2021

Guida alla risoluzione degli esercizi di ETC

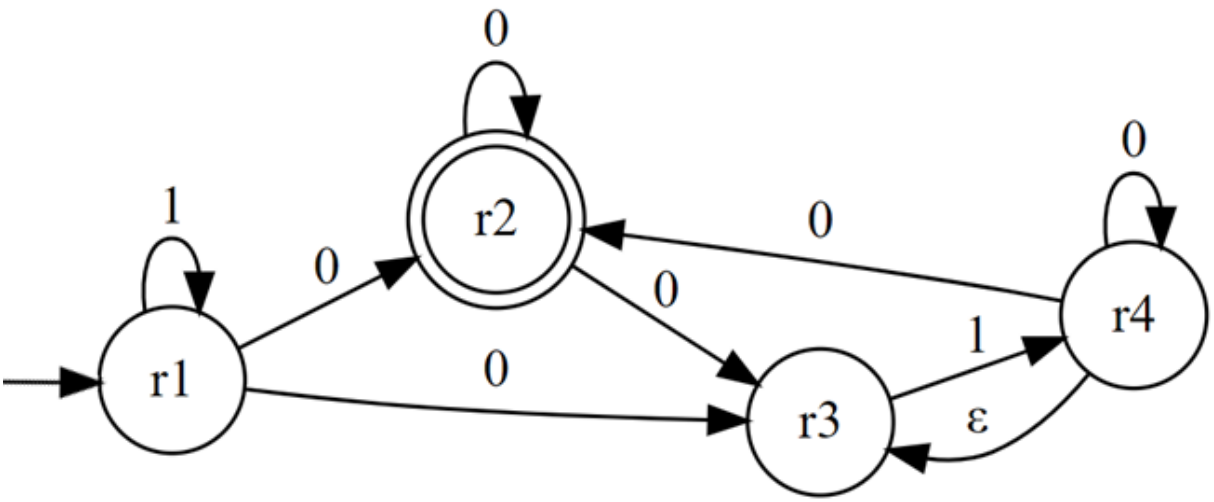
ESERCIZI SVOLTI E SPIEGAZIONI

MATTIA LIMONE

Indice

Definire una quintupla (NFA/DFA)

Una quintupla è definita da $(Q, \sum, \delta, q_0, F)$



Di conseguenza

Q = {r1, r2, r3, r4}
∑ = {0, 1}
q_0 = r1
F = {r2}

δ sarà definita dalla tabella

	ε	0	1
r1	{r2}	{r3}	{r1}
r2	Φ	{r2, r3}	Φ
r3	Φ	Φ	{r4}
r4	{r3}	{r4}	{r2}

Q rappresenta l'insieme degli stati

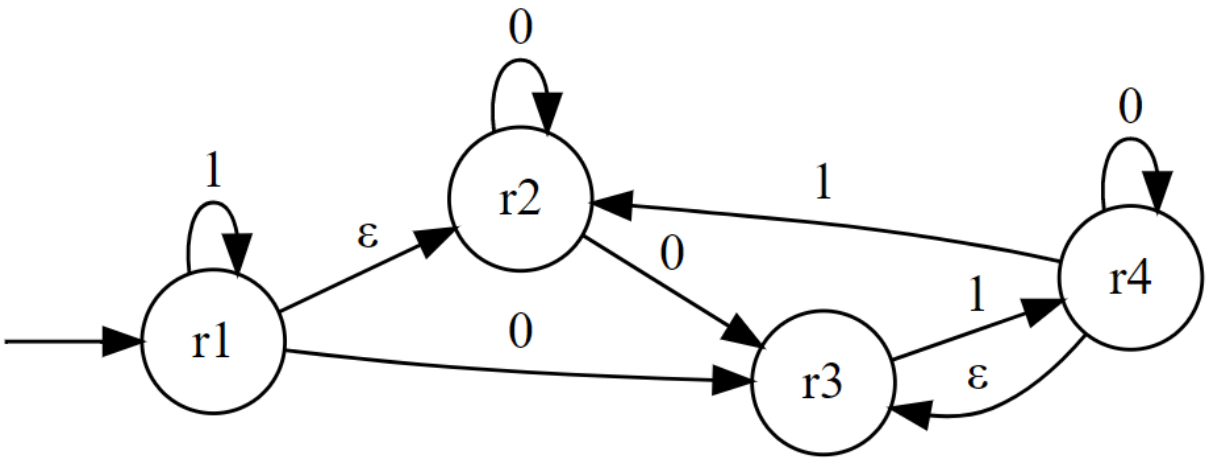
∑ l'alfabeto

q_0 lo stato iniziale

F l'insieme degli stati finali

δ la funzione di transizione

Determinare DFA a partire da un NFA



Prendiamo in esempio il seguente NFA

I passi da seguire sono i seguenti

- Determinare l'insieme degli stati iniziali
- Seguo il percorso degli insiemi di stati fino all'esaurimento dei percorsi
- Determino gli stati finali, scegliendo tra quelli che presentano almeno uno stato finale dell'NFA

Di conseguenza lo stato iniziale sarà {r1,r2} data la presenza dell'ε transition.

