

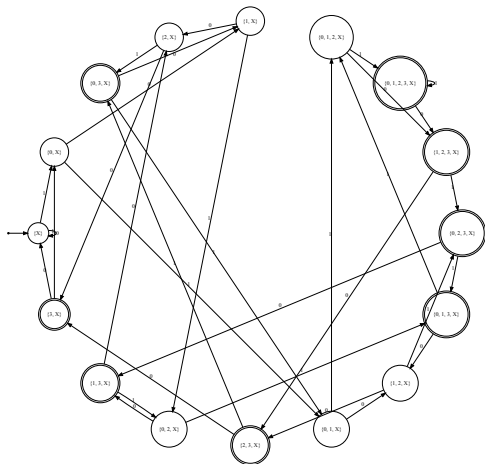
# Automata, Languages and Computation

## Chapter 2 : Finite Automata

Master Degree in Computer Engineering  
University of Padua  
Lecturer : Giorgio Satta

Lecture based on material originally developed by :  
Gösta Grahne, Concordia University

# Finite Automata



- 1 Deterministic finite automata : this is the simplest and most efficient type of FA
- 2 Nondeterministic finite automata : FAs with choices activating independent computations
- 3 Nondeterministic finite automata with  $\epsilon$ -transitions : nondeterministic automata with special moves that do not consume the input

# Deterministic finite automata

These devices read input from left to right

They can only store a quantity of information limited by a constant, using the important notion of **state**

They are easy to implement in a computer (table)

# Deterministic finite automata

A **deterministic finite automaton** (DFA) is a 5-tuple

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

where :

- $Q$  is a **finite** set of **states**
- $\Sigma$  is the input symbol **alphabet**
- $\delta$  is a **transition** function  $Q \times \Sigma \rightarrow Q$
- $q_0 \in Q$  is the initial state
- $F \subseteq Q$  is a set of final states

Comment on the notion of determinism

## Example

A DFA  $A$  that **accepts** the language of all strings of 0 and 1 containing the substring 01 :

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

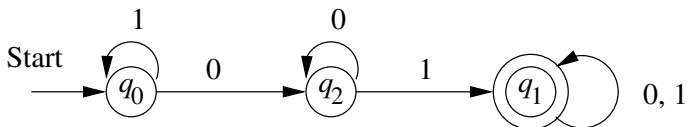
$$Q = \{q_0, q_1, q_2\}, \Sigma = \{0, 1\}, F = \{q_1\}$$

$\delta$  function, specified by means of a **transition table** :

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

# Example

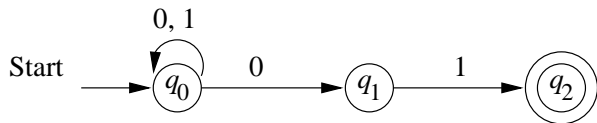
Our DFA  $A$  specified by means of a **transition diagram** :



What is the meaning of  $q_0$ ,  $q_1$ ,  $q_2$ ?

# Test

Does the following transition diagram correspond to a DFA ?





# Acceptance

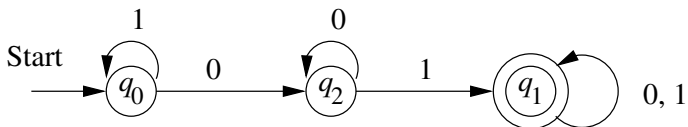
A DFA **accepts** a string  $w = a_1a_2 \cdots a_n$  if there is a path in the transition diagram that

- starts in the initial state
- ends in some final state
- has a sequence of transitions with labels  $a_1a_2 \cdots a_n$

Note that the above is **not** a mathematical definition

# Test

Is the string 01101 accepted by the DFA below ?



## Extended transition function

The  $\delta$  transition function can be **extended** to function  $\hat{\delta}$  defined over state and string pairs (as opposed to state and alphabet symbol pairs)

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

**Induction**  $\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$

## Example

Given a string  $w$  over the alphabet  $\Sigma$  and a symbol  $a$ , let  $\#_a(w)$  denote the number of **occurrences** of  $a$  in  $w$

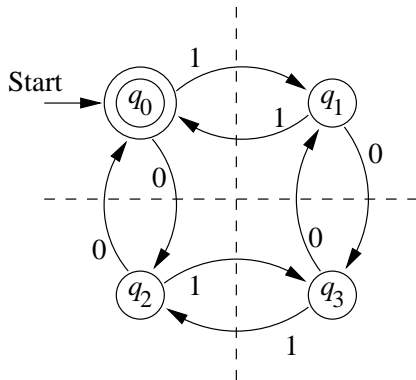
Specify a DFA  $A$  accepting all and only the strings in the following language

$$L = \{w \mid w \in \{0,1\}^*, \#_0(w) \text{ even}, \#_1(w) \text{ even}\}$$

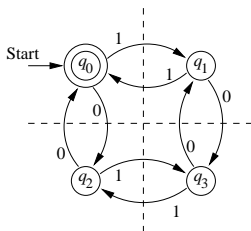
In words,  $L$  contains all and only the binary strings with an even number of 0's and an even number of 1's

What is the shortest string in  $L$ ? Are there strings in  $L$  with odd length?

