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

$x = a_1 a_2 \ldots a_i, y = b_1 b_2 \ldots b_j$

$xy = a_1 a_2 \ldots a_i b_1 b_2 \ldots b_j$

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

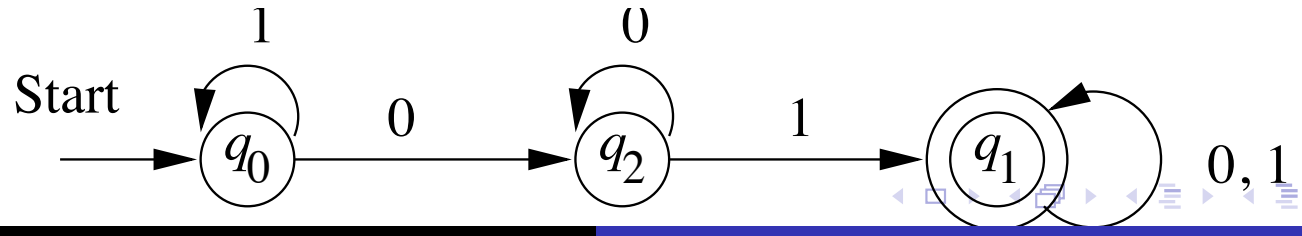
q_2 rappresenta lo stato in cui non si è visto 01, ma si è appena visto 0

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*:

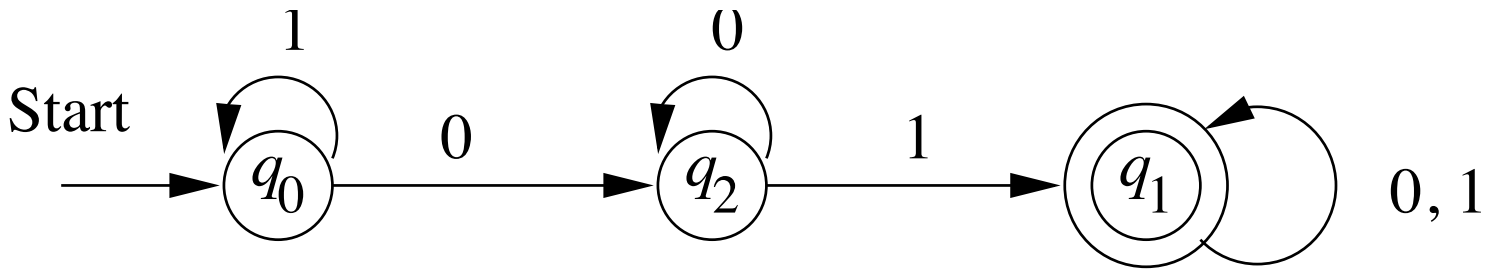


Accettazione

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 $a_1 ... a_n$ che formano la stringa w

Definizione

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

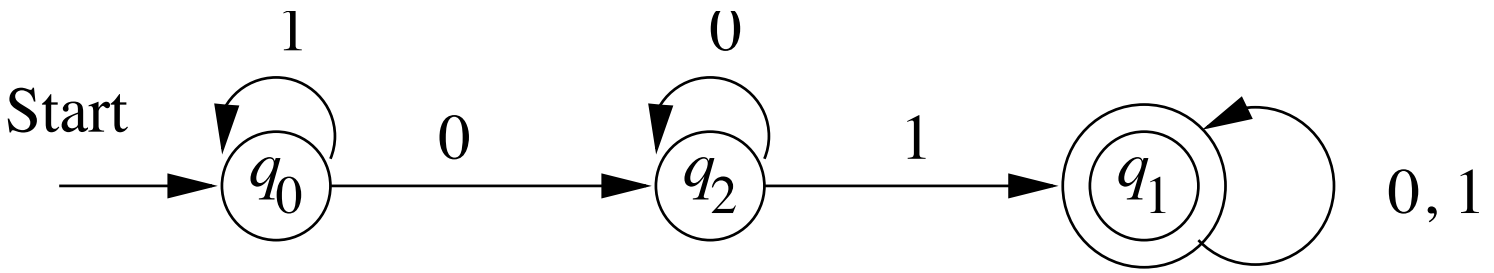
Induzione: $\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$

Definizione

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

Esempio di stringa accettata



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 $q_1 \in F$