{r1,r2} con 1 cicla su se stesso

{r1,r2} con 0 va in {r2,r3}

{r2,r3} con 1 va in {r3,r4}, fare attenzione all'ε transition in r4

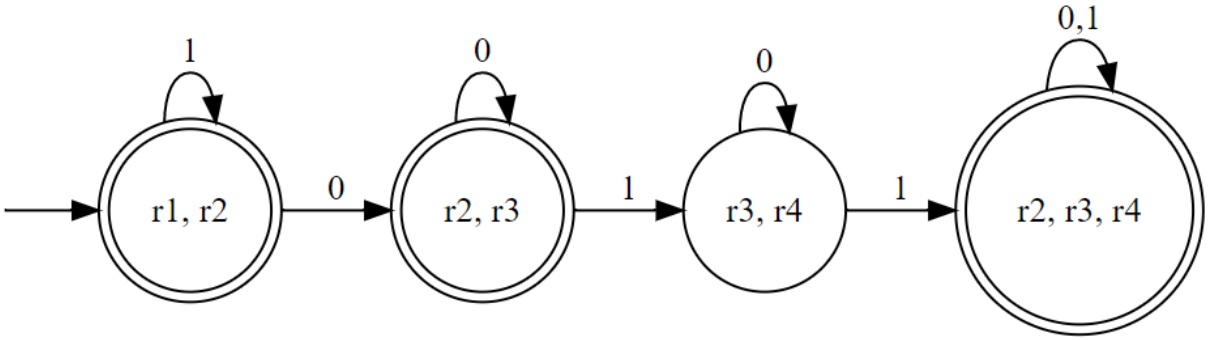
{r2,r3} con 0 cicla su se stesso

{r3,r4} con 1 va in {r2,r3,r4} va in r3 sempre per l'ε transition in r4

{r2,r3,r4} con 1 cicla su se stesso

{r2,r3,r4} con 0 cicla su se stesso

Segno come stati finali quelli in cui è presente r2 ed ottengo il seguente DFA


1
0
0
0,1
r1, r2
0
r2, r3
1
r3, r4
1
r2, r3, r4

Determinare diagramma DFA con chiusura per l'intersezione

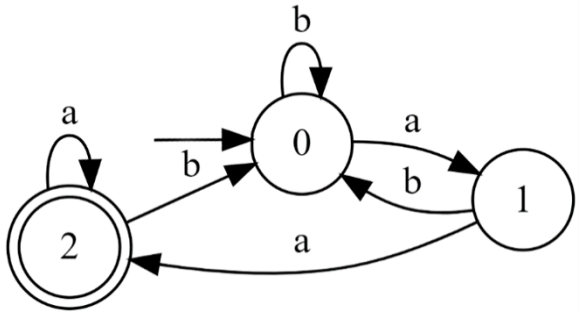
Alfabeto: {a, b}

L = {w | w termina con aa e non contiene sottostringa bb}

1) Divido il problema in proprietà

Proprietà 1: {w | w termina con aa}
Proprietà 2: {w | w non contiene sottostringa bb}

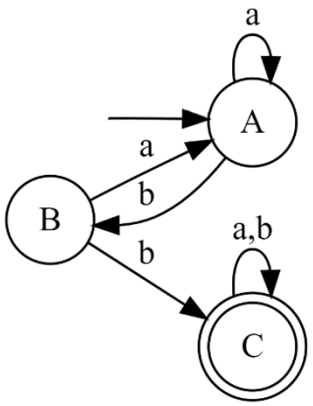
Automa per la proprietà 1



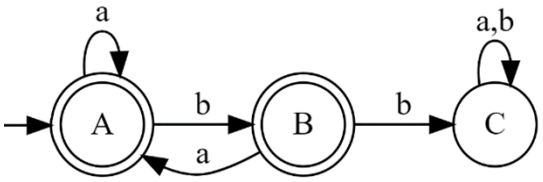
Automa per la proprietà 2

(Scrivo il complementare e poi inverto gli stati finali in non accettanti e non accettanti in stati finali)

Accetta bb

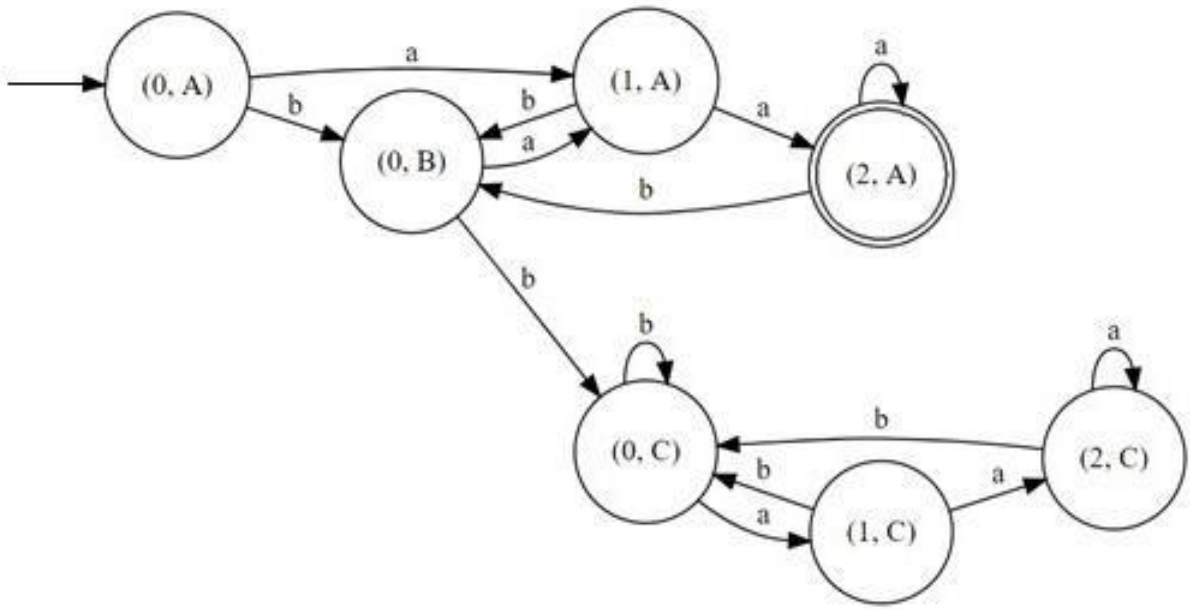


Non accetta bb



Effettuo a questo punto l'intersezione fra i due automi, si parte dalla coppia di stati iniziali e si analizza il comportamento al variare dell'alfabeto.

