Fondamenti di Programmazione: AUTOMI E LINGUAGGI FORMALI Corso di Laurea in MATEMATICA a.a. 2021/2022

Chiara Bodei, Roberta Gori, Damiano De Francesco Maesa

Dipartimento di Informatica
chiara.bodei@unipi.it

Testi di riferimento

- **Libro di testo principale**: *Automi, linguaggi e calcolabilità*, J. E. Hopcroft, R. Motwani, and J. D. Ullman, Addison-Wesley, 2003.
- **Altro libro**: *Compilers: principles, techniques, and tools*, A.H. Aho, M. S. Lam, R. Sethi, J. D. Ullman, Pearson/Addison-Wesley, 2007 (seconda edizione).
- I lucidi su questa parte del corso sono basati su quelli della Prof. Francesca Rossi (basati a loro volta su quelli di Gösta Grahne e David Ford).

Perché studiare gli automi e i linguaggi formali?

- Le espressioni regolari sono usate in molti sistemi, ad esempio in UNIX: `rm hello*.c`
- Gli automi sono utilizzati per modellare protocolli e circuiti elettronici.
- Le grammatiche sono usate per descrivere la sintassi dei linguaggi di programmazione.

Cosa studieremo

- Linguaggi regolari
  - Loro descrizione tramite automi finiti deterministici e non deterministici, espressioni regolari.
  - Proprietà di decisione e di chiusura
- Linguaggi liberi
  - Loro descrizione tramite grammatiche libere
  - Proprietà di decisione e di chiusura

Inoltre impareremo a trattare in modo formale i sistemi discreti e a familiarizzare con i modelli astratti.

Qual è il pattern?

Un robot lancia in aria una moneta e occorre indovinare se viene testa o croce. Sembra esserci tuttavia una sequenza, un pattern, ripetuta nel modo in cui la moneta appare. È un gioco truccato? Dati i seguenti risultati relativi ai tiri della moneta (t=testa, c=croce):

ttcttctttcctttttcctcccttttttcttt
cccttttcccttttttcctccccctcctccc
tttcctttcttttttttttcctttccccttt
ttcccccccc

qual è il pattern ricorrente dei risultati dei tiri?

Qual è il pattern?

ttc / ttc / ttt / cct / ttt / cct / ccc / ttt / ttc / ttt
ccc / ttt / ccc / ttt / ttt / cct / ccc / cct / cct / ccc
ttt / cct / ttc / ttt / ttt / ttt / cct / ttc / ccc / ttt
ttc / ccc / ccc

I primi due tiri ogni tre danno sempre lo stesso risultato

Automi a stati finiti

Gli automi a stati finiti sono usati come modello per

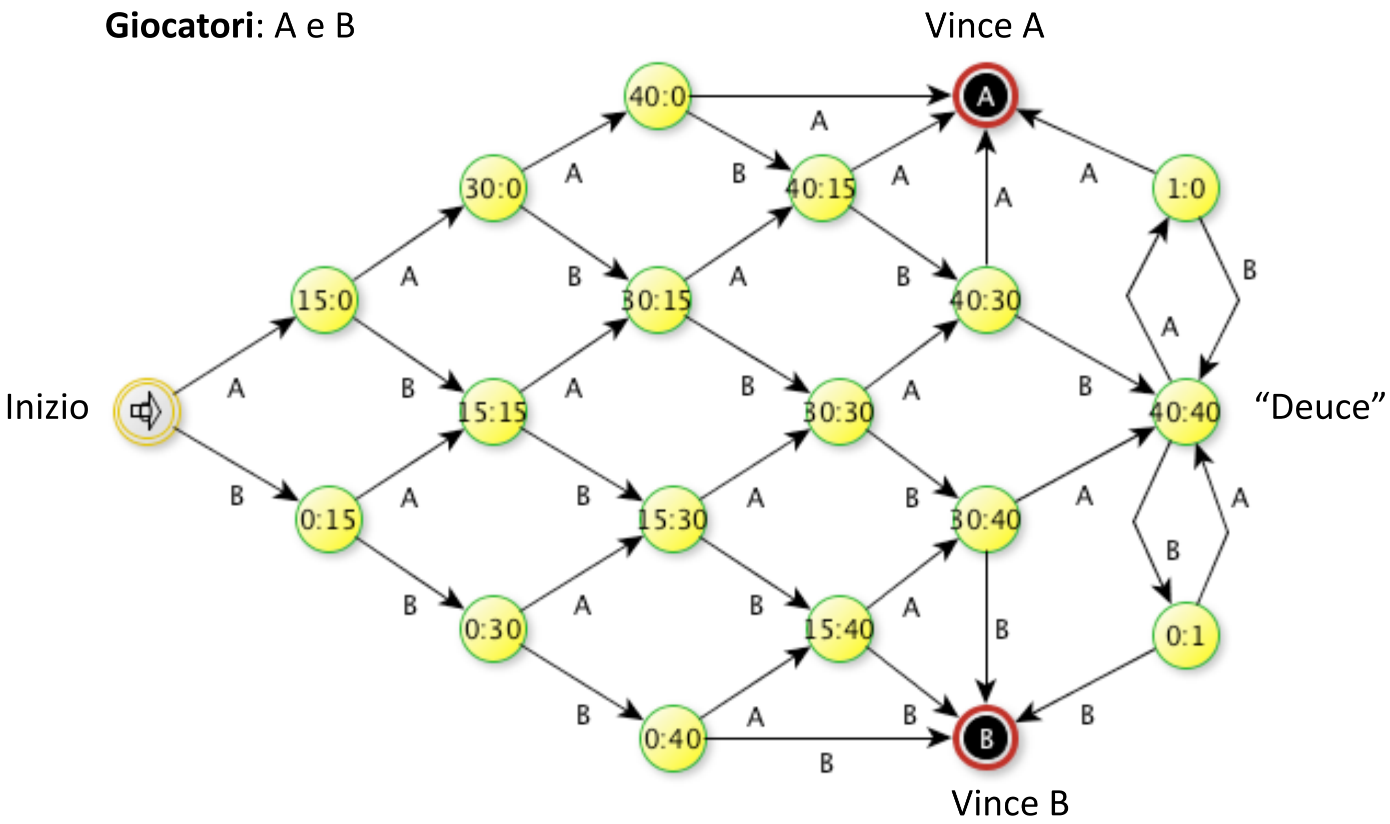
- Software per la progettazione di circuiti digitali.
- Analizzatori lessicali di un compilatore.
- Applicazioni per l'elaborazione di testi, ad esempio per la ricerca di parole chiave in un file o sul web.
- Software per verificare sistemi a stati finiti, come protocolli di comunicazione.

