Espressioni regolari

Espressioni Regolari

- Un FA (NFA o DFA) e' una *macchina a stati finiti* che riconosce linguaggi regolari.
- Una *espressione regolare* e' un modo **dichiarativo** (o algebrico) per descrivere un linguaggio regolare.
- Esempio: $\mathbf{01}^* + \mathbf{10}^*$ denota il linguaggio delle stringhe composte da uno 0 seguito da qualsiasi numero di 1 o delle stringhe composte da un 1 seguito da qualsiasi numero di 0.

A Cosa Servono?

- La descrizione dichiarativa delle stringhe che compongono un linguaggio regolare fornisce qualcosa di diverso rispetto agli automi
- Le espressioni regolari per esempio vengono utilizzate come linguaggio di input per vari sistemi che trattano stringhe:
  - comandi UNIX (`grep`) per la ricerca di stringhe
  - generatori di *analizzatori lessicali*), tipo (Lex (Lexical analyzer generator) e Flex (Fast Lex)).
- In queste applicazioni viene preso un input, nella forma di una espressione regolare, e viene convertita l'espressione regolare in un DFA (o NFA)

Cosa vedremo?

- La definizione formale delle **espressioni regolari**
- Vedremo la relazione con i linguaggi regolari definiti da FA (DFA o NFA), dando delle traduzioni sistematiche da un formalismo nell'altro
- Tali traduzioni sono importanti per capire il **potere espressivo** del formalismo, ma sono anche utili **a livello pratico**

Operazioni sui linguaggi

- **Unione:**
  $L \cup M = \{w : w \in L \text{ o } w \in M\}$
- **Concatenazione:**
  $L.M = \{w : w = xy, x \in L, y \in M\}$
- **Potenze:**
  $L^0 = \{\epsilon\},\ L^1 = L,\ L^{k+1} = L.L^k$
- **Chiusura di Kleene:**
  $L^* = \bigcup_{i=0}^{\infty} L^i$

Esempio: operazioni sui linguaggi

- $L = \{0, 1\}$ allora L^* sono tutte le stringhe di $0, 1$
- $L = \{0, 11\}$ allora L^* sono tutte le stringhe di $0, 1$ in cui gli 1 compaiono a coppie. Inoltre

$$L^0 = \{\epsilon\},\ L^1 = L,\ L^2 = \{00, 011, 110, 1111\}$$

  e cosi' via
- Inoltre, $\emptyset^0 = \{\epsilon\}$, $\emptyset^i = \{\epsilon\}$, e $\emptyset^* = \{\epsilon\}$.

Definizione induttiva di espressioni regolari

Diamo la definizione delle espressioni regolari rispetto ad un dato alfabeto Σ. Indicheremo con $L(E)$ il linguaggio definito dall'espressione regolare E.

- **Base:**
  - ϵ e $\emptyset$ sono espressioni regolari.
    $L(\epsilon) = \{\epsilon\}$ e $L(\emptyset) = \emptyset$.
  - Se $a \in \Sigma$, allora $\mathbf{a}$ e' un'espressione regolare.
    $L(\mathbf{a}) = \{a\}$.

Definizione induttiva di espressioni regolari

- **Induzione:**
  - Se E e' un'espressione regolare, allora (E) e' un'espressione regolare. $L((E)) = L(E)$.
  - Se E e F sono espressioni regolari, allora $E + F$ e' un'espressione regolare. $L(E + F) = L(E) \cup L(F)$.
  - Se E e F sono espressioni regolari, allora $E.F$ e' un'espressione regolare. $L(E.F) = L(E).L(F)$.
  - Se E e' un'espressione regolare, allora E^* e' un'espressione regolare. $L(E^*) = (L(E))^*$.

