M \cup L$.
  L'unione è *commutativa*.
- $(L \cup M) \cup N = L \cup (M \cup N)$.
  L'unione è *associativa*.
- $(LM)N = L(MN)$.
  La concatenazione è *associativa*.

Nota: La concatenazione non è commutativa, *cioè*, esistono L e M tali che $LM \neq ML$.

- $\emptyset \cup L = L \cup \emptyset = L$.
  $\emptyset$ è *l'identità* per l'unione.
- $\{\epsilon\}L = L\{\epsilon\} = L$.
  $\{\epsilon\}$ è l'identità *sinistra* e *destra* per la concatenazione.
- $\emptyset L = L\emptyset = \emptyset$.
  $\emptyset$ è l'annichilatore *sinistro* e *destro* per la concatenazione.

- $L(M \cup N) = LM \cup LN$.
  La concatenazione è *distributiva a sinistra* sull'unione.
- $(M \cup N)L = ML \cup NL$.
  La concatenazione è *distributiva a destra* sull'unione.
- $L \cup L = L$.
  L'unione è *idempotente*.
- $\emptyset^* = \{\epsilon\},\ \{\epsilon\}^* = \{\epsilon\}$.
- $L^+ = LL^* = L^*L,\ L^* = L^+ \cup \{\epsilon\}$

- $(L^*)^* = L^*$. La chiusura è *idempotente*.

Dimostrazione:

$$w \in (L^*)^* \iff w \in \bigcup_{i=0}^{\infty} \left(\bigcup_{j=0}^{\infty} L^j \right)^i$$

$$\iff \exists k, m \in \mathrm{N} : w \in (L^m)^k$$

$$\iff \exists p \in \mathrm{N} : w \in L^p$$

$$\iff w \in \bigcup_{i=0}^{\infty} L^i$$

$$\iff w \in L^*$$

Espressioni Regolari in UNIX

Le espressioni regolari fanno parte del sistema operativo UNIX fin dall'inizio e sono usate in vari comandi che elaborano testi (ad esempio `grep`, `sed`, `lex`, `vi`, `awk`, `ad`, `ex`, `pg`) per:

- elencare/eliminare righe che contengono un'espressione regolare
- sostituire un'espressione regolare (find/replace)
- ...

La sintassi è leggermente diversa da quella vista finora.

Espressioni Regolari in UNIX

- insiemi di caratteri: pattern elementari che specificano la presenza un carattere appartenente ad un certo insieme.
- ancore: legano il pattern a comparire una posizione specifica della riga (es. inizio, fine).
- gruppi: "racchiudono" l'espressione regolare che può quindi essere riferita come una singola entità.
- modificatori: specificano ripetizioni dell'espressione che precede il modificatore stesso.

Espressioni Regolari in UNIX: sintassi

- `[abc]` match any of the characters enclosed
- `[a-d]` match any character in the enclosed range
- `[set]` match any character not in the following set
- `.` match any single character except `<newline>`
- `[:alpha:]` some predefined character sets
- `[:digit:]`
- `\` treat the next char literally. Normally used to escape special characters such as "." and "*"