*Scritti da Gabriel: gli appunti comprendono anche le domande a quiz interattive e tutti gli esercizi (sezione in fondo al file). Gli esercizi fatti in classe sono dentro la lezione apposita.*

***02/03/2022: Introduzione: automi a stati finiti deterministici e concetti base***

La domanda fondamentale di inizio corso è determinare cosa effettivamente possa o non possa fare un calcolatore, un automa o altro, capendo la struttura di un certo problema e quindi risolverlo possibilmente. Per descrivere un *problema* noi dobbiamo descrivere i possibili input, i possibili output o risultati che coi aspettiamo e la relazione tra di essi.

A tale scopo sappiamo di avere un certo *algoritmo*, quindi una procedura tale a risolvere un dato problema, sulla base di un certo tipo di calcoli e computazioni.

Analogamente, importante valutare correttamente la *complessità* dell’algoritmo, quindi che l’algoritmo risolva effettivamente il problema presente, valutandone poi la *complessità spaziale/temporale.*

Allo stesso modo, poter risolvere un problema riguarda l’astrazione del problema esprimendolo sotto forma di *linguaggio*, visto come *insieme di stringhe*. Le soluzioni trasformano quindi le linee di input con linee di output. L’esempio di problema può essere l’insieme dei numeri primi.

Tutti i processi computazionali possono essere ridotti ad uno tra la determinazione di *appartenenza* ad un insieme, eseguendo una *mappatura* tra insiemi di stringhe.

A questo punto introduciamo il concetto di *automa*, dispositivo matematico astratto in grado di determinare l’appartenenza di una stringa ad un insieme di stringhe e in grado di trasformare una stringa in un’altra stringa. Esso ha tutti gli aspetti di un computer, avendo I/O, una memoria, è in grado di prendere decisioni e può trasformare l’input in output.

Descriviamo in particolare l’aspetto della *memoria*, distinguendo tra *memoria finita ed infinita* e, naturalmente, anche il tipo di accesso alla memoria, distinguendo tra *limitato e illimitato*.

Parliamo ora di *automi a stati finiti*, che sono il modello computazionale più semplice e dispongono di una quantità di memoria *finita*. Gli automi sono usati come *modello* per ricerche di parole chiave, analizzatori lessicali, progettazione di circuiti digitali e scopi similari.

Un esempio semplice è la porta automatica, capedo tramite un sensore se ci sia o meno una persona rilevando la presenza di una persona. In questo caso gli stati sono: *chiusa* o *aperta*.

Quattro possibili input: *fronte/retro/ambo/nessuna.*

In maniera più precisa, possiamo definire alcuni concetti, come il concetto di *alfabeto*, che è un insieme finito e non vuoto di simboli. Il simbolo che lo identifica è il *∑*, quello di serie/sommatoria (ad esempio *∑ = {0,1},* alfabeto binario, oppure *∑* = {a, b, c, d…. z} insieme delle lettere minuscole, insieme dei caratteri ASCII, ecc.).

Definiamo anche una *stringa (o parola)*, cioè una sequenza finita di simboli da un alfabeto. La *stringa vuota* sarà definita con *ε,* contenuto in *∑0.* La stringa è composta da un certo numero di simboli e questa è la *lunghezza di una stringa*, definita come*|w|*.

Vi sono poi le *potenze di un alfabeto*, Σk = insieme delle stringhe di lunghezza k con simboli da Σ.

Ad esempio: Σ = {0, 1}

Σ 0 = {ε} Σ 1 = {0, 1} Σ 2 = {00, 01, 10, 11}

L’insieme di tutte le stringhe su Σ è denotato da Σ\* (unione di tutte le stringhe matematicamente parlando) e più in generale, dato un alfabeto, un *linguaggio* è ogni sottoinsieme L ⊆ Σ∗.

Il *linguaggio vuoto* non contiene nessuna parola ed è definito dal simbolo ∅; si ricordi che il linguaggio vuoto è diverso da una parola vuota.

Un *Automa a Stati Finiti Deterministico (DFA)* è una quintupla *A = (Q, Σ, δ, q0, F),* dove:

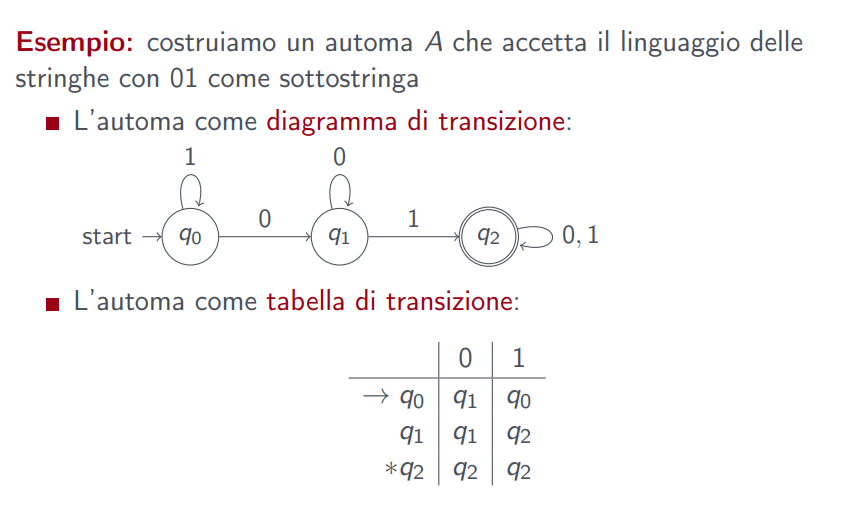
* Q è un insieme finito di *stati*
* Σ è un *alfabeto finito* (= simboli in input)
* δ è una *funzione di transizione* (q, a) → q’
* q0 ∈ Q è lo *stato iniziale*
* F ⊆ Q è un insieme di *stati finali*

In un automa deterministico ogni relazione di transizione ha uno stato di destinazione per ogni simbolo (evento). Non possono esserci relazioni di transizione nello stesso stato che usano lo stesso simbolo.

Una *funzione di transizione* è una funzione con dominio finito, prendendo come finiti l’alfabeto di input e l’insieme degli stati che l’automa può assumere viene rappresentato in forma tabellare come *tabella di transizione* oppure sotto forma di grafo orientato, i cui archi rappresentano le transizioni tra uno stato e l’altro; qui si parla di *diagramma di transizione*.

Possiamo quindi rappresentare gli automi sia come diagramma di transizione che come tabella di transizione.

Esempio pratico:



Data una parola w = w1w2 . . .wn, la computazione dell’automa *A* con input *w* è una sequenza di stati r0,r1 ..r*n*che rispetta due condizioni:

1. r0 = q0 (inizia dallo stato iniziale)
2. δ(ri ,wi+1) = ri+1 per ogni i = 0, … , n − 1 (rispetta la funzione di transizione)

Diciamo che la computazione accetta la parola w se: rn ∈ F (la computazione termina in uno stato finale).

Un DFA *A* accetta la parola *w* se la computazione accetta *w*. Formalmente, il linguaggio accettato da A è L(A) = {w ∈ Σ∗ | A accetta w}.

Domande interattive della lezione:

* Quali sono gli input della funzione di transizione di un DFA? 🡪 Uno stato e un simbolo
* Un DFA deve avere un solo stato finale? 🡪 Falso
* Qual è l’input del problema “È un numero primo?” 🡪 {0,1,2,3,4,5..}
* Quale insieme di stringhe rappresenta il problema nel caso “È un numero primo?” 🡪 {2,3,5,7,11,13..}
* Per ogni elemento, stabilire se è carattere, parola, linguaggio:

1. a 🡪 carattere
2. abracadabra 🡪 parola
3. {abracadabra, apostrofo, a} 🡪 linguaggio di tre parole
4. {abracadabra} 🡪 linguaggio composto da una parola

* Dato l’alfabeto ∑ = {0,1} quante sono le stringhe appartenenti al linguaggio? 🡪 16
* Quale dei seguenti linguaggi sull’alfabeto {0,1} contiene un numero infinito di stringhe? 🡪 Tutte le stringhe che iniziano con 1
* Dato l’alfabeto ∑ = {0,1} quante sono le stringhe contenute in Σ 0? 🡪 ε
* Quante stringhe ci sono nel linguaggio sull’alfabeto {0,1} di tutte le stringhe di lunghezza *n*? 🡪 2n
* Un DFA possiede una quantità di memoria molto limitata 🡪 Vero
* L’insieme di tutti gli stati di un automa si indica con 🡪 Q
* Quali sono gli input della funzione di transizione di un DFA? 🡪 Uno stato e un simbolo
* Un DFA deve avere un solo stato finale 🡪 Falso

***03/03/2022: Automi non deterministici/utilizzo di JFLAP/NFA***

Una volta costruita tutta la computazione è nello stato finale e la parola è *corretta*, essa viene *accettata*, altrimenti se non è uno stato finale l’automa *rifiuta* questa parola.