Automa intersezione

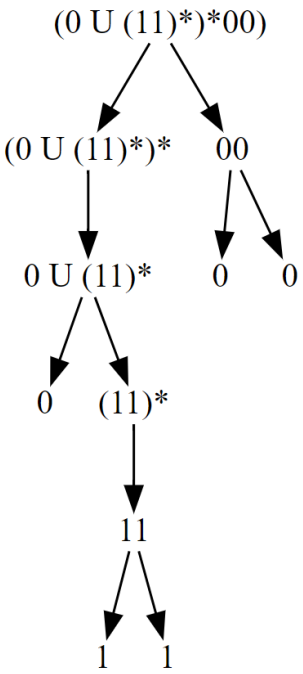


Ciò che ottengo è un automa di questo tipo, dove gli stati finali sono gli stati che contengono entrambi gli stati finali dei due automi di partenza (chiusura rispetto all'intersezione), quindi in questo caso avrebbero potuto essere solo la coppia (2, A) e (2, B), ma il secondo non è presente.
Nel caso fosse richiesta la chiusura rispetto all'unione gli stati finali sarebbero allora stati quelli con almeno uno dei due stati finali dei precedenti automi (2, x), (x, A), (x, B).

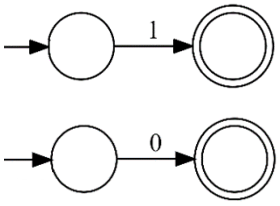
Determinare NFA da Espressione Regolare

Prendiamo in esempio la RegEx = (0 ∪ (11)*)*00

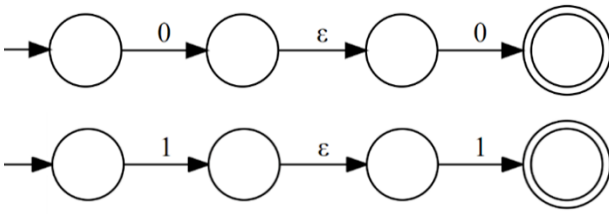
L'esercizio si svolge con la scomposizione dell'espressione regolare



Partiamo dagli automi 0 e 1 perché i più semplici



Risaliamo l'albero dell'espressione regolare e passiamo a 11 e 00

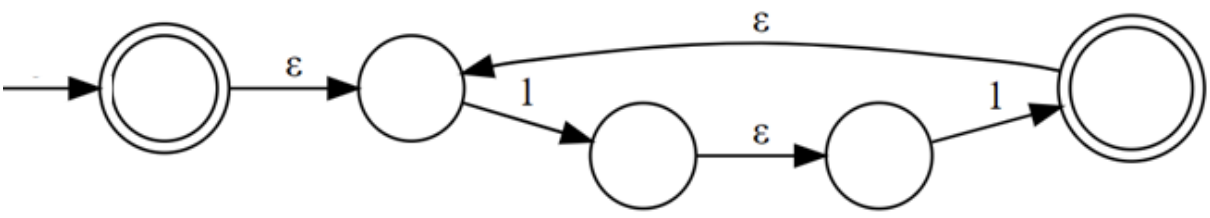


La procedura prevede di aggiungere una ε transition dallo stato finale del primo automa a quello iniziale dell'altro automa.

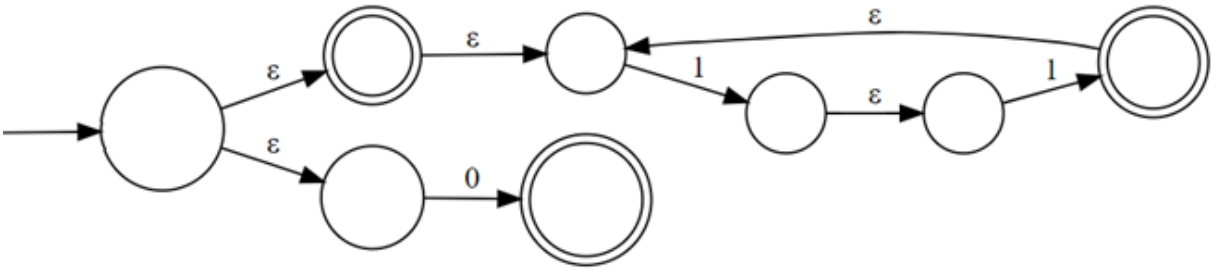
Adesso si passa a (11)*.

