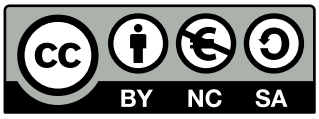
Riconoscitori a stati finiti
Esercizi

Prof. Enrico Denti
a cura di Andrea Giovine

novembre, 2021



Indice

0.1 Esercizio 1

$$L = ab^* + c$$

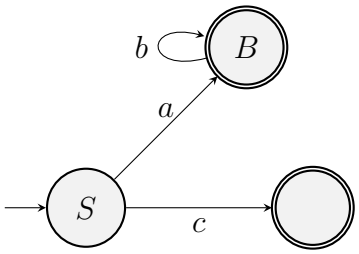
Osserviamo che L è l'unione di due linguaggi L_1 e L_2: quindi anche la grammatica che lo genera avrà regole per L_1 e regole per L_2:

$$\begin{array}{rl} L = L_1 \cup L_2 & \\ L_1 = ab^* \longrightarrow & \left\{ \begin{array}{c} S \to a \mid aB \\ B \to b \mid bB \end{array} \right. \\ L_2 = c \longrightarrow & \mathrm{S} \to \mathrm{c} \end{array}$$

da cui:

$$\left\{ \begin{array}{l} S \to a \mid c \mid aB \\ B \to b \mid bB \end{array} \right. \tag{1}$$

L'automa corrispondente risulta:

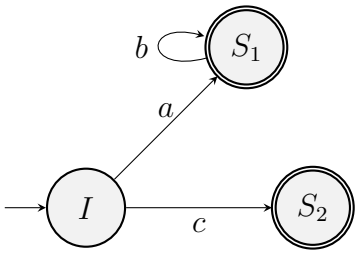


Ciò è coerente con la regexp iniziale, da cui avrebbe potuto essere dedotto direttamente. Infatti nelle regexp iniziale:

- L è l'unione di due linguaggi
- nel secondo percorso, è già finita: c'è direttamente uno stato finale (non importa il nome)
- nel primo occorre produrre $b^* \Longrightarrow$ *autoanello*

Per concludere, rileggiamo tale automa bottom-up, al fine di ottenere la grammatica regolare *sinistra* equivalente:

1. rinominare gli stati, per evitare confusione
2. estrarre le regole per il mapping bottom-up



Poiché ci sono due stati finali, S_1 e S_2, occorre trattarli separatamente:

- $S_1 \to S_1 b \mid a$
- $S_2 \to c$

Per comporli si scrive concettualmente $S \to S_1 \mid S_2$, quindi per sostituzione:

$$S \to S_1 b \mid a \mid c \qquad \begin{cases} S \to Ab \mid a \mid c \\ A \to Ab \mid a \end{cases}$$
$$S_1 \to S_1 b \mid a$$

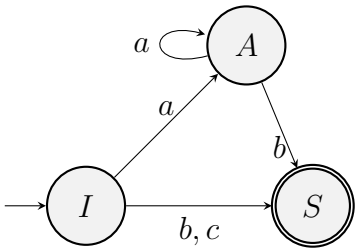
0.2 Esercizio 2

$$L = a^*b + c$$

Simile a prima, ma leggermente più complesso da gestire per la presenza dell' $*$ iniziale, che "nasconde" le possibili iniziali delle frasi.
Essendo la ripetizione a sinistra, è naturale esprimerla con una grammatica regolare sinistra

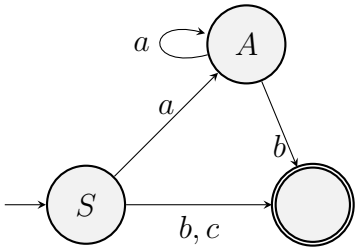
- $S \to Ab \mid b \mid c$
- $A \to Aa \mid a$

a cui corrisponde il seguente automa:



La seconda regola si potrebbe anche scrivere con ricorsione destra, ma in tal caso la grammatica sarebbe di *tipo 2*, per la contemporanea presenza di regole sinistre e destre. Benché possibile, ciò sarebbe inappropriato, dato che il linguaggio, provenendo da una regexp, è certamente di tipo 3.

Come utile esercizio possiamo rileggerlo top-down. A tal fine, come primo passo è opportuno procedere a un renaming:



- $S \to b \mid c \mid aA$
- $A \to aA \mid b$

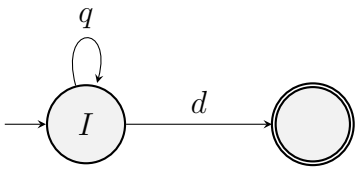
La si confronti con quella, regolare sinistra, iniziale.