I linguaggi accettati da automi a stati finiti sono detti *linguaggi regolari.*

Seguono esempi di automi DFA costruibili con il software *JFLAP*.



A tale scopo *piccolo tutorial sul software* disponibile in formato jar (descrizione icone da sx verso dx)

1. icona della freccia, che permette di spostarsi liberamente. Cliccando in questo stato con il tasto dx su uno stato è possibile impostarlo come *Initial* o *Final* o eventualmente cambiarne l’etichetta. In questa modalità, inoltre, facendo doppio clic sul valore di una freccia, esso può essere modificato graficamente (viene evidenziato in rosso) e si può immettere un valore senza doverlo eliminare.
2. icona degli stati, messi ad ogni clic del mouse.
3. icona di vettore. Cliccando sull’icona freccia e subito dopo su uno stato senza trascinare la freccia su altri stati, la freccia verrà rivolta verso lo stato attuale (quindi verso lo stato stesso). Altrimenti se si trascina verso un altro stato si imposta la freccia verso quello. Una volta inserito il valore, la freccia viene disegnata. È quindi possibile disegnare sia frecce che vanno avanti ma anche frecce che vanno indietro da uno stato ad un altro in maniera facile.
4. icona del teschio, che elimina frecce e/o stati. Passando la freccia, una volta selezionata questa icona su un valore, nel caso in cui uno stato abbia valori multipli, ne viene eliminato uno singolo, in qualunque posizione esso si trovi.
5. torna indietro
6. torna avanti

Attenzione anche ad *Automation Size* che funziona da zoom e si può selezionare tutto l’automa con il tasto sinistro e spostarlo poi per l’area di lavoro con il tasto destro.

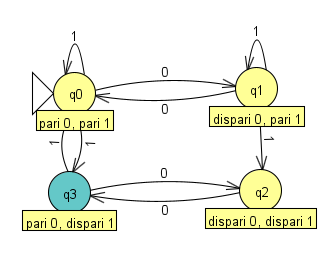


Sulla barra del menù, cliccando *Input* e poi su *Step by state* è possibile, dando un certo input come stringa, verificare il comportamento effettivo dell’automa. Ciò avviene se è stato impostato almeno uno stato iniziale. Si entra quindi nello stato Simulate e si avanza step by step.

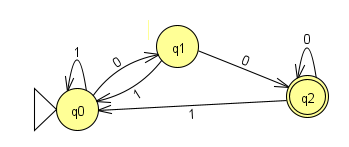
Se il comportamento finale del testo è verde significa che l’automa impostato è corretto; se è rosso, ci sta qualche problema o l’automa è non deterministico (si veda sotto). Per uscire dalla scheda di simulazione attuale e tornare nell’editor si preme la freccia X in alto a dx.

In conclusione, si può salvare l’automa nel formato compatibile con JFLAP oppure in immagine JPG o altro (cliccando su *Convert* per convertirlo, naturalmente, e su File per salvarlo).

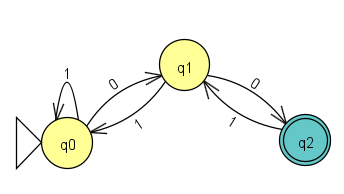
*- Insieme di tutte e sole le stringhe con un numero pari di zeri e un numero pari di uni*



*- Insieme di tutte le stringhe che finiscono con 00*



Un’idea non corretta su questo esempio sarebbe:



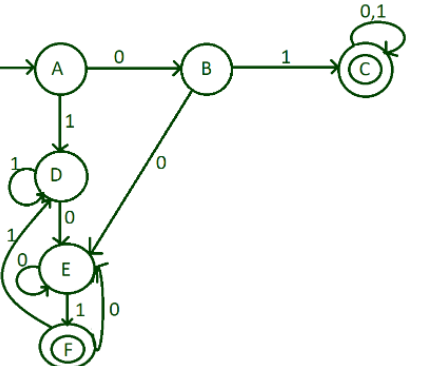
Di fatto è sbagliata come idea perché non c’è una fine vera e propria agli input esistenti (caso rosso su JFLAP come spiegato sopra; l’automa non accetta la stringa fornita); si nota che i due zeri ci sono, ma nel caso avessi un 1 e poi uno 0, ecco che si potrebbe generare un loop per cui la situazione non si sblocca mai.

*- Insieme di tutte le stringhe che contengono esattamente tre zeri (anche non consecutivi)*

Immagine che contiene testo, orologio

Descrizione generata automaticamente

*- Insieme delle stringhe che cominciano o finiscono (o entrambe le cose) con 01 (assegnato dal prof)*

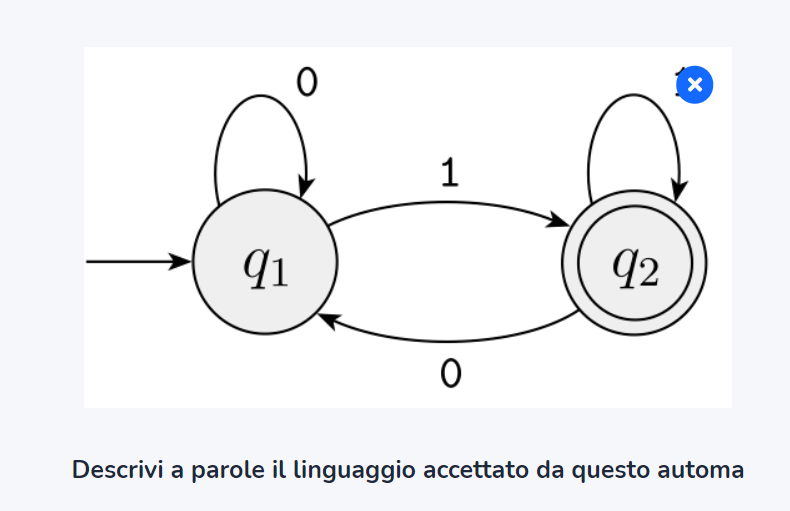


*Altra soluzione:*

Immagine che contiene neve, orologio, busta

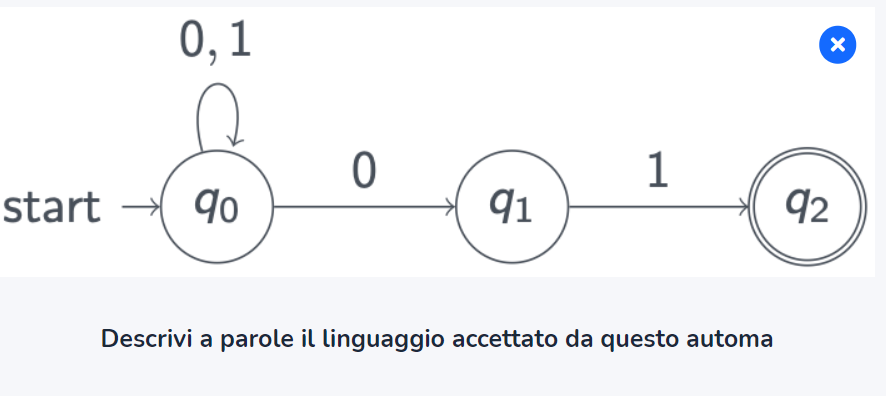
Descrizione generata automaticamente

*- Quale linguaggio accetta il seguente automa? (c’è un numero nascosto dalla X di chiusura ed è un 1)*



Risposta: Tutte le stringhe composte da 0 e 1 che terminano con 1

*Quale linguaggio accetta il seguente automa?*



Risposta (estesa, soluzione più in basso nella pagina):

È un automa non deterministico, perché *la relazione di transizione può avere uno o più stati di destinazione con lo stesso simbolo (evento).*

Se avessi come input 01101, un ramo avrebbe un ramo finale corrispondente, mentre l’altro no.

