AUTOMATA THEORY AND FORMAL LANGUAGES

2022-23

UNIT 3: FINITE AUTOMATA

José Luis Mira Peidro jmira@inf.uc3m.es
Universidad Carlos III de Madrid

Sequential Machines and Finite Automata. Bibliography

- Enrique Alfonseca Cubero, Manuel Alfonseca Cubero, Roberto Moriyón Salomón. Teoría de Autómatas y Lenguajes Formales. McGraw-Hill (2007). Chapters 3 and 7.

- John E. Hopcroft, Rajeev Motwani, Jeffrey D.Ullman. Introduction to Automata Theory, Languages, and Computation (3rd edition). Ed, Pearson Addison Wesley. Sects. 2.1-2.2; Sects. 2.3-2.8; Chap. 4; Sects. 3-1-3.7

- Manuel Alfonseca, Justo Sancho, Miguel Martínez Orga. Teoría de Lenguajes, Gramáticas y Autómatas. Publicaciones R.A.E.C. 1997 Capítulos 4,5,y 8

OUTLINE

- Sequential machines
- Finite Automata
- Deterministic Finite Automata (DFA)
  - Representation and Basic Concepts
  - Equivalence and Minimization
- Nondeterministic Finite Automata (NFA)
- DFA equivalent to a NFA (NFA → DFA)

OUTLINE

- **Sequential machines**
- Finite Automata
- Deterministic Finite Automata (DFA)
  - Representation and Basic Concepts
  - Equivalence and Minimization
- Nondeterministic Finite Automata (NFA)
- DFA equivalent to a NFA (NFA → DFA)

Sequential Machines. Definitions

- **Sequential Machine = (Σ_I, Σ_O, Q, f, g)**
  - Σ_I: Input Alphabet
  - Σ_O: Output Alphabet
  - Q : Finite nonempty set of states (alphabet or set of states)
  - f : Transition function

    $f : Q \times \Sigma_i \rightarrow Q \quad , \quad f(q,a) = q'$
  - g: **Output function**

Sequential Machines. Definitions

- Device that it is able of:
  - Taking different states $\in Q$
  - Receiving environmental information, words $\in \Sigma_I$
  - Acting on the environment, words $\in \Sigma_O$
  - Time is quantified, for each time t:
    - It can only be in a state $\in Q$
    - Receive a stimulus, symbol $\in \Sigma_I$
    - Generate an output, symbol $\in \Sigma_O$
    - Given the input and the current state, we can predict the output and the next state.

Sequential Machines. Definitions

- Two types of sequential machines considering *g*:

- Mealy sequential machine
  - $g : Q \times \Sigma_I \rightarrow \Sigma_O$
  - $g(q, a) = b$
- Moore sequential machine
  - $g : Q \rightarrow \Sigma_O$
  - $g(q) = b$

<u>Rate for transmitting information in the sequential machine</u>

- Infinite, the output only depends on the input and the state

- Finite, the output only depends on the state.
- Moore SM: specific case of a Mealy SM.

Sequential Machines. Definitions

- Sequential machines can be represented by:
  - Two tables:
    - Transitions table, table of *f*
      - Table of double-inputs.
    - Outputs table, table of *g*
      - Mealy sequential machine: Table of double inputs.
      - Moore sequential machine: Table of simple inputs.
  - Transition diagram.

Sequential Machines. Definitions

- Table of transitions and outputs, only one table:
  - Rows: possible states, $q_i \in Q$
  - Cols: Symbols of the input alphabet, $a_m \in \Sigma_I$

f,g

$Q \backslash \Sigma_I$	a_1	...	a_m
q_1	q_i / b_j		
...			
q_n			

Mealy Sequential Machine

$f(q, a) = q'$ $g(q, a) = b$

f,g

$Q \backslash \Sigma_I$	a_1	...	a_m
q_1/b_j	q_i		
...			
q_n/b_k			

Moore Sequential Machine

$f(q, a) = q'$ $g(q) = b$

Sequential Machines. Definitions

- **Transitions diagram:**

➡ Directed graph:

  - Each node is a state in *Q.*
  - Branches link states, represent transitions between states, the inputs of the SM are also represented.

  - Outputs:
    - Mealy SM: Outputs are represented in the transitions.
    - Moore SM: Outputs are represented in the states.