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,

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

Parentesi

- Le parentesi hanno significato semantico.
- Per esempio, le espressioni regolari $(01)^*$ e $0(1)^*$ denotano linguaggi diversi
- Per semplificare la notazione e' possibile eliminare le parentesi in base ad un ordine di precedenza degli operatori (in modo standard)

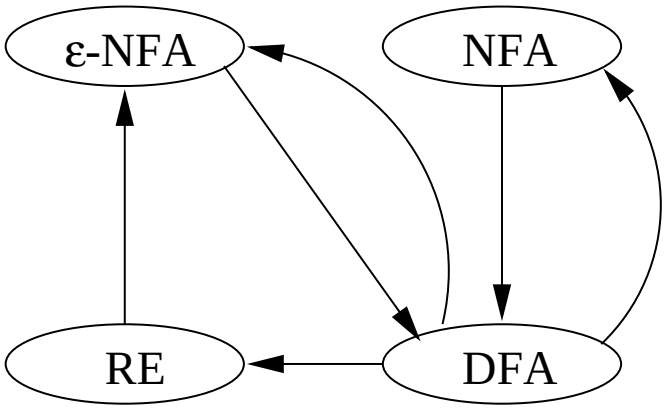
Ordine di precedenza per gli operatori

1. Chiusura
2. Concatenazione (punto)
3. Piu' (+)

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

Equivalenza di FA e espr. regolari

Abbiamo gia' mostrato che DFA, NFA, e ϵ-NFA sono formalismi equivalenti.



Per mostrare che gli FA sono equivalenti alle espressioni regolari, faremo vedere che (tramite costruzioni effettive)

1. Per ogni espressione regolare R esiste un ϵ-NFA A, tale che $L(A) = L(R)$.

Da espressioni regolari a ϵ-NFA

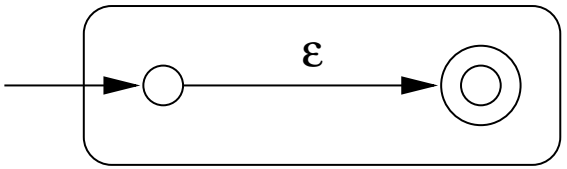
Teorema 3.7: Per ogni espressione regolare R possiamo costruire un ϵ-NFA A, tale che $L(A) = L(R)$.
Prova: Per induzione strutturale su R. Faremo vedere come si costruisce l'automa A per

- espressioni piu' semplici, carattere, stringa vuota, linguaggio vuoto
- come si possono combinare automi per realizzare l'unione, la concatenazione, e la chiusura

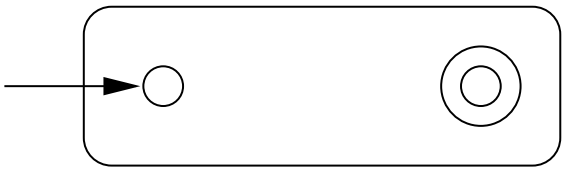
Nota: tutti gli automi che costruiamo sono ϵ-NFA con un unico stato finale (altrimenti la costruzione non funziona)

Da espressioni regolari a ϵ-NFA

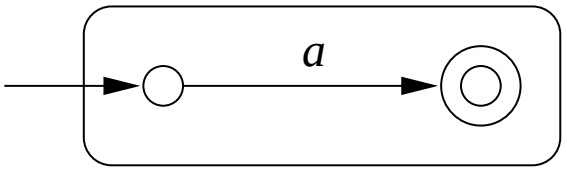
Base: Automa per ϵ, $\emptyset$, e **a**.



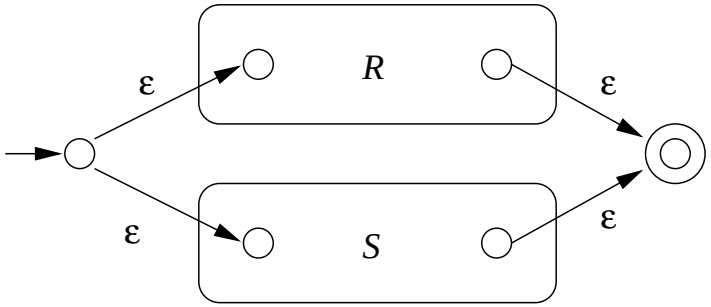
(a)