0.3 Esercizio 3

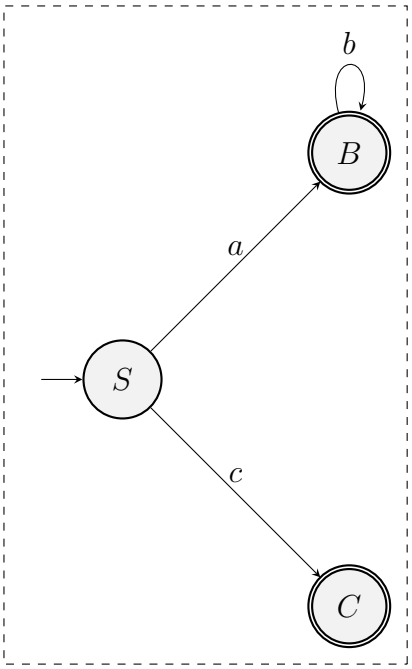
$$L = (ab^* + c)^* d$$

Regexp più complessa, con una sotto espressione (quella fra parentesi) che si può ripetere: si noti infatti l' $*$ che segue il gruppo parentesizzato.

Approccio: per un momento, immaginiamo che al posto della sotto espressione ci sia un terminale q, ossia che la regexp abbia la forma $L = q^* d$. In tal caso, il mapping sull'automa sarebbe immediato:



In realtà però al posto di q c'è un'intera sotto-espressione, che di per sé sarebbe schematizzabile con l'automa:

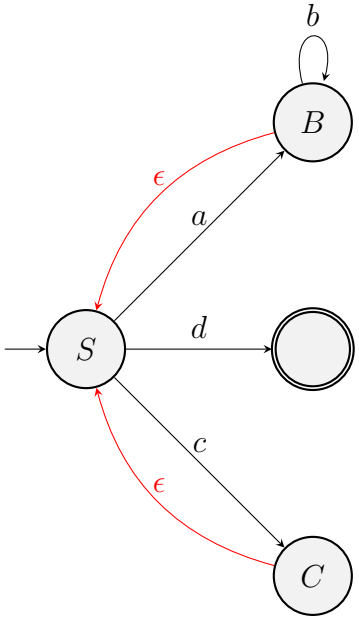


Poiché la sotto-espressione $(ab^* + c)^*$ è solo una sottoparte di un'espressione più ampia, anche questo automa dovrà essere parte dell'automa più ampio schematizzato sopra: di fatto, *l'intero sotto-automa a lato equivale allo stato* I dell'automa iniziale.

Quindi, per avere l'automa completo dobbiamo idealmente (ma anche praticamente) "sostituire" ad I il sotto-automa: per farlo dobbiamo "richiudere" il sotto-automa in un loop, che equivalga all'autoanello presente su I.

Per simulare quell'autoanello occorre introdurre un artificio, la ϵ-*mossa*: una mossa "spontanea", che l'automa può compiere senza consumare input. Essa rappresenta il "goto" che riporta il loop all'inizio, catturando la semantica dell' $*$.

Concretamente, dato che il sotto-automa termina nei due stati B e C, che quindi corrispondono ad aver raggiunto la "chiusa parentesi" della sotto-espressione, poter "tornare spontaneamente all'inizio" (autoanello su I) significa tornare da B (o C) allo stato iniziale del sotto-automa: si inserisce a tal fine una ϵ-*mossa* da ciascuno di tali stati ad S.

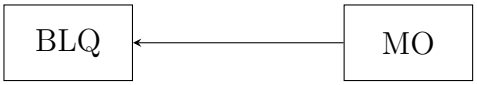


Ovviamente, lo stato finale è ora solo quello dell'automa complessivo, che rappresenta la regexp data nella sua totalità.

Tuttavia, la ϵ-*mossa*, in quanto mossa spontanea a cui non corrisponde alcun input, è di norma indesiderabile: costituisce fondamentalmente un artificio per comporre i pezzi.

Per eliminare le ϵ-mosse garantendo *invarianza di effetti* occorre, per ogni epsilon-arco rimosso, inserire archi che producano lo stesso risultato.

Per capire come e dove può essere utile un'analogia, quella dell'aeroporto.

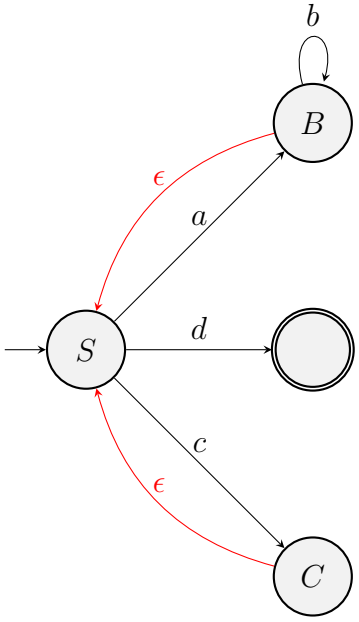


Modena (MO) non ha aeroporto: tuttavia, se ai modenesi viene offre un transfer bus gratuito verso l'aeroporto di Bologna (BLQ), essi potranno andare negli stessi luoghi dei bolognesi, *come se* avessero un loro aeroporto.