# Example



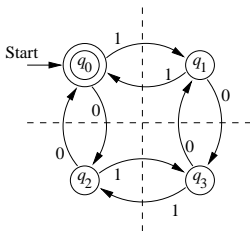
# Example



States have the following meaning

- $q_0$ :  $\#_0(w)$  and  $\#_1(w)$  even
- $q_1$ :  $\#_0(w)$  even,  $\#_1(w)$  odd
- $q_2$ :  $\#_0(w)$  odd,  $\#_1(w)$  even
- $q_3$ :  $\#_0(w)$  and  $\#_1(w)$  odd

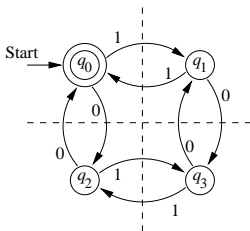
# Example



Tabular representation of the DFA

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

# Example



Is string  $w = 0101$  accepted by  $A$  ?

- $\hat{\delta}(q_0, \epsilon) = q_0$
- $\hat{\delta}(q_0, 0) = \delta(\hat{\delta}(q_0, \epsilon), 0) = \delta(q_0, 0) = q_2$
- $\hat{\delta}(q_0, 01) = \delta(\hat{\delta}(q_0, 0), 1) = \delta(q_2, 1) = q_3$
- $\hat{\delta}(q_0, 010) = \delta(\hat{\delta}(q_0, 01), 0) = \delta(q_3, 0) = q_1$
- $\hat{\delta}(q_0, 0101) = \delta(\hat{\delta}(q_0, 010), 1) = \delta(q_1, 1) = q_0 \in F$



# Language recognized by a DFA

The language **recognized** by DFA  $A$  is

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

The languages accepted by the class of DFAs are called **regular languages**

# Notational conventions

Commonly used notation for DFAs

- $a, b, c, \dots$  alphabet symbols
- $u, v, w, x, y, z$  strings over input alphabet
- $p, q, r, s, q_0, q_1, q_2, \dots$  states

# Test

Specify DFAs for the following languages over the alphabet  $\{0, 1\}$  :

- set of all strings ending in 00
- set of all strings with three consecutive 0's
- set of all strings with 011 as a substring
- set of all strings that start or end (or both) with 01

## Exercise

Consider the language  $L$  of strings over the alphabet  $\{0, 1\}$  with **exactly** one occurrence of string 00

Carry out the following points :

- draw the transition diagram of a DFA  $A$  such that  $L(A) = L$
- state the meaning of each of  $A$ 's states (i.e. for each state of  $A$  describe the strings leading to it)

Hint: define a “failure state” that can never reach any final state

# Nondeterministic finite automata

These automata accept only regular languages

Easier to design than DFAs

Later on we will see several examples of this fact

Very useful for implementing the **search** for a pattern in a text

# Nondeterministic finite automata

A nondeterministic finite automaton can **simultaneously** be in different states

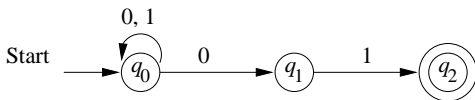
The automaton **accepts** if at least one final state is reached at the end of the scan of the input string

Equivalently, in a given state the automaton can **guess** which next state will lead to acceptance

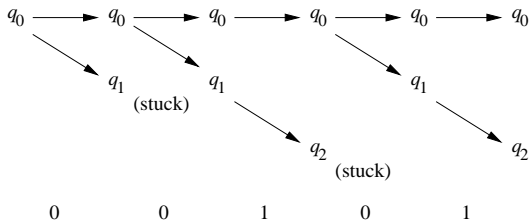
This interpretation is not in the textbook

## Example

Nondeterministic automaton  $N$  accepting all and only the strings ending in 01



Simultaneous computations of  $N$  on input string 00101



# Nondeterministic finite automaton

A **nondeterministic finite automata** (NFA) is a 5-tuple

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

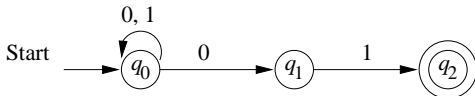
where :

- $Q$  is a **finite** set of **states**
- $\Sigma$  is the **alphabet** of input symbols
- $\delta$  is a **transition** function  $Q \times \Sigma \rightarrow 2^Q$ , where  $2^Q$  is the set of all subsets of  $Q$  (power set)
- $q_0 \in Q$  is the initial state
- $F \subseteq Q$  is the set of final states



## Example

The transition diagram



represents the nondeterministic automaton

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

with transition function  $\delta$

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

# Extended transition function $\hat{\delta}$

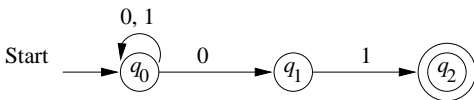
**Base**  $\hat{\delta}(q, \epsilon) = \{q\}$

**Induction**

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

Notice the difference with the case of DFA in the induction part. Can you explain this?