Automi a stati finiti (cont.)

Che cosa è un automa a stati finiti?

- È un sistema **formale**
- È un sistema in grado di ricordare solo un quantitativo finito di informazione.
- L'informazione viene rappresentata da stati.
- Gli stati cambiano in risposta agli input.
- È un sistema costituito da un insieme finito di stati e da regole che dicono come passare da uno stato all'altro, in risposta all'input.

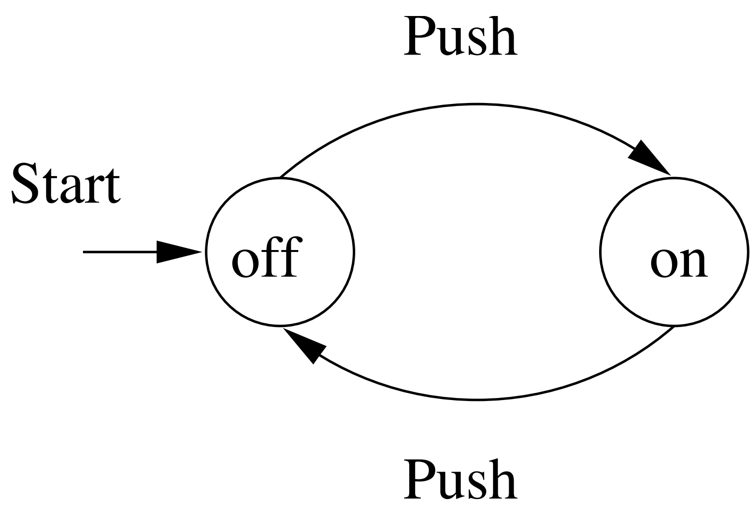
Esempio: automa a stati finiti per il game del tennis*



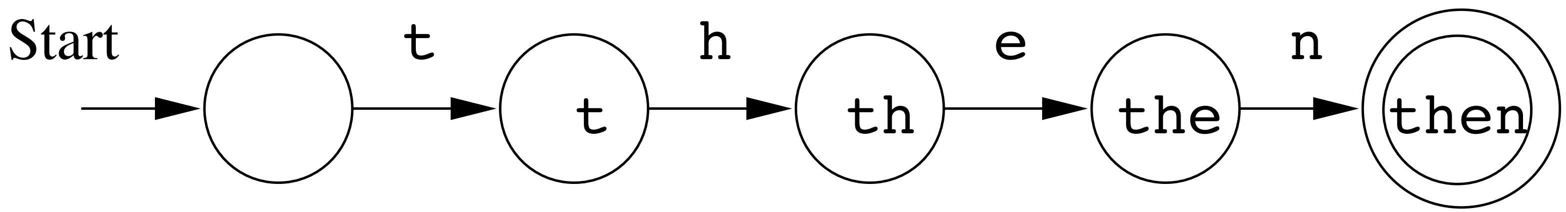
* Da "Automatic Java Code Generator for Regular Expression and Finite Automata" di Suejb Memeti, riprendendo un esempio di J. D. Ullman

Esempi

- Esempio: automa a stati finiti per un interruttore on/off



- Esempio: automa a stati finiti che riconosce la stringa `then`



Accettare input

- Data una sequenza (o stringa) di input, si comincia dallo stato iniziale e si seguono gli archi delle transizioni di simbolo in simbolo.
- L''input viene accettato quando dopo aver letto tutti i simboli in input si finisce in uno stato finale. Nell'esempio di prima, se la stringa in ingresso è proprio `then`, l'input è accettato, mentre non lo è se la stringa è `ten`.