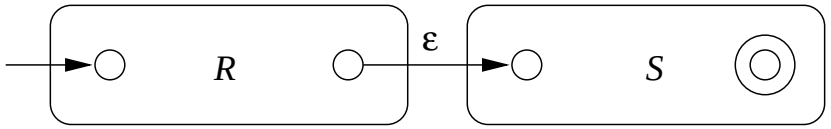
(b)



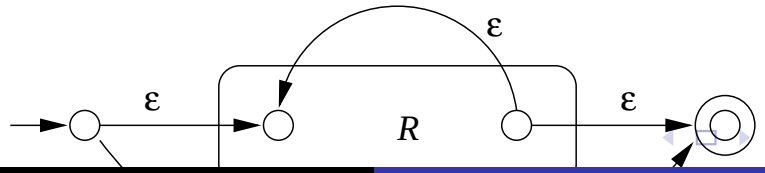
Induzione: Automa per $R + S$, RS, e R^*



(a)

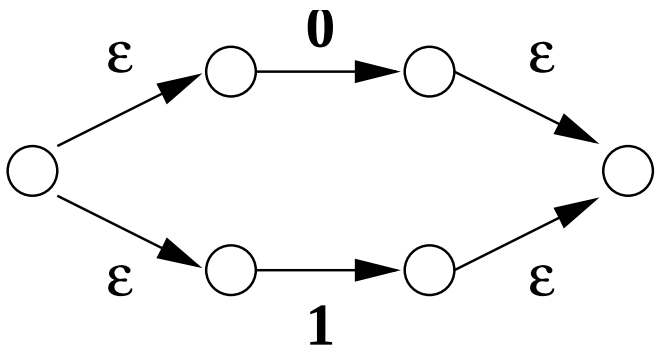


(b)