# Example



Computation of  $\hat{\delta}(q_0, 00101)$

- $\hat{\delta}(q_0, \epsilon) = \{q_0\}$
- $\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$
- $\hat{\delta}(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
- $\hat{\delta}(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$
- $\hat{\delta}(q_0, 0010) = \delta(q_0, 0) \cup \delta(q_2, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
- $\hat{\delta}(q_0, 00101) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

## Accepted language for NFA

The **accepted** language for an NFA  $A$  is

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

In words,  $L(A)$  is the set of all strings  $w \in \Sigma^*$  such that  $\hat{\delta}(q_0, w)$  contains **at least one** final state. This amounts to say that at least one computation for  $w$  leads to acceptance

# Test

Consider the following language over  $\Sigma = \{0, 1\}$

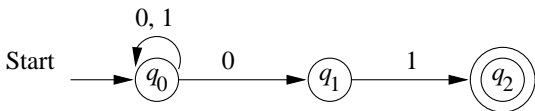
$$L = \{w \mid w = x1ab, x \in \Sigma^*, a, b \in \Sigma\}$$

Informally,  $L$  is the set of all strings with 1 as **third to last** symbol

Specify a NFA  $A$  such that  $L(A) = L$

## Exercise with solution

Show that the NFA



accepts the language  $L = \{x01 \mid x \in \Sigma^*\}$

We prove the following three statements using **mutual induction**:

- (i)  $q_0 \in \hat{\delta}(q_0, w) \Leftrightarrow w \in \Sigma^*$
- (ii)  $q_1 \in \hat{\delta}(q_0, w) \Leftrightarrow w = x0, x \in \Sigma^*$
- (iii)  $q_2 \in \hat{\delta}(q_0, w) \Leftrightarrow w = x01, x \in \Sigma^*$

## Exercise with solution

**Base** If  $|w| = 0$  then  $w = \epsilon$ , and statement (i) follows from the definition of  $\hat{\delta}$ . As for statements (ii) and (iii), both hand sides hold false for  $\epsilon$

**Induction** Assume  $w = xa$ , with  $a \in \{0, 1\}$ ,  $|x| = n$ , and assume statements (i)–(iii) hold true for  $x$ .

- (i) We know that  $q_0 \in \hat{\delta}(q_0, x)$ . From state  $q_0$  we have transitions to  $q_0$  for both 0 and 1

## Exercise with solution

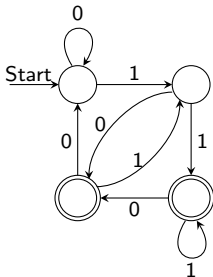
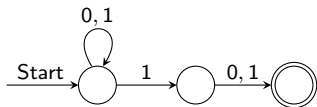
- (ii) (if)  $w$  ends with 0. From (i) we know that  $q_0 \in \hat{\delta}(q_0, x)$ . There is a transition from  $q_0$  to  $q_1$  on symbol 0. Hence  $q_1 \in \hat{\delta}(q_0, w)$
- (only if)  $q_1 \in \hat{\delta}(q_0, w)$ . In order to reach  $q_1$ , the only possible transition is from  $q_0$  upon reading 0. Thus  $w = x0$
- (iii) (if)  $w$  ends with 01, and thus  $a = 1$  and  $x$  ends with 0. From (ii) we have  $q_1 \in \hat{\delta}(q_0, x)$ . From  $q_1$  we can reach  $q_2$  upon reading 1. Then  $q_2 \in \hat{\delta}(q_0, w)$
- (only if)  $q_2 \in \hat{\delta}(q_0, w)$ . In order to reach  $q_2$ , the only possible transition is from  $q_1$  upon reading 1. From statement (ii) for  $x$  we have that  $x$  ends with 0. We then conclude that  $w$  ends with 01 □



## Equivalence for DFA and NFA

NFAs are **easier** than DFAs to “program”, since nondeterminism makes it possible to simplify the structure of the automaton

**Example** : compare NFA and DFA accepting strings in  $\{0, 1\}^*$  with penultimate symbol 1

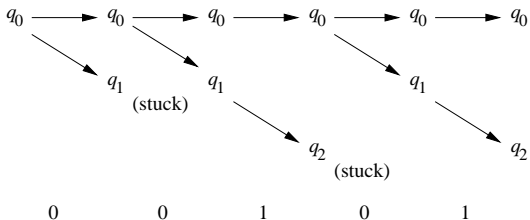


With an increase in the distance between 1 and the end of the string, the gap gets exponentially larger

# Equivalence for DFA and NFA

Quite surprisingly, for every NFA  $N$  there exists some DFA  $D$  such that  $L(D) = L(N)$ . The proof involves the **subset construction**