Rappresentazioni strutturali

Ci sono vari modi di specificare una macchina

- **Grammatiche**:
  Una regola come $E \Rightarrow E + E$ specifica un'espressione aritmetica
  $Coda \Rightarrow Persona.Coda$
  dice che una coda è costituita da una persona seguita da una coda.
- **Espressioni regolari**:
  Denotano la struttura dei dati, per esempio:
  `'[A-Z][a-z]*[][A-Z][A-Z]'`
  è compatibile con (matches) `Ithaca NY`
  non è compatibile con `Palo Alto CA`
- Domanda: Quale espressione è compatibile con
  `Palo Alto CA`?

Concetti di base

- **Alfabeto**: Insieme finito e non vuoto di simboli
  - Esempio: $\Sigma = \{0, 1\}$ alfabeto binario
  - Esempio: $\Sigma = \{a, b, c, \ldots, z\}$ insieme di tutte le lettere minuscole
  - Esempio: Insieme di tutti i caratteri ASCII
- **Stringa:** Sequenza finita di simboli presi da un alfabeto Σ, per es. 0011001 (i simboli li scriviamo di seguito)
- **Stringa vuota:** La stringa con zero occorrenze di simboli da Σ
  - La stringa vuota è denotata con ϵ

Lunghezza di una stringa: Numero di posizioni per i simboli nella stringa.
$|w|$ denota la lunghezza della stringa w
$|0110| = 4, |\epsilon| = 0$
Potenze di un alfabeto: Σ^k = insieme delle stringhe di lunghezza k con simboli da Σ
Esempio: $\Sigma = \{0, 1\}$ $\Leftarrow$ 0 rappresenta un simbolo (in C '0')
$\Sigma^1 = \{0, 1\}$ $\Leftarrow$ 0 (in C "0")
$\Sigma^2 = \{00, 01, 10, 11\}$
$\Sigma^0 = \{\epsilon\}$
Domanda: Quante stringhe ci sono in Σ^3?

L'insieme di tutte le stringhe su Σ è denotato da Σ^*
$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots$
Ad esempio, $\{0,1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001...\}$ è l'insieme di tutte le stringhe composte di 0 e di 1.
Anche:
$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \cdots$
$\Sigma^* = \Sigma^+ \cup \{\epsilon\}$
Se Σ è finito, Σ^* è *numerabile*, ovvero esiste una corrispondenza biunivoca tra Σ^* e $\mathbb{N}$
Concatenazione: Se x e y sono stringhe, allora xy è la stringa ottenuta rimpiazzando una copia di y immediatamente dopo una copia di x
$x = a_1a_2 \ldots a_i, y = b_1b_2 \ldots b_j$
$xy = a_1a_2 \ldots a_ib_1b_2 \ldots b_j$
Esempio: $x = 01101, y = 110, xy = 01101110$
Nota: Per ogni stringa x

$$x\epsilon = \epsilon x = x$$

Linguaggi

Definizione: Se Σ è un alfabeto (finito), e $L \subseteq \Sigma^*$, allora L è un linguaggio

Esempi di linguaggi:

- L'insieme delle parole italiane legali
- L'insieme dei programmi C legali
- L'insieme delle stringhe che consistono di n zeri seguiti da n uni

$$\{\epsilon, 01, 0011, 000111, \ldots\}$$

Altri esempi

- L'insieme delle stringhe con un numero uguale di zeri e di uni

$$\{\epsilon, 01, 10, 0011, 0101, 1001, \ldots\}$$

- L_P = insieme dei numeri binari il cui valore è primo

$$\{10, 11, 101, 111, 1011, \ldots\}$$

- Il linguaggio vuoto $\emptyset$
- Il linguaggio $\{\epsilon\}$ consiste della stringa vuota

Nota: $\emptyset \neq \{\epsilon\}$

Nota: L'alfabeto Σ è sempre finito

Problemi

- La stringa w è un elemento di un linguaggio L?
- Esempio: Dato un numero binario, è primo = è un elemento di L_P?
- È $11101 \in L_P$? Che risorse computazionali sono necessarie per rispondere a questa domanda?
- Di solito non pensiamo ai problemi come delle decisioni sì/no, ma come qualcosa che trasforma un input in un output.
- Esempio: Fare il parsing di un programma C = controllare se il programma è corretto, e se lo è, produrre un albero di parsing.

