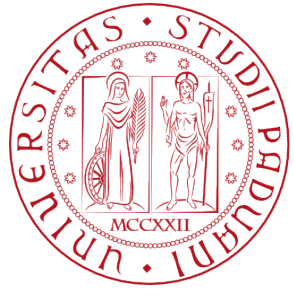
Automi e Linguaggi Formali

1. Automi a Stati Finiti

Davide Bresolin
a.a. 2018/19



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Facile o difficile?

Per ognuno dei seguenti problemi, indicate se è un problema facile o difficile da risolvere per un computer:

1. Trovare il percorso più breve per andare da casa all'Università
2. Trovare bug in un programma
3. "Rompere" un codice crittografico
4. Colorare una mappa
5. Ottimizzare la consegna della posta
6. Ottimizzare la consegna delle pizze

Informatica teorica — disciplina scientifica al confine tra informatica e matematica

- si pone domande generali sugli algoritmi e sugli strumenti di computazione
- studio di diversi tipi di formalismi per la descrizione degli algoritmi
- studio di diversi approcci per la descrizione della sintassi e della semantica dei linguaggi formali (principalmente linguaggi di programmazione)
- usa un approccio matematico all'analisi e alla soluzione dei problemi (prove di proprietà matematiche generali riguardanti algoritmi)

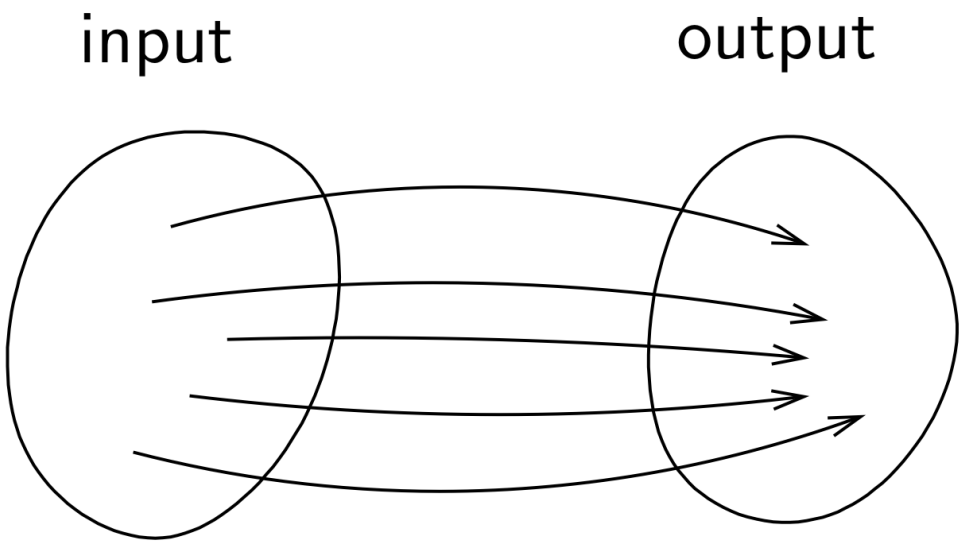
Esempi di domande tipiche studiate nell'informatica teorica:

- È possibile risolvere un certo problema usando un algoritmo?
- Se il problema può essere risolto da un algoritmo, qual è la complessità computazionale di questo algoritmo?
- Esiste un algoritmo efficiente per risolvere il problema?
- Come verificare che un dato algoritmo sia davvero una soluzione corretta del problema dato?
- Quali tipi di istruzioni sono sufficienti affinché una determinata macchina possa eseguire un determinato algoritmo?

Problema

Per descrivere un problema dobbiamo specificare:

- l'insieme dei possibili input
- l'insieme dei possibili output
- la relazione tra input e output



Algoritmi e Problemi

- Algoritmo – procedura meccanica che esegue delle computazioni (e può essere eseguita da un calcolatore)
- Un algoritmo risolve un dato problema se:
  - Per ogni input, il calcolo dell'algoritmo si interrompe dopo un numero finito di passaggi.
  - Per ogni input, l'algoritmo produce un output corretto.
- Correttezza di un algoritmo – verificare che l'algoritmo risolva realmente il problema dato
- Complessità computazionale di un algoritmo:
  - **complessità temporale** – come varia il tempo di esecuzione dell'algoritmo rispetto alla dimensione dei dati di input
  - **complessità spaziale** – come varia la quantità di memoria utilizzata dall'algoritmo rispetto alla dimensione dei dati di input

Linguaggi Formali

- Astrazione della nozione di problema
- I problemi possono sono espressi come linguaggi (= insiemi di stringhe)
  - Le soluzioni determinano se una determinata stringa è nell'insieme o no
    - ad esempio: un certo intero n è un numero primo?
- Oppure, come trasformazioni tra linguaggi
  - Le soluzioni trasformano la stringa di input in una stringa di output
    - ad esempio: quanto fa $3 + 5$?

Linguaggi Formali

- Quindi in sostanza tutti i processi computazionali possono essere ridotti ad uno tra:
  - Determinazione dell'appartenenza a un insieme (di stringhe)
  - Mappatura tra insiemi (di stringhe)
- Formalizzeremo il concetto di computazione meccanica:
  - dando una definizione precisa del termine "algoritmo"
  - caratterizzando i problemi che sono o non sono adatti per essere risolti da un calcolatore.

Di cosa parla questo corso?

Automi

- Gli automi (singolare automa) sono dispositivi matematici astratti che possono:
  - determinare l'appartenenza di una stringa ad un insieme di stringhe
  - trasformare una stringa in un'altra stringa
- Hanno tutti gli aspetti di un computer:
  - input e output
  - memoria
  - capacità di prendere decisioni
  - trasformare l'input in output

Automi

- Il tipo di memoria è cruciale:
  - memoria finita
  - memoria infinita:
    - con accesso limitato
    - con accesso illimitato
- Abbiamo diversi tipi di automi per diversi classi di linguaggi
- I diversi tipi di automi si differenziano per
  - la quantità di memoria (finita vs infinita)
  - il tipo di accesso alla memoria (limitato vs illimitato)

Organizzazione del corso

Docenti del Corso

Prima parte + Laboratorio + Terza parte

Docente: Davide Bresolin

e-mail: davide.bresolin@unipd.it

ufficio: Stanza 320, III Piano, Scala C della Torre Archimede, Dipartimento di Matematica, via Trieste

ricevimento: lunedì 16:30-18:30 oppure su appuntamento

Seconda parte + Terza parte

Docente: Gilberto Filè

Programma del Corso

- **Parte 1:** linguaggi regolari
  - automi a stati finiti
  - espressioni e linguaggi regolari
- **Parte 2:** linguaggi liberi da contesto
  - grammatiche e linguaggi liberi dal contesto
  - automi a pila
- **Laboratorio:** due lezioni di esercitazione
  - costruzione di un parser e di un interprete per un linguaggio di programmazione
  - **Lunedì 29 Aprile** e **venerdì 3 Maggio**, 12:30-14:30, LabP140
- **Parte 3:** indecidibilità e intrattabilità
  - macchine di Turing
  - concetto di indecidibilità
  - problemi intrattabili
  - classi P e NP

Calendario delle prime quattro settimane

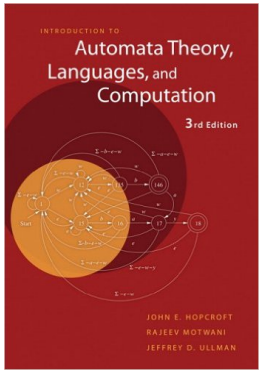
I Settimana	Lun 25/2, 12:30–14:30, Aula LuM250
	Mar 26/2, 12:30–14:30, Aula LuM250
	Ven 1/3, 12:30–14:30, Aula LuM250
II Settimana	Lun 4/3, 12:30–14:30, Aula LuM250
	Mar 5/3, 12:30–14:30, Aula LuM250
	Ven 8/3, 12:30–14:30, Aula LuM250
III Settimana	Lun 11/3, 12:30–14:30, Aula LuM250
	Mar 12/3, 12:30–14:30, Aula LuM250
	Ven 15/3, 12:30–14:30, Aula LuM250
IV Settimana	Lun 18/3, 12:30–14:30, Aula LuM250
	Mar 19/3, 12:30–14:30, Aula LuM250
	Ven 22/3, 12:30–14:30, Aula LuM250

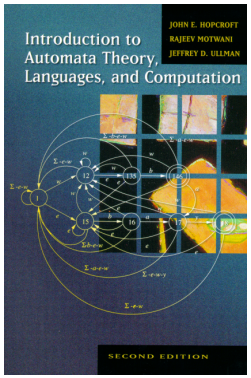
Libro di testo

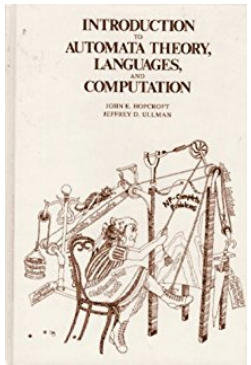
J. E. Hopcroft, R. Motwani, J. D. Ullman
Automi, linguaggi e calcolabilità

J. E. Hopcroft, R. Motwani, J. D. Ullman
Introduction to Automata Theory, Languages, and Computation

Va bene qualsiasi edizione (1a, 2a, 3a)







Moodle del corso

- Vi si accede da `https://elearning.unipd.it/math`
  - selezionando prima Informatica – Triennale
  - e poi Automi E Linguaggi Formali A.A. 2018/2019
- Autenticazione tramite le proprie credenziali UniPD
- Pubblicazione di slide e altro materiale del corso
- Esercizi e soluzioni
- Comunicazioni e aggiornamenti

Tutorato

Università degli Studi di Padova

Tutor:	Linpeng Zhang
Contatti:	gruppo Facebook del Tutorato di Informatica
Incontri:	tutti i mercoledì a partire dal 06/03/19 fino al 05/06/19, dalle ore 10:30 alle ore 12:30 in aula 2BC60 Torre Archimede

Esami, compitini ed esercizi

- **Esame:** Scritto e, se richiesto dai docenti, colloquio orale. Cinque appelli, tra Luglio, Settembre 2018 e Febbraio 2019.
- **Compitini:** Due compitini che sostituiscono l'esame (maggiori informazioni nella slide successiva!)
- **Esercizi (prima parte del corso):** test di autovalutazione sul Moodle + esercizi pubblicati su `www.automatatutor.com`.