L’esempio esteso con rappresentazione ad albero è:

0

1

1

Strada che si blocca

0

1

Terminazione finale

Un *Automa a Stati Finiti Non Deterministico (NFA)* è una quintupla *A = (Q, Σ, δ, q0, F),* dove:

* Q è un insieme finito di *stati*
* Σ è un *alfabeto finito* (= simboli in input)
* δ è una *funzione di transizione* che prende in input (q, a) e restituisce *un sottoinsieme di Q*
* q0 ∈ Q è lo *stato iniziale*
* F ⊆ Q è un insieme di *stati finali*

Una parola viene accettata se c’è un ramo che arriva ad uno stato finale, altrimenti viene rifiutata; questo avviene se arriva in uno stato che non è finale.

*Conclusione*: accetta parole che terminano esattamente con 01.

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene tavolo

Descrizione generata automaticamente

Data una parola w = w1w2 . . .wn, una computazione di un NFA *A* con input *w* è una sequenza di stati r0, r1 ...rn che rispetta due condizioni:

1. r0 = q0 (inizia dallo stato iniziale)
2. δ(ri ,wi+1) = ri+1 per ogni i = 0, . . . , n − 1 (rispetta la funzione di transizione)

Diciamo che una computazione accetta la parola w se: rn ∈ F (la computazione *termina in uno stato finale*) A causa del nondeterminismo, ci può essere più di una computazione per ogni parola.

Un NFA *accetta* la parola se esiste una computazione che accetta w (quindi arrivo alla fine correttamente); un NFA *rifiuta* la parola se tutte le computazioni la rifiutano. Le computazioni sono rappresentabili come albero, sulla base dell’esempio di prima, definendo una particolare parola.

A tale scopo, vediamo in dettaglio esempi fatti dal prof e presenti sulle slide come esercizio.

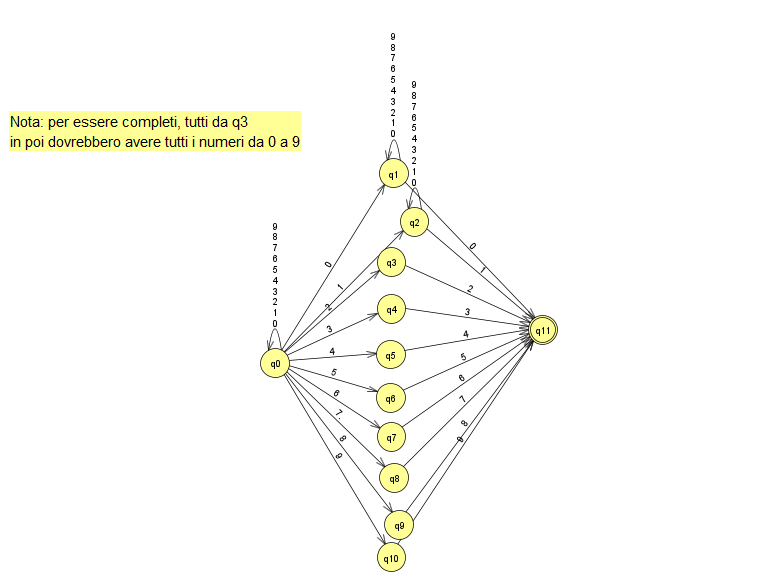
Differenza riassuntiva tra NFA e DFA:

Immagine che contiene testo

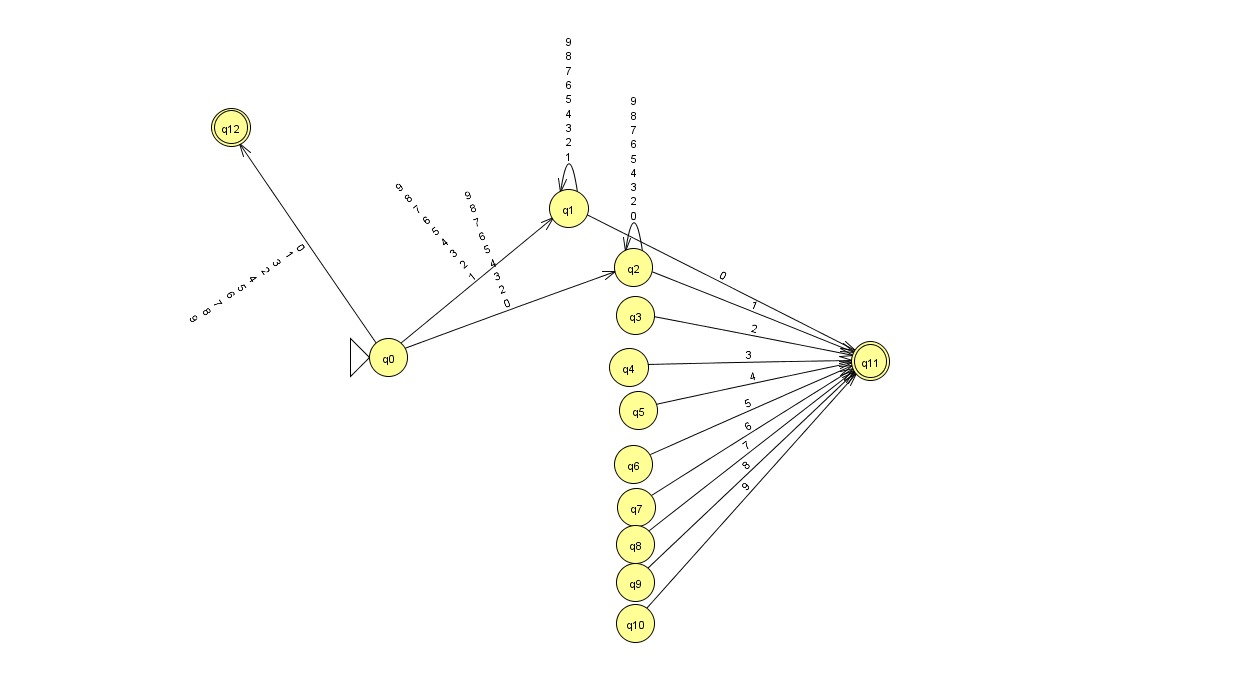
Descrizione generata automaticamente

*Primo esercizio:*

*(L’insieme delle parole sull’alfabeto {0, 1, . . . , 9} tali che la cifra finale sia comparsa in precedenza)*



*Secondo esercizio:*

*(L’insieme delle parole sull’alfabeto {0, 1, . . . , 9} tali che la cifra finale non sia comparsa in precedenza)*

*Terzo esercizio:*

*(L’insieme delle parole di 0 e 1 tali che esistono due 0 separati da un numero di posizioni multiplo di 4 (si considera che 0 sia multiplo di 4) )*

Immagine che contiene testo, orologio

Descrizione generata automaticamente

***09/03/2022: Equivalenza di DFA e NFA***

NFA e DFA sono in grado di riconoscere gli stessi linguaggi in quanto, per ogni NFA *N* c’è un DFA *D* tale che L(D) = L(N) e viceversa. Un esempio schematico:

Immagine che contiene testo, orologio

Descrizione generata automaticamente

Rappresentato ad albero:

{q0}

| 1

{q0}

| 0

{q0, q1}

| 0

{q0, q1}

| 1

\*{q0, q2}

1

0

0

1

La costruzione è definita equivalentemente come *costruzione a sottoinsiemi*.