Automi a stati finiti deterministici

Un DFA (Deterministic Finite Automaton, in italiano ASFD) è un formalismo per definire linguaggi che consiste di:

- Q è un insieme finito di *stati*
- Σ è un *alfabeto finito* ($=$ simboli in input)
- δ è una *funzione di transizione* $(q, a) \mapsto p$
- $q_0 \in Q$ è lo *stato iniziale*
- $F \subseteq Q$ è un insieme di *stati finali*

Un DFA è quindi una quintupla

$$A = (Q, \Sigma, \delta, q_0, F)$$

Funzione di transizione

- La funzione di transizione δ prende in ingresso uno stato e un simbolo e restituisce uno stato
- $\delta(q, a) = p$ denota che p è lo stato raggiunto a partire dallo stato q quando si riceve in input il simbolo a
- Per ogni stato deve essere sempre definito uno stato successivo per ogni simbolo di input (esistono anche stati pozzo).

Esempio

Esempio: Un automa che accetta

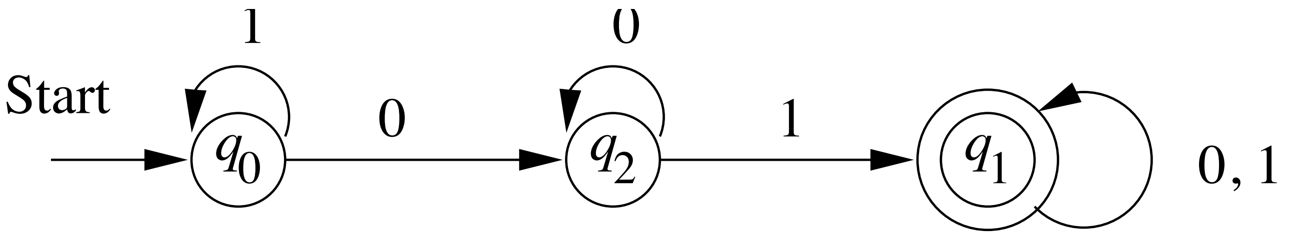
$$L = \{x01y : x, y \in \{0,1\}^*\}$$

è l'automa $A = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_1\})$, dove
q_0 rappresenta lo stato in cui non si è ancora visto 01
q_2 rappresenta lo stato in cui non si è visto 01, ma si è appena visto 0
q_1 rappresenta lo stato in cui si è visto 01

- L'automa come una *tabella di transizione*(stati/simboli di input):

	0	1
$\rightarrow q_0$	q_2	q_0
$\star q_1$	q_1	q_1
q_2	q_2	q_1

- L'automa come un *diagramma di transizione*:

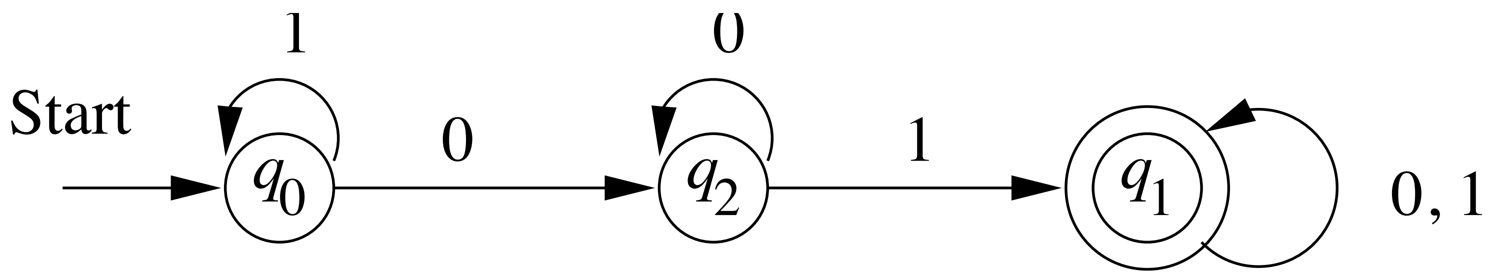


Accettazione

Un automa a stati finiti (FA) *accetta* una stringa $w = a_1 a_2 \cdots a_n$ se esiste un cammino nel diagramma di transizione che

1. Inizia nello stato iniziale
2. Finisce in uno stato finale (di accettazione)
3. Ha una sequenza di etichette $a_1 a_2 \cdots a_n$

Esempio: L'automa a stati finiti



accetta ad esempio la stringa 01101

Funzione di transizione estesa $\hat{\delta}$ e il linguaggio accettato

- La funzione di transizione δ può essere estesa a $\hat{\delta}$ che opera su stati e stringhe (invece che su stati e simboli)
- $\hat{\delta}(q, w) = p$ denota che p è lo stato raggiunto a partire dallo stato q quando si riceve in input uno ad uno i simboli $a_1 \ldots a_n$ che formano la stringa w