**Idea** : build a state in  $D$  for every state set representing a “configuration” in a computation of  $N$ . The collection of all configurations is still a **finite set**



# Equivalence for DFA and NFA

Given an NFA

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

the subset construction produces a DFA

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

such that  $L(D) = L(N)$

# Equivalence for DFA and NFA

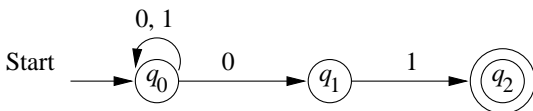
Subset construction :

- $Q_D = \{S \mid S \subseteq Q_N\}$
- $F_D = \{S \subseteq Q_N \mid S \cap F_N \neq \emptyset\}$
- For every  $S \subseteq Q_N$  and  $a \in \Sigma$ ,

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

**Note** :  $|Q_D| = 2^{|Q_N|}$ . Nonetheless, the large majority of states in  $Q_D$  turn out to be **garbage**, that is, they cannot be reached from the initial state

# Example



Construction of  $\delta_D$  :

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

## Example

**Note** :  $D$  states correspond to subsets of  $N$  states, but we could denote  $D$  states in any other way, for instance using the letters  $A, B, \dots, F$

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

## Equivalence for DFA and NFA

We can often avoid **exponential growth** of states in  $Q_D$  using a technique called **lazy evaluation** (or deferred evaluation)

State  $q$  of DFA  $A$  is **accessible** if there is at least one string  $w$  such that  $\hat{\delta}_A(q_0, w) = q$

We build the transition table of  $D$  **only** for the accessible states of  $D$

## Equivalence for DFA and NFA

Construction of DFA  $D$  through lazy evaluation

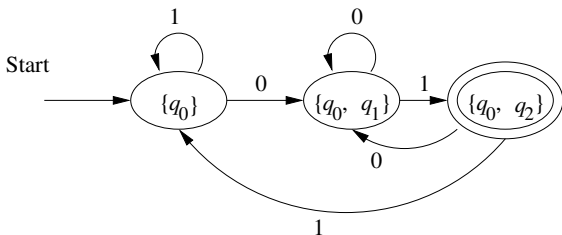
**Base**  $S = \{q_0\}$  is accessible in  $D$

**Induction** If state  $S$  is accessible in  $D$ , then state  $\delta_D(S, a)$  is also accessible in  $D$ , for every  $a \in \Sigma$



# Example

DFA  $D$  with only accessible states



In several **practical applications**  $D$  has about as many states as  $N$

## Equivalence for DFA and NFA

**Theorem** Let  $D$  be the DFA obtained from an NFA  $N$  using the subset construction. Then  $L(D) = L(N)$

**Proof** We first prove that, for every string  $w \in \Sigma^*$ , we have

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

Check that both sides in the above equation are sets!

We use induction on  $|w|$

**Base**  $w = \epsilon$ . The claim follows from the definition

# Equivalence for DFA and NFA

## Induction

$$\begin{aligned}
 \hat{\delta}_D(\{q_0\}, xa) &= \delta_D(\hat{\delta}_D(\{q_0\}, x), a) && \text{definition of } \hat{\delta}_D \\
 &= \delta_D(\hat{\delta}_N(q_0, x), a) && \text{induction} \\
 &= \bigcup_{p \in \hat{\delta}_N(q_0, x)} \delta_N(p, a) && \text{definition of } \delta_D \\
 &= \hat{\delta}_N(q_0, xa) && \text{definition of } \hat{\delta}_N
 \end{aligned}$$

$L(D) = L(N)$  now follows from the definition of  $F_D$



## Equivalence for DFA and NFA

**Theorem** A language  $L$  is accepted by a DFA if and only if  $L$  is accepted by an NFA

**Proof** (If) Previous theorem

(Only if) Any DFA can be converted into an equivalent NFA by modifying  $\delta_D$  into  $\delta_N$  according to the following rule

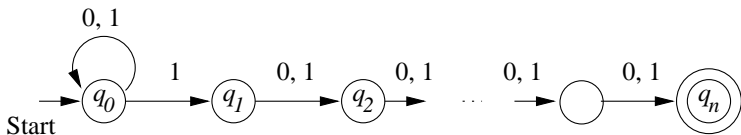
$$\text{If } \delta_D(q, a) = p, \text{ then } \delta_N(q, a) = \{p\}$$

By induction on  $|w|$  one can show that  $\hat{\delta}_D(q_0, w) = p$  if and only if  $\hat{\delta}_N(q_0, w) = \{p\}$  □

## Exponential growth of the state set

**Theorem** There exists an NFA  $N$  with  $n + 1$  states that has no equivalent DFA with less than  $2^n$  states

**Proof** Let  $N$  be the NFA



$$L(N) = \{x1c_2c_3 \cdots c_n \mid x \in \{0, 1\}^*, c_i \in \{0, 1\}\}$$

Intuitively, an equivalent DFA must “remember” the last  $n$  symbols it has read