Dunque, dato un NFA *N* = (QN, Σ, q0, δN, FN) costruiremo un DFA *D* = (QD, Σ, S0, δD, FD) tale che L(D) = L(N)

Immagine che contiene testo

Descrizione generata automaticamente

Prende quindi in input un simbolo e ritorna come output uno stato del DFA, costruendo una funzione di transizione che dimostri che posso percorrere tutte le possibili strade.

Matematicamente, si rappresenta l’unione dei singoli stati scorrendo tutti gli stati che stanno in S e mettendo almeno uno di questi come stato destinazione.

Immagine che contiene testo, orologio

Descrizione generata automaticamente

Sotto riportato gradualmente come costruire la tabella corretta di transizione.

Si segnala che nella quinta riga ci sarebbe l’unione di q0 e q1 con l’insieme vuoto, che viene trascurato.

Nelle righe dove ci sono due stati si considerano gli stato da entrambe le parti per 0 e 1, piccola sintesi di come ragionarla.

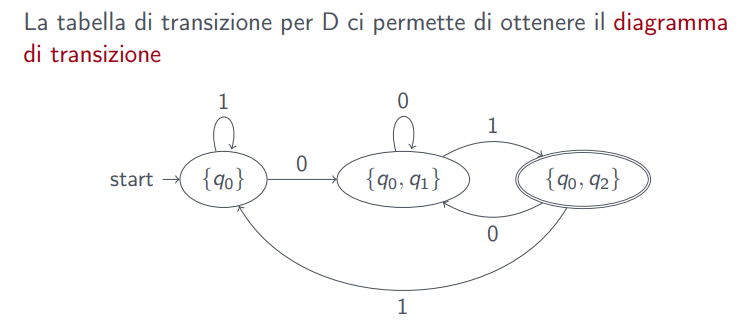
Note utili:

* *L’asterisco indica gli stati finali e negli stati possibili rientra anche lo stato vuoto. Questo perché, al contrario dei DFA, dove deve sempre esserci una transizione, non è obbligatorio negli NFA e per questo potrebbe non esistere una transizione verso altri stati.*
* *Inoltre, posso intuire i risultati delle transizioni composte da più stati (per es. {q0, q1}), essendo degli insiemi, partendo da quelli precedenti.*
* *Infine, nella scrittura degli stati di transizione degli stati composti, si fa l’unione degli stati visti singolarmente sull’insieme degli stati composti*

Immagine che contiene testo, tavolo

Descrizione generata automaticamente

Si nota che l’insieme degli stati, insieme di tutti i possibili sottoinsiemi, cresce esponenzialmente ed il prezzo che si paga nella trasformazione da NFA a DFA è un costo notevole.



Si nota che alcuni stati possono essere inutili, quindi non raggiungibili dallo stato iniziale.

In questo caso vi sono solo tre stati non raggiungibili e gli altri possono essere omessi.

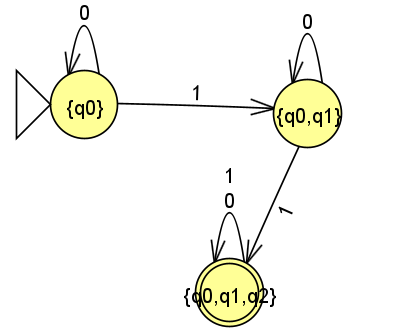


Immagine che contiene tavolo

Descrizione generata automaticamente

1. Qui sopra a destra ho rappresentato l’automa DFA, segue la sua tabella di riferimento:

|  |  |  |
| --- | --- | --- |
| Qd | 0 | 1 |
| Insieme vuoto | Insieme vuoto | Insieme vuoto |
| {q0} | {q0} | {q0, q1} |
| {q1} | {q1} | {q0, q2} |
| \*{q2} | {q1, q2} | {q0, q1, q2} |
| {q0, q1} | {q0, q1} | {q0, q1, q2} |
| {q0, q1} | {q0, q1, q2} | {q0, q1, q2} |
| {q0, q1} | {q0, q1} | {q0, q1, q2} |
| \*{q0, q2} | {q0, q1, q2} | {q0, q1, q2} |
| \*{q1, q2} | {q1, q2} | {q0, q1, q2} |
| \*{q0, q1, q2} | {q0, q1, q2} | {q0, q1, q2} |

1. Accetta come linguaggio tutte le stringhe che contengono almeno due 1

*Nota*: l’automa si ottiene, guardando la tabella, per unione delle precedenti. Parto da q0 e interpreto la prima riga. Sono in {q0,q1} e da lì faccio l’unione dei risultati di q0 e q1 (q0 va a 0, q1 va a 0, quindi unisco e q0,q1 andrà a 0 su sé stesso). Poi vado verso q0,q1,q2: questo perché q0 va ad 1 verso q0,q1 mentre q1 va ad 1 con q0,q2. Unisco i due risultati ed ottengo esattamente q0,q1,q2. A questo punto, unisco q0,q1,q2 (avendo q0 che va a q0 per 0, q1 che va a q1 per 0 e q1 che va a 0 per q1, q2) e i loro risultati portano proprio a q0,q1,q2. Ad 1 si nota che ci sta già una transizione verso q0,q1,q2 quindi non serve metterne altre e sarà quella finale.

Immagine che contiene testo

Descrizione generata automaticamente

Questo automa fa parte della domanda presente anche sotto che riassumo:

*“Prova a costruire un DFA partendo dall’NFA (foto, ndr) che riconosce il linguaggio di tutte le stringhe con un 1 nella terza posizione dalla fine”*

*Immagine che contiene testo, orologio

Descrizione generata automaticamente*

Ecco quindi il DFA di riferimento costruito.

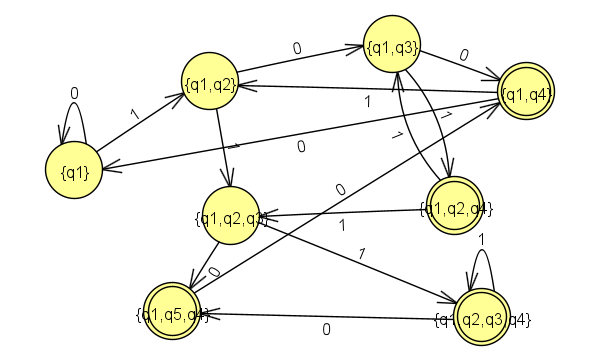


Immagine che contiene testo, orologio

Descrizione generata automaticamenteSi può anche usare JFLAP per convertire un NFA ad un DFA e verificare in autonomia se è corretta l’implementazione con tutti gli stati come descritto sopra.

Rappresentiamo su JFLAP l’automa NFA qui a destra:

Clicchiamo poi nel menù *Convert* e successivamente *Convert to DFA.*

Nella schermata che si apre *NFA to DFA* avremo un menù di questo tipo in alto, assieme a *Complete* e *Done.*



A parte la selezione si ha:

* il secondo pulsante, *Expand Group on Terminal*, da cui si apre una finestra di contesto che chiede di immettere dei valori all’automa specifici per espanderne gli stati e formare tutte le transizioni.

A questo punto clicco su un automa trascinando fuori con il cursore e sarà chiesto il valore con cui espandere stati/transizioni e l’elenco degli stati su cui operare (scritti senza spazi tra le virgole).

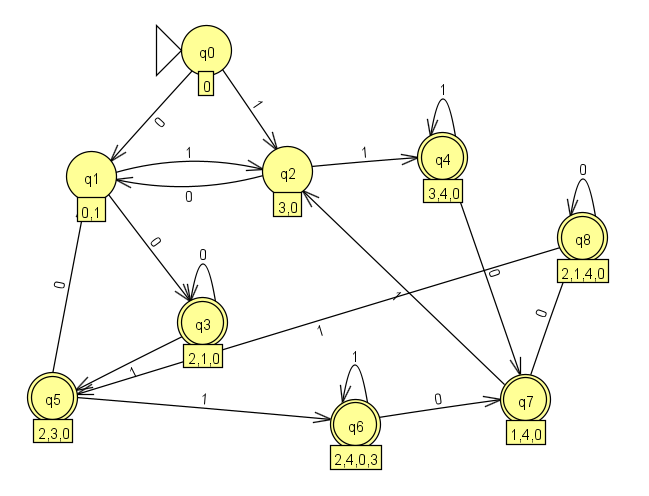
Se immetto stati errati, JFLAP dirà *The list of states is incorrect.*

* il terzo pulsante, *State Expander*, che permette, cliccando su uno stato, la creazione automatica di transizioni verso gli altri stati esistenti. Fa tutto lui in questo caso.