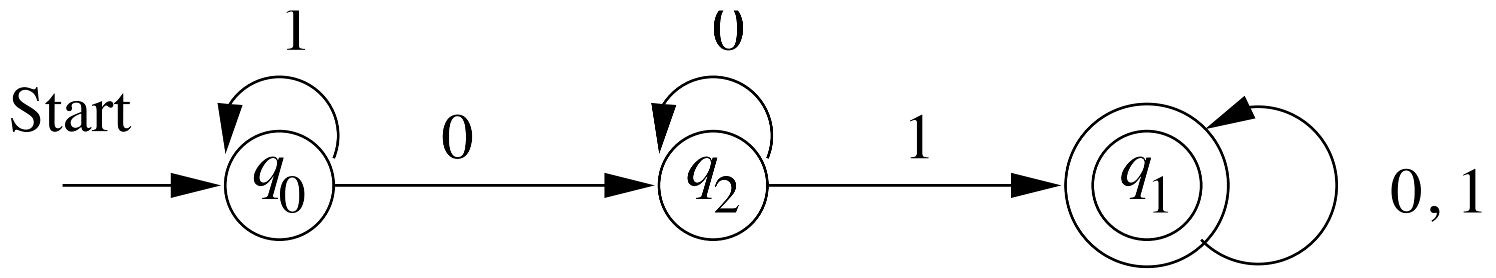
Base: $\hat{\delta}(q, \epsilon) = q$

Induzione: $\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$

- Formalmente, il *linguaggio accettato da A* è

$$L(A) = \{w : \hat{\delta}(q_0, w) \in F\}$$

- Insiemi diversi di stati finali portano a linguaggi diversi.
- I linguaggi accettati da automi a stati finiti sono detti linguaggi regolari

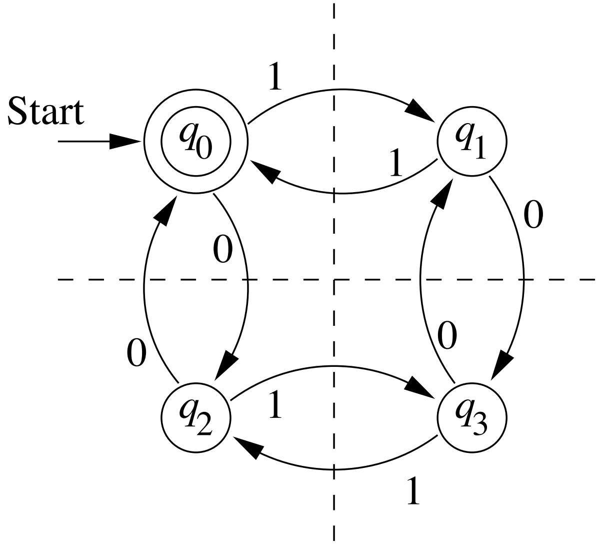


Applichiamo la funzione di transizione estesa $\hat{\delta}$ all'input 01101:

1. $\hat{\delta}(q_0, \epsilon) = q_0$
2. $\hat{\delta}(q_0, 0) = q_2$
3. $\hat{\delta}(q_0, 01) = \delta(\hat{\delta}(q_0, 0), 1) = \delta(q_2, 1) = q_1$
4. $\hat{\delta}(q_0, 011) = \delta(\hat{\delta}(q_0, 01), 1) = \delta(q_1, 1) = q_1$
5. $\hat{\delta}(q_0, 0110) = \delta(\hat{\delta}(q_0, 011), 0) = \delta(q_1, 1) = q_1$
6. $\hat{\delta}(q_0, 01101) = \delta(\hat{\delta}(q_0, 0110), 1) = \delta(q_1, 1) = q_1$

con $q_1 \in F$

Esempio: DFA che accetta tutte e sole le stringhe con un numero pari di zeri e un numero pari di uni (compresa quindi la stringa ϵ)



Esempio

Rappresentazione tabulare dell'automa

	0	1
$\star \rightarrow q_0$	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

q_0 rappresenta lo stato in cui sia il numero di 0 che di 1 è pari
q_1 rappresenta lo stato in cui il numero di 0 è pari e quello di 1 è dispari
q_2 rappresenta lo stato in cui il numero di 0 è dispari e quello di 1 è pari
q_3 rappresenta lo stato in cui sia il numero di 0 che di 1 è dispari

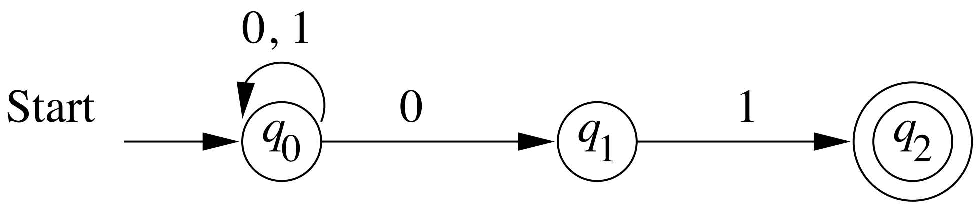
Alcuni esercizi

- DFA per i seguenti linguaggi sull'alfabeto $\{0, 1\}$:
  - Insieme di tutte le stringhe che finiscono con 00
  - Insieme di tutte le stringhe con tre zeri consecutivi
  - Insieme delle stringhe con 011 come sotto-stringa
  - Insieme delle stringhe che cominciano o finiscono (o entrambe le cose) con 01 [Provare a fare prima un automa per le stringhe che cominciano con 01 e uno per quelle che finiscono con 01]

