

Automati e Linguaggi Formali

Espressioni regolari

A.A. 2014-2015
Enrico Mezzetti
emezzett@math.unipd.it



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Espressioni regolari

- Un FA (NFA o DFA) e' un metodo per costruire una macchina che riconosce linguaggi regolari
- Una *espressione regolare* e' un modo dichiarativo per descrivere un linguaggio regolare
 - Descrizione *algebraica* vs. *comportamentale*
- Esempio: $01^* + 10^*$
 - $L(01^* + 10^*) = \{0111111111111 \vee 1000000000 \vee \dots\}$
- Le espressioni regolari sono usate, ad esempio, in
 - Comandi UNIX (grep)
 - Strumenti per l'analisi lessicale presenti nelle distro Linux (E.g., `lex` - Lexical analyzer generator e `flex` - Fast Lex)



Notazione delle espressioni regolari: Operatori

■ Espressioni comprendono **operatori** e **simboli**

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

Es.: $L = \{001, 10, 111\}$ e $M = \{\epsilon, 001\}$

$L \cup M = \{\epsilon, 001, 10, 111\}$

Concatenazione $L.M = \{w : w = xy, x \in L, y \in M\}$

Es.: $L = \{001, 10, 111\}$ e $M = \{\epsilon, 001\}$

$L.M = \{001, 10, 111, 001001, 10001, 111001\}$

Potenza $L^0 = \{\epsilon\}$, $L^1 = L$, $L^{k+1} = L.L^k$

Es.: $L = \{001, 10\}$

$L^0 = \{\epsilon\}$, $L^1 = \{001, 10\}$, $L^2 = \{00110, 10001, 1010, 001001\}$

Chiusura di Kleene $L^* = \bigcup_{i=0}^{\infty} L^i$

Es.: $L = \{0, 11\}$

$L^* = \{\text{stringhe di 0 e 1 tali che gli 1 compaiono a coppie}\}$

👁 L^* include tutte le stringhe che possono essere formate concatenando elementi in L

Notazione delle espressioni regolari: Costruzione

■ Definizione induttiva delle espressioni

Base

- Costanti: ϵ e \emptyset sono espressioni regolari
 $L(\epsilon) = \{\epsilon\}$ e $L(\emptyset) = \emptyset$.
- Simboli: Se $a \in \Sigma$, allora a e' un'espressione regolare.
 $L(a) = \{a\}$ \uparrow (in grassetto)
- Variabili: L rappresenta un linguaggio arbitrario

Induzione

- Unione: se E e F sono espressioni regolari \Rightarrow
 $E + F$ e' un'espressione regolare per $L(E + F) = L(E) \cup L(F)$
- Concatenazione: se E e F sono espressioni regolari \Rightarrow
 $E.F$ e' un'espressione regolare per $L(E.F) = L(E).L(F)$
- Chiusura: se E e' un'espressione regolare \Rightarrow
 E^* e' un'espressione regolare per $L(E^*) = (L(E))^*$
- Parentesizzazione: se E e' un'espressione regolare \Rightarrow
 (E) e' un'espressione regolare per $L((E)) = L(E)$

Esempio costruzione espressione regolare

- Espressione regolare per le stringhe di 0 e 1 alternati
 $L = \{w \in \{0,1\}^* : 0 \text{ e } 1 \text{ alternati in } w\}$
- Costruzione
 - **0** e **1** \in regexp (base - simboli)
 - **01** \in regexp per concatenazione
 - **(01)*** \in regexp per chiusura
 - **(01)*** + **(10)*** \in regexp per unione
 - **(01)*** + **(10)*** + **0(10)*** + **1(01)*** \in regexp per unione

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

equivalentemente:

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

👁 $L = (\textit{epsilon} + 1) = L(\epsilon) + L(1) = \{\epsilon\} \cup \{1\} = \{\epsilon, 1\}$

Ordine di precedenza per gli operatori

- Algebre definiscono precedenza tra operatori
 - $x \cdot y + z \rightarrow (x \cdot y) + z$ e non $x \cdot (y + z)$
- Precedenza in espressioni regolari
 - 1 Chiusura ($*$)
 - 2 Concatenazione (punto)
 - 3 Unione ($+$)
 - Esempio: $01^* + 1$ e' raggruppato in $(0(1)^*) + 1$
- Derogabile con uso di parentesi
 - Esempio: $(01)^*$