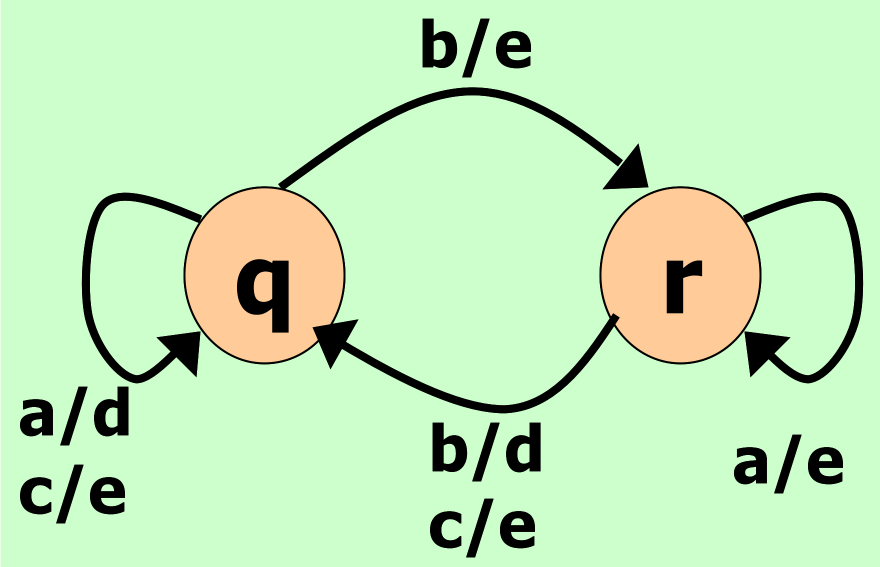
Sequential Machines. Example of representation of a Mealy SM

{(a,b,c), (e,d), (q,r), *f*, *g*)}

- f (q, a) = q
- f (q, b) = r
- f (q, c) = q
- f (r, a) = r
- f (r, b) = q
- f (r, c) = q

- g (q, a) = d
- g (q, b) = e
- g (q, c) = e
- g (r, a) = e
- g (r, b) = d
- g (r, c) = e

Q \ Σ_E	a	b	c
q	q /d	r /e	q /e
r	r /e	q /d	q /e



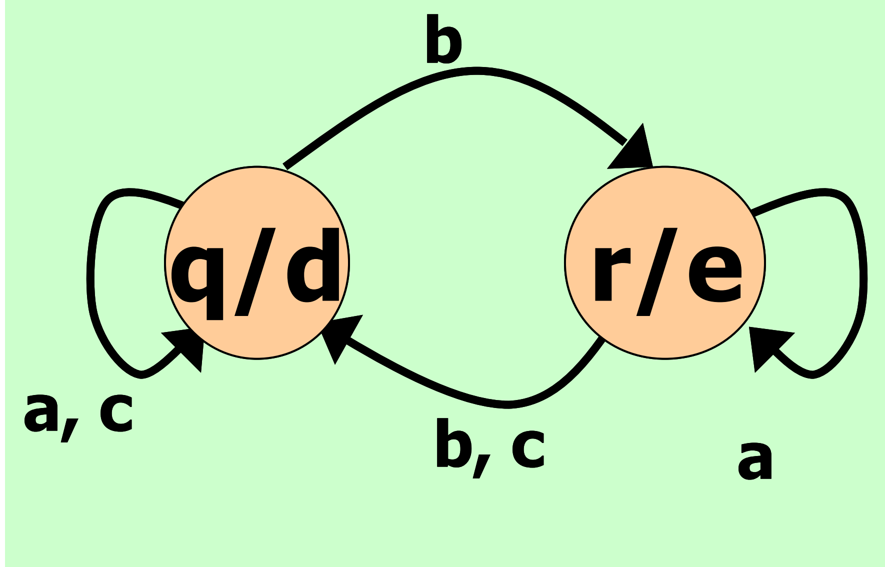
Sequential Machines. Example of representation of a Moore SM

{(a,b,c), (e,d), (q,r), *f*, *g*)}

- *f (q, a) = q*
- f (q, b) = r
- f (q, c) = q
- f (r, a) = r
- f (r, b) = q
- f (r, c) = q