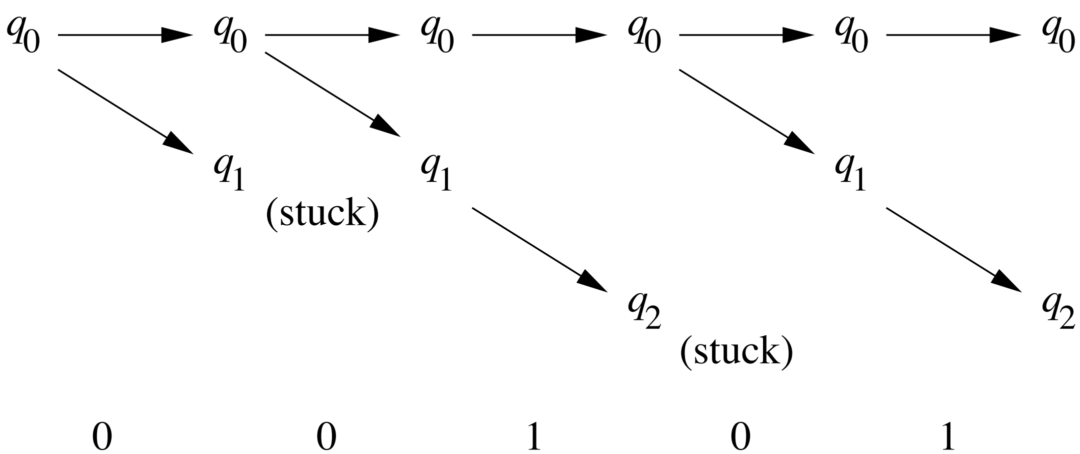
Automi a stati finiti non deterministici (NFA)

- Un NFA può essere in vari stati nello stesso momento, intuitivamente può “scommettere” su quale sarà il prossimo stato.
- Da uno stato, dato un certo input, posso infatti raggiungere un insieme di stati.

Es: automa che accetta tutte e solo le stringhe che finiscono in 01.



Ecco cosa succede quando l'automa elabora l'input 00101 con l'*albero delle alternative*



Definizione formale di NFA

Formalmente, un NFA è una quintupla

$$A = (Q, \Sigma, \delta, q_0, F)$$

- Q è un insieme finito di stati
- Σ è un alfabeto finito
- δ è una funzione di transizione da $Q \times \Sigma$ all'insieme dei sottoinsiemi di Q
- $q_0 \in Q$ è lo *stato iniziale*
- $F \subseteq Q$ è un insieme di *stati finali*

Funzione di transizione

- La funzione di transizione δ prende in ingresso uno stato e un simbolo e restituisce un insieme di stati (che può essere anche vuoto).
- Lo stato successivo non è quindi univocamente determinato dallo stato corrente e dall'input. L'automa può *scegliere* tra stati diversi.
- $\delta(q, a) = \{p_1, ..., p_n\}$ denota che ogni stato p_i può essere raggiunto dallo stato q quando si riceve in input il simbolo a

$$q \rightarrow p_i \Leftrightarrow p_i \in \delta(q, a)$$

- Non è detto che per ogni stato sia sempre definito uno stato successivo per ogni simbolo di input.

Esempio

L' NFA di due pagine fa è

$$(\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$$

dove δ è la funzione di transizione

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$\star q_2$	$\emptyset$	$\emptyset$

Funzione di transizione estesa $\hat{\delta}$ e il linguaggio accettato

Definizione induttiva di $\hat{\delta}$.

Base: $\hat{\delta}(q, \epsilon) = \{q\}$

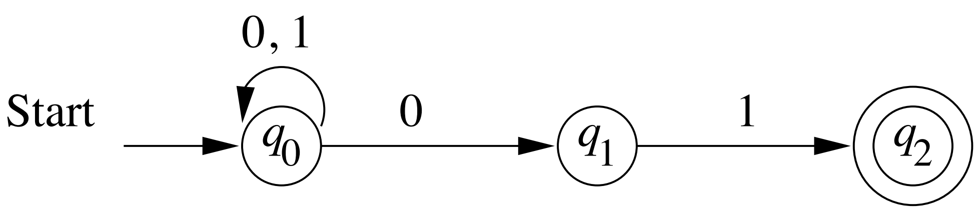
Induzione:

$$\hat{\delta}(q, xa) = \bigcup_{p \in \hat{\delta}(q,x)} \delta(p, a)$$

cioè l'unione di tutti gli stati $\delta(p, a)$ per p che appartiene a $\hat{\delta}(q, x)$