Il bus gratuito è l'analogo della ϵ-mossa: per toglierlo senza causare danni ai modenesi, occorre che Modena abbia un aeroporto con gli stessi voli e destinazioni di Bologna.

Analogamente, per togliere un ϵ-arco occorre aggiungere, in compensazione, archi che portino negli stessi stati che si sarebbero potuti raggiungere con l'ϵ-arco soppresso.

Nell'automa di poco fa, gli ϵ-archi (in rosso) erano due:



- dallo stato B (analogia: Modena) verso S (analogia: BLQ)
- dallo stato C (analogia: un'altra Modena) verso S (analogia: BLQ)

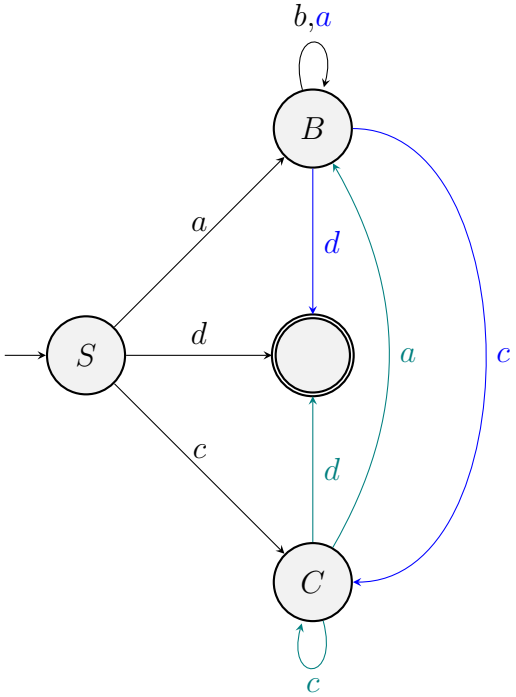
Poiché da S si possono raggiungere:

- lo stato B, con ingresso a
- lo stato C, con ingresso c
- lo stato finale, con ingresso d

occorre che la soppressione dei due ϵ-archi (in rosso) sia accompagnata dal simultaneo inserimento di archi:

- dallo stato B, verso tutti quegli stati a cui si giungeva da S (B,C,F)
- dallo stato C, verso tutti quegli stati a cui si giungeva da S (B,C,F)

Il risultato è mostrato in figura:

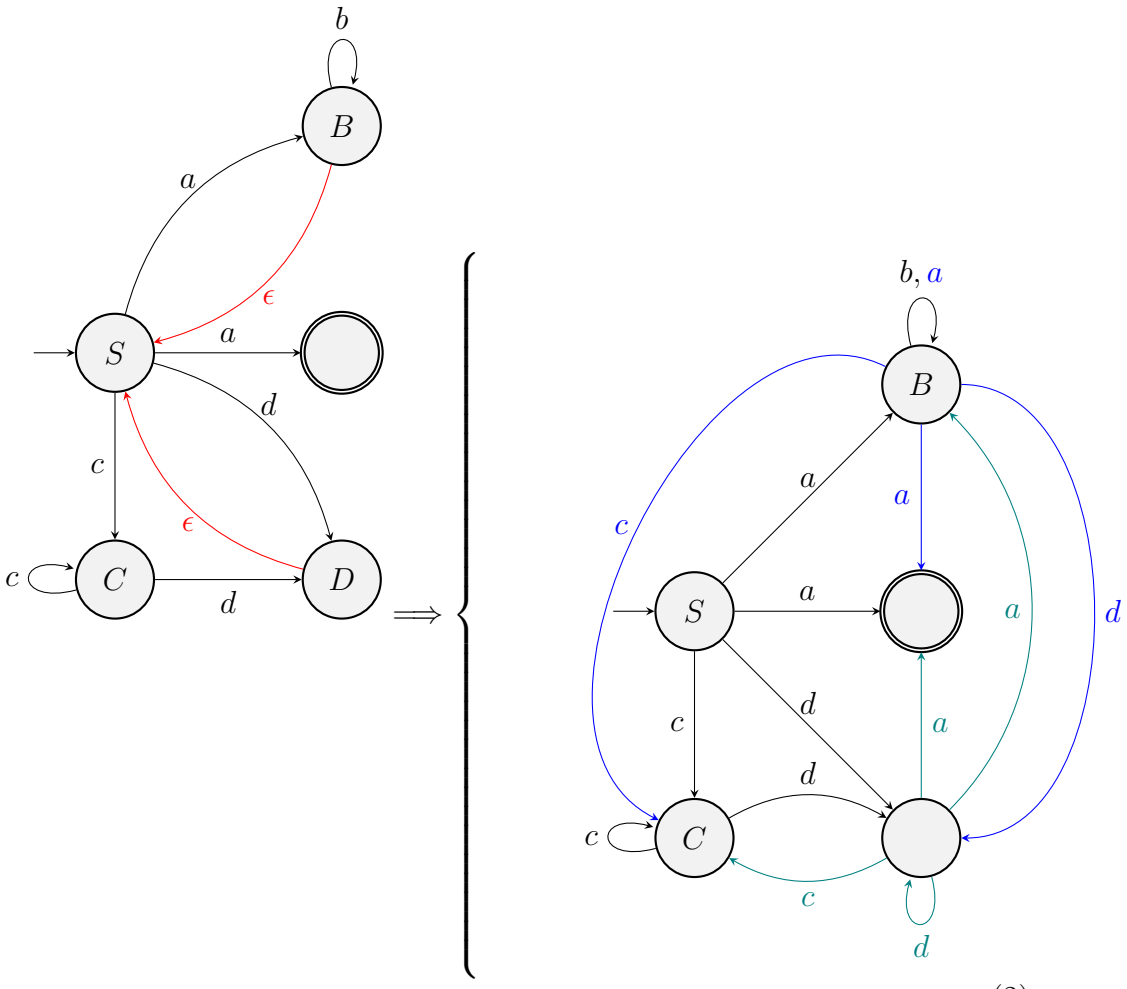


- blu: archi inseriti in compensazione dell'epsilon-arco di B
- verde: archi inseriti in compensazione dell'epsilon-arco di C

L'automa così ottenuto può essere, se necessario, minimizzato.

0.4 Esercizio 4

$$L = (ab^* + c^*d)^*a$$



(2)

0.5 Esercizio 5

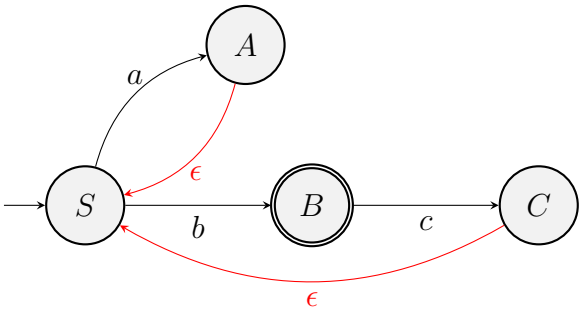
$$L = (a + bc)^* b$$

Seguendo la stessa metodologia degli esercizi precedenti, è immediato disegnare l'automa con le epsilon-mosse.

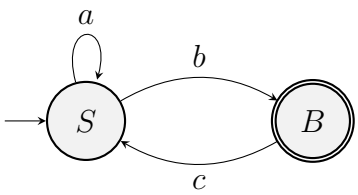
Per eliminarle, sono possibili due approcci:

- il metodo "classico", che agisce meccanicamente, dando luogo in prima battuta a un automa non minimo
- un metodo "veloce", che si basa sull'osservazione diretta di percorsi "evidenti" in questo caso specifico.

Metodo veloce

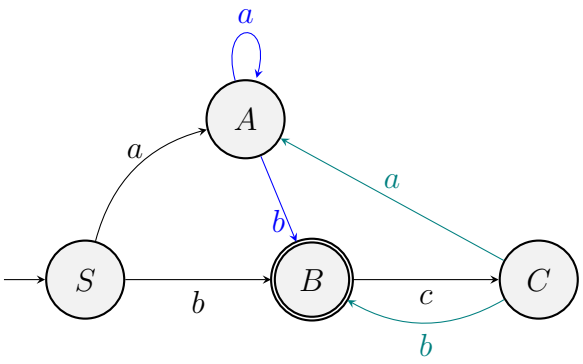


si osserva che A e C sono puri stati di transito per la sequenza "a ϵ" e "c ϵ", dunque si possono facilmente eliminare "fondendo" i due archi in uno solo



automa risultante (minimo)

Metodo classico

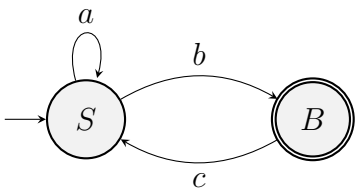


automa non minimo

Tabella di minimizzazione:

S	NO	<u>SI</u>	
A	NO	SI	<u>SI</u>
B	SI	NO	NO
	B	A	C

Stati equivalenti: $S \equiv A,\ A \equiv C$



automa minimo