Compitini

- Due compitini:
  - il primo durante la settimana di sospensione delle lezioni
    - 8–12 Aprile
  - il secondo alla fine del corso
  - I compitini sostituiscono l'esame
    - devono essere entrambi sufficienti
- Per gli appelli di Giugno e Luglio e Settembre:
  - i voti dei compitini rimangono validi
  - compito diviso in due parti
  - si può svolgere una sola delle due parti
  - si può recuperare un compitino insufficiente
- Per l'appello di Febbraio 2020:
  - i voti dei compitini e delle singole parti non sono più validi
  - si deve fare l'esame completo

Automi a Stati Finiti Deterministici

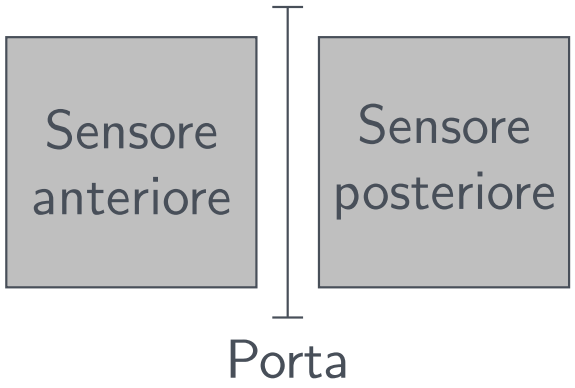
Gli Automi a Stati Finiti

- Sono il più semplice modello computazionale
- Dispongono di una quantità di memoria finita
- Gli automi a stati finiti sono usati come modello per:
  - Software per la progettazione di circuiti digitali
  - Analizzatori lessicali di un compilatore
  - Ricerca di parole chiave in un file o sul web
  - Software per verificare sistemi a stati finiti, come protocolli di comunicazione

Esempio: una porta automatica

Costruiamo un esempio di controllore di una porta automatica:

- La porta si apre quando una persona si avvicina
- Un sensore di fronte alla porta rileva la presenza della persona
- Un sensore sul retro della porta rileva quando la persona ha attraversato la porta e se c'è qualcuno dietro la porta

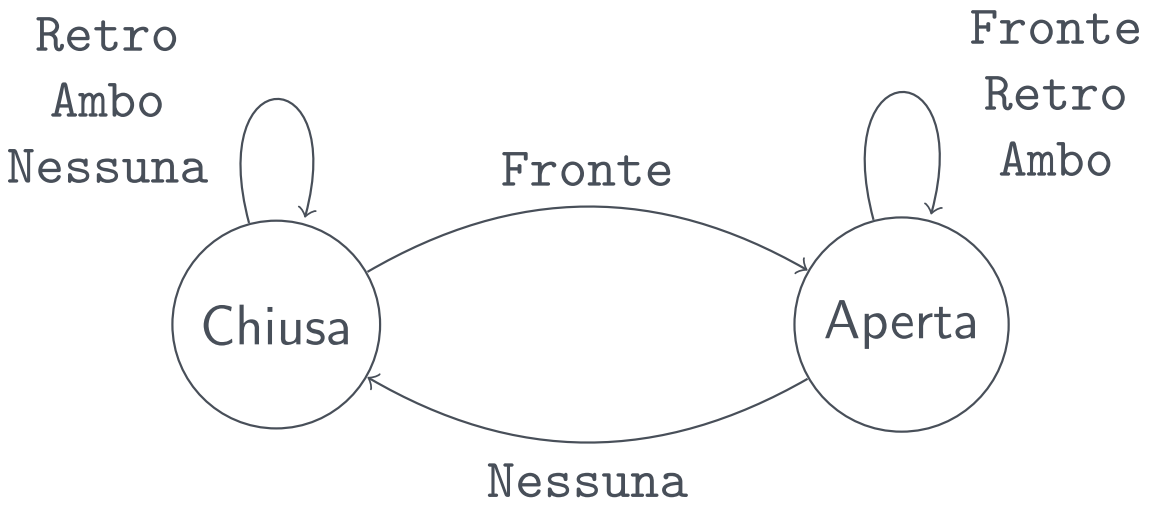


Esempio: una porta automatica

- La porta si può trovare in due stati: Chiusa o Aperta
- Ci sono quattro possibili input dai sensori:
  - `Fronte`: c'è una persona di fronte alla porta
  - `Retro`: c'è una persona dietro alla porta
  - `Ambo`: ci sono persone sia di fronte che dietro alla porta
  - `Nessuna`: non ci sono persone né davanti né dietro la porta

Esempio: una porta automatica

- La porta si può trovare in due stati: Chiusa o Aperta
- Ci sono quattro possibili input dai sensori:
  - `Fronte`: c'è una persona di fronte alla porta
  - `Retro`: c'è una persona dietro alla porta
  - `Ambo`: ci sono persone sia di fronte che dietro alla porta
  - `Nessuna`: non ci sono persone né davanti né dietro la porta



Alfabeti, linguaggi e automi a stati finiti

Per rappresentare in maniera precisa l'esempio, dobbiamo definire alcuni concetti di base:

- Che cos'è un alfabeto (di simboli/messaggi/azioni)
- Che cos'è un linguaggio formale
- Che cos'è un Automa a stati finiti deterministico
- Cosa vuol dire che un automa accetta un linguaggio

Alfabeti e stringhe

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: (o parola) Sequenza finita di simboli da un alfabeto Σ, e.g. 0011001

Stringa vuota: La stringa con zero occorrenze di simboli da Σ

- La stringa vuota è denotata con ε

Lunghezza di una stringa: Numero di simboli nella stringa.

- $|w|$ denota la lunghezza della stringa w
- $|0110| = 4$, $|\varepsilon| = 0$

Potenze di un alfabeto

- Potenze di un alfabeto: Σ^k = insieme delle stringhe di lunghezza k con simboli da Σ
  - Esempio: $\Sigma = \{0, 1\}$

$$\Sigma^0 = \{\varepsilon\}$$
$$\Sigma^1 = \{0, 1\}$$
$$\Sigma^2 = \{00, 01, 10, 11\}$$

  - Domanda: Quante stringhe ci sono in Σ^3?
- L'insieme di tutte le stringhe su Σ è denotato da Σ^*
  - $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$

Linguaggi

- **Linguaggio:** dato un alfabeto Σ, chiamiamo linguaggio ogni sottoinsieme $L \subseteq \Sigma^*$
- Esempi di linguaggi:
  - L'insieme delle parole italiane
  - L'insieme dei programmi C sintatticamente corretti
  - L'insieme delle stringe costituite da n zeri seguiti da n uni:

    $$\{\varepsilon, 01, 0011, 000111, \dots\}$$

  - Il **linguaggio vuoto** $\emptyset$ non contiene nessuna parola
  - Il linguaggio che contiene solo la parola vuota:

    $$\{\varepsilon\}$$

  - ...

Automi a Stati Finiti Deterministici

Un Automa a Stati Finiti Deterministico (DFA) è una quintupla

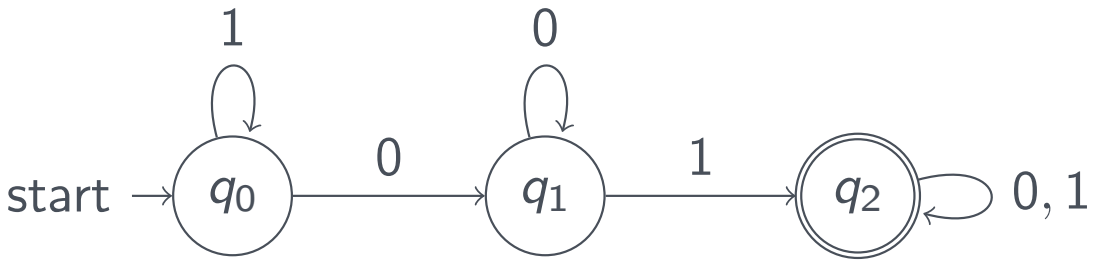
$$A = (Q, \Sigma, \delta, q_0, F)$$

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

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

Esempio: costruiamo un automa A che accetta il linguaggio delle stringhe con 01 come sottostringa

- L'automa come diagramma di transizione:



- L'automa come tabella di transizione:

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

Linguaggio accettato da un DFA

- La funzione di transizione δ può essere estesa a $\hat{\delta}$ che opera su stati e parole (invece che su stati e simboli):

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

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

  con $w = xa$ (parola x seguita dal simbolo a)

- Formalmente, il linguaggio accettato da A è

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

- I linguaggi accettati da automi a stati finiti sono detti linguaggi regolari

Esempi

DFA per i seguenti linguaggi sull'alfabeto $\{0, 1\}$:

- 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
- Insieme di tutte le stringhe che contengono esattamente tre zeri (anche non consecutivi)
- Insieme delle stringhe che cominciano o finiscono (o entrambe le cose) con 01