Those symbols might all be relevant for the final decision

## Exponential growth of the state set

Suppose there exists a DFA  $D$  equivalent to  $N$  with fewer than  $2^n$  states

There are  $2^n$  binary strings of length  $n$ . Since  $D$  has fewer than  $2^n$  states, there must be

- a state  $q$ ,
- binary strings  $a_1a_2 \cdots a_n \neq b_1b_2 \cdots b_n$ ,

such that

$$\hat{\delta}_D(q_0, a_1a_2 \cdots a_n) = \hat{\delta}_D(q_0, b_1b_2 \cdots b_n) = q$$

The above reasoning uses the so-called pigeonhole principle

## Exponential growth of the state set

Since  $a_1 a_2 \cdots a_n \neq b_1 b_2 \cdots b_n$ , there exists  $i$  with  $1 \leq i \leq n$  such that  $a_i \neq b_i$ ; we assume  $a_i = 1$  and  $b_i = 0$  (the other case being symmetrical)

Case 1:  $i = 1$ ; we have

$$\hat{\delta}_D(q_0, 1a_2 \cdots a_n) \in F$$

$$\hat{\delta}_D(q_0, 0b_2 \cdots b_n) \notin F$$

which is a contradiction

## Exponential growth of the state set

Case 2:  $i > 1$ ; since  $\hat{\delta}_D(q_0, a_1 a_2 \cdots a_n) = \hat{\delta}_D(q_0, b_1 b_2 \cdots b_n)$  and  $D$  is deterministic, we have

$$\begin{aligned}\hat{\delta}_D(q_0, a_1 \cdots a_{i-1} 1 a_{i+1} \cdots a_n 0^{i-1}) &= \\ \hat{\delta}_D(q_0, b_1 \cdots b_{i-1} 0 b_{i+1} \cdots b_n 0^{i-1}) &\end{aligned}$$

From the definition of  $L$ , we must have

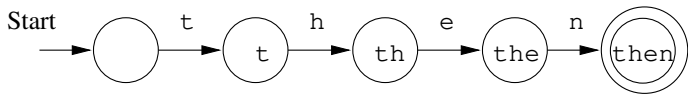
$$\begin{aligned}\hat{\delta}_D(q_0, a_1 \cdots a_{i-1} 1 a_{i+1} \cdots a_n 0^{i-1}) &\in F \\ \hat{\delta}_D(q_0, b_1 \cdots b_{i-1} 0 b_{i+1} \cdots b_n 0^{i-1}) &\notin F\end{aligned}$$

which is a contradiction





# Partial DFA



This is not a DFA, since for some symbols in  $\Sigma$  transitions are not specified

A **partial** DFA has **at most** one outgoing transition for each state in  $Q$  and for each symbol in  $\Sigma$

A partial DFA can be completed to a DFA if we add one non-accepting state having the status of a **trap state**, from which you cannot escape

## Exercise with solution

Consider the NFA

$$N = (\{q_0, q_1\}, \{0, 1\}, \delta_N, q_0, \{q_1\}),$$

where  $\delta_N(q_0, 0) = \{q_0, q_1\}$ ,  $\delta_N(q_0, 1) = \{q_1\}$ ,  $\delta_N(q_1, 0) = \emptyset$ ,  
 $\delta_N(q_1, 1) = \{q_0, q_1\}$

- check whether strings  $w_1 = 101$  and  $w_2 = 0010$  are in  $L(N)$ , showing all steps in the computations
- construct the transition diagram of the DFA equivalent to  $N$
- using a set-former, define the language accepted by the automaton; **suggestion**: this is easier if you look at the DFA

## Exercise with solution

$w_1 = 101 \in L(A)$  ?

- $\hat{\delta}(q_0, \epsilon) = \{q_0\}$
- $\hat{\delta}(q_0, 1) = \delta(q_0, 1) = \{q_1\}$
- $\hat{\delta}(q_0, 10) = \delta(q_1, 0) = \emptyset$ , then  $w_1 = 101 \notin L(A)$

$w_2 = 0010 \in L(A)$  ?

- $\hat{\delta}(q_0, \epsilon) = \{q_0\}$
- $\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$
- $\hat{\delta}(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
- $\hat{\delta}(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_1\} \cup \{q_0, q_1\} = \{q_0, q_1\}$
- $\hat{\delta}(q_0, 0010) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
- since  $\{q_0, q_1\} \cap \{q_1\} \neq \emptyset$ ,  $w_2 = 0010 \in L(A)$

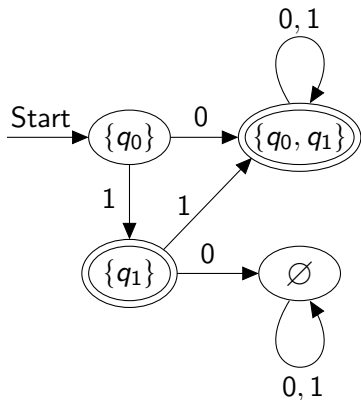
## Exercise with solution

We construct the transition diagram of the equivalent DFA using the subset construction and lazy evaluation