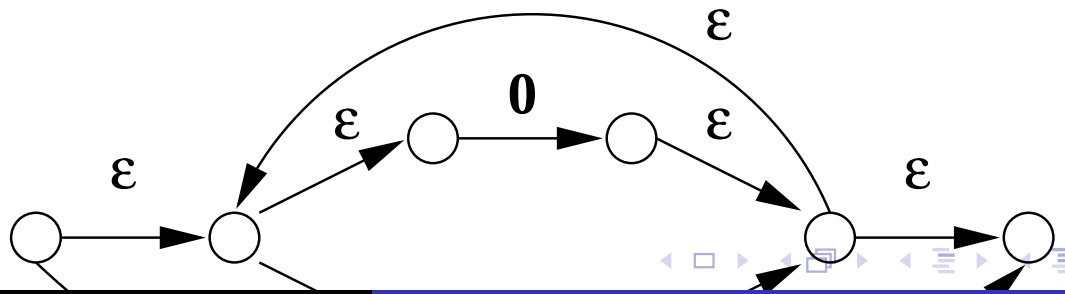


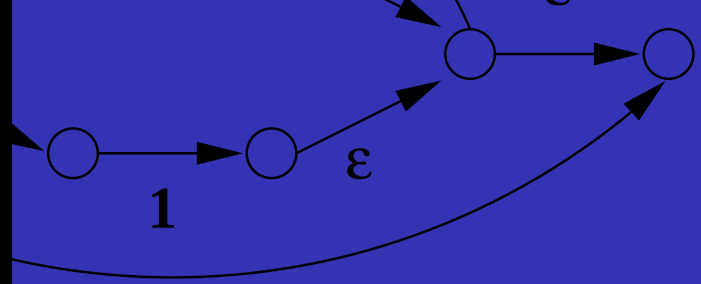
Esempio

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



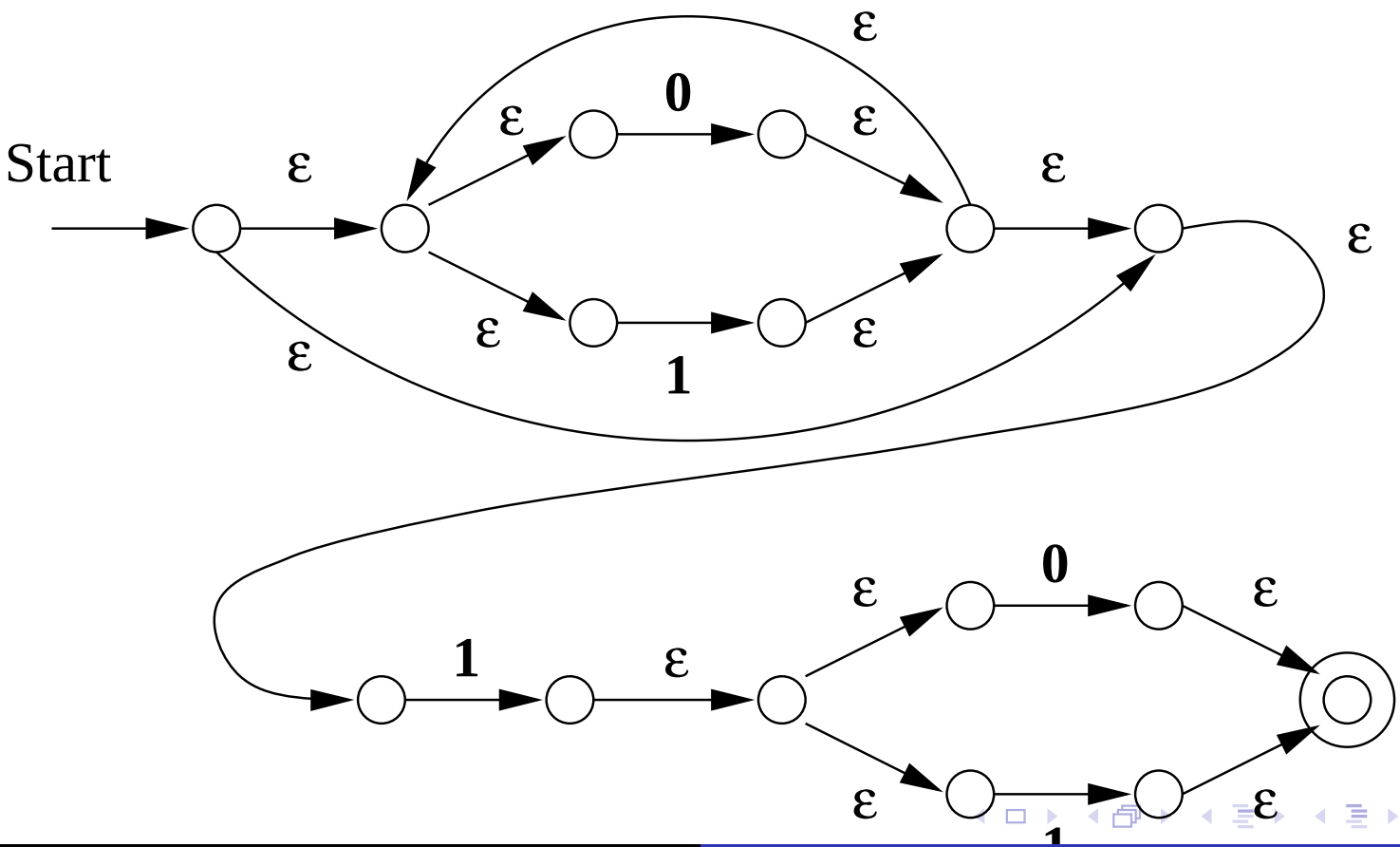
(a)





(b)