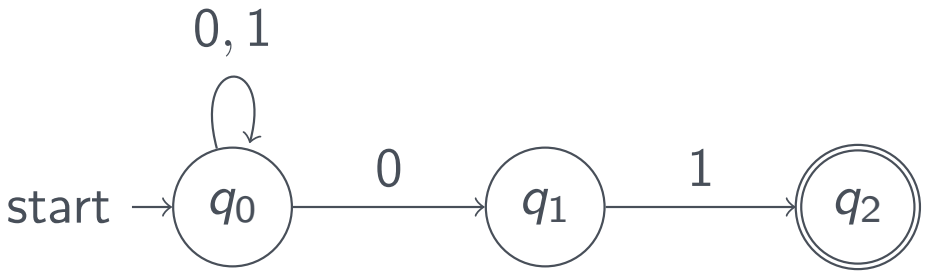
Il distributore di Bibite

Modellare il comportamento di un distributore di bibite con un DFA. Il modello deve rispettare le seguenti specifiche:

- Costo della bibita: 40 centesimi
- Monete utilizzabili: 10 centesimi, 20 centesimi
- Appena le monete inserite raggiungono o superano il costo della bibita, il distributore emette una lattina
- Il distributore dà il resto (se serve) subito dopo aver emesso la lattina

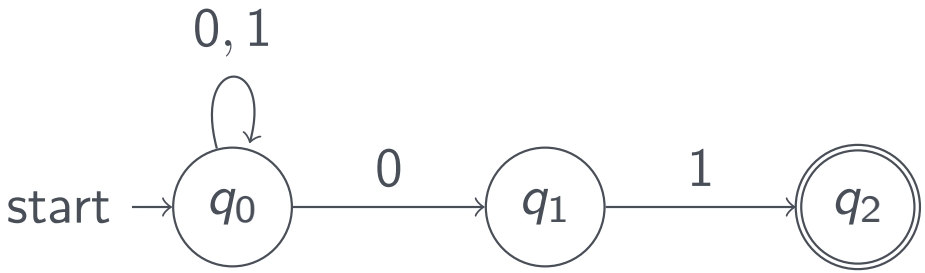
Automi a Stati Finiti Non Deterministici

- Cosa fa questo automa?



Automi a stati finiti non deterministici (NFA)

- Cosa fa questo automa?



  Riconosce le parole che terminano con 01 "scommettendo" se sta leggendo gli ultimi due simboli oppure no
- È un esempio di automa a stati finiti non deterministico:
  - può trovarsi contemporaneamente in più stati diversi
  - le transizioni non sono necessariamente complete:
    - da q_1 si esce solo leggendo 1
    - q_2 non ha transizioni uscenti

    in questi casi il percorso si blocca, ma può proseguire lungo gli altri percorsi

Definizione formale di NFA

Un Automa a Stati Finiti Non Deterministico (NFA) è una quintupla

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

- 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
- $q_0 \in Q$ è lo stato iniziale
- $F \subseteq Q$ è un insieme di stati finali

Tabella delle transizioni per l'esempio

L'NFA che riconosce le parole che terminano con 01 è

$$A = (Q, \{0, 1\}, \delta, q_0, \{q_2\})$$

dove δ è la funzione di transizione

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

Linguaggio riconosciuto da un NFA

- La funzione di transizione estesa $\hat{\delta}$ per gli NFA:

  Base:

  $$\hat{\delta}(q, \varepsilon) = \{q\}$$

  Induzione:

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

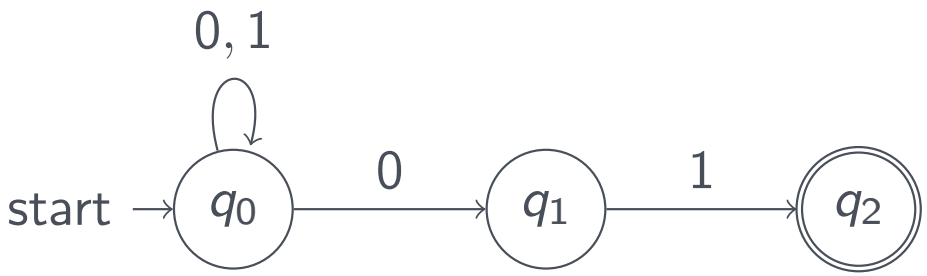
  con $w = xa$ (parola x seguita dal simbolo a)

- **Esempio:** calcoliamo $\hat{\delta}(q_0, 00101)$ alla lavagna
- Formalmente, il linguaggio accettato da A è

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

Dimostriamo che l'esempio è corretto

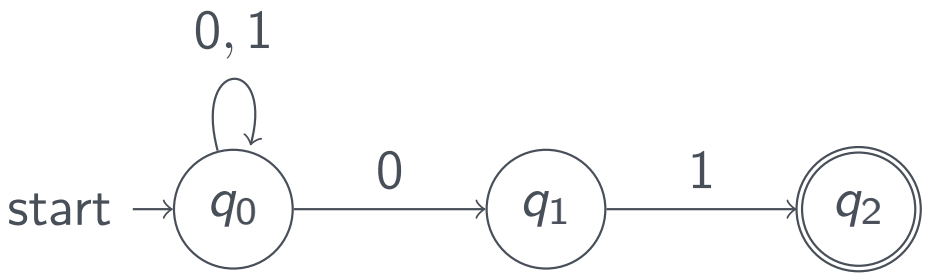
- Dimostriamo che l'automa d'esempio



accetta il linguaggio $L = \{x01 : x \in \Sigma^*\}$.

Dimostriamo che l'esempio è corretto

- Dimostriamo che l'automa d'esempio

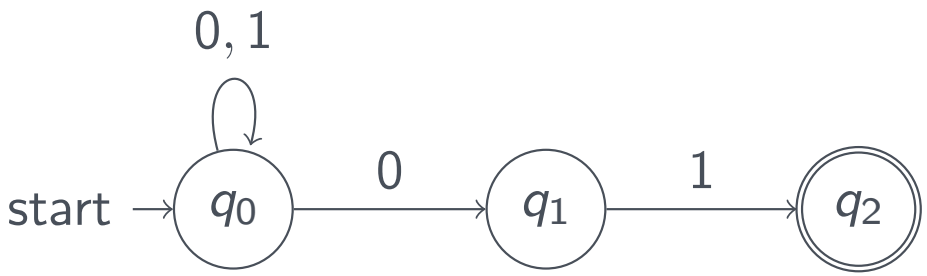


  accetta il linguaggio $L = \{x01 : x \in \Sigma^*\}$.

- Lo faremo dimostrando che valgono tre enunciati che danno le proprietà degli stati:
  1. per ogni $w \in \Sigma^*$, $q_0 \in \hat{\delta}(q_0, w)$
  2. $q_1 \in \hat{\delta}(q_0, w)$ se e solo se $w = x0$
  3. $q_2 \in \hat{\delta}(q_0, w)$ se e solo se $w = x01$

Dimostriamo che l'esempio è corretto

- Dimostriamo che l'automa d'esempio



  accetta il linguaggio $L = \{x01 : x \in \Sigma^*\}$.

- Lo faremo dimostrando che valgono tre enunciati che danno le proprietà degli stati:
  1. per ogni $w \in \Sigma^*$, $q_0 \in \hat{\delta}(q_0, w)$
  2. $q_1 \in \hat{\delta}(q_0, w)$ se e solo se $w = x0$
  3. $q_2 \in \hat{\delta}(q_0, w)$ se e solo se $w = x01$

- La dimostrazione è per induzione sulla lunghezza $|w|$ della parola in ingresso

Esercizi

Definire degli automi a stati finiti non deterministici che accettino i seguenti linguaggi:

- L'insieme delle parole sull'alfabeto $\{0, 1, \ldots, 9\}$ tali che la cifra finale sia comparsa in precedenza
- L'insieme delle parole sull'alfabeto $\{0, 1, \ldots, 9\}$ tali che la cifra finale *non* sia comparsa in precedenza
- 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)

Esercizio

Consideriamo l'alfabeto $\Sigma = \{a, b, c, d\}$ e costruiamo un automa non deterministico che riconosce il linguaggio di tutte le parole tali che uno dei simboli dell'alfabeto non compare mai:

- tutte le parole che non contengono a
- $+$ tutte le parole che non contengono b
- $+$ tutte le parole che non contengono c
- $+$ tutte le parole che non contengono d

Equivalenza di DFA e NFA

- Sorprendentemente, NFA e DFA sono in grado di riconoscere gli stessi linugaggi
- Per ogni NFA N c'è un DFA D tale che $L(D) = L(N)$, e viceversa
- L'equivalenza di dimostra mediante una costruzione a sottoinsiemi:

Equivalenza di DFA e NFA

- Sorprendentemente, NFA e DFA sono in grado di riconoscere gli stessi linugaggi
- Per ogni NFA N c'è un DFA D tale che $L(D) = L(N)$, e viceversa
- L'equivalenza di dimostra mediante una costruzione a sottoinsiemi:
  Dato un NFA

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

  costruiremo un DFA

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

  tale che

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

La costruzione a sottoinsiemi

- $Q_D = \{S : S \subseteq Q_N\}$
  Ogni stato del DFA corrisponde ad un insieme di stati dell'NFA
- $F_D = \{S \subseteq Q_N : S \cap F_N \neq \emptyset\}$
  Uno stato del DFA è finale se c'è almeno uno stato finale corrispondente nell'NFA
- Per ogni $S \subseteq Q_N$ e per ogni $a \in \Sigma$

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

La funzione di transizione "percorre tutte le possibili strade"