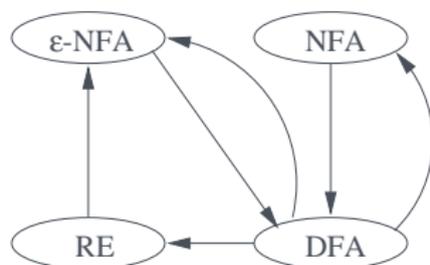


Esempi di regexp



Equivalenza di FA e regexp

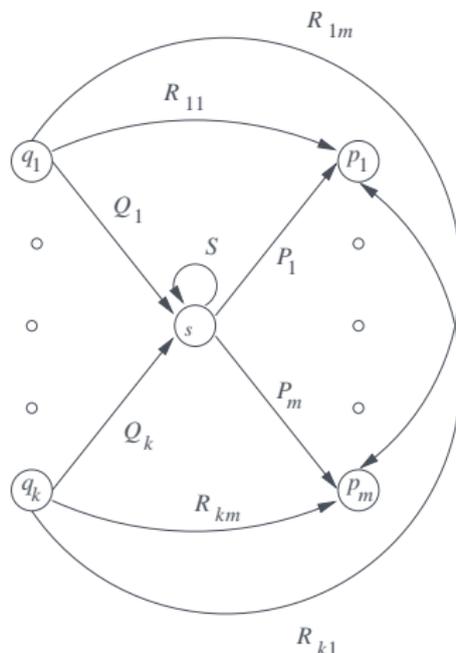
- DFA, NFA e ϵ -NFA sono equivalenti
 - Accettano la stessa classe di linguaggi (l. regolari)



- Per sostenere equivalenza tra regexp FA dimostriamo che
 - 1 Per ogni DFA A possiamo trovare (costruire) un'espressione regolare R , tale che $L(R) = L(A)$.
 - 2 Per ogni espressione regolare R esiste un ϵ -NFA A , tale che $L(A) = L(R)$.

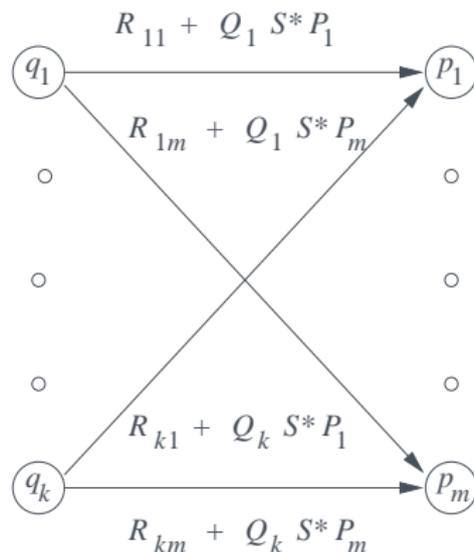
Da DFA a RE: tecnica per eliminazione di stati

- 1 Etichettiamo gli archi con espressioni regolari di simboli



La tecnica di eliminazione di stati - 2

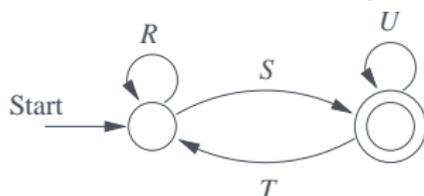
2 Ora eliminiamo lo stato s



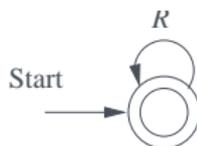
3 $\forall \star q$ eliminiamo da FA originale tutti gli stati eccetto q_0 e $\star q$.

La tecnica di eliminazione di stati - 3

- Per ogni $q \in F$ saremo rimasti con A_q in due forme possibili
 - Corrispondente all'espressione regolare $E_q = (R + SU^*T)^*SU^*$



- Corrispondente all'espressione regolare $E_q = R^*$



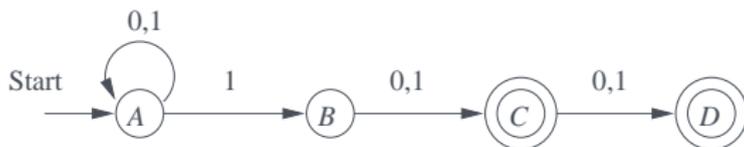
- L'espressione finale e'

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

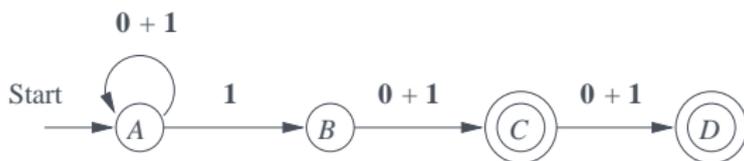
Esempio riduzione

- Ridurre l'automa NFA \mathcal{A} che accetta il linguaggio delle stringhe con 1 come penultimo o terzultimo elemento