- $\delta_D(\{q_0\}, 0) = \delta_N(q_0, 0) = \{q_0, q_1\}$
- $\delta_D(\{q_0\}, 1) = \delta_N(q_0, 1) = \{q_1\}$
- $\delta_D(\{q_0, q_1\}, 0) = \delta_N(q_0, 0) \cup \delta_N(q_1, 0)$   
 $= \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
- $\delta_D(\{q_0, q_1\}, 1) = \delta_N(q_0, 1) \cup \delta_N(q_1, 1)$   
 $= \{q_1\} \cup \{q_0, q_1\} = \{q_0, q_1\}$
- $\delta_D(\{q_1\}, 0) = \delta_N(q_1, 0) = \emptyset$
- $\delta_D(\{q_1\}, 1) = \delta_N(q_1, 1) = \{q_0, q_1\}$
- $\delta_D(\emptyset, 0) = \delta_D(\emptyset, 1) = \emptyset$
- $\{q_0\}$  initial state,  $\{q_0, q_1\}$  e  $\{q_1\}$  final states

## Exercise with solution

Graphical representation of the transition diagram



## Exercise with solution

Using a set-former, define the language accepted by the automaton

$$L(A) = \{w \in \{0,1\}^+ \mid w = 1 \text{ or } w = 0x \\ \text{or } w = 11y, x, y \in \{0,1\}^*\}$$

# Exercises

Specify an NFA  $A$  for each of the following languages defined on the alphabet  $\{0, 1\}$

- set of strings with two consecutive 0 or two consecutive 1
- set of strings such that at least one of the last three symbols is 1

## NFA with $\epsilon$ -transitions

Extension of NFAs where transitions labelled with symbol  $\epsilon$  are allowed; this means that the automaton can change state **without consuming** any of its input

They accept all and only the regular languages

Easier to design than NFAs

$\epsilon$ -NFA widely used in compilers and for search of patterns in a text



## Example

A fractional number consists of

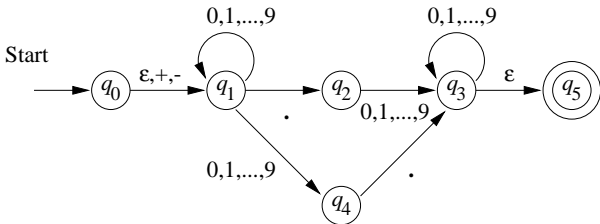
- + or - sign, optional
- a first string of digits
- one decimal point
- a second string of digits

with the first or the second strings **optional**, but not both

This example comes from a lexical analyser in compiler theory

# Example

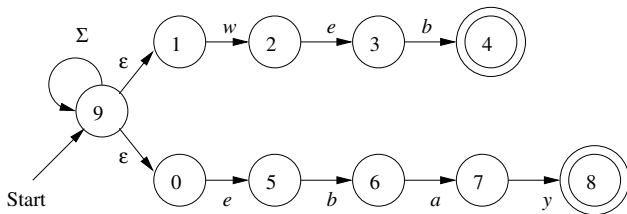
$\epsilon$ -NFA accepting fractional numbers



The  $\epsilon$ -transition makes operators + and - optional

# Example

$\epsilon$ -NFA accepting set of keywords {ebay, web}



The  $\epsilon$ -transition makes it easy to combine several automata

# NFA with $\epsilon$ -transitions

A **nondeterministic finite automaton with  $\epsilon$ -transitions**

( $\epsilon$ -NFA) is a 5-tuple

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

where

- $Q$ ,  $\Sigma$ ,  $q_0$ , and  $F$  are defined as for NFAs
- $\delta$  is a **transition** function  $Q \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^Q$ , with  $2^Q$  denoting the class of subsets of  $Q$

# Example

$\epsilon$ -NFA accepting fractional numbers

$$E = (\{q_0, q_1, \dots, q_5\}, \{., +, -, 0, 1, \dots, 9\}, \delta, q_0, \{q_5\})$$

Transition function  $\delta$

	$\epsilon$	$+, -$	$.$	$0, \dots, 9$
$\rightarrow q_0$	$\{q_1\}$	$\{q_1\}$	$\emptyset$	$\emptyset$
$q_1$	$\emptyset$	$\emptyset$	$\{q_2\}$	$\{q_1, q_4\}$
$q_2$	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_3\}$
$q_3$	$\{q_5\}$	$\emptyset$	$\emptyset$	$\{q_3\}$
$q_4$	$\emptyset$	$\emptyset$	$\{q_3\}$	$\emptyset$
$\star q_5$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

# Test

Specify an  $\epsilon$ -NFA accepting the language of strings over  $\{a, b, c\}$  with zero or more  $a$ 's, followed by zero or more  $b$ 's, followed by zero or more  $c$ 's