La costruzione a sottoinsiemi

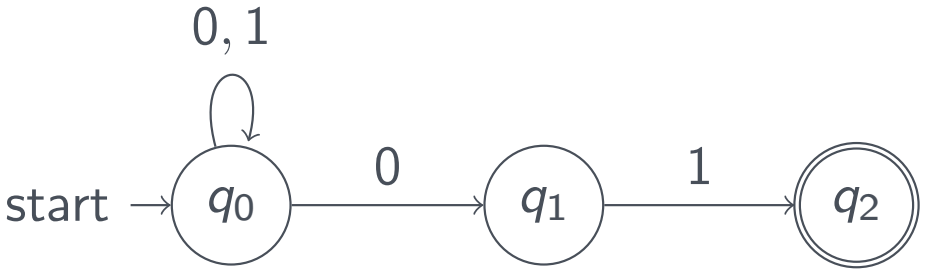
- $Q_D = \{S : S \subseteq Q_N\}$
  Ogni stato del DFA corrisponde ad un insieme di stati dell'NFA
- $F_D = \{S \subseteq Q_N : S \cap F_N \neq \emptyset\}$
  Uno stato del DFA è finale se c'è almeno uno stato finale corrispondente nell'NFA
- Per ogni $S \subseteq Q_N$ e per ogni $a \in \Sigma$

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

La funzione di transizione "percorre tutte le possibili strade"

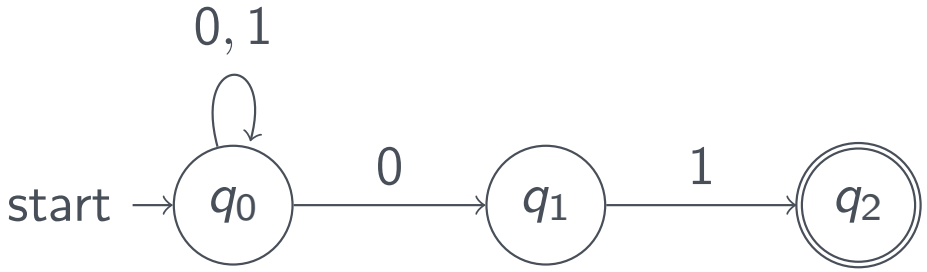
Nota: $|Q_D| = 2^{|Q_N|}$, anche se spesso la maggior parte degli stati in Q_D sono "inutili", cioè non raggiungibili dallo stato iniziale.

Esempio di costruzione a sottoinsiemi



Costruiamo δ_D per l'NFA qui sopra:

Esempio di costruzione a sottoinsiemi

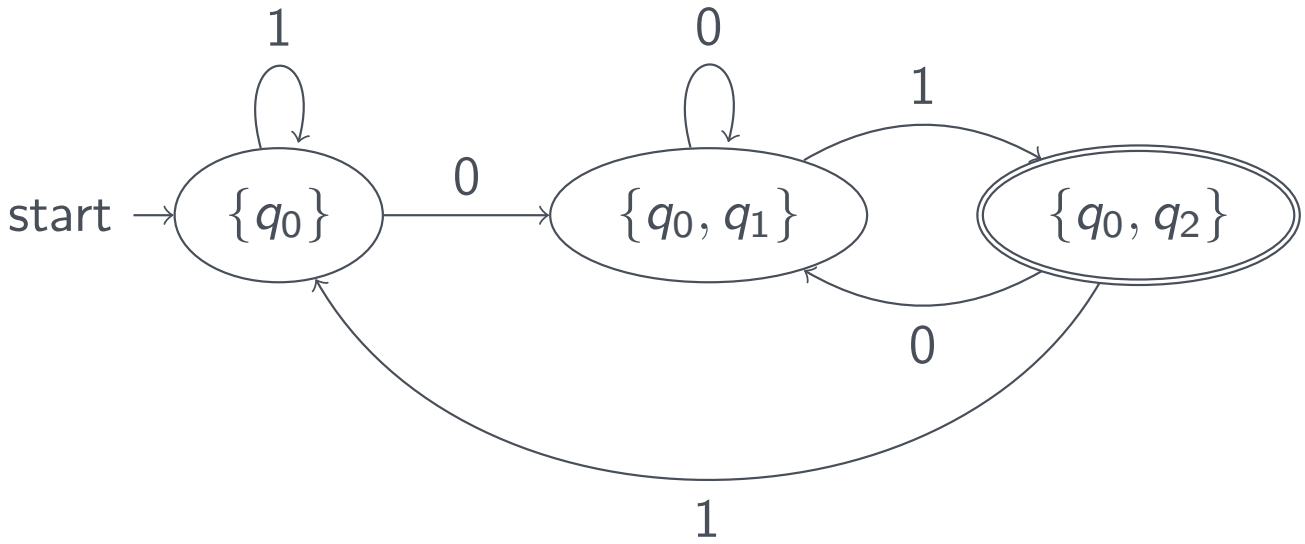


Costruiamo δ_D per l'NFA qui sopra:

	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\to \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
$*\{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$*\{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
$*\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

Diagramma degli stati

La tabella di transizione per D ci permette di ottenere il diagramma di transizione



Per semplificare il disegno, ho omesso gli stati non raggiungibili

Correttezza della costruzione a sottoinsiemi

Theorem

Sia D il DFA ottenuto da un NFA N con la costruzione a sottoinsiemi. Allora $L(D) = L(N)$.

Dimostrazione:

Theorem

Sia D il DFA ottenuto da un NFA N con la costruzione a sottoinsiemi. Allora $L(D) = L(N)$.

Dimostrazione: Prima mostriamo per induzione su $|w|$ che

$$\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)$$

Theorem

Sia D il DFA ottenuto da un NFA N con la costruzione a sottoinsiemi. Allora $L(D) = L(N)$.

Dimostrazione: Prima mostriamo per induzione su $|w|$ che

$$\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)$$

Base: $w = \varepsilon$. L'enunciato segue dalla definizione.

Induzione:

- Sia $|w| = n+1$ e supponiamo vero l'enunciato per la lunghezza n. Scomponiamo w in $w = xa$ (con $|x| = n$ e a simbolo finale)
- Per ipotesi induttiva $\hat{\delta}_D(\{q_0\}, x) = \hat{\delta}_N(q_0, x) = \{p_1, \ldots, p_k\}$
- Per la definizione di $\hat{\delta}$ per gli NFA

$$\hat{\delta}_N(q_0, xa) = \bigcup_{i=1}^{k} \delta_N(p_i, a)$$

- Per la costruzione a sottoinsiemi

$$\delta_D(\{p_1, \ldots, p_k\}, a) = \bigcup_{i=1}^{k} \delta_N(p_i, a)$$

Induzione (continua):

- Per la definizione di $\hat{\delta}$ per i DFA

$$\hat{\delta}_D(\{q_0\}, xa) = \delta_D(\{p_1, \ldots, p_k\}, a) = \bigcup_{i=1}^{k} \delta_N(p_i, a)$$

- Quindi abbiamo mostrato che $\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)$

Poiché sia D che N accettano se solo se $\hat{\delta}_D(\{q_0\}, w)$ e $\hat{\delta}_N(q_0, w)$ contengono almeno un stato in F_N, allora abbiamo dimostrato che $L(D) = L(N)$

Teorema di equivalenza tra DFA e NFA

Theorem

Un linguaggio L è accettato da un DFA se e solo se è accettato da un NFA.

Dimostrazione:

- La parte "se" è il teorema precedente
- La parte "solo se" si dimostra osservando che ogni DFA può essere trasformato in un NFA modificando δ_D in δ_N con la seguente regola:

  Se $\delta_D(q, a) = p$ allora $\delta_N(q, a) = \{p\}$

Esercizio 1

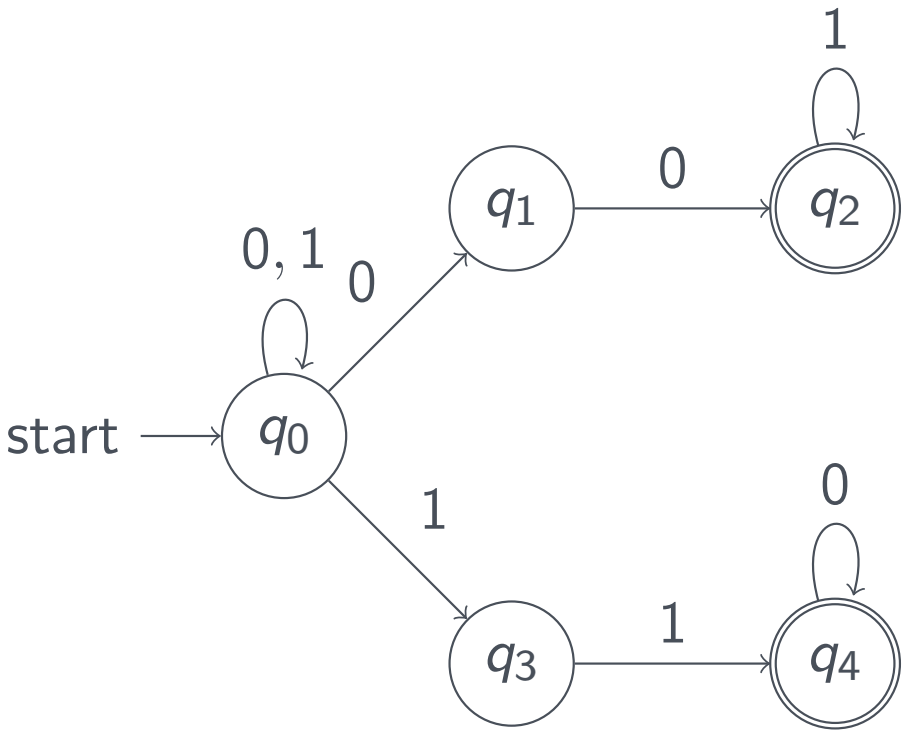
1. Determinare il DFA equivalente all'NFA con la seguente tabella di transizione:

	0	1
$\rightarrow q_0$	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_1\}$	$\{q_0, q_2\}$
$*q_2$	$\{q_1, q_2\}$	$\{q_0, q_1, q_2\}$

2. Qual è il linguaggio accettato dall'automa?

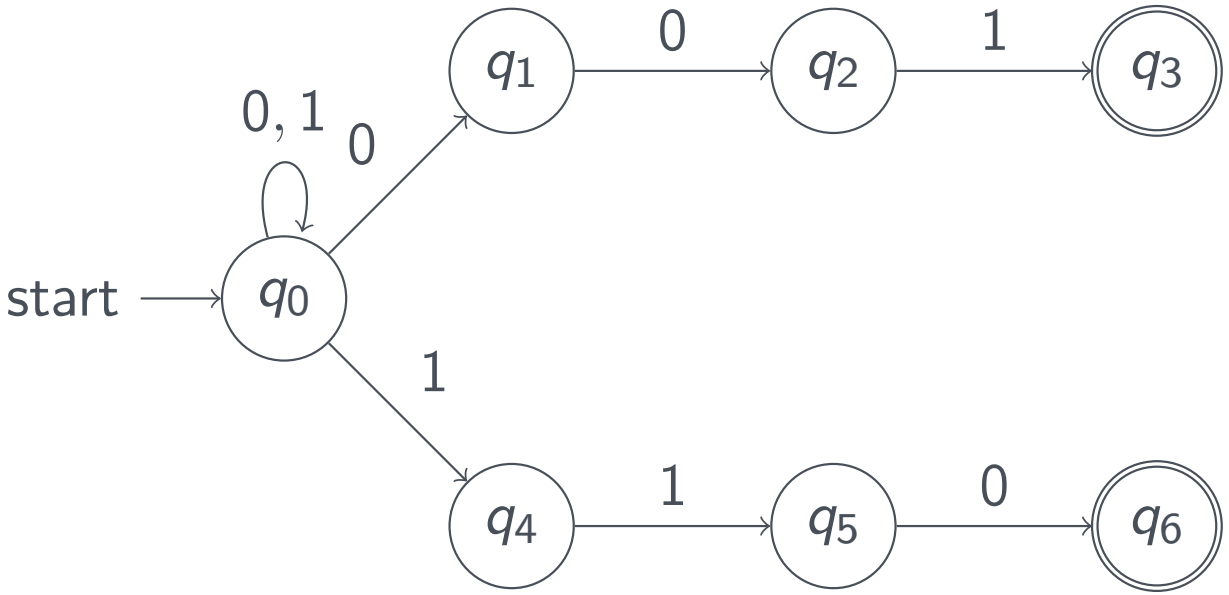
Esercizio 2

Trasformare il seguente NFA in DFA



Esercizio 3

Dato il seguente NFA



1. determinare il linguaggio riconosciuto dall'automa
2. costruire un DFA equivalente

Esercizio

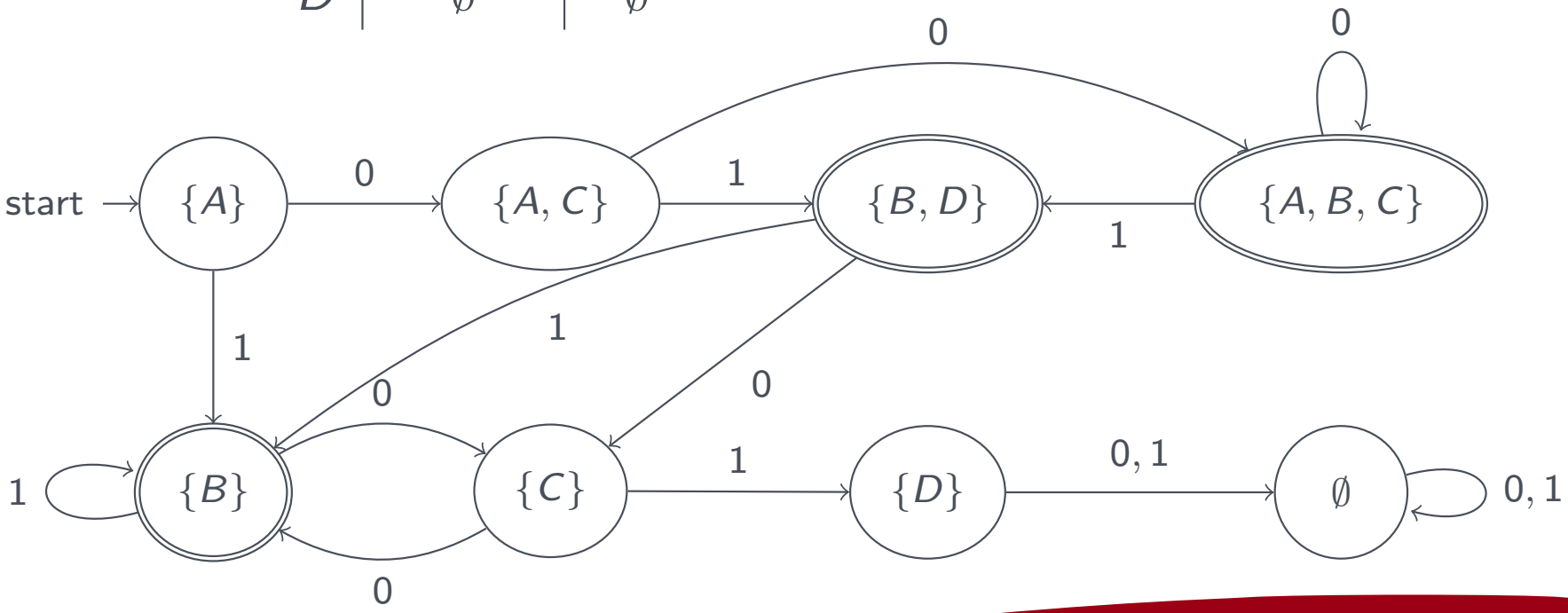
Convertire il seguente NFA in DFA:

	0	1
$\rightarrow A$	$\{A, C\}$	$\{B\}$
$*B$	$\{C\}$	$\{B\}$
C	$\{B\}$	$\{D\}$
D	$\emptyset$	$\emptyset$

Esercizio

Convertire il seguente NFA in DFA:

	0	1
$\rightarrow A$	$\{A, C\}$	$\{B\}$
$*B$	$\{C\}$	$\{B\}$
C	$\{B\}$	$\{D\}$
D	$\emptyset$	$\emptyset$



NFA con epsilon-transizioni

Esercizio: costruiamo un NFA che accetta numeri decimali:

1. Un segno $+$ o $-$, opzionale
2. Una stringa di cifre decimali $\{0, \ldots, 9\}$
3. un punto decimale .
4. un'altra stringa di cifre decimali

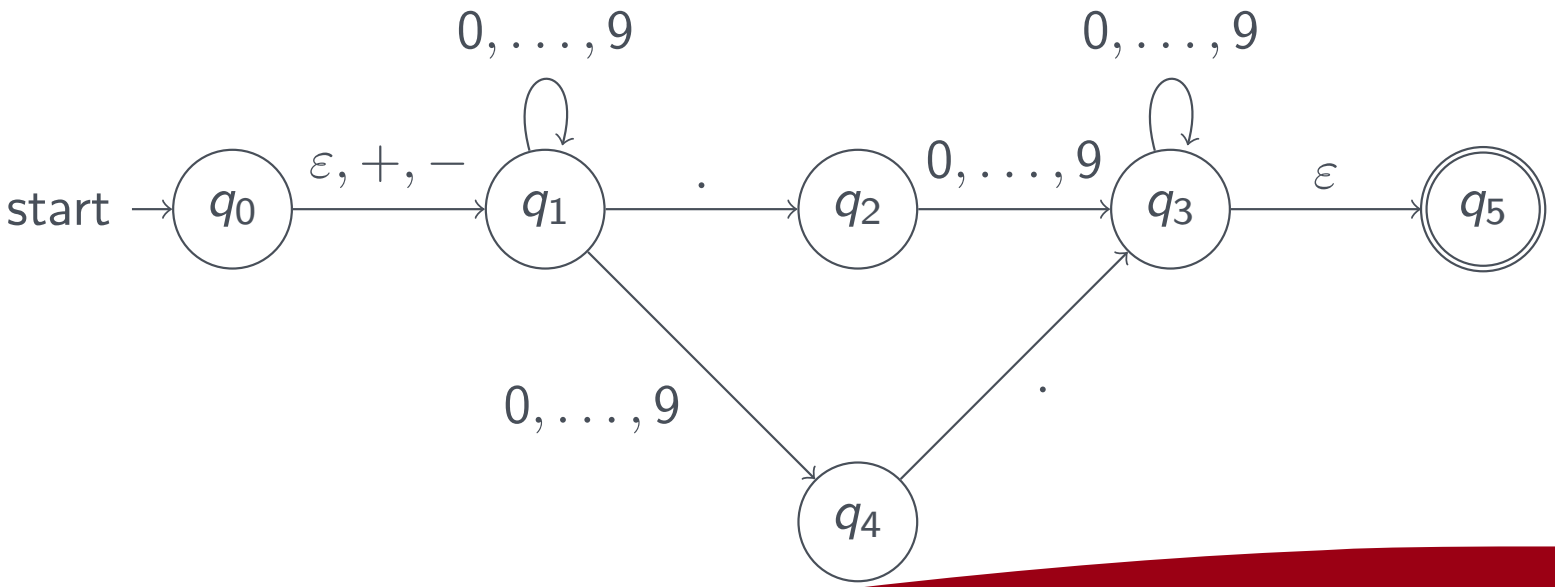
Una delle stringhe (2) e (4) può essere vuota, ma non entrambe