- *g* (q) = d
- *g* (r) = e

$Q \backslash \Sigma_E$	a	b	c
q/d	q	r	q
r/e	r	q	q



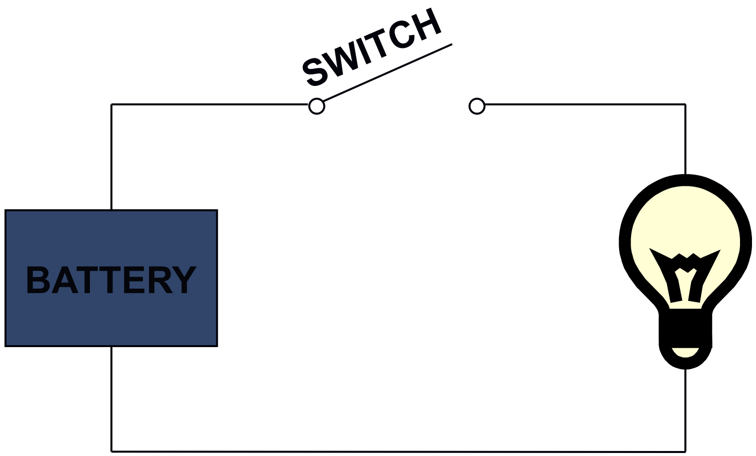
OUTLINE

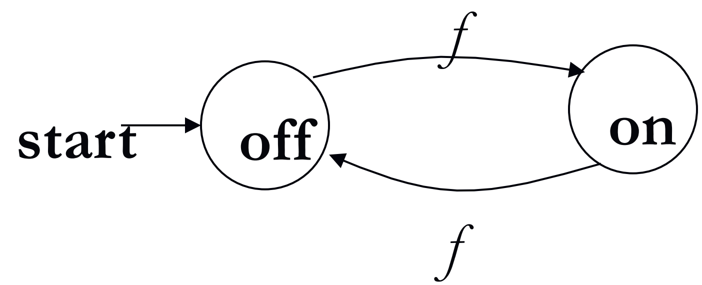
- Sequential machines
- **Finite Automata**
- Deterministic Finite Automata (DFA)
  - Representation and Basic Concepts
  - Equivalence and Minimization
- Nondeterministic Finite Automata (NFA)
- DFA equivalent to a NFA (NFA → DFA)

Finite Automata: Introduction

- A finite automata consists of:
  - A finite set of states, including a **start state** and one or more final states.
  - An alphabet Σ of possible input symbols.
  - A finite set of transitions.

Finite Automata: Introduction





input: switch

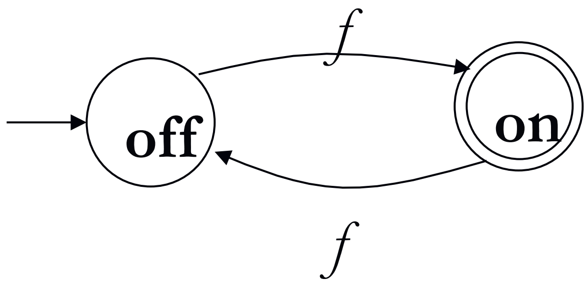
output: light bulb

actions: f **for "flip switch"**

states: on, off

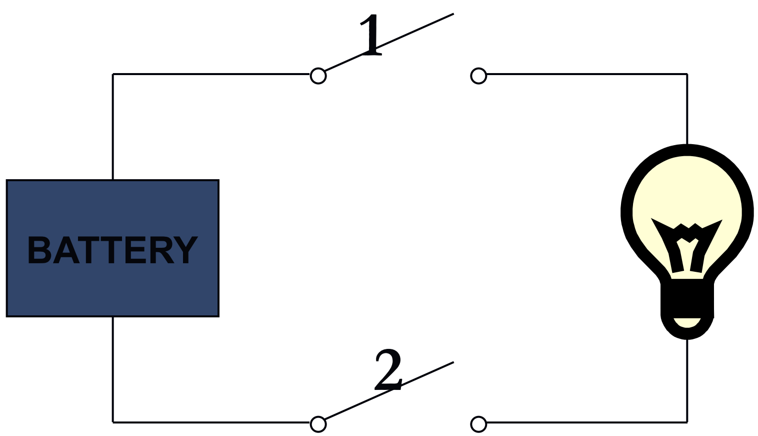
bulb is on if and only if there was an odd number of flips