## $\epsilon$ -closure

Let us compute the  **$\epsilon$ -closure** of a state  $q$ , written  $\text{ECLOSE}(q)$ , adding all the states reachable from  $q$  itself through a sequence of one or more symbols  $\epsilon$

Needed later in the definition of  $\hat{\delta}$  function

**Base**  $q \in \text{ECLOSE}(q)$

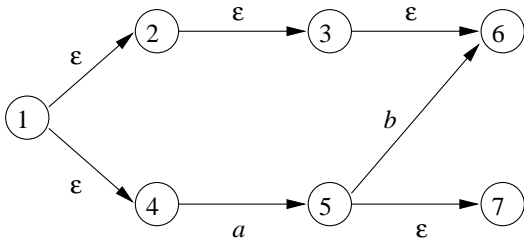
**Induction**  $(p \in \text{ECLOSE}(q) \wedge r \in \delta(p, \epsilon)) \Rightarrow r \in \text{ECLOSE}(q)$

Extension to set of states  $S$

$$\text{ECLOSE}(S) = \bigcup_{q \in S} \text{ECLOSE}(q)$$

# Example

$\epsilon$ -NFA fragment



$$\text{ECLOSE}(1) = \{1, 2, 3, 4, 6\}$$

$$\text{ECLOSE}(\{4, 5\}) = \{4\} \cup \{5, 7\} = \{4, 5, 7\}$$



# Extended transition function $\hat{\delta}$

**Base**  $\hat{\delta}(q, \epsilon) = \text{ECLOSE}(q)$

**Induction**  $\hat{\delta}(q, xa)$  is computed as

- $\{p_1, \dots, p_k\} = \hat{\delta}(q, x)$
- $\{r_1, \dots, r_m\} = \bigcup_{i=1}^k \delta(p_i, a)$
- $\hat{\delta}(q, xa) = \text{ECLOSE}(\{r_1, \dots, r_m\})$

Note that processing of  $\epsilon$  symbols is accounted for after the processing of each symbol in  $\Sigma$

## Example

We compute  $\hat{\delta}(q_0, 5.6)$  for the  $\epsilon$ -NFA accepting fractional numbers

$$\hat{\delta}(q_0, \epsilon) = \text{ECLOSE}(q_0) = \{q_0, q_1\}$$

Computation of  $\hat{\delta}(q_0, 5)$  :

- $\delta(q_0, 5) \cup \delta(q_1, 5) = \emptyset \cup \{q_1, q_4\} = \{q_1, q_4\}$
- $\text{ECLOSE}(q_1) \cup \text{ECLOSE}(q_4) = \{q_1\} \cup \{q_4\} = \{q_1, q_4\} = \hat{\delta}(q_0, 5)$

## Example

Computation of  $\hat{\delta}(q_0, 5.)$  :

- $\delta(q_1, .) \cup \delta(q_4, .) = \{q_2\} \cup \{q_3\} = \{q_2, q_3\}$
- $\text{ECLOSE}(q_2) \cup \text{ECLOSE}(q_3) = \{q_2\} \cup \{q_3, q_5\} = \{q_2, q_3, q_5\} = \hat{\delta}(q_0, 5.)$

Computation of  $\hat{\delta}(q_0, 5.6)$  :

- $\delta(q_2, 6) \cup \delta(q_3, 6) \cup \delta(q_5, 6) = \{q_3\} \cup \{q_3\} \cup \emptyset = \{q_3\}$
- $\text{ECLOSE}(q_3) = \{q_3, q_5\} = \hat{\delta}(q_0, 5.6)$

# Accepted language for $\epsilon$ -NFA

The language **accepted** by  $\epsilon$ -NFA  $E = (Q, \Sigma, \delta, q_0, F)$  is

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

## From $\epsilon$ -NFA to DFA

Given the  $\epsilon$ -NFA

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

we construct a DFA

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

such that  $L(D) = L(E)$

Construction details :

- $Q_D = \{S \mid S \subseteq Q_E, S = \text{ECLOSE}(S)\}$
- $q_D = \text{ECLOSE}(q_0)$
- $F_D = \{S \mid S \in Q_D, S \cap F_E \neq \emptyset\}$

# From $\epsilon$ -NFA to DFA

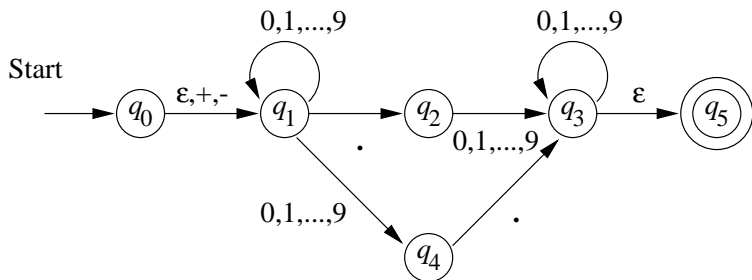
Construction details (cont'd)