NFA con epsilon-transizioni

Esercizio: costruiamo un NFA che accetta numeri decimali:

1. Un segno $+$ o $-$, opzionale
2. Una stringa di cifre decimali $\{0, \ldots, 9\}$
3. un punto decimale .
4. un'altra stringa di cifre decimali

Una delle stringhe (2) e (4) può essere vuota, ma non entrambe



ε-NFA: definizione

Un Automa a Stati Finiti Non Deterministico con ε-transizioni (ε-NFA) è una quintupla

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

dove:

- Q, Σ, q_0, F sono definiti come al solito
- δ è una funzione di transizione che prende in input:
  - uno stato in Q
  - un simbolo nell'alfabeto $\Sigma \cup \{\varepsilon\}$

  e restituisce un sottoinsieme di Q

Esempio di ε-NFA

L'automa che riconosce le cifre decimali è definito come

$$A = (\{q_0, q_1, \ldots, q_5\}, \{+, -, ., 0, \ldots, 9\}, \delta, q_0, \{q_5\})$$

dove δ è definita dalla tabella di transizione

Esempio di ε-NFA

L'automa che riconosce le cifre decimali è definito come

$$A = (\{q_0, q_1, \ldots, q_5\}, \{+, -, ., 0, \ldots, 9\}, \delta, q_0, \{q_5\})$$

dove δ è definita dalla tabella di transizione

	ε	$+, -$	$.$	$0, \ldots, 9$
$\rightarrow q_0$	$\{q_1\}$	$\{q_1\}$	$\emptyset$	$\emptyset$
q_1	$\emptyset$	$\emptyset$	$\{q_2\}$	$\{q_1, q_4\}$
q_2	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_3\}$
q_3	$\{q_5\}$	$\emptyset$	$\emptyset$	$\{q_3\}$
q_4	$\emptyset$	$\emptyset$	$\{q_3\}$	$\emptyset$
$*q_5$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Epsilon chiusura: definizione

L'eliminazione delle ε-transizioni procede per ε-chiusura degli stati:

- tutti gli stati raggiungibili da q con una sequenza $\varepsilon\varepsilon\ldots\varepsilon$

La definizione di $\mathrm{ECLOSE}(q)$ è per induzione:

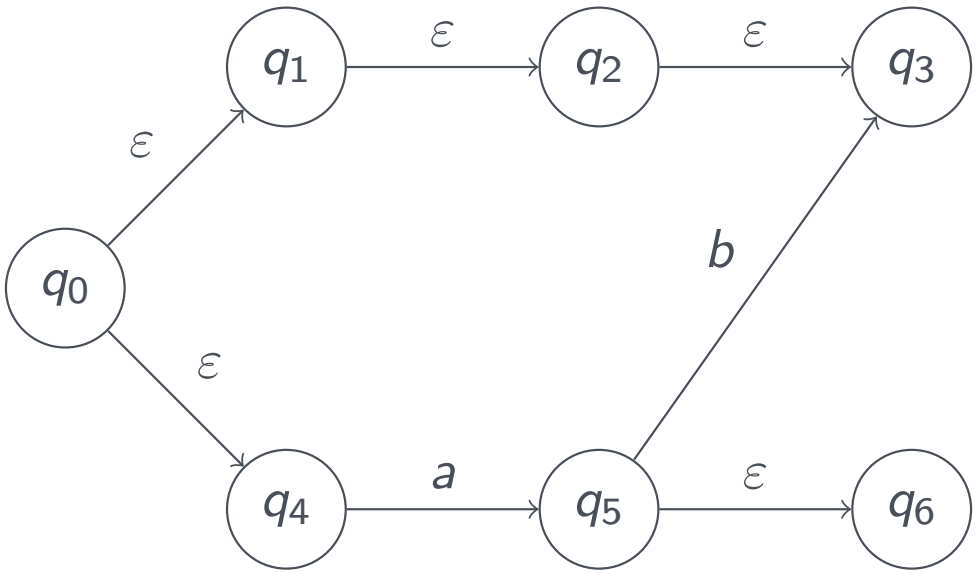
Caso base:

$$q \in \mathrm{ECLOSE}(q)$$

Caso induttivo:

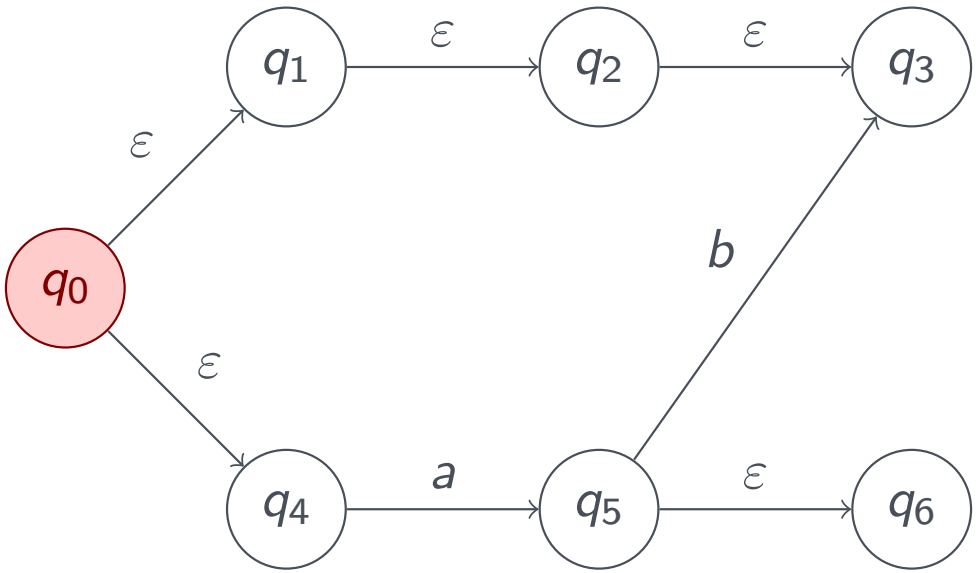
$$\text{se } p \in \mathrm{ECLOSE}(q) \text{ e } r \in \delta(p, \varepsilon) \text{ allora } r \in \mathrm{ECLOSE}(q)$$