Cliccando sul pulsante *Done* si verifica se l’NFA corrispondente è stato costruito con tutti gli stati possibili. JFLAP lo segnala aprendo l’automa in un’altra finestra e dicendo *The NFA is fully built*.

Segnalerà altrimenti il numero di transizioni mancanti.

In questo esempio ecco quindi il DFA corrispondente al precedente NFA:



*Domande interattive della lezione:*

1. L'NFA mostrato nella figura riconosce il linguaggio di tutte le stringhe con un 1 nella terza posizione dalla fine. Prova a costruire un DFA che riconosce lo stesso linguaggio: quanti stati possiede?

Immagine che contiene testo, orologio

Descrizione generata automaticamente

*Risposta:* Si risponde costruendo tutte le transizioni da tutte le combinazioni stati quindi *2^4 = 16*

*Per capire gli stati effettivamente utili non si fa la tabella di transizione (qui trovi tutti gli stati, quindi al completo), ma il diagramma di transizione. Qui gli stati utili sono 8.*

1. Qual è l'OUTPUT della funzione di transizione di un NFA?

*Risposta*: Un insieme di stati

1. In un NFA, quante transizioni con lo stesso simbolo possiamo avere in uno stato?

*Risposta*: Un numero a piacere, zero compreso.

1. Come si rappresenta la computazione di un NFA?

*Risposta*: Con un albero di possibili transizioni

1. Un NFA accetta una stringa quando….?

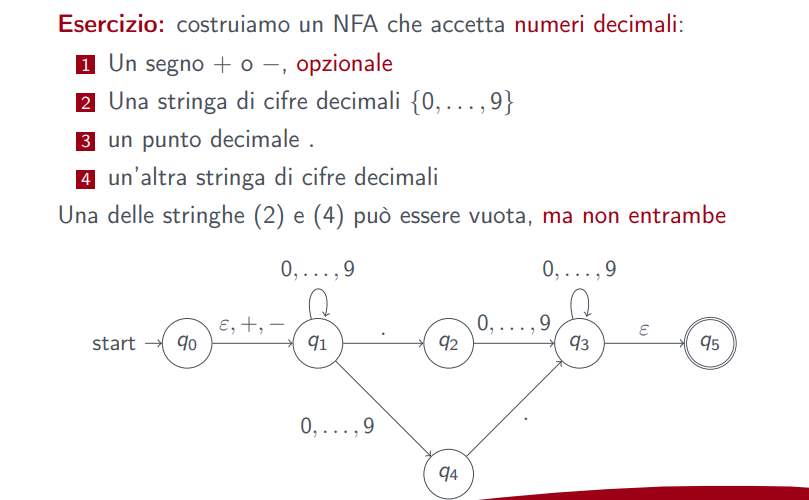
*Risposta*: Esiste una computazione che termina in uno stato finale

1. Un NFA rifiuta una stringa quando….?

*Risposta*: Nessuna computazione termina in uno stato finale

***10/03/2022: Automi a stati finiti con epsilon-transizioni***

Abbiamo quindi automi a stati finiti non deterministici, che sono appunto gli automi ad ε-transizioni. Iniziamo con un esempio:



Finora tutti gli automi visti eseguivano una transizione per ogni singolo input; altrimenti possiamo indicare più simboli per non consumare tutto l’input, combinandone vari per una singola transizione.

Nota: se avessi messo nell’immagine sopra q3 come stato finale, poteva andare bene lo stesso.

La computazione che indica che potrebbe non avvenire una transizione viene data da ε.

*Qui la rappresentazione ad albero*

*1*

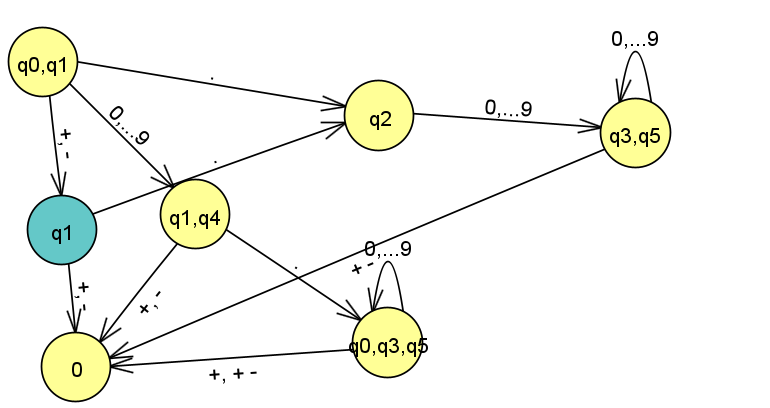
*0*

*.*

*2*

*3*

L’automa chiesto dall’immagine della slide mostrata è il seguente (si noti che nelle frecce da {q0,q1} verso q2 e da {q3,q5} verso 0.



In pratica le ε-transizioni permettono di andare oltre i possibili valori presenti non consumando ulteriori transizioni ed eventualmente proseguendo oltre la fine.

La lista delle epsilon-chiusure in questo caso è data da:

ECLOSE (q0) = {q0,q1}

ECLOSE (q1) = {q1}

ECLOSE ({q0, q4}) = {q1, q4}

ECLOSE (q2) = {q2}

ECLOSE ({q2,q5}) = {q2,q3,q5}

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene tavolo

Descrizione generata automaticamente

È possibile eliminare le transizioni, ma noi non lo vedremo; modificheremo invece la conversione da NFA ad un DFA per poi realizzare una trasformazione da ε-NFA a ε-DFA.

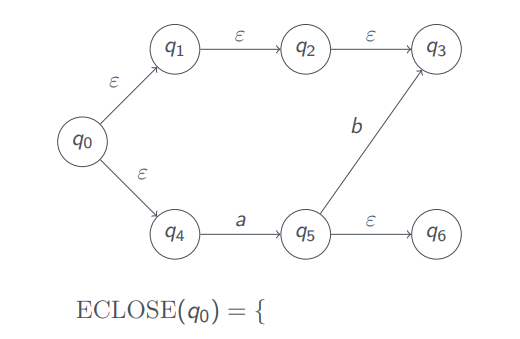
L’eliminazione all’interno del DFA degli stati raggiungibili/possibili rappresenta l’insieme degli stati ottenibili con le operazioni date. Calcoliamo quindi l’insieme delle ε-transizioni, per poi effettuarne l’eliminazione induttivamente.

Questa operazione viene definita come *ε-chiusura*.

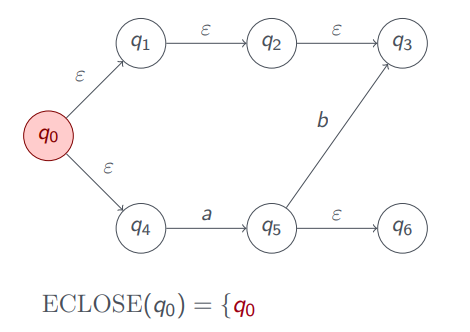
Immagine che contiene testo

Descrizione generata automaticamente

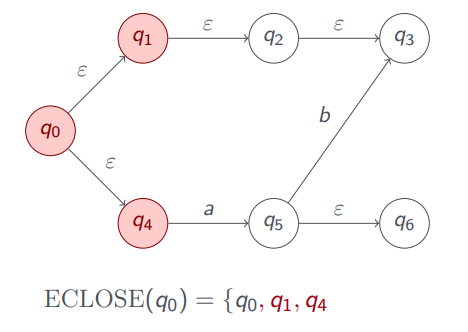
Calcoliamo quindi induttivamente la ε-chiusura partendo da:



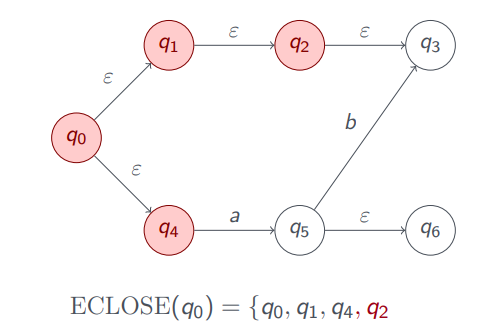
Ad esempio, partendo da q0, avremo:



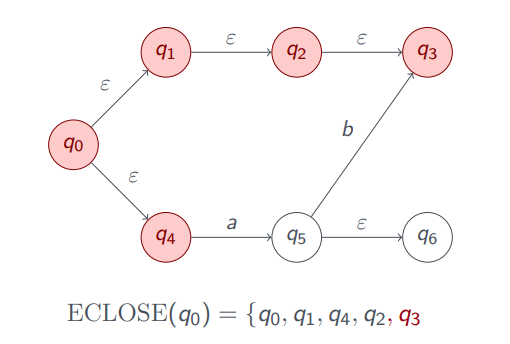
Seguono poi le altre possibili:



Verifico poi se posso applicare il caso induttivo su q5 e q2; quindi proseguo solo su q2, perché dalla tabella di transizione sopra si vede che su q5 muore (dead state):



E così via quindi, inoltre concludendo:



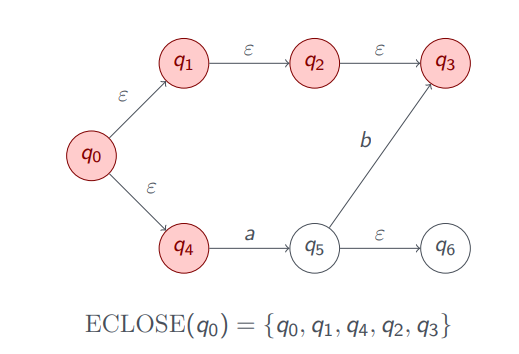


Immagine che contiene testo

Descrizione generata automaticamente

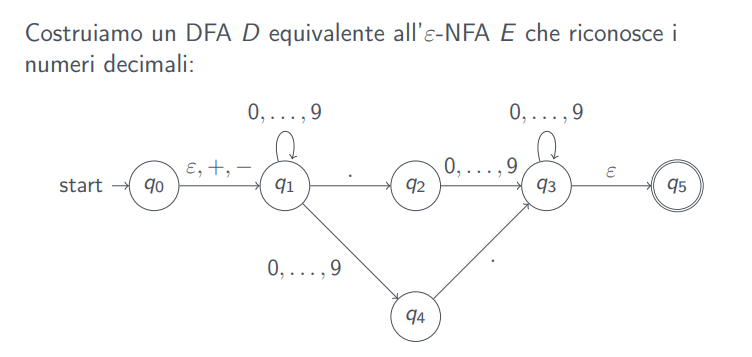
Definiamo poi come sempre i membri matematicamente, prendendo quindi in sintesi ogni singola ε-transizione, realizzandone poi la chiusura e rappresentando gli stati finali, qualora ci sia almeno uno stato finale, realizzando quindi ogni singolo stato ed inoltre quelli raggiungibili.

Anche in questo caso l’insieme degli stati possibili risulta essere esponenziale.

Immagine che contiene testo

Descrizione generata automaticamente

Andiamo quindi ad un esempio pratico:



Si applicano le regole delle ε-chiusure, descrivendo quali sono gli stati iniziali, considerando anche lo stato stesso q0 nel primo caso e poi prendendo più stati nel caso in cui si abbia ε tra questi.

Immagine che contiene testo

Descrizione generata automaticamente

Per arrivare a questo automa usamo la tabella presente a pagina 16. La ε-chiusura mi permette di stabilire lo stato iniziale utile (quindi q0,q1) e poi da quello sviluppare tutti gli altri stati per mezzo della tabella citata. Gli stati unione anche di stati precedenti sono dati dalle ε, dove si può notare che q0 è ε di q1, da cui sono uniti nello stato iniziali, simlmente anche q3 con q5. q1,q4,q5 si forma dal fatto di avere q5 come stato finale unico.

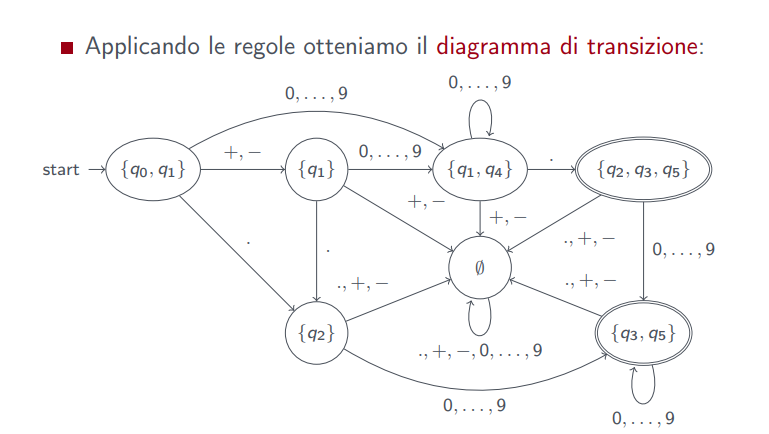


Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

Le espressioni sui linguaggi regolari prevedono una serie di operazioni insiemistiche, descritte di fianco:

*Domande interattive:*

1. Una epsilon-transizione è:

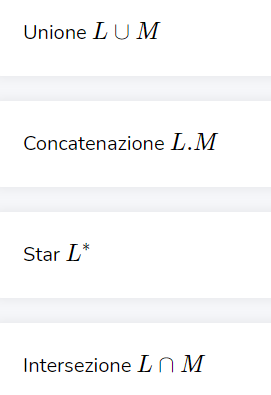
*Risposta*: Una transizione che non consuma l’input

1. NFA ed ε-NFA riconoscono la stessa classe di linguaggi?

*Risposta*: Vero

1. Siano *L* e *M* linguaggi regolari. Per ognuna delle operazioni seguenti, indica se puoi costruire un automa che riconosce il linguaggio. Puoi scegliere più di una risposta.

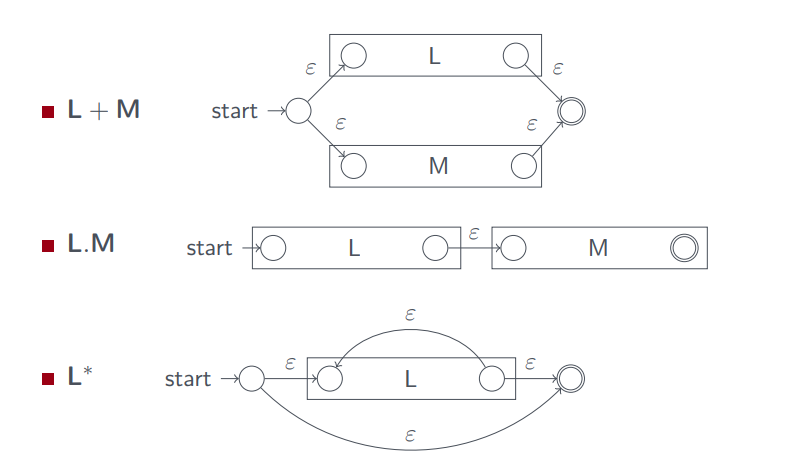
Risposte:



***Immagine che contiene testo

Descrizione generata automaticamente***

Ecco esempi possibili di automi (risposta quindi anche al file di esercizi).