Esempio



Da automi ad espressioni regolari

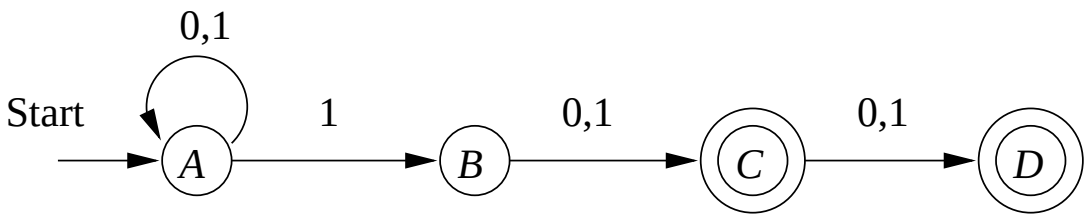
- la derivazione di una espressione regolare R da un automa e' parecchio complicata
- esistono vari algoritmi per derivare una espressione regolare che descrive le stringhe che etichettano i cammini del diagramma di transizione
- faremo vedere una delle tecniche che si applica a DFA (altre tecniche piu' generali si possono trovare in Hopcroft, Motwani, Ullman, capitolo 3)

Automi che hanno come etichette epressioni regolari

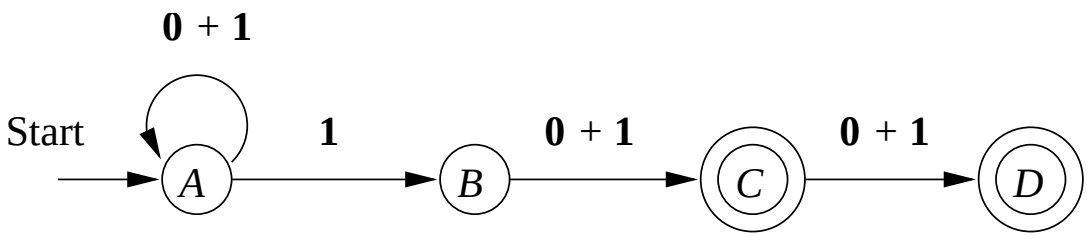
- per applicare l'algoritmo si usa una generalizzazione degli automi a stati finiti, in cui gli archi sono etichettati da espressioni regolari, invece che da simboli dell'alfabeto
- Intuitivamente, il linguaggio definito da un automa di questo tipo e' l'unione, su tutti i cammini dallo stato iniziale ad uno stato finale, dei linguaggi ottenuti concatenando i linguaggi delle espressioni regolari lungo il cammino
- Notiamo che la definizione e' una generalizzazione della definizione standard: ogni DFA e' un caso particolare di questi automi

Esempio

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



Lo possiamo vedere come un automa con espressioni regolari come etichette

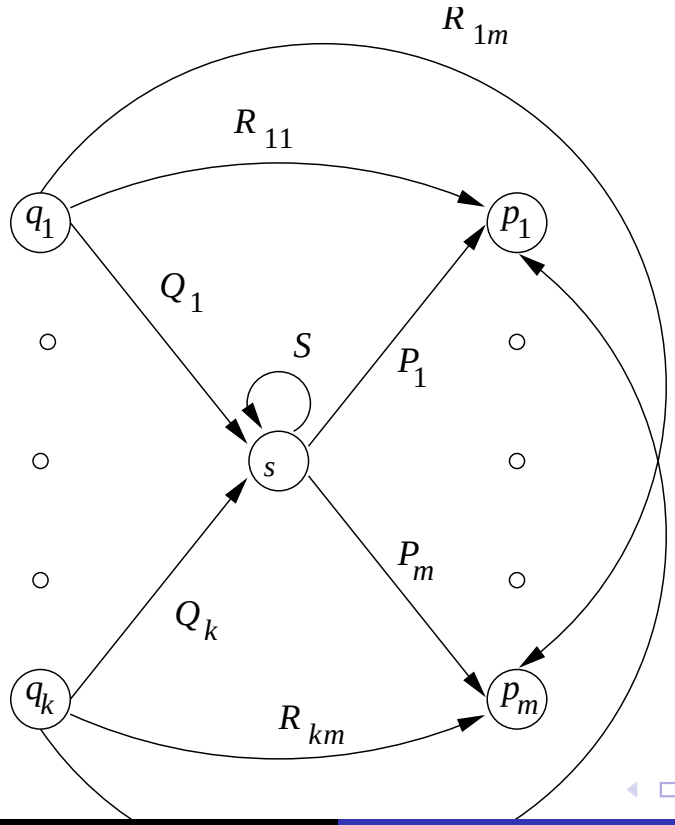


Come si usano questi automi?

