ER - Slide

Gabriel Rovesti

1 Esercizio 1

Consegna: Trasformare l'espressione regolare

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

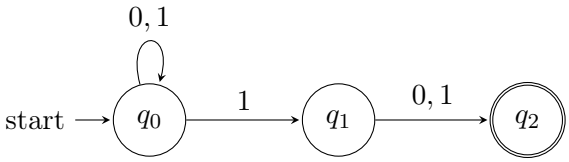
in un NFA equivalente.

Soluzione

L'espressione regolare $(0+1)^*1(0+1)$ descrive le stringhe (su $\{0,1\}$) che contengono almeno un simbolo 1 non necessariamente all'ultimo posto, e *seguite* da *almeno un* simbolo (0 o 1).

Un NFA equivalente si può costruire come segue:

- Stato iniziale q_0. Da q_0 c'è un loop su $\{0,1\}$ per realizzare $(0+1)^*$.
- Poi, per esprimere il fattore "1", da q_0 si va in un nuovo stato q_1 consumando 1.
- Infine, serve almeno un simbolo $(0+1)$ dopo il 1. Quindi da q_1 si va in uno stato finale q_2 leggendo un solo $\{0,1\}$.



Stato finale: q_2. Osserviamo che in molte costruzioni si preferisce una "chiusura" ε-transizione per il pezzo $(0+1)^*$, ma non è strettamente necessario. L'idea illustrata è sufficiente a realizzare la stessa RE.

2 Esercizio 2

Consegna: Scrivere *un'espressione regolare* sull'alfabeto $\{a, b, c\}$ che descriva:

- tutte le stringhe che *iniziano con* a e sono composte solo da a o b;
- la stringa singola c.

Soluzione

Le stringhe che *iniziano con* a e poi usano solo a o b si descrivono con:

$$a\,(a+b)^*$$

Inoltre vogliamo includere la singola stringa c. L'unione dei due insiemi produce:

$$(\,c\,) \;+\; a\,(a+b)^*$$

o, in notazione alternativa:

$$c \mid a(a|b)^*$$

che è appunto l'espressione regolare richiesta.

3 Esercizio 3

Consegna: Trasformare in NFA l'espressione regolare dell'esercizio precedente:

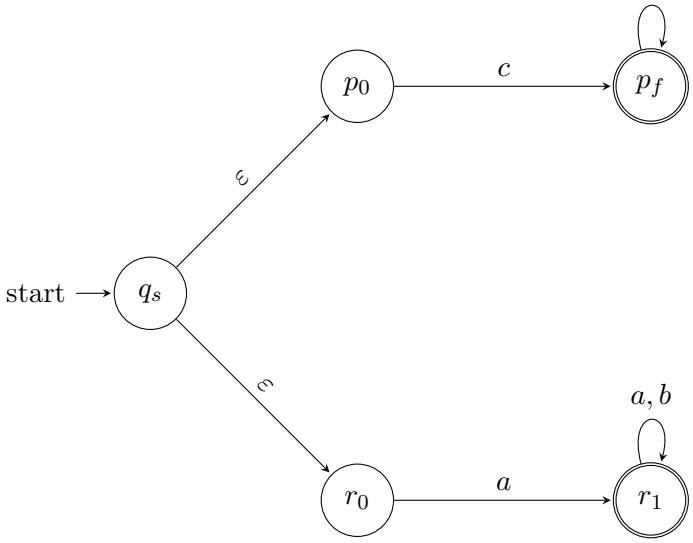
$$c \mid a(a+b)^*.$$

Soluzione

Basta costruire due piccoli automi, uno per c (accetta solo la stringa "c") e uno per $a(a+b)^*$. Poi si uniscono con uno stato iniziale e ε-transizioni verso i due "rami".

- NFA_1: Stato p_0 iniziale, lettura di c per andare a uno stato finale p_f. Nient'altro.
- NFA_2: Stato r_0 iniziale, lettura di a per passare a r_1. Poi in r_1 un loop su $\{a, b\}$. Stato finale r_1 stesso (in quanto $(a+b)^*$ include anche ε come stringa *successiva* all'aver letto a).

NFA globale:



Gli stati finali sono p_f e r_1. Le ε-transizioni dal super-iniziale q_s ai due stati iniziali dei due sotto-NFA implementano l'unione $(\,c\,) + a(a+b)^*$.

4 Esercizio 4

Consegna: Scrivere un'espressione regolare per tutte le stringhe binarie *che cominciano e finiscono con 1.*

Soluzione

Se una stringa comincia con 1 e finisce con 1, allora la forma generale è:

$$1\,(0+1)^*\,1.$$

Dobbiamo leggere almeno 2 simboli "1 all'inizio e 1 alla fine"; in mezzo può esserci una qualsiasi sequenza (anche vuota) di 0 o 1.

5 Esercizio 5

Consegna: Scrivere un'espressione regolare per le stringhe binarie *che contengono almeno tre 1 consecutivi.*

Soluzione

Per avere almeno "111" consecutivi in qualche punto, possiamo scomporre la stringa in "tutto ciò che precede i tre 1" + "i tre 1 consecutivi" + "tutto ciò che segue i tre 1". Formalmente:

$$(0+1)^*\ \ 111\ \ (0+1)^*.$$

È irrilevante quante altre cifre ci sono prima o dopo, purché *almeno* un blocco di tre 1 di fila sia presente.