Epsilon chiusura: esempio



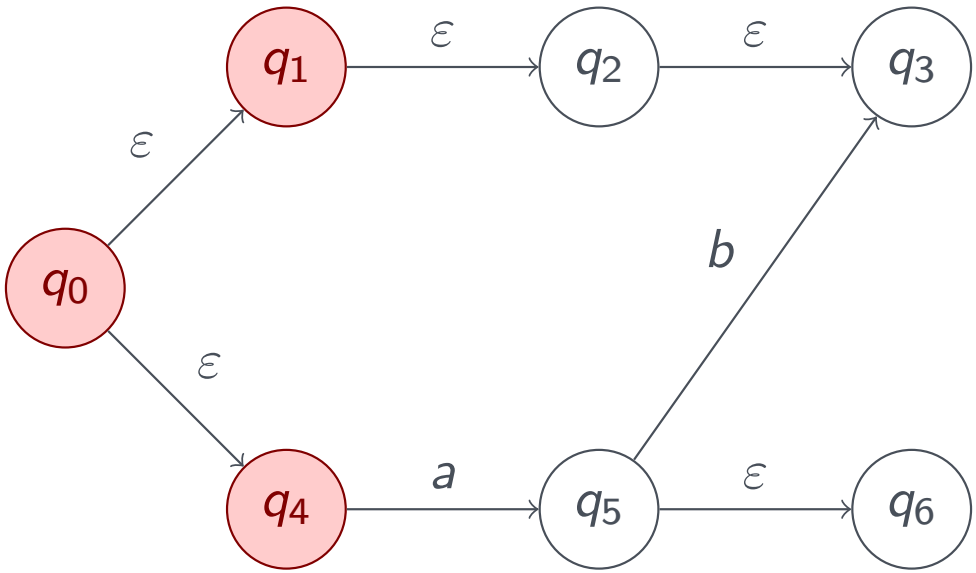
$$\text{ECLOSE}(q_0) = \{$$

Epsilon chiusura: esempio



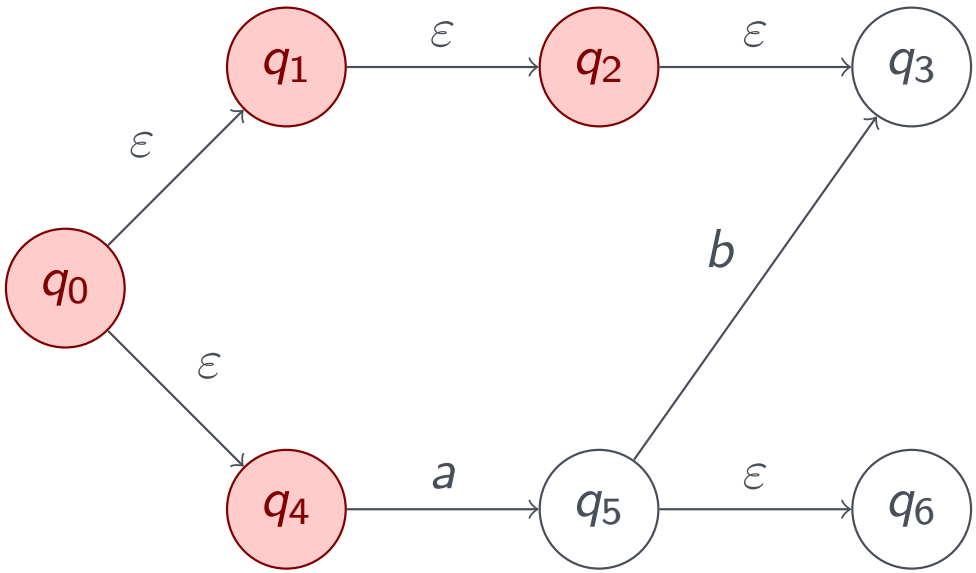
$$\text{ECLOSE}(q_0) = \{q_0$$

Epsilon chiusura: esempio



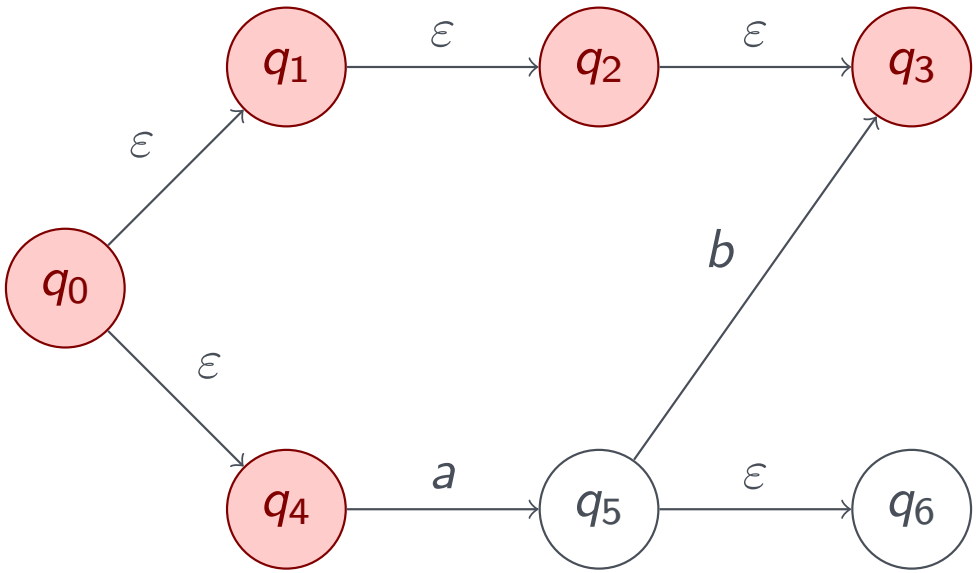
$$\text{ECLOSE}(q_0) = \{q_0, q_1, q_4$$

Epsilon chiusura: esempio



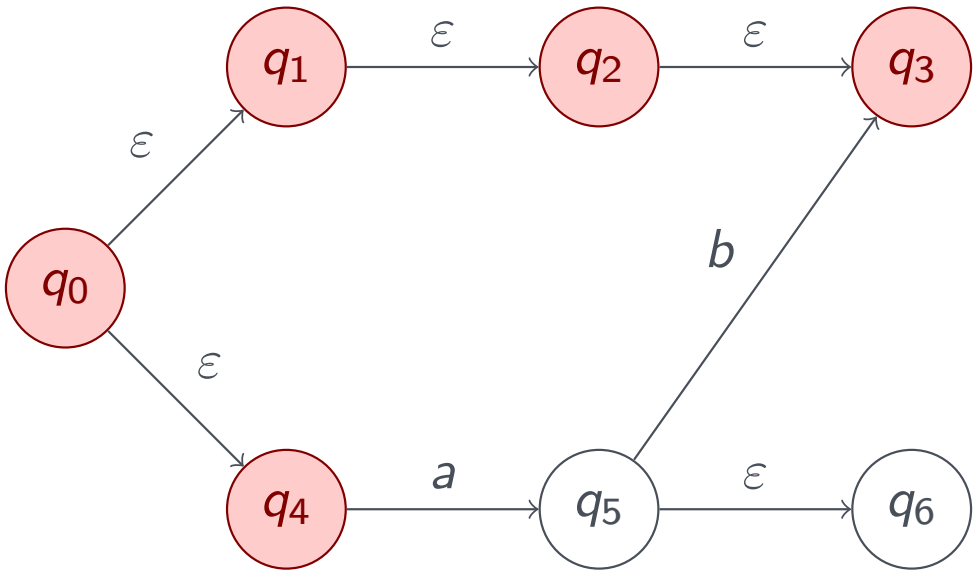
$$\text{ECLOSE}(q_0) = \{q_0, q_1, q_4, q_2$$

Epsilon chiusura: esempio



$$\text{ECLOSE}(q_0) = \{q_0, q_1, q_4, q_2, q_3$$

Epsilon chiusura: esempio



$$\text{ECLOSE}(q_0) = \{q_0, q_1, q_4, q_2, q_3\}$$

Funzione di transizione estesa per ε-NFA

- La funzione di transizione estesa $\hat{\delta}$ per gli ε-NFA:

  Base:

  $$\hat{\delta}(q, \varepsilon) = \text{ECLOSE}(q)$$

  Induzione:

  $$\hat{\delta}(q, w) = \text{ECLOSE}\left(\bigcup_{p \in \hat{\delta}(q,x)} \delta(p, a)\right)$$

  con $w = xa$ (parola x seguita dal simbolo a)

- **Esempio:** calcoliamo $\hat{\delta}(q_0, 5.6)$ alla lavagna
- Formalmente, il linguaggio accettato da A è

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

Equivalenza di DFA e ε-NFA

- Anche in questo caso abbiamo definito una classe di automi che è equivalente ai DFA
- Per ogni ε-NFA E c'è un DFA D tale che $L(E) = L(D)$, e viceversa
- Lo si dimostra modificando la costruzione a sottoinsiemi:

Equivalenza di DFA e ε-NFA

- Anche in questo caso abbiamo definito una classe di automi che è equivalente ai DFA
- Per ogni ε-NFA E c'è un DFA D tale che $L(E) = L(D)$, e viceversa
- Lo si dimostra modificando la costruzione a sottoinsiemi: Dato un ε-NFA

$$E = (Q_E, \Sigma, q_0, \delta_E, F_E)$$

costruiremo un DFA

$$D = (Q_D, \Sigma, S_0, \delta_D, F_D)$$

tale che

$$L(D) = L(E)$$

La costruzione a sottoinsiemi modificata

- $Q_D = \{S \subseteq Q_E : S = \text{ECLOSE}(S)\}$
  Ogni stato è un insieme di stati chiuso per ε-transizioni
- $S_0 = \text{ECLOSE}(q_0)$
  Lo stato iniziale è la ε-chiusura dello stato iniziale di E
- $F_D = \{S \in Q_D : S \cap F_E \neq \emptyset\}$
  Uno stato del DFA è finale se c'è almeno uno stato finale di E
- Per ogni $S \in Q_D$ e per ogni $a \in \Sigma$:

$$\delta_D(S, a) = \text{ECLOSE}\Big(\bigcup_{p \in S} \delta_E(p, a)\Big)$$

  La funzione di transizione "percorre tutte le possibili strade" (comprese quelle con ε-transizioni)

La costruzione a sottoinsiemi modificata

- $Q_D = \{S \subseteq Q_E : S = \text{ECLOSE}(S)\}$
  Ogni stato è un insieme di stati chiuso per ε-transizioni
- $S_0 = \text{ECLOSE}(q_0)$
  Lo stato iniziale è la ε-chiusura dello stato iniziale di E
- $F_D = \{S \in Q_D : S \cap F_E \neq \emptyset\}$
  Uno stato del DFA è finale se c'è almeno uno stato finale di E
- Per ogni $S \in Q_D$ e per ogni $a \in \Sigma$:

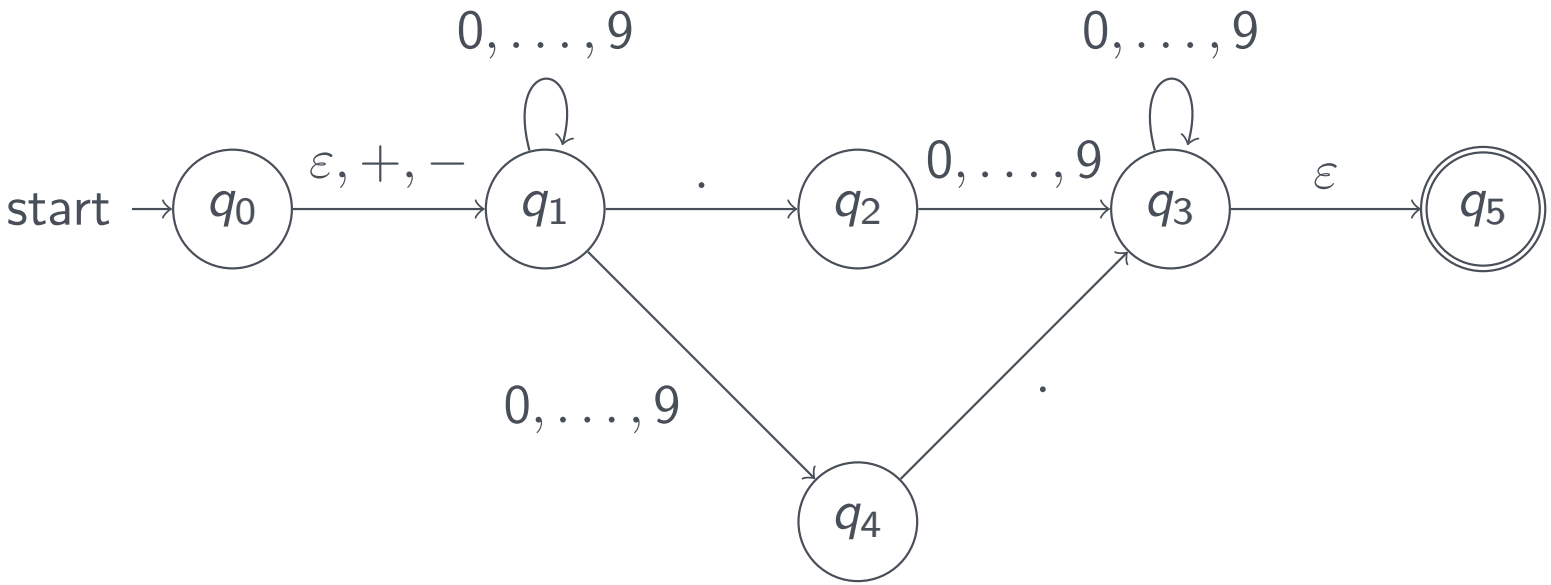
$$\delta_D(S, a) = \text{ECLOSE}\Big(\bigcup_{p \in S} \delta_E(p, a)\Big)$$