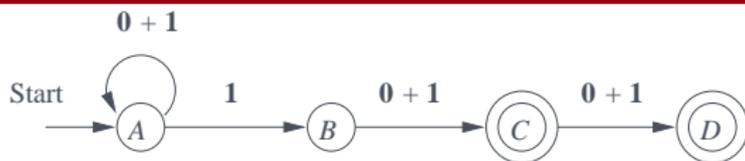
$$L(\mathcal{A}) = \{w : \exists \alpha = \xi\infty[, \circ \exists \beta = \xi\infty], \xi \in \{0,1\}^*, \{[,]\} \subseteq \{0,1\}\}$$



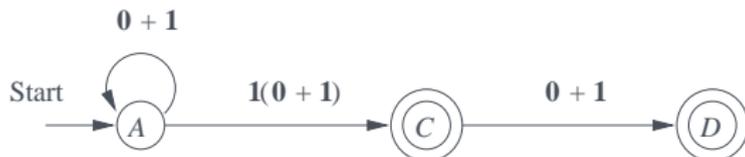
- La trasformiamo in un automa con espressioni regolari come etichette



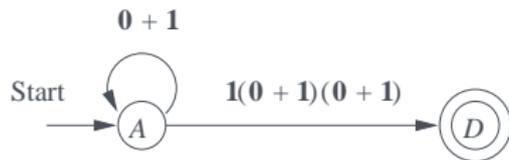
Esempio riduzione - 2



2 Eliminiamo lo stato B (non terminale)

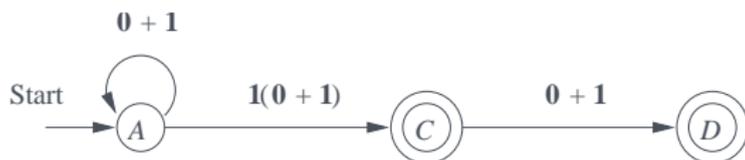


3 Poi eliminiamo lo stato C e otteniamo \mathcal{A}_D

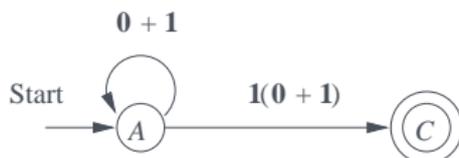


con espressione regolare $(0 + 1)^*1(0 + 1)(0 + 1)$

Esempio riduzione - 3



- 4 Alternativamente possiamo eliminare D e ottenere \mathcal{A}_C



con espressione regolare $(0 + 1)^*1(0 + 1)$

- 5 L'espressione finale e' la *somma* delle due espressioni precedenti

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

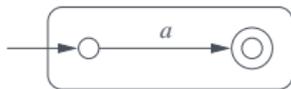
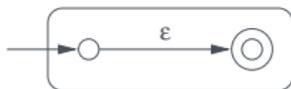
Da espressioni regolari a ϵ -NFA

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

- 1 Un solo stato accettante
- 2 Nessun arco entrante nello stato iniziale
- 3 Nessun arco uscente dallo stato finale

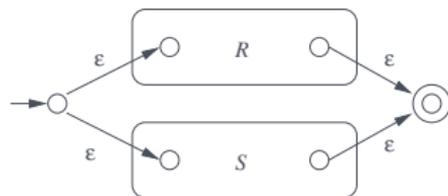
Prova Per induzione strutturale su R

Base ϵ -NFA per ϵ , \emptyset , e a



Da espressioni regolari a ϵ -NFA - 2

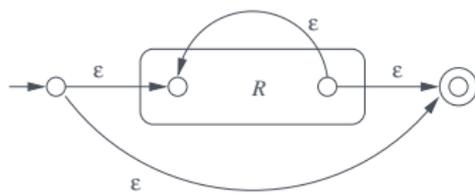
Induzione ϵ -NFA per $R + S$, RS , e R^* (caso parentesi non rilevante)



(a)



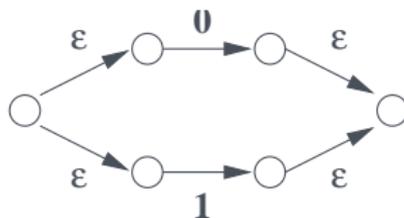
(b)



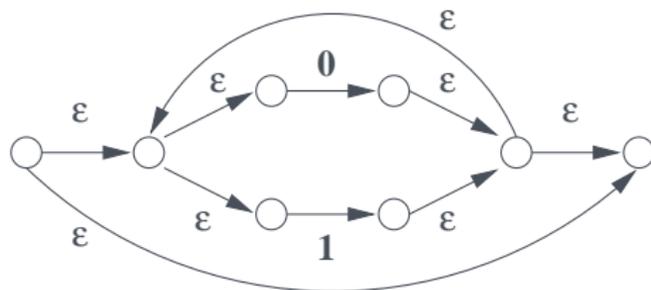
(c)