Per la chiusura di Kleene il passaggio è semplice:

- aggiungo un nuovo stato iniziale, che è anche finale per riconoscere la stringa vuota
- aggiungo una ε transition da questo nuovo stato iniziale al precedente stato iniziale
- aggiungo una ε transition dagli stati finali del precedente automa allo stato iniziale prima dell'aggiunta del nuovo stato iniziale

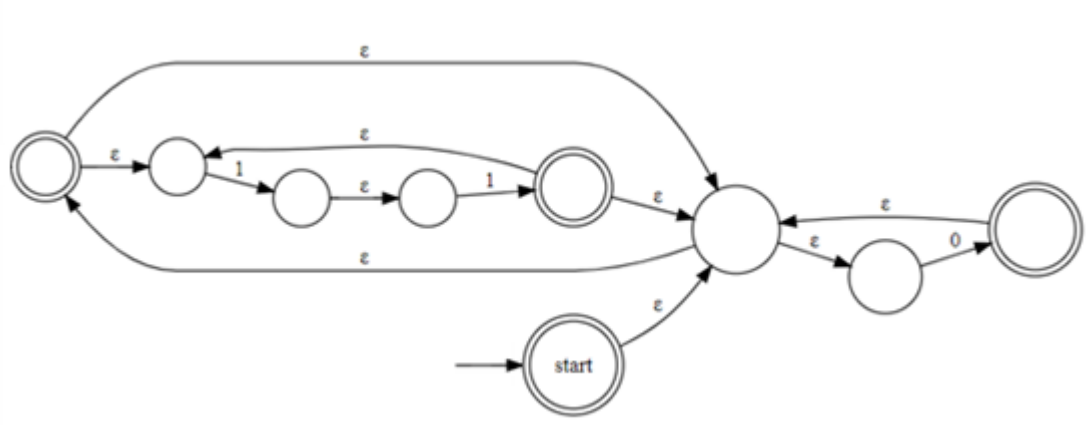


Adesso si passa a 0 ∪ (11)*
Per l'unione dato che abbiamo già generato i due automi da unire aggiungiamo un nuovo stato iniziale collegato con ε transition agli stati iniziali dei due automi da unire

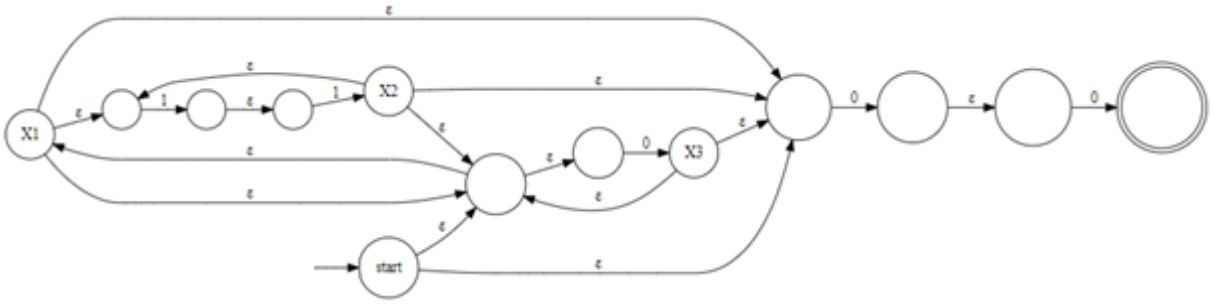


Adesso si passa a (0 ∪ (11)*)*
Il procedimento è lo stesso spiegato in precedenza, quindi:

- aggiungo un nuovo stato iniziale, che è anche finale per riconoscere la stringa vuota
- aggiungo una ε transition da questo nuovo stato iniziale al precedente stato iniziale
- aggiungo una ε transition dagli stati finali del precedente automa allo stato iniziale prima dell'aggiunta del nuovo stato iniziale



Adesso si passa a (0 ∪ (11)*)*00
La concatenazione avviene collegando gli stati finali del precedente automa con lo stato iniziale dell'automa da concatenare tramite ε transition. Ricorda che si conserva come stato finale solo quello dell'automa a cui concateniamo.



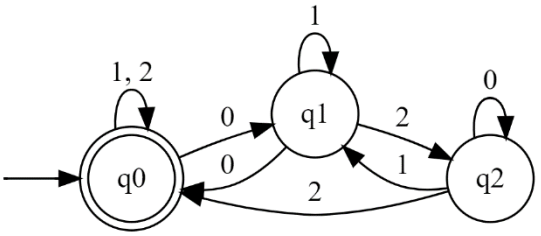
X1, X2, X3 e start erano i precedenti stati finali collegati con ε transition allo stato iniziale dell'automa 00 a cui andava concatenato.

Il procedimento è ridondante si, ma semplice da eseguire.

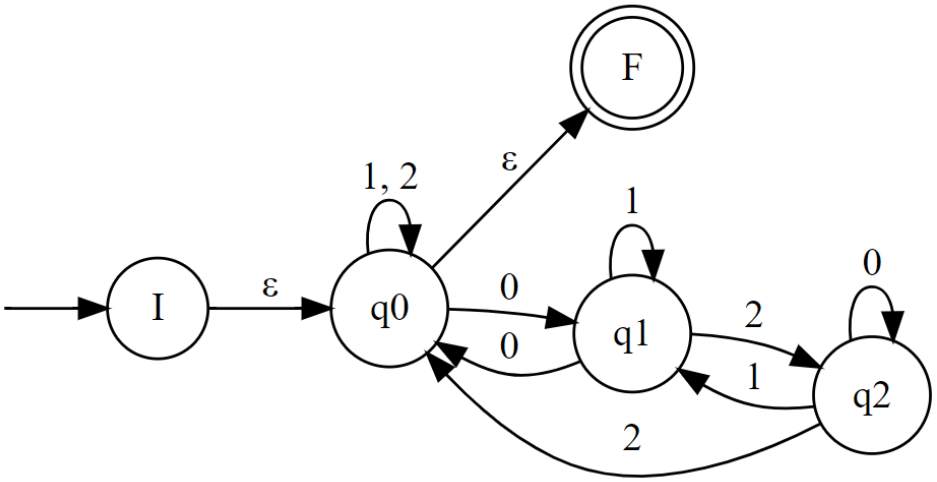
Determinare Espressione Regolare da Automa