- Un NFA *accetta* una stringa w se $\hat{\delta}(q_0, w) \in F$ contiene almeno uno stato finale
- Formalmente, il *linguaggio accettato da A* è

$$L(A) = \{w : \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$



Applichiamo la funzione di transizione estesa $\hat{\delta}$ all'input 00101:

1. $\hat{\delta}(q_0, \epsilon) = \{q_0\}$
2. $\hat{\delta}(q_0, 0) = \{q_0, q_1\}$
3. $\hat{\delta}(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
4. $\hat{\delta}(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$
5. $\hat{\delta}(q_0, 0010) = \delta(q_0, 0) \cup \delta(q_2, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
6. $\hat{\delta}(q_0, 00101) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

con $\{q_0, q_2\} \cap F \neq \emptyset$

Dimostrazioni di equivalenza

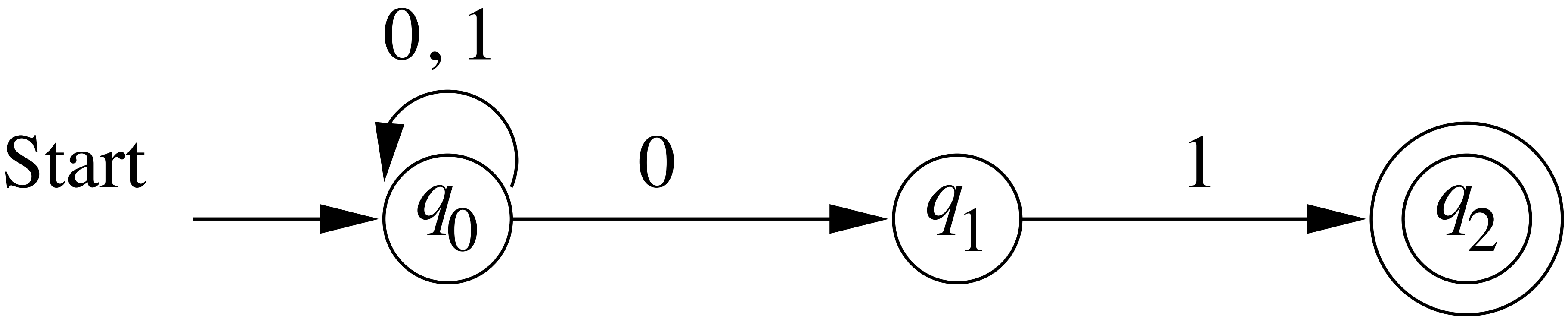
- Spesso ci servirà dimostrare che due descrizioni di insiemi, descrivono lo stesso insieme, ad esempio che il linguaggio accettato dall'automa visto prima coincide con il linguaggio $\{x01 : x \in \Sigma^*\}$
- Per dimostrare che due insiemi S e T sono uguali, si deve dimostrare che $S \subseteq T$ e che $T \subseteq S$:
  - $w \in S \Rightarrow w \in T$, and
  - $w \in T \Rightarrow w \in S$
- Nel nostro esempio dobbiamo dimostrare che

$$w \in L(A) \quad \Leftrightarrow \quad w \in \{x01 : x \in \Sigma^*\}$$

ovvero

$$q_2 \in \hat{\delta}(q_0, w) \quad \Leftrightarrow \quad w \in \{x01 : x \in \Sigma^*\}$$

Per dimostrare formalmente che l'NFA



accetta il linguaggio $\{x01 : x \in \Sigma^*\}$, ci conviene espandere le ipotesi. La dimostrazione si fa per mutua induzione, sulla lunghezza della stringa w, sui tre enunciati seguenti

1. $w \in \Sigma^* \Rightarrow q_0 \in \hat{\delta}(q_0, w)$
2. $q_1 \in \hat{\delta}(q_0, w) \Leftrightarrow w = x0$
3. $q_2 \in \hat{\delta}(q_0, w) \Leftrightarrow w = x01$

Dimostrazione

1. $w \in \Sigma^* \Rightarrow q_0 \in \hat{\delta}(q_0, w)$
2. $q_1 \in \hat{\delta}(q_0, w) \Leftrightarrow w = x0$
3. $q_2 \in \hat{\delta}(q_0, w) \Leftrightarrow w = x01$

Base: Se $|w| = 0$ allora $w = \epsilon$. Allora l'enunciato (1) segue dalla definizione. Per (2) e (3) le ipotesi di entrambi i lati sono false per ϵ e quindi entrambe le implicazioni sono banalmente vere, dato che in logica:

$$(False \Rightarrow False) \equiv True$$

Induzione: Ipotizziamo che $w = xa$, dove $a \in \{0, 1\}$, $|x| = n$ e gli enunciati (1)–(3) valgono per x. Si mostra che gli enunciati valgono per xa (che è lunga $n + 1$).

Dimostrazione (cont.)