6 Esercizio 6

Consegna: Scrivere un'espressione regolare per le stringhe binarie *che contengono almeno tre 1 in totale* (non necessariamente consecutivi).

Soluzione

Un classico modo per dire "almeno tre 1 nel corso della stringa" è forzare l'esistenza di tre `1` in posizioni (non fissate) separate da eventuali `0` e `1`. In RE:

$$(0+1)^*\,1\,(0+1)^*\,1\,(0+1)^*\,1\,(0+1)^*.$$

Qui i quattro blocchi $(0+1)^*$ rappresentano la parte prima del primo 1, la parte fra il primo e il secondo 1, ecc.

7 Esercizio 7

Consegna: Scrivere un'espressione regolare per descrivere *date* nel formato `GG/MM/AAAA` (giorno, mese, anno) usando cifre decimali. Ammesso che ci si limiti a una verifica "semplice" senza controllare la validità logica (es. 30/02) oltre al pattern di cifre.

Soluzione

Uno schema tipico:

- `GG`: 01–31, in formato 2 cifre (da `00` a `31`). Ma volendo imporre la regola "giorno 01–31":

$$(0[1-9]) \mid ([12][0-9]) \mid (3[0-1])$$

- `MM`: 01–12

$$(0[1-9]) \mid (1[0-2])$$

- `AAAA`: 4 cifre decimali (da 0000 a 9999, per semplicità)

$$[0-9]\,[0-9]\,[0-9]\,[0-9].$$

Separando con "`/`" si ottiene:

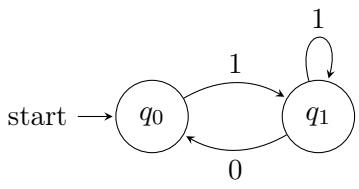
$$\big(0[1-9]|[12][0-9]|3[0-1]\big) \,/\, \big(0[1-9]|1[0-2]\big) \,/\, [0-9]4.$$

È un'espressione regolare tipicamente usata per validare un pattern "DD/MM/YYYY".

8 Esercizio 8

Consegna: Costruire *un'espressione regolare equivalente* ai seguenti automi (rappresentati schematicamente):

8.a)



Sono due stati, con transizioni: $q_0 \xrightarrow{1} q_1$, $q_1 \xrightarrow{1} q_1$, $q_1 \xrightarrow{0} q_0$. Non è indicato chi sia finale: supponendo q_1 *finale*, ad esempio, si può interpretare come la RE

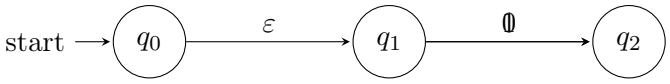
$$1\,(1^*\,0\,1^*)^*,$$

se si parte da q_0 (iniziale), per accettare, occorre almeno un 1 per entrare in q_1 (finale), e poi si può effettuare qualunque numero di cicli ($0 \to q_0 \xrightarrow{1} q_1$) in mezzo a possibili 1-loop su q_1. In forma più estesa: $(1^*)^*$ *non* aggiunge nulla, quindi la scrittura più pulita è

$$1\left((1^*)\,0\,(1^*)\right)^*$$

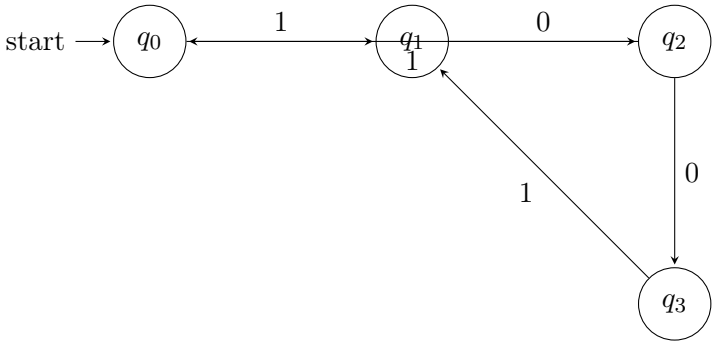
(anche $(1 + 1^*)$ si può unificare in 1^*). Se invece fosse q_0 finale, la RE cambierebbe. Dipende dai dettagli non mostrati.

8.b)



Se q_2 è finale (per esempio), allora la RE corrispondente è "ε-transizione da q_0 a q_1" e da q_1 a q_2 con $\{0, 1\}$. In breve, $\{0, 1\}$. Ma l'ε da q_0 a q_1 consente di non consumare nulla prima di andare a q_1. Quindi la NFA ammette esattamente una lettera 0 o 1 prima di raggiungere q_2. L'insieme delle stringhe è $\{0, 1\}$. RE: $(0 + 1)$.

8.c)



Se q_3 fosse finale, ad esempio, notiamo un giro:

$$q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{0} q_3(\text{finale}),$$

oppure si può passare $q_2 \xrightarrow{1} q_0$ e rifare i loop. Un'eventuale RE (dipende di nuovo da chi è finale) potrebbe essere:

$$(1(0(1(0(1(0\ldots)))))^*\ldots$$

In genere si procede con la "eliminazione di stati" (GNFA) per ottenere la forma testuale. Comunque, ciascuno di questi automi è riscrivibile come espressione regolare con i metodi standard.