Esempio trasformazione - 1

■ $R = (0 + 1)^*1(0 + 1)$

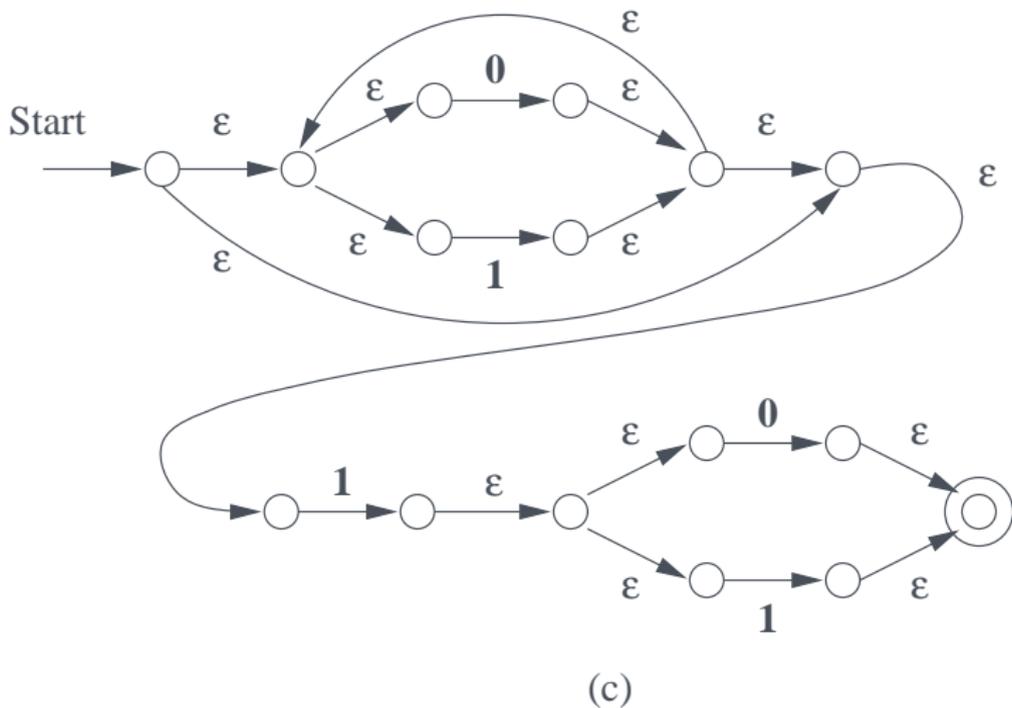


(a)



(b)

Esempio trasformazione - 2 (b)



Esempi



Proprietà algebriche per le espressioni regolari

- regexp presentano proprietà algebriche
 - Similitudine con espressioni aritmetiche

- **Associatività** e **commutatività**

Commutatività L'unione è *commutativa*

$$L \cup M = M \cup L$$

Associatività L'unione è *associativa*

$$(L \cup M) \cup N = L \cup (M \cup N)$$

La concatenazione è *associativa*

$$(LM)N = L(MN)$$

Nota: concatenazione non è commutativa cioè $\exists L, M$ tali che $LM \neq ML$.



■ Identità' e annichilatori

Identità' \emptyset e' l'*identità'* per l'unione
$$\emptyset \cup L = L \cup \emptyset = L$$

$\{\epsilon\}$ e' l'*identità'* *sinistra* e *destra* per la concatenazione
$$\{\epsilon\}L = L\{\epsilon\} = L$$

Annichilatore \emptyset e' l'*annichilatore* *sinistro* e *destro* per la concatenazione
$$\emptyset L = L\emptyset = \emptyset$$

■ Distributività' Idempotenza

Distributività' La concatenazione e' *distributiva a sinistra* sull'unione
$$L(M \cup N) = LM \cup LN$$

La concatenazione e' *distributiva a destra* sull'unione
$$(M \cup N)L = ML \cup NL$$

Idempotenza L'unione e' *idempotente*
$$L \cup L = L$$

■ Proprietà' relative alla chiusura

Idempotenza La chiusura e' idempotente

$$(L^*)^* = L^*$$

Valori notevoli $\emptyset^* = \{\epsilon\}$

$$\{\epsilon\}^* = \{\epsilon\}$$

$$L^+ = LL^* = L^*L$$

$$L^* = L^+ \cup \{\epsilon\}$$

Dimostrazione

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

$$\begin{aligned}w \in (L^*)^* &\iff w \in \bigcup_{i=0}^{\infty} \left(\bigcup_{j=0}^{\infty} L^j \right)^i \\ &\iff \exists k, m \in \mathbb{N} : w \in (L^m)^k \\ &\iff \exists p \in \mathbb{N} : w \in L^p \\ &\iff w \in \bigcup_{i=0}^{\infty} L^i \\ &\iff w \in L^*\end{aligned}$$

Esempi