Disegnare l'automa

	0	1	2
q_0	q_1	q_0	q_0
q_1	q_0	q_1	q_2
q_2	q_2	q_1	q_0

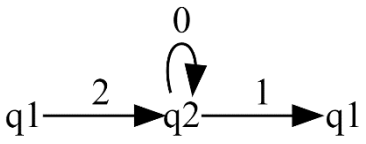


Aggiungere stato iniziale (I) e finale (F) fittizi

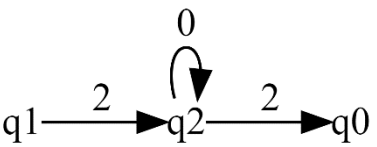


Scegliere uno stato da eliminare, ad esempio q2 e calcolare i percorsi.

In q2 si arriva solo da q1 ed ha due uscite q1 e q0 ed una ridondanza; quindi i percorsi saranno i seguenti

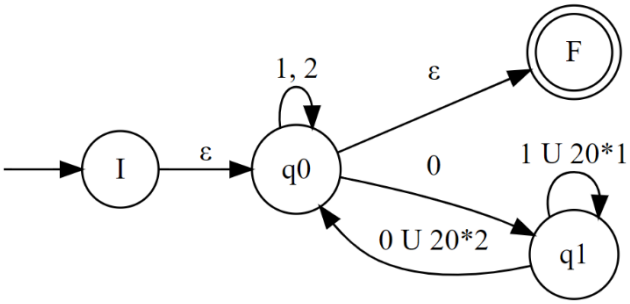


2
0* (ciclo)
1

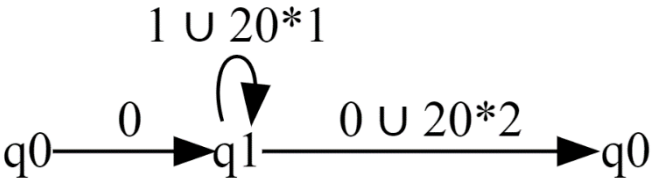


2
0* (ciclo)
2

Ed ottengo di conseguenza dal primo percorso un aggiunta al ciclo su q1 e dal secondo un'aggiunta al collegamento q1->q0



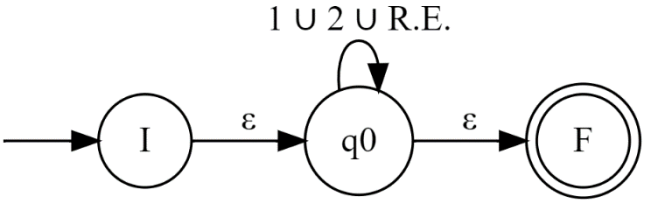
Si reitera il processo scegliendo un nuovo nodo da eliminare, banalmente q1 e ne calcolo i percorsi. Questa volta il percorso è solo uno.



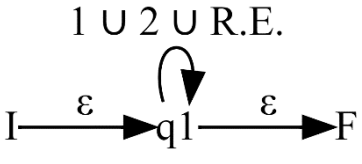
E quindi ottengo un'aggiunta al ciclo su q0, dato che l'espressione che ne esce è lunga la riassumo ed da qui in avanti sarà chiamata R.E.

R.E. = 0(1 ∪ 20*1)*(0 ∪ 20*2)

Di conseguenza ottengo il seguente automa



Non rimane che eliminare l'ultimo stato q0, ma dato che qui il passo è banale avendo ε transition in entrata e uscita dal nodo e quindi è solo un ciclo, l'unico percorso è



(1 ∪ 2 ∪ R.E.)*
ovvero
(1 ∪ 2 ∪ (0(1 ∪ 20*1)*(0 ∪ 20*2)))*

Funzione di transizione estesa F*

La funzione di transizione F può essere estesa a F* che opera su stati e stringhe.

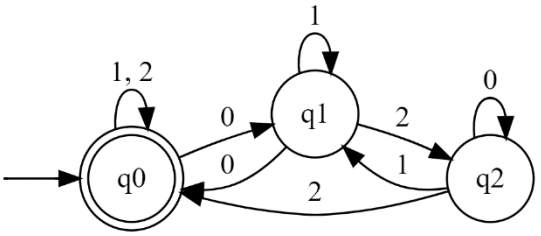
Esercizio del tipo dato l'automa (con stato iniziale q0 e F ={q0})

- determinare F*(q0, 0122)

Prendiamo ad esempio il seguente automa.