- La costruzione che mostreremo parte dal DFA (ovvero dall'automa corrispondente con espressioni reagolari come etichette)
- E' basata su una **tecnica di riduzione**, che dato un automa con etichette costruisce un altro automa con etichette
- **Idea:** ad ogni passo viene eliminato uno stato s, le espressioni regolari degli archi rimanenti vengono modificate in modo da tenere traccia dei cammini che passavano per s e che sono stati cancellati

La tecnica di eliminazione di stati

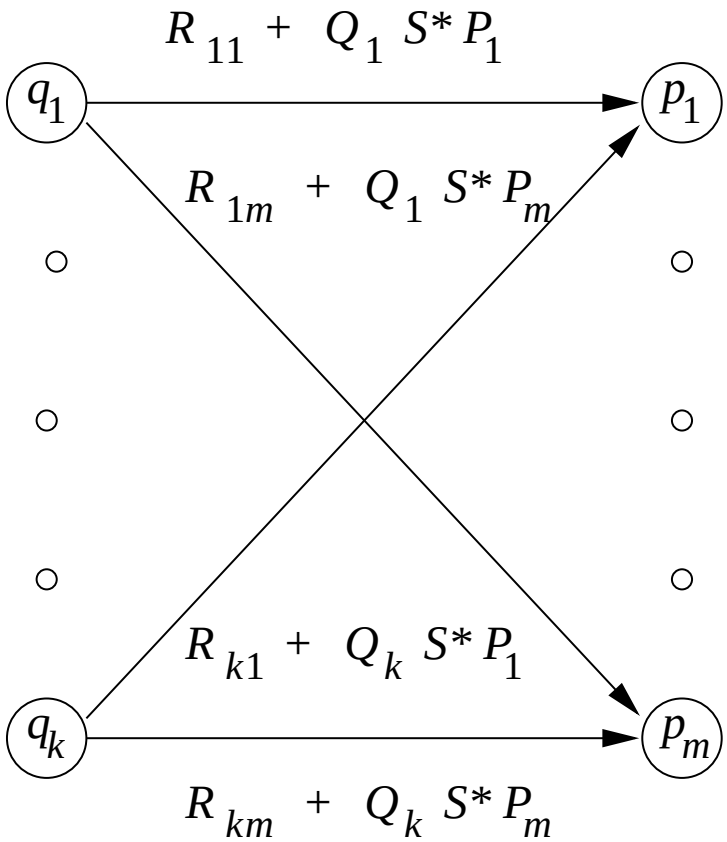
La seguente figura mostra uno stato s che sta per essere eliminato



Notare che

- $q_1, \ldots, q_k$ sono *predecessori* di s, mentre $p_1, \ldots, p_m$ sono *successori* di s;
- c'e' un ciclo su s con etichetta S
- per tutti i valori di i, j esiste un arco tra q_i e p_j con etichetta $R_{i,j}$
- Nota: la situazione e' generale nel senso che tutti questi archi potrebbero non esistere nell'automa (in tal caso l'espressione regolare corrispondente sara' $\emptyset$)

L'automa ottenuto dopo avere eliminato lo stato s.



Notare che

Quando viene eliminato lo stato s vengono modificate le espressioni regolari di tutte le coppie predecessore q_i e successore p_j. In particolare,

- viene considerata l'unione di $R_{i,j}$ con una espressione regolare che denota le stringhe che potrebbero etichettare cammini tra q_i e p_j, che sono stati cancellati
- questi cammini partivano da q_i, facevano una mossa in s, un tot di cicli in S, e poi una mossa in p_j

Nell'automa ottenuto il linguaggio definito, considerando i cammini tra q_i e p_j, e' equivalente a quello definito dai cammini tra q_i e p_j nell'automa di partenza