  La funzione di transizione "percorre tutte le possibili strade" (comprese quelle con ε-transizioni)

Nota: anche in questo caso $|Q_D| = 2^{|Q_E|}$

Esempio di costruzione a sottoinsiemi (1)

Costruiamo un DFA D equivalente all'ε-NFA E che riconosce i numeri decimali:



Esempio di costruzione a sottoinsiemi (2)

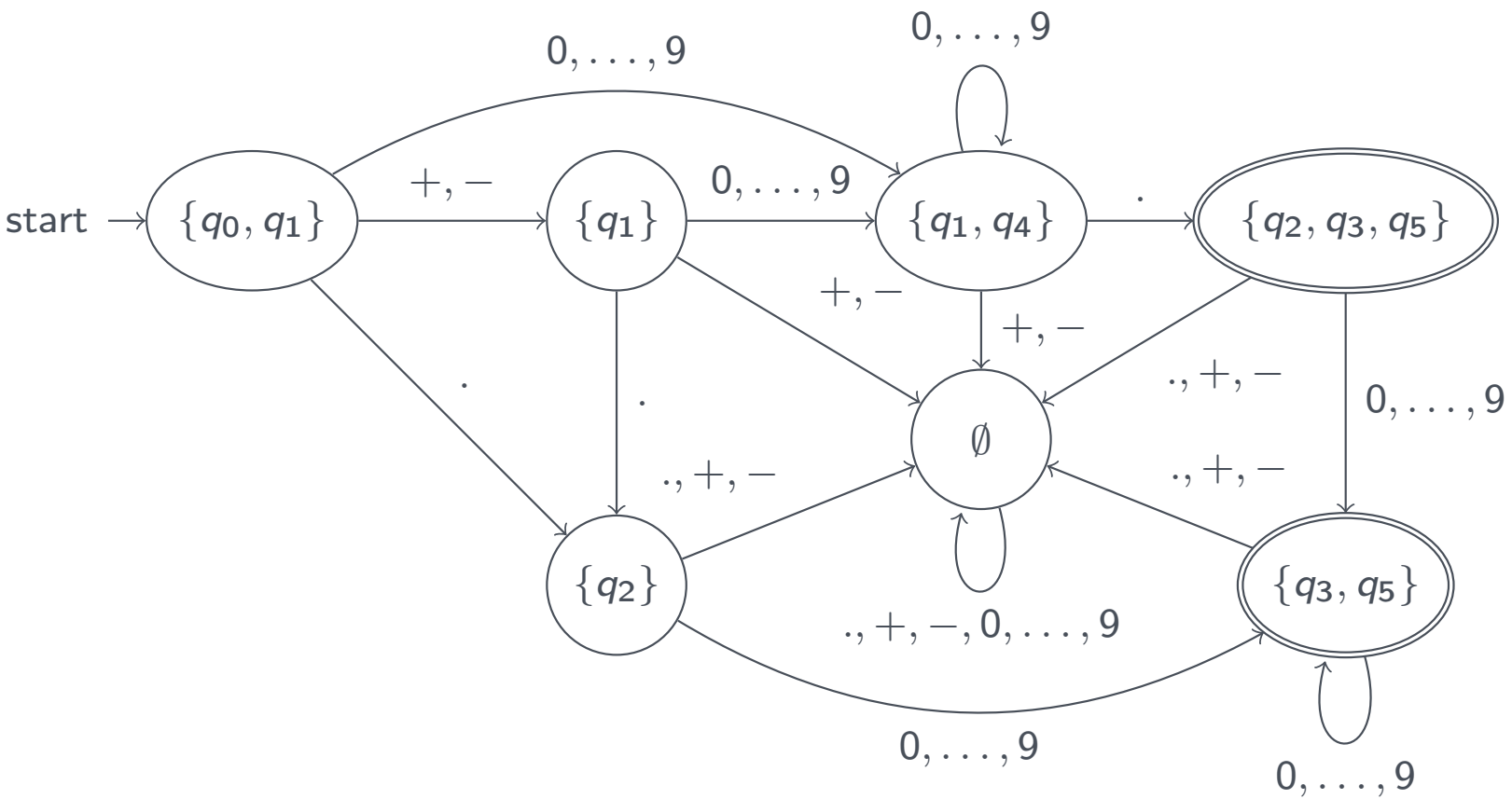
- Come prima cosa costruiamo la ε-chiusura di ogni stato:

$$\begin{aligned} \text{ECLOSE}(q_0) &= \{q_0, q_1\} & \text{ECLOSE}(q_1) &= \{q_1\} \\ \text{ECLOSE}(q_2) &= \{q_2\} & \text{ECLOSE}(q_3) &= \{q_3, q_5\} \\ \text{ECLOSE}(q_4) &= \{q_4\} & \text{ECLOSE}(q_5) &= \{q_5\} \end{aligned}$$

- Lo stato iniziale di D è $\{q_0, q_1\}$

Esempio di costruzione a sottoinsiemi (1)

- Applicando le regole otteniamo il diagramma di transizione:



Theorem

Sia $D = (Q_D, \Sigma, S_0, F_D)$ il DFA ottenuto da un ε-NFA E con la costruzione a sottoinsiemi modificata. Allora $L(D) = L(E)$.

Dimostrazione:

Theorem

Sia $D = (Q_D, \Sigma, S_0, F_D)$ il DFA ottenuto da un ε-NFA E con la costruzione a sottoinsiemi modificata. Allora $L(D) = L(E)$.

Dimostrazione: Prima mostriamo per induzione su $|w|$ che

$$\hat{\delta}_D(S_0, w) = \hat{\delta}_E(q_0, w)$$

Theorem

Sia $D = (Q_D, \Sigma, S_0, F_D)$ il DFA ottenuto da un ε-NFA E con la costruzione a sottoinsiemi modificata. Allora $L(D) = L(E)$.

Dimostrazione: Prima mostriamo per induzione su $|w|$ che

$$\hat{\delta}_D(S_0, w) = \hat{\delta}_E(q_0, w)$$

Base: $w = \varepsilon$. L'enunciato segue dalla definizione:

- Lo stato iniziale di D è $S_0 = \mathrm{ECLOSE}(q_0)$;
- $\hat{\delta}_D(S_0, \varepsilon) = S_0 = \mathrm{ECLOSE}(q_0)$;
- $\hat{\delta}_E(q_0, \varepsilon) = \mathrm{ECLOSE}(q_0)$.

Induzione:

- Sia $|w| = n+1$ e supponiamo vero l'enunciato per la lunghezza n. Scomponiamo w in $w = xa$ (con $|x| = n$ e a simbolo finale)
- Per ipotesi induttiva $\hat{\delta}_D(S_0, x) = \hat{\delta}_E(q_0, x) = \{p_1, \ldots, p_k\}$
- Per la definizione di $\hat{\delta}$ per gli ε-NFA

$$\hat{\delta}_E(q_0, xa) = \text{ECLOSE}\left(\bigcup_{i=1}^{k} \delta_E(p_i, a)\right)$$

- Per la costruzione a sottoinsiemi

$$\delta_D(\{p_1, \ldots, p_k\}, a) = \text{ECLOSE}\left(\bigcup_{i=1}^{k} \delta_E(p_i, a)\right)$$

Induzione (continua):

- Per la definizione di $\hat{\delta}$ per i DFA

$$\hat{\delta}_D(S_0, xa) = \delta_D(\{p_1, \ldots, p_k\}, a) = \text{ECLOSE}\left(\bigcup_{i=1}^{k} \delta_E(p_i, a)\right)$$

- Quindi abbiamo mostrato che $\hat{\delta}_D(S_0, w) = \hat{\delta}_E(q_0, w)$

Poiché sia D che E accettano se solo se $\hat{\delta}_D(S_0, w)$ e $\hat{\delta}_E(q_0, w)$ contengono almeno un stato in F_E, allora abbiamo dimostrato che $L(D) = L(N)$.

Teorema di equivalenza tra DFA e NFA

Theorem

Un linguaggio L è accettato da un DFA se e solo se è accettato da un ε-NFA.

Dimostrazione:

- La parte "se" è il teorema precedente
- La parte "solo se" si dimostra osservando che ogni DFA può essere trasformato in un ε-NFA modificando δ_D in δ_E con la seguente regola:

  Se $\delta_D(q, a) = p$ allora $\delta_E(q, a) = \{p\}$

Esercizio

1. Costruiamo un ε-NFA che riconosce le parole costituite da
   - zero o più a
   - seguite da zero o più b
   - seguite da zero o più c
2. Calcolare ECLOSE di ogni stato dell'automa
3. Convertire l'ε-NFA in DFA