	0	1	2
q_0	q_1	q_0	q_0
q_1	q_0	q_1	q_2
q_2	q_2	q_1	q_0

Disegnare l'automa



Passo 1
- F*(q0, 0122) = F(F*(q0, 012), 2))

Risultato
= F(q2, 2) = q0

Passo 2
- F*(q0, 012) = F(F*(q0, 01), 2))

= F(q1, 2) = q2

Passo 3
- F*(q0, 01) = F(F*(q0, 0), 1))

= F(q1, 1) = q1

Passo 4
- F*(q0, 0) = F(F*(q0, ε), 0))

= F(q0, 0) = q1

Procedimento

In q0 con input 0 si ottiene lo stato q1
- F*(q0, 0) = F(F*(q0, ε), 0)) = F(q0, 0) = q1

sostituisco F*(q0, 0) con q1 nel Passo 3
- F*(q0,01) = F(F*(q0, 0), 1)) = F(q1, 1) = q1

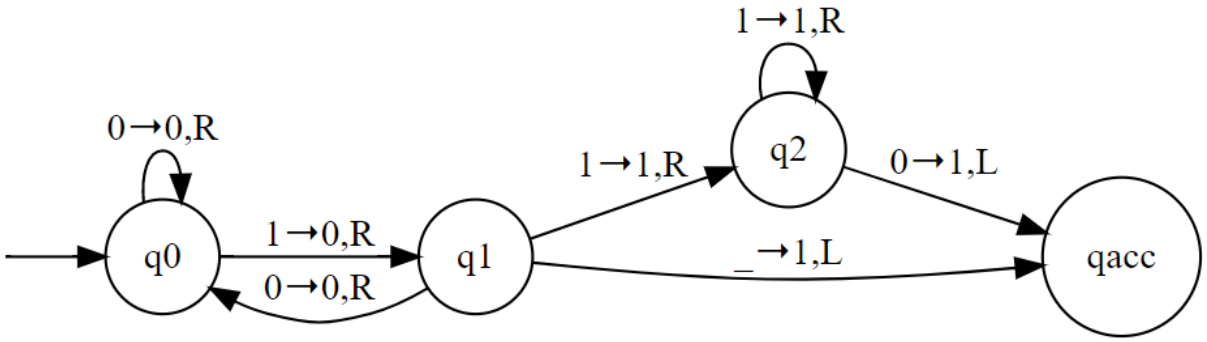
sostituisco F*(q0, 01) con q1 nel Passo 2
- F*(q0, 012) = F(F*(q0, 01), 2)) = F(q1, 2) = q2

sostituisco F*(q0, 012) con q2 nel Passo 1
- F*(q0, 0122) = F(F*(q0, 012), 2)) = F(q2, 2) = q0

Determinare configurazione MdT

In questo tipo di esercizio viene fornita una MdT ed una o più stringhe di cui calcolare la configurazione.

Prendiamo in esempio il seguente automa e calcoliamo la configurazione per le seguenti stringe 00110 e 0111



[00110]

Si parte ovviamente da

- $C_1 = q_000110$ “=> la testina ora punta al primo 0 “

Una volta letto 0 la testina si sposta a destra, lo 0 rimane e lo stato resta q_0

- $C_2 = 0q_00110$ => la testina ora punta al secondo 0

Una volta letto 0 la testina si sposta a destra, lo 0 rimane e lo stato resta q_0

- $C_3 = 00q0110$ => la testina ora punta al primo 1

Una volta letto 1 la testina si sposta a destra, sostituisce 1 con 0 e lo stato ora è q_1

- $C_4 = 000q_110$ => la testina ora punta al “secondo” 1

Una volta letto 1 la testina si sposta a destra, l’1 rimane e lo stato ora è q_2

- $C_5 = 0001q_20$ => la testina ora punta all’ultimo 0

Una volta letto la testina si sposta a sinistra, sostituisce 0 con 1 e lo stato ora è q_{acc}

- $C_6 = 000q_{acc}11$ => CONF. ACCEPT

[0111]

Si parte ovviamente da

- $C_1 = q_0 0111$ => la testina punta al primo 0

Una volta letto 0 la testina si sposta a destra, lo 0 rimane e lo stato resta q_0

- $C_2 = 0q_0 111$ => la testina ora punta al primo 1

Una volta letto 1 la testina si sposta a destra, sostituisce 1 con 0 e lo stato ora è q_1

- $C_3 = 00q_1 11$ => la testina ora punta al “secondo” 1

Una volta letto 1 la testina si sposta a destra, l’1 rimane e lo stato ora è q_2

- $C_4 = 001q_2 1$ => la testina ora punta al “terzo” 1

Una volta letto 1 la testina si sposta a destra, l’1 rimane e lo stato resta q_2

- $C_5 = 0011q_2$ => non è stato raggiunto lo stato q_{acc} quindi CONF. REJECT

Definizione Ricorsiva per le Espressioni Regolari

Diciamo che R è un'**espressione regolare** se R è:

- α per qualche α nell'alfabeto X,
- ε
- $\emptyset$
- $(R_1 \cup R_2)$, dove R_1 ed R_2 sono espressioni regolari,
- $(R_1 \circ R_2)$, dove R_1 ed R_2 sono espressioni regolari, 0
- (R_1^*), dove R_1 è un'espressione regolare.

Nei punti 1 e 2, le espressioni regolari α e ε rappresentano i linguaggi $\{\alpha\}$ e $\{\varepsilon\}$, rispettivamente. Nel punto 3, l'espressione regolare $\emptyset$ rappresenta il linguaggio vuoto. Nei punti 4, 5, e 6, le espressioni rappresentano i linguaggi ottenuti prendendo l'unione o la concatenazione dei linguaggi R_1 ed R_2, o lo star del linguaggio R_1, rispettivamente.

Teorema di Kleene

Un linguaggio L è regolare ⇔ si può individuare con un'espressione regolare.