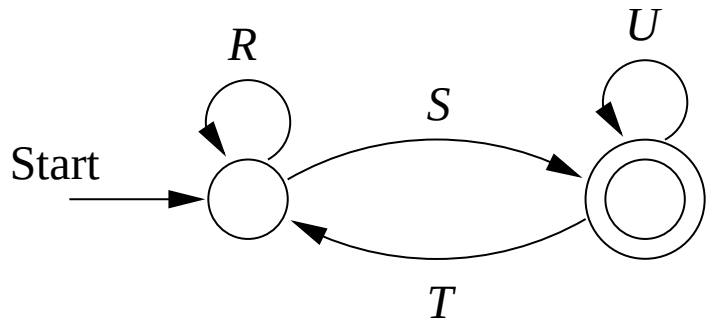
Come si applica l'algoritmo di eliminazione degli stati?

- si parte con l'automa DFA, opportunamente trasformato con espressioni regolari, chiamiamolo A
- per ogni *stato finale* q, si applica la tecnica di riduzione, fino ad eliminare tutti gli stati eccetto q_0 e q
- chiamiamo A_q l'automa ottenuto per lo stato finale q

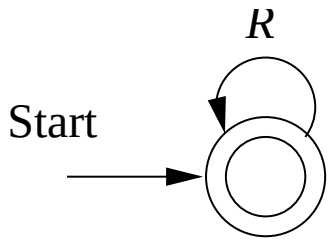
Derivare l'espressione regolare

- per ogni A_q ridotto facciamo vedere come derivare una corrispondente espressione regolare E_q
- intuitivamente, E_q denota le stringhe di $L(A)$ che etichettano i cammini tra lo stato iniziale q_0 e lo stato accettante q

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



e che corrisponde all'espressione regolare $E_q = (R + SU^*T)^*SU^*$
o con A_q della forma



che corrisponde all'espressione regolare $E_q = R^*$

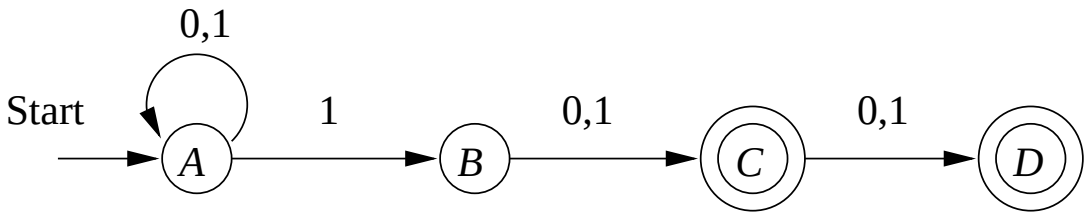
In conclusione

- E' chiaro che per ogni stato finale q, E_q denota le stringhe di $L(A)$ che etichettano i cammini tra lo stato iniziale q_0 e lo stato accettante q
- Quindi l'espressione finale che denota $L(A)$ e'