Computation of  $\delta_D(S, a)$ ,  $a \in \Sigma$  and  $S \in Q_D$

- $S = \{p_1, \dots, p_k\}$
- $\{r_1, \dots, r_m\} = \bigcup_{i=1}^k \delta_E(p_i, a)$
- $\delta_D(S, a) = \text{ECLOSE}(\{r_1, \dots, r_m\})$

# Example

$\epsilon$ -NFA  $E$



## Example

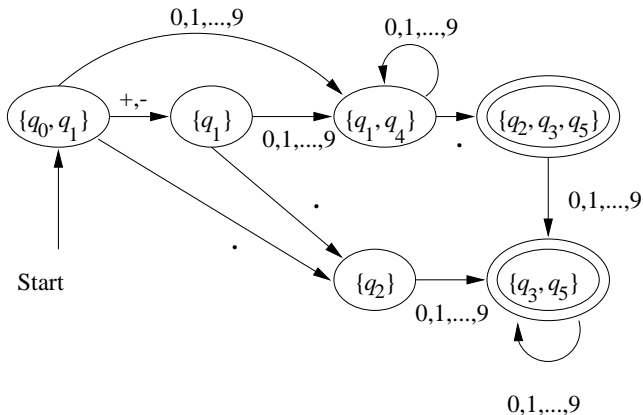
Computation of some of the values of  $\delta_D$

- $\delta_D(\{q_0, q_1\}, +) = \text{ECLOSE}(\delta_E(q_0, +) \cup \delta_E(q_1, +)) = \text{ECLOSE}(\{q_1\}) = \{q_1\}$
- $\delta_D(\{q_1\}, 0) = \text{ECLOSE}(\delta_E(q_1, 0)) = \text{ECLOSE}(\{q_1, q_4\}) = \{q_1, q_4\}$
- $\delta_D(\{q_1, q_4\}, \cdot) = \text{ECLOSE}(\delta_E(q_1, \cdot) \cup \delta_E(q_4, \cdot)) = \text{ECLOSE}(\{q_2, q_3\}) = \{q_2, q_3, q_5\}$
- $\delta_D(\{q_2, q_3, q_5\}, 0) = \text{ECLOSE}(\delta_E(q_2, 0) \cup \delta_E(q_3, 0) \cup \delta_E(q_5, 0)) = \text{ECLOSE}(\{q_3\} \cup \{q_3\} \cup \emptyset) = \text{ECLOSE}(\{q_3\}) = \{q_3, q_5\}$
- ...



## Example

DFA  $D$  constructed from  $E$ ; the DFA has been further simplified, omitting the trap state and all transitions leading to that state



## Equivalence between $\epsilon$ -NFA and DFA

**Theorem** A language  $L$  is recognized by  $\epsilon$ -NFA  $E$  if and only if  $L$  is recognized by DFA  $D$

### Proof

(If) Convert  $\delta_D(q, a) = p$  into  $\delta_E(q, a) = \{p\}$ . Then add  $\delta_E(q, \epsilon) = \emptyset$  for each state  $q$  of  $D$

(Only if) Using our construction for  $D$ , we prove  $\hat{\delta}_E(q_0, w) = \hat{\delta}_D(q_D, w)$  by induction on  $|w|$

**Base**  $\hat{\delta}_E(q_0, \epsilon) = \text{ECLOSE}(q_0) = q_D = \hat{\delta}(q_D, \epsilon)$

## Equivalence between $\epsilon$ -NFA and DFA

**Induction** Let  $w = xa$ . We show  $\hat{\delta}_E(q_0, xa) = \hat{\delta}_D(q_D, xa)$  using the inductive hypothesis  $\hat{\delta}_E(q_0, x) = \hat{\delta}_D(q_D, x)$

Let  $\hat{\delta}_E(q_0, x) = \{p_1, \dots, p_k\}$

From the definition of  $\hat{\delta}_E$

- $\{r_1, \dots, r_m\} = \bigcup_{i=1}^k \delta_E(p_i, a)$
- $\hat{\delta}_E(q_0, xa) = \text{ECLOSE}(\{r_1, \dots, r_m\})$

## Equivalence between $\epsilon$ -NFA and DFA

From the inductive hypothesis  $\hat{\delta}_D(q_D, x) = \{p_1, \dots, p_k\}$

Using the definition of  $D$  we compute  $\delta_D(\{p_1, \dots, p_k\}, a)$

- $\{r_1, \dots, r_m\} = \bigcup_{i=1}^k \delta_E(p_i, a)$
- $\delta_D(\{p_1, \dots, p_k\}, a) = \text{ECLOSE}(\{r_1, \dots, r_m\})$

We can now write

$$\begin{aligned}\hat{\delta}_D(q_D, xa) &= \delta_D(\hat{\delta}_D(q_D, x), a) \\ &= \delta_D(\{p_1, \dots, p_k\}, a) \\ &= \text{ECLOSE}(\{r_1, \dots, r_m\}) \\ &= \hat{\delta}_E(q_0, xa)\end{aligned}$$