Dimostrazione

Sapendo che:
Linguaggio L è regolare ⇔ L riconosciuto da NFA ⇔ L riconosciuto da DFA
Si mostra che:
Per ogni DFA A possiamo costruire una e.r. R con L(R) = L(A). Per ogni e.r. R esiste un NFA A, tale che L(A) = L(R). Cioè:
L riconosciuto da DFA ⇔ L ammette un'e.r.
Quindi:
Linguaggio L è regolare ⇔ L ammette un'e.r.

Definizione di configurazione MdT

Supponiamo di avere *a*, *b* e *c* in Γ, così come *u* e *v* in Γ* e gli stati q_i e q_j. In tal caso uaq_ibv e uq_jacv sono due configurazioni. Diciamo che

$$uaq_ibv \quad \text{produce} \quad uq_jacv$$

se nella funzione di transizione $\delta(q_i, b) = (q_j, c, L)$. Questo nel caso in cui la macchina di Turing effettua uno spostamento verso sinistra. Per lo spostamento a destra, diciamo che

$$uaq_ibv \quad \text{produce} \quad uacq_jv$$

se $\delta(q_i, b) = (q_j, c, R)$.

Definizione di linguaggio riconosciuto da MdT

L'insieme di stringhe che una MdT, che chiamiamo M, accetta rappresenta il linguaggio di M, o il linguaggio riconosciuto da M, denotato con L(M).

Un linguaggio può essere definito **Turing-riconoscibile** o anche **linguaggio ricorsivamente enumerabile** se esiste una Macchina di Turing che lo riconosce.

Quando una Macchina di Turing si ferma su ogni singolo input e non ciclano mai, vengono chiamate **decisori**, quindi tale macchina ***"decide"*** un certo linguaggio se riconosce tale linguaggio.

Un linguaggio si dice **Turing-decidibile** o anche **linguaggio ricorsivo** o semplicemente decidibile se esiste una macchina di Turing che lo decide.

Dimostrazione per ogni DFA esiste una MdT equivalente

Per ogni Automa Finito Deterministico esiste una Macchina di Turing equivalente

Dimostrazione

Sapendo che:

Linguaggio L è regolare ⇔ L riconosciuto da NFA ⇔ L riconosciuto da DFA

Si mostra che:

Presa una stringa χ posso sempre decidere se appartiene o meno al linguaggio L, perché posso sempre farla "riconoscere" o "non riconoscere" dall'automa di L, allora il linguaggio L è decidibile.

Linguaggio L è decidibile ⇔ L deciso da MdT

Il linguaggio L regolare è decidibile e quindi esiste una Macchina di Turing che lo decide.

Se un linguaggio è decidibile allora esiste una macchina di Turing che lo riconosce. Quindi il linguaggio è anche Turing riconoscibile.

Di conseguenza per ogni DFA che riconosce un linguaggio regolare L esiste una MdT che riconosce lo stesso linguaggio L ed è quindi ad essa equivalente.

Funzioni con MdT

Cominciamo col definire due tipologie di funzioni generiche con MdT.

Le funzioni di conta/calcolo e le funzioni di comparazione.

Definiamo come funzioni di comparazione quelle del tipo

$$f(x) = \{\ 1,\ se\ \ x > y;\ 0,\ se\ \ x \leq y$$

Definiamo come funzioni di conta/calcolo quelle del tipo

$$f(x) = x + 2$$

Funzione di comparazione

$f(x) = \{\ 1,\ se\ \ x > y;\ 0,\ se\ \ x \leq y$

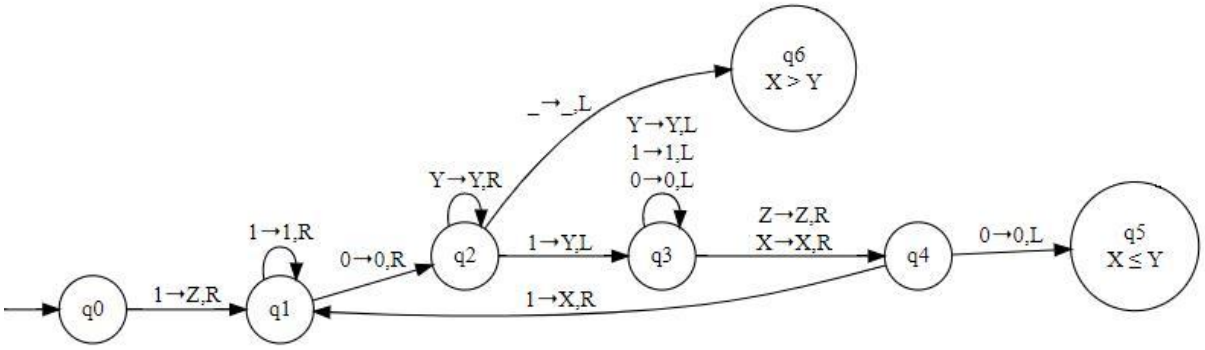
1) Marcare con un nuovo carattere (Z) il primo 1 che incontriamo in modo da salvare l'inizio della stringa, mi sposto a destra e cambio stato (q_1)
2) Ciclo su sé stesso il nuovo stato (q1) finchè incontro 1 spostandosi a destra sulla stringa, quando leggo 0 mi sposto a destra ed in un nuovo stato (q_2) per analizzare il secondo numero
3) Quando incontro il primo 1 lo marco con un nuovo carattere (Y) e inizio a scorrere la stringa verso sinistra e mi sposto in un nuovo stato (q_3)
4) Ciclo su me stesso (q_3) qualsiasi carattere posso incontrare del tipo 1,0 e Y perché cerco l'inizio della stringa. Quando incontro Z mi sposto in un nuovo stato (q_4) riscrivo Z e mi sposto a destra perché sono all'inizio della stringa
5) A questo punto se incontro 0 mi sposto nel nuovo stato q_5 spostando la testina a sinistra e so che la stringa alla destra dello 0 contiene un numero maggiore o uguale di 1 di quella a sinistra situazione del tipo ZX0Y1. Se incontro 1 sovrascrivo con un nuovo

carattere (X) mi sposto a destra e vado nello stato (q_1) per scorrere la stringa verso destra come nel punto 2

