*Scritti da Gabriel: gli appunti comprendono anche le domande a quiz interattive*

*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*, che sarebbe niente altro che 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:

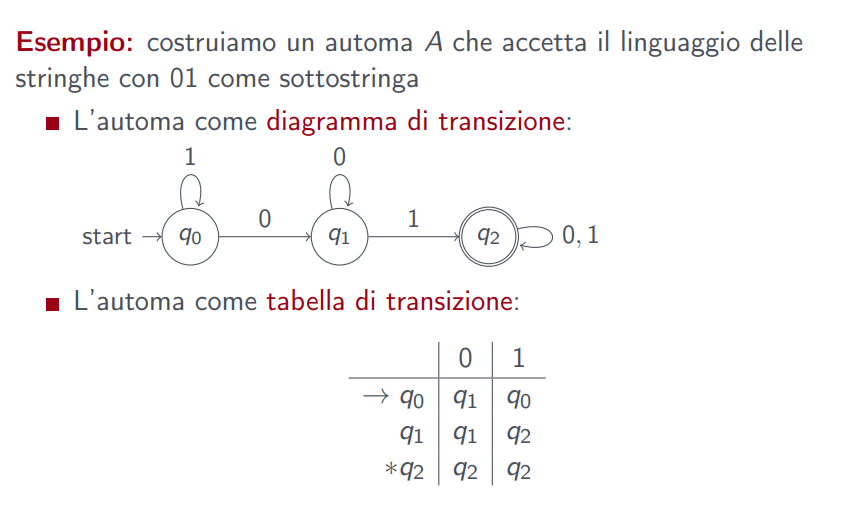
* Q è un insieme finito di *stati*
* Σ è un *alfabeto finito* (= simboli in input)
* δ è una *funzione di transizione* (q, a) → q’
* q0 ∈ Q è lo *stato iniziale*
* F ⊆ Q è un insieme di *stati finali*

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 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.

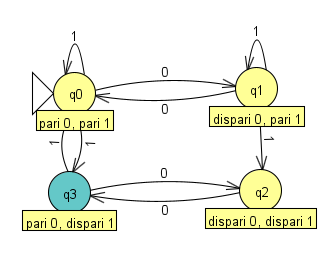


Sulla barra del menù, cliccando *Input* e poi su *Step by state* è possibile, dando un certo input come stringa, verificare il comportamento effettivo dell’automa. Ciò avviene se è stato impostato almeno uno stato iniziale. Si entra quindi nello stato Simulate e si avanza step by step.

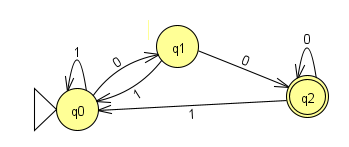
Se il comportamento finale del testo è verde significa che l’automa impostato è corretto; se è rosso, ci sta qualche problema o l’automa è non deterministico (si veda sotto). Per uscire dalla scheda di simulazione attuale e tornare nell’editor si preme la freccia X in alto a dx.

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).

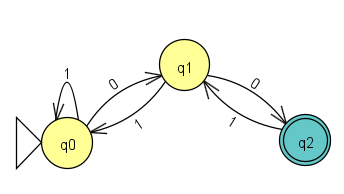
- Insieme di tutte e sole le stringhe con un numero pari di zeri e un numero pari di uni



- Insieme di tutte le stringhe che finiscono con 00



Un’idea non corretta su questo esempio sarebbe:



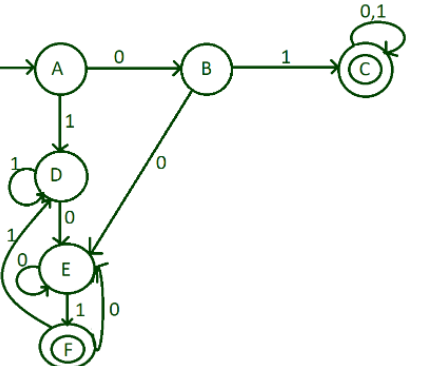
Di fatto è sbagliata come idea perché non c’è una fine vera e propria agli input esistenti; si nota che i due zeri ci sono, ma nel caso avessi un uno e poi uno zero, ecco che si potrebbe generare un loop per cui la situazione non si sblocca mai.