Induzione: Ipotizziamo che $w = xa$, dove $a \in \{0, 1\}$, $|x| = n$ e gli enunciati (1)–(3) valgono per x. Si mostra che gli enunciati valgono per xa (che è lunga $n + 1$).

1. $w \in \Sigma^* \Rightarrow q_0 \in \hat{\delta}(q_0, w)$
   Per ipotesi induttiva $q_0 \in \hat{\delta}(q_0, x)$. Dato che $\forall a \in \Sigma . q_0 \in \delta(q_0, a)$, allora $\hat{\delta}(q_0, w) = \delta(\hat{\delta}(q_0, x), a) \ni q_0$

Dimostrazione (cont.)

Induzione: Ipotizziamo che $w = xa$, dove $a \in \{0, 1\}$, $|x| = n$ e gli enunciati (1)–(3) valgono per x. Si mostra che gli enunciati valgono per xa (che è lunga $n + 1$).

2.
   - **(se)** $q_1 \in \hat{\delta}(q_0, w) \Leftarrow w = x0$
     Supponiamo che $w = xa$ finisca per 0 (quindi $a = 0$).
     Per l'enunciato (1) sappiamo che $q_0 \in \hat{\delta}(q_0, x)$ e
     dato che $q_1 \in \delta(q_0, 0)$, possiamo concludere che $q_1 \in \hat{\delta}(q_0, w)$.
   - **(solo se)** $q_1 \in \hat{\delta}(q_0, w) \Rightarrow w = x0$
     Supponiamo che $q_1 \in \hat{\delta}(q_0, w)$. Dal diagramma vediamo che l'unico modo di raggiungere q_1 è che $w = x0$.

Dimostrazione (cont.)

3.
- **(se)** $q_2 \in \hat{\delta}(q_0, w) \Leftarrow w = x01$
  Supponiamo che $w = xa$ finisca per 01 (quindi $a = 1$ e x finisce per 0).
  Per l'enunciato (2) sappiamo che $q_1 \in \hat{\delta}(q_0, x)$ e
  dato che $q_2 \in \delta(q_1, 1)$, possiamo concludere che $q_2 \in \hat{\delta}(q_0, w)$.
- **(solo se)** $q_2 \in \hat{\delta}(q_0, w) \Rightarrow w = x01$
  Supponiamo che $q_2 \in \hat{\delta}(q_0, w)$. Dal diagramma vediamo che
  l'unico modo di raggiungere q_2 è che $w = x1$, dove $q_1 \in \hat{\delta}(q_0, x)$.
  Per l'enunciato (2), x finisce per 0 e quindi w finisce per 01.

$\square$

Equivalenza di DFA e NFA

- Gli NFA sono di solito più facili da "programmare".
- Sorprendentemente, per ogni NFA N c'è un DFA D, tale che $L(D) = L(N)$, e viceversa.
- Questo comporta una *costruzione per sottoinsiemi*, un esempio importante di come un automa B può essere costruito a partire da un altro automa A.
- Dato un NFA

$$N = (Q_N, \Sigma, \delta_N, q_0, F_N)$$

costruiremo un DFA

$$D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$$

tali che

$$L(D) = L(N).$$

I dettagli della costruzione per sottoinsiemi:

- $Q_D = \{S : S \subseteq Q_N\}$.

Nota: $|Q_D| = 2^{|Q_N|}$, anche se la maggior parte degli stati in Q_D sono “garbage”, cioè non raggiungibili dallo stato iniziale.

- $F_D = \{S \subseteq Q_N : S \cap F_N \neq \emptyset\}$
- Per ogni $S \subseteq Q_N$ e $a \in \Sigma$,

$$\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a)$$

Ad esempio l'insieme di stati $\{q_0, q_1\}$ nel nostro esempio diventa un singolo stato del DFA corrispondente, raggiungibile dallo stato $\{q_0\}$.

Costruiamo δ_D dall'NFA già visto, quello cioè che accetta le stringhe che terminano con 01.

	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
$\star\{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\star\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\star\{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
$\star\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

Nota: Gli stati di D corrispondono a sottoinsiemi di stati di N, ma potevamo denotare gli stati di D in un altro modo, per esempio $A - F$.

	0	1
A	A	A
$\rightarrow B$	E	B
C	A	D
$\star D$	A	A
E	E	F
$\star F$	E	B
$\star G$	A	D
$\star H$	E	F

Per evitare la crescita esponenziale degli stati può essere utile costruire la tabella di transizione per D solo per gli stati raggiungibili (o accessibili) S come segue:
Base: $S = \{q_0\}$ è raggiungibile in D
Induzione: Se lo stato S è raggiungibile, lo sono anche gli stati in $\bigcup_{a \in \Sigma} \delta_D(S, a)$ raggiungibile a partire da S.

Esempio: Il "sottoinsieme" DFA con i soli stati raggiungibili: B ($\{q_0\}$), E ($\{q_0, q_1\}$) e F ($\{q_0, q_2\}$).