A questo punto aggiungo le condizioni generate dai nuovi simboli aggiunti:

6) Ciclo q_2 su sé stesso finché incontro Y riscrivendo Y e spostandosi a destra.
7) Se nello stato q_2 leggo un blank (_) vuol dire che mi trovo in una situazione del genere “ZXX0YY” e quindi la stringa alla sinistra dello 0 contiene un numero maggiore di 1 di quella alla destra, quindi sposto la testina a sinistra e mi sposto in un nuovo stato q_6.
8) Aggiungo lo spostamento a destra leggendo X e scrivendo X nel collegamento q_3-q_4 perchè potrei trovarmi nella situazione in cui la stringa ZX10Y11 e quindi devo ricominciare il ciclo o dedurre una soluzione

Di conseguenza l’automa che si ottiene con questa spiegazione è il seguente:

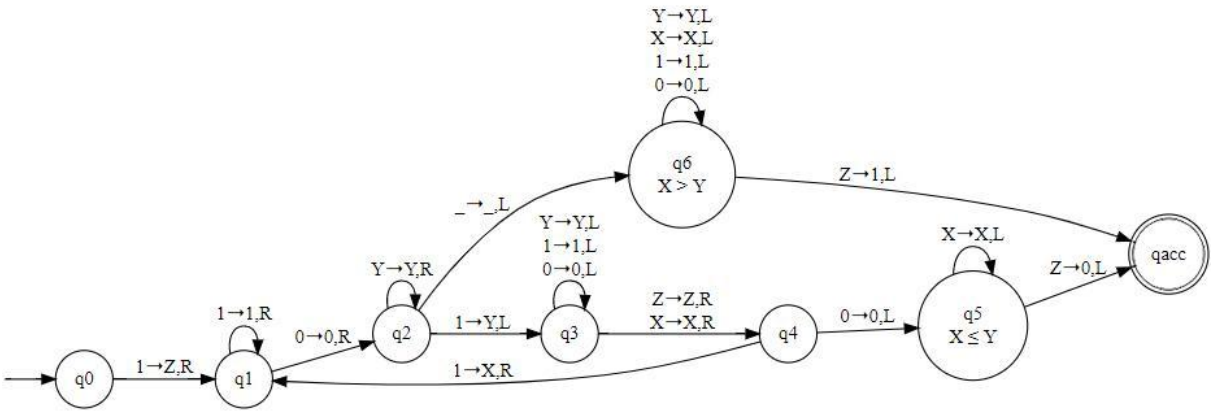


Calcolata la base del comparatore aggiungiamo la computazione ovvero, se X > Y scrivo 1, se X ≤ Y scrivo 0

1) Ciclo q_5 su sé stesso spostandosi verso sinistra per spostarmi fino all’inizio della stringa, in questo caso posso solo incontrare delle X.

A questo punto quando incontro il carattere Z lo sostituisco con 0 e mi sposto a sinistra essendo arrivati all'inizio la testina punterà sempre a questo carattere e vado nello stato q_{acc}

2) Ciclo q_6 su sé stesso spostandosi verso sinistra per spostarmi fino all'inizio della stringa qualsivoglia carattere incontro eccezion fatta per la Z. A questo punto quando incontro il carattere Z lo sostituisco con 1 e mi sposto a sinistra essendo arrivati all'inizio la testina punterà sempre a questo carattere e vado nello stato q_{acc}



Funzione Somma

$f(x) = x + 2$

1) Il primo stato q_0 come per il comparatore svolge il ruolo di marker quindi, appena legge il primo 1 lo sostituisce con il carattere blank (_) spostando la testina a destra e andando nello stato q_1
2) Il secondo stato cicla su sé stesso fino alla fine della stringa quindi se legge 1 scrive 1 e si sposta a destra, se legge blank (_) scrive 1 allora si sposta a destra andando nello stato q_2 aggiungendo alla stringa il primo 1 richiesto
3) Trovandoci nello stato q_2, l'unico carattere che potremo incontrare sarà il blank (_). Quindi lo sostituisco con 1 per aggiungere il secondo 1 richiesto, ma mi comincio a spostare verso sinistra con la testina dato che devo ritornare ad inizio stringa e quindi vado nello stato q_3
4) In q_3 effettuo un ciclo leggendo 1 scrivendo 1 e spostandomi verso sinistra per tornare ad inizio stringa, quando la macchina legge blank (_) questo sarà il blank scritto al punto 1 dallo stato "marker" quindi lo sostituisco con 1 e mi sposto a sinistra, essendo arrivati ad inizio stringa la testina punterà sempre a questo carattere e vado nello stato q_{acc}
5) Per terminare aggiungo il caso stringa vuota ovvero x + 2 con x= 0. Molto semplicemente, aggiungo un nuovo stato e 2 movimenti speculari a quelli di entrata e uscita di q_2 che terminano direttamente nello stato q_{acc}

   L'automa ottenuto è il seguente