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

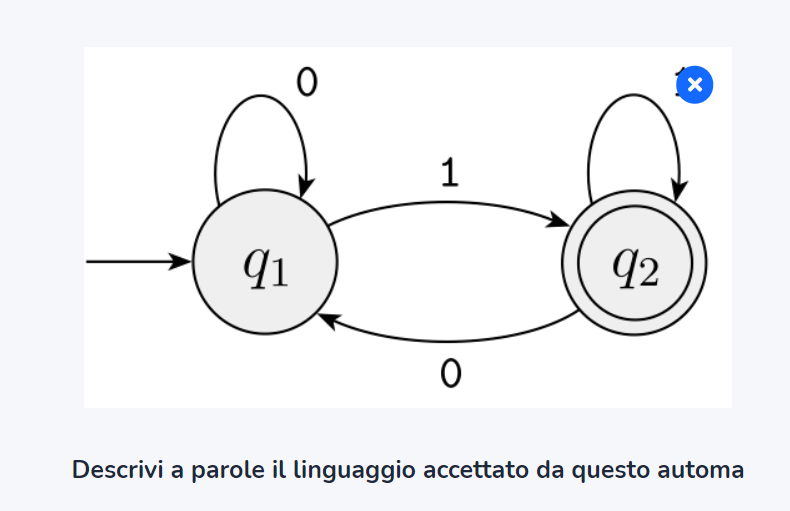
Immagine che contiene testo, orologio

Descrizione generata automaticamente

- Insieme delle stringhe che cominciano o finiscono (o entrambe le cose) con 01

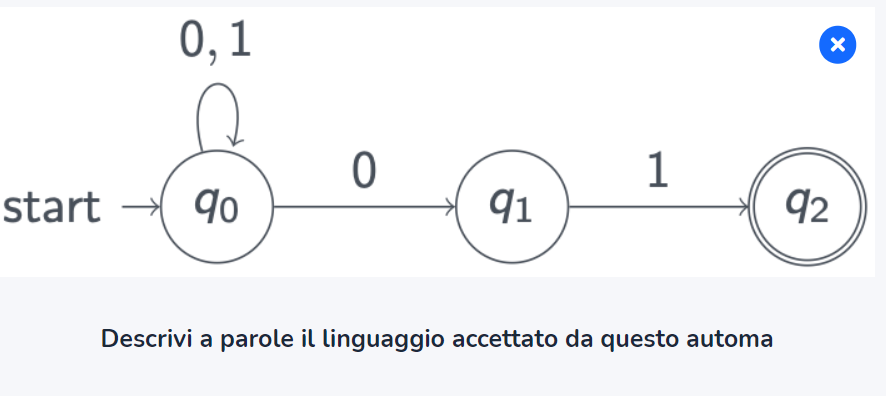


Quale linguaggio accetta il seguente automa? (il numero nascosto dalla X è un 1 per la cronaca)



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

Quale linguaggio accetta il seguente automa?



Risposta:

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:

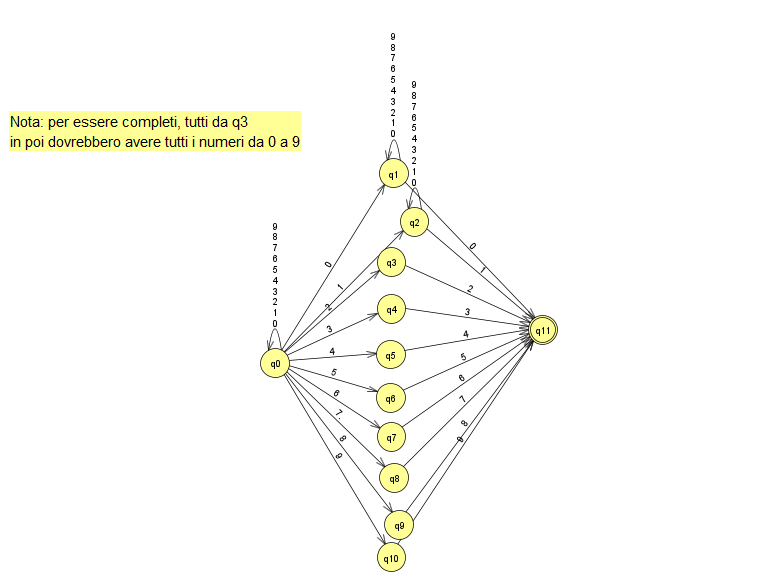
1. r0 = q0 (inizia dallo stato iniziale)
2. δ(ri ,wi+1) = ri+1 per ogni i = 0, . . . , n − 1 (rispetta la funzione di transizione)

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.

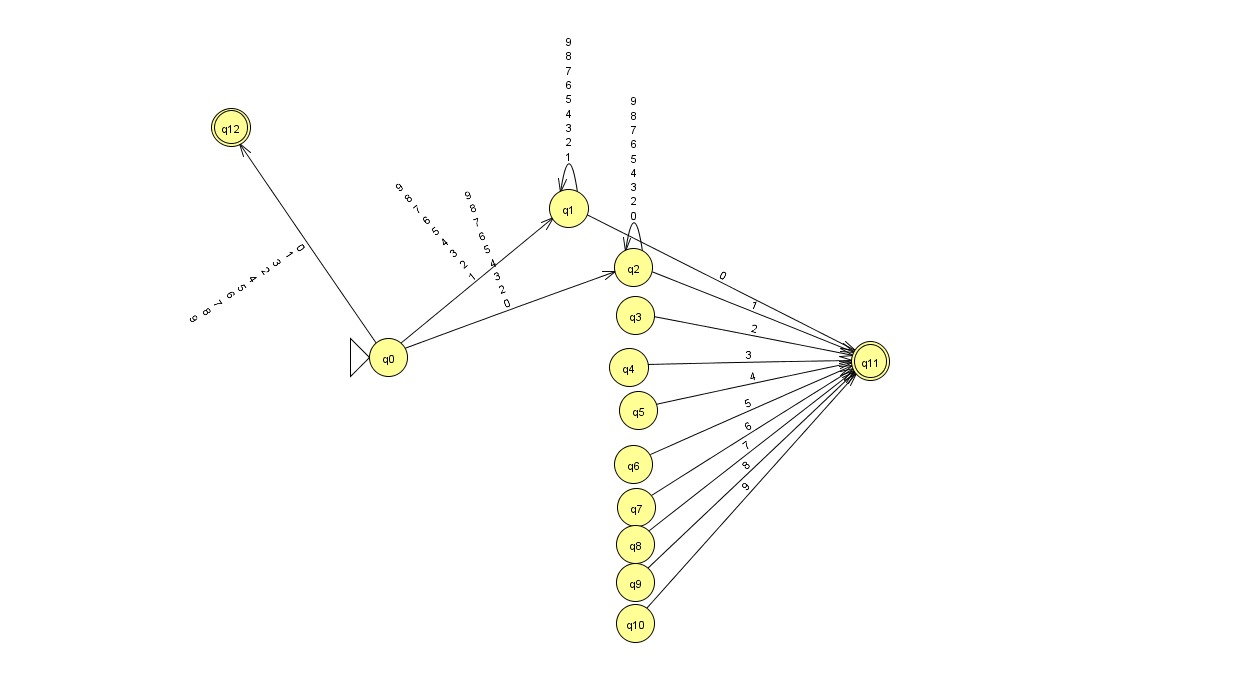
Un NFA *accetta* la parola se esiste una computazione che accetta w; un NFA *rifiuta* la parola se tutte le computazioni la rifiutano. Le computazioni sono rappresentabili come albero, sulla base dell’esempio di prima, definendo una particolare parola.

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

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 (0 è un multiplo di 4))