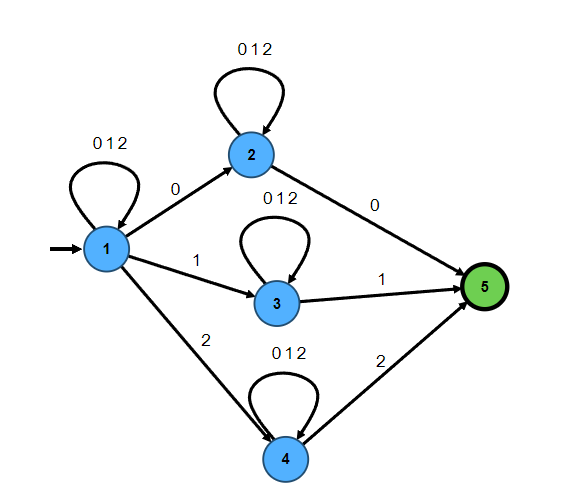


***Sezione esercizi***

***Primo set di esercizi Automata Tutor***

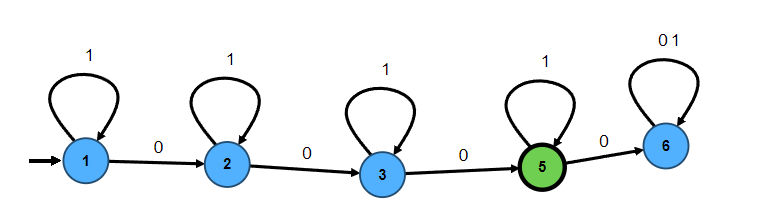
1. *Automa NFA con alfabeto {0,1,2} che ha come linguaggio:*

*la cifra finale sia comparsa in precedenza*



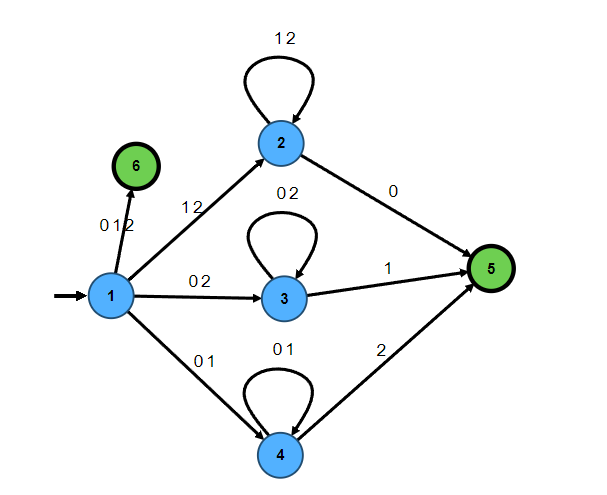
1. *Automa DFA con alfabeto {0, 1} che ha come linguaggio:*

*tutte e sole le stringhe che contengono esattamente tre zeri (anche non consecutivi)*



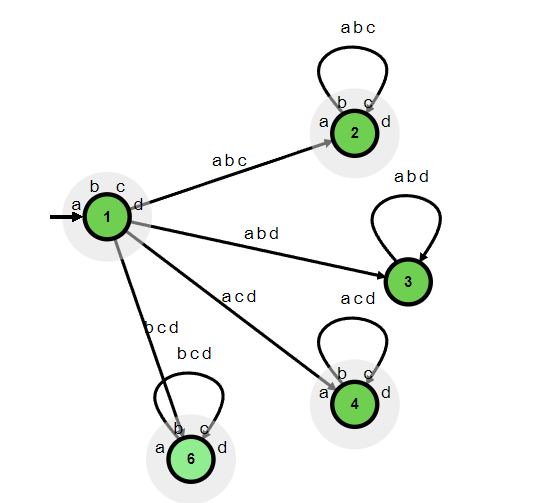
1. *Automa NFA con alfabeto {0,1,2} che ha come linguaggio le stringhe in cui:*

*la cifra finale non sia comparsa in precedenza*



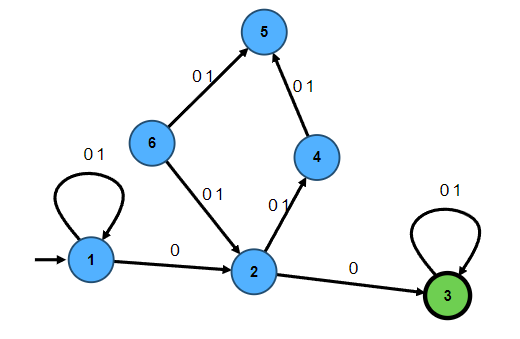
1. *Automa NFA con alfabeto {a, b, c, d} che ha come linguaggio le stringhe in cui:*

*uno dei simboli dell’alfabeto non compare mai*

**

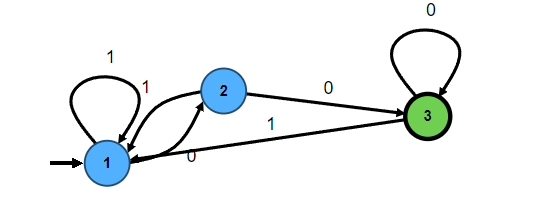
1. *Automa NFA con alfabeto {0, 1} che ha come linguaggio le stringhe in cui:*

*esistono due 0 separati da un numero di posizioni multiplo di 4*



1. *Automa DFA con alfabeto {0, 1} che ha come linguaggio:*

*tutte e sole le stringhe che terminano con 00*



1. *Automa DFA con alfabeto {0, 1} che ha come linguaggio:*

*tutte e sole le stringhe che cominciano o finiscono con 01 (o entrambe le cose)*

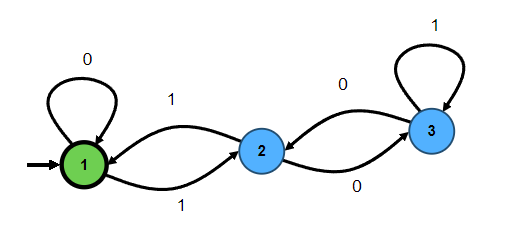
*Immagine che contiene dispositivo

Descrizione generata automaticamente*

1. *Automa DFA con alfabeto {0, 1} che ha come linguaggio:*

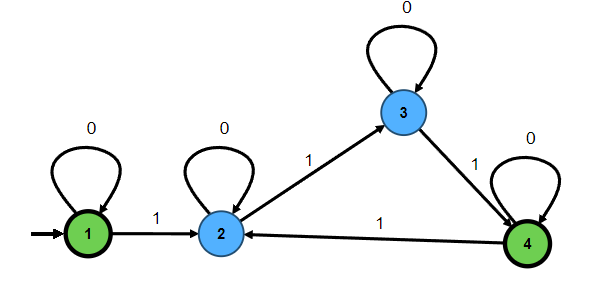
*tutte le stringhe che rappresentano la codifica binaria di un numero multiplo di 3. La stringa vuota non rappresenta nessun numero.*

*Dà 8/10 però è necessario mettere come stato finale il primo.*

**

1. *Automa DFA con alfabeto {0, 1} che ha come linguaggio:*

*tutte e sole le stringhe che un numero di 1 multiplo di 3*

**

1. *Automa DFA con alfabeto {0, 1} che ha come linguaggio:*

*tutte e sole le stringhe con un numero pari di “a” e di “b”*

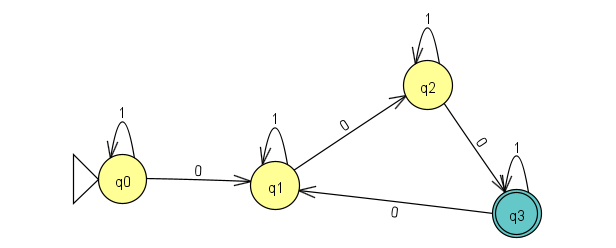
Immagine che contiene filo, dispositivo

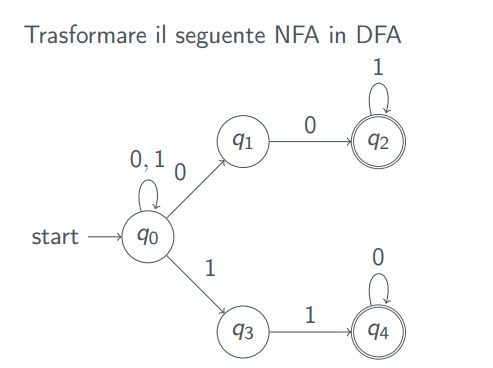
Descrizione generata automaticamente

*Dal file – Esercizi01 (uno dei due del file è stato fatto qui sopra)*

Immagine che contiene testo

Descrizione generata automaticamente



*Conversioni da NFA a DFA (esercizi delle slide)*

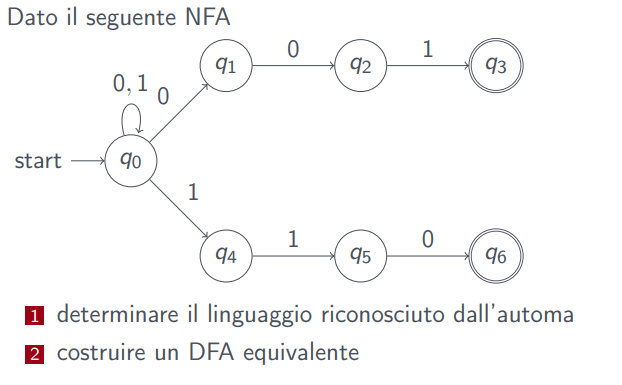
Per poterlo fare:

* prima si costruisce la tabella di transizione (con tutti gli stati effettivamente utili, gli stati unione di altri si considerano solo quelli che servono)
* successivamente si costruisce l’automa sulla base della tabella

|  |  |  |
| --- | --- | --- |
| Qd | 0 | 1 |
| ∅ | ∅ | ∅ |
| {q0} | {q0, q1} | {q0, q3} |
| {q1} | {q2} | ∅ |
| \*{q2} | ∅ | {q2} |
| {q3} | ∅ | {q4} |
| \*{q4} | {q4} | ∅ |
| {q0, q1} | {q0, q1} | {q0} |
| {q0, q3} | {q0, q1} | {q3} |

Immagine che contiene testo, orologio

Descrizione generata automaticamente

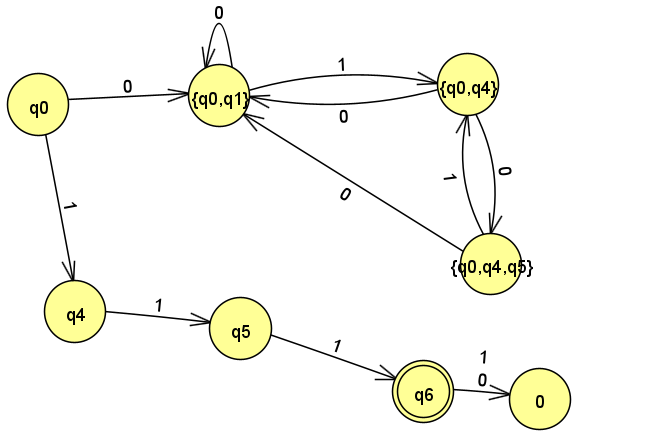


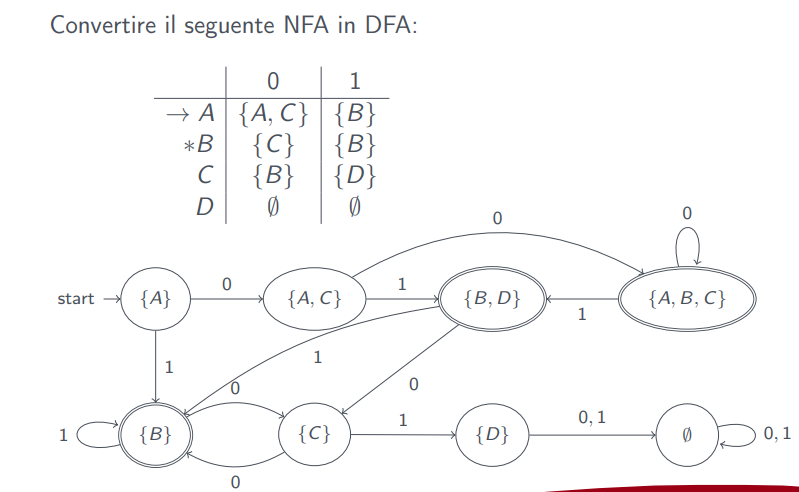
1) L’automa riconosce come linguaggio:

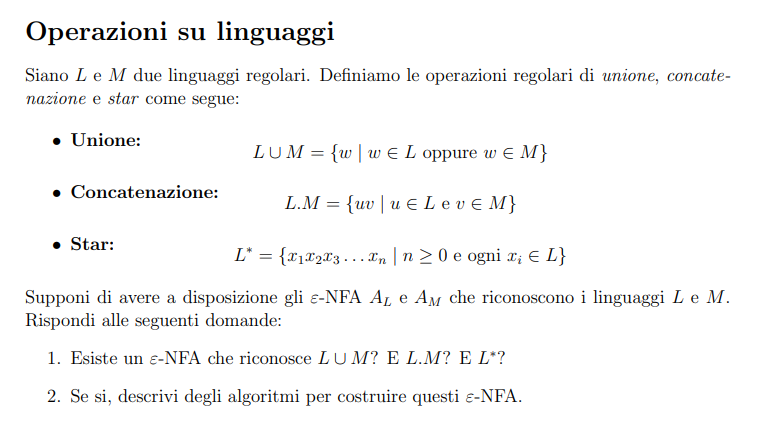
Un alfabeto che comprende delle stringhe con almeno due 0 e due 1

2)

|  |  |  |
| --- | --- | --- |
| Qd | 0 | 1 |
| ∅ | ∅ | ∅ |
| {q0} | {q0, q1} | {q4} |
| {q1} | {q2} | ∅ |
| {q2} | ∅ | {q3} |
| \*{q3} | ∅ | ∅ |
| {q4} | ∅ | {q5} |
| {q5} | {q6} | ∅ |
| \*{q6} | ∅ | ∅ |
| {q0, q1} | {q0, q1} | {q0, q4} |
| {q0, q4} | {q0, q1} | {q0, q4, q5} |
| {q0, q4, q5} | {q0, q1} | {q0, q4} |

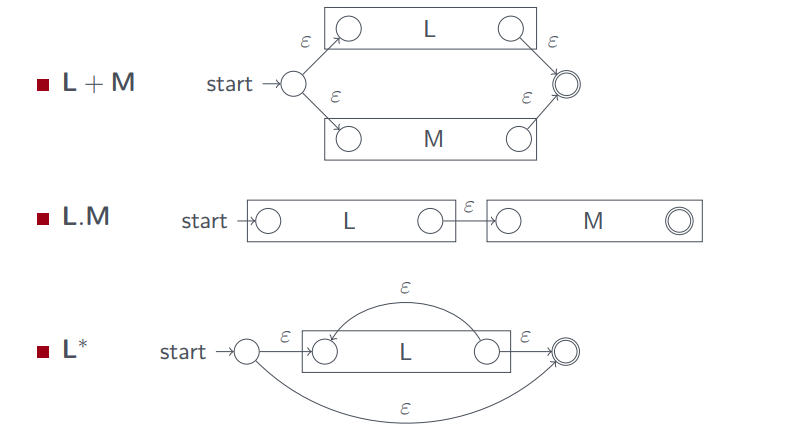




*Tratto dal file 03-esercizi e visto in classe:*

1. Sì a tutte e 3 le domande
2. Intuitivamente (poi seguono gli automi di riferimento):

* per l’unione basta avere uno stato iniziale comune ed una biforcazione verso due stati
* per la concatenazione si avrà uno stato iniziale seguito da uno stato finale oppure uno non finale
* per lo star, basta avere tutte le combinazioni da e verso altri stati



*Esercizio con ε-chiusure (da slide 2)*

Immagine che contiene testo

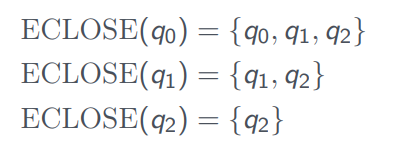
Descrizione generata automaticamente

1)

Immagine che contiene orologio

Descrizione generata automaticamente

2)



3) Si ottiene l’automa anche qui applicando le regole e il ragionamento visto fino ad ora, modificando gli stati sulla base delle ε-chiusure.