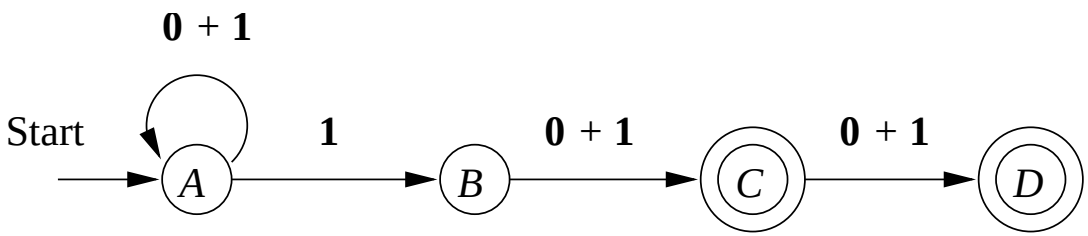
$$\sum_{q \in F} E_q$$

Esempio

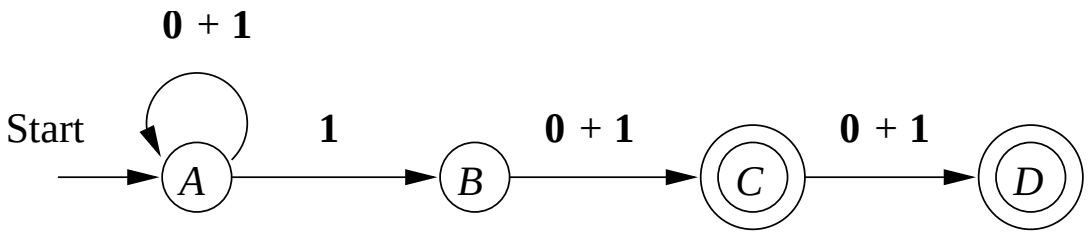
$\mathcal{A}$, dove $L(\mathcal{A}) = \{W : w = x1b,\ \text{o}\ w = x1bc,\ \ x \in \{0,1\}^*, \{b,c\} \subseteq \{0,1\}\}$



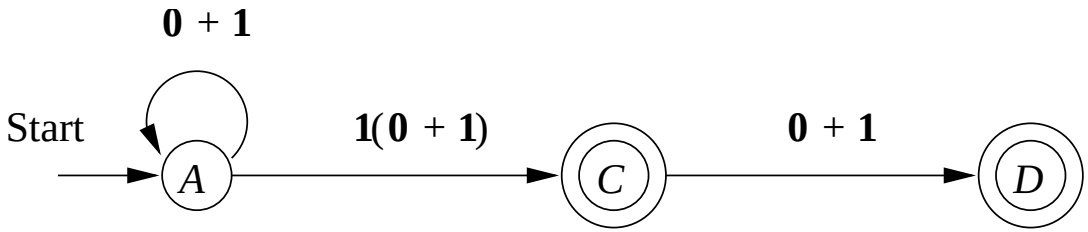
La trasformiamo in un automa con espressioni regolari come etichette



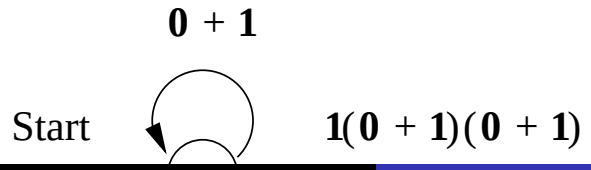
Esempio



Eliminiamo lo stato B

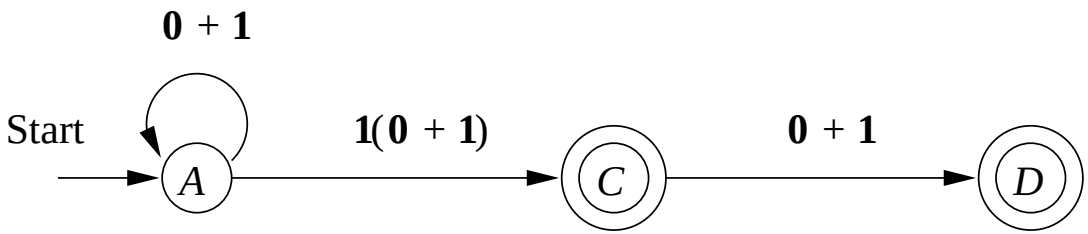


Poi eliminiamo lo stato C e otteniamo $\mathcal{A}_D$

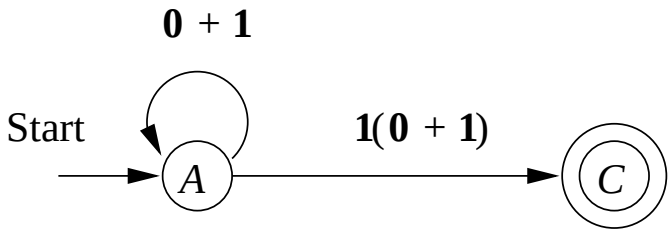


Esempio

Da



possiamo eliminare D e ottenere $\mathcal{A}_C$



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

- L'espressione finale e' la somma delle due espressioni regolari

Leggi algebriche per i linguaggi

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

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

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

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

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

Prova:

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

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

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

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