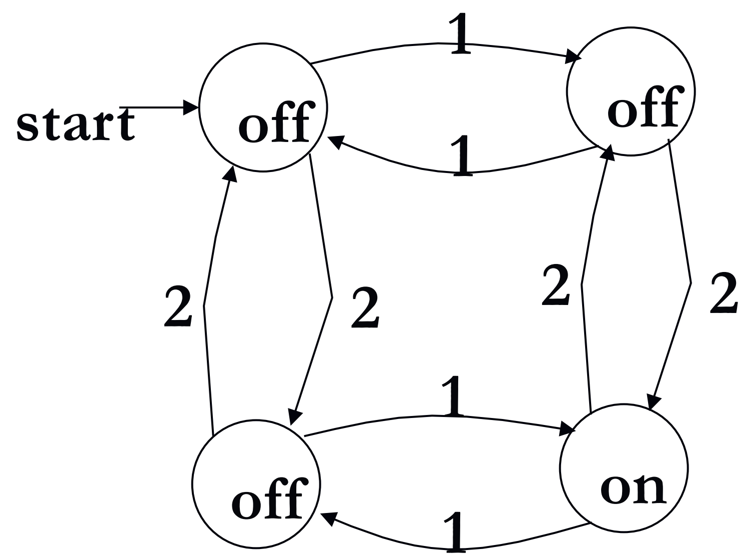
Finite Automata: Introduction



- There are states off and on, the automaton starts in off and tries to reach the "good state" on
- What sequences of fs lead to the good state?
- Answer: $\{f, fff, fffff, \ldots\} = \{f^{\,n}\!: n \text{ is odd}\}$
- This is an example of a deterministic finite automaton over alphabet $\{f\}$

Finite Automata: Introduction





inputs: switches 1 and 2

actions: 1 **for "flip switch 1"**
2 **for "flip switch 2"**

states: on, off

bulb is on if and only if both switches were flipped an odd number of times

OUTLINE

- Sequential machines
- Finite Automata
- **Deterministic Finite Automata (DFA)**
  - **Representation and Basic Concepts**
  - Equivalence and Minimization
- Nondeterministic Finite Automata (NFA)
- DFA equivalent to a NFA (NFA → DFA)

Finite Automata: DFA and NFA

- Types of finite automata:
  - **Deterministic:**
    - Each combination **(State, input symbol)** produces a single (State)
  - **Nondeterministic:**
    - Each combination **(state, input symbol)** produces several **(state$_1$, state$_2$, ..., states$_i$)**
    - Transitions with λ are valid.

Deterministic Finite Automata

Deterministic finite automata (DFA):

$$DFA=(\Sigma, Q, f, q_0, F)$$

- Σ is the alphabet of possible input symbols.
- *Q is the set of states*
- $q_0 \in Q$ *is the start state*
- $F \subseteq Q$ *is the set of final states*
- *f is the transition function*

$$f : Q \times \Sigma \rightarrow Q$$

There are not outputs (Moore Machine)

Nondeterministic Finite Automata

Nondeterministic finite automata:

$$NFA=(\Sigma, Q, f, q_0, F)$$

- Σ is the alphabet of possible input symbols.
- *Q is the set of states*
- *$q_0 \in Q$ is the start state*
- *$F \subseteq Q$ is the set of final states*
- *f is the transition fuction*

$$f : Q \times (\Sigma \cup \{\lambda\}) \rightarrow P(Q)$$

There are not outputs (Moore Machine)

Finite Automata: DFA and NFA

Deterministic finite automata (DFA):

1. There are not moves on input λ.
2. For each state *s* and input symbol *a*, there is exactly one edge out of *s* labeled as *a*.

Nondeterministic finite automata (NFA):

1. More than one edge with the same label from any state is allowed.
2. Some states for which certain input symbols have no edge are allowed.
3. λ -NFA: λ transitions allowed.

DFA: Representation

- DFA can be represented using transition tables or transition diagrams:

1. **Transition tables**:
   - rows contain States($q_i \in Q$)
   - columns contain input symbols ($e_i \in \Sigma$)

	e_1	e_2	…	e_n
→ q_1		$f(q_1, e_2)$		
…				
$*q_m$				

DFA: Representation

- DFA can be represented using transition tables or transition diagrams:
  - **Transition diagrams**:
    - nodes labeled by States ($q_i \in Q$)
    - arcs between nodes q_i to q_j labeled with e_i if exists $f(q_i,e_i) = q_j$
    - q_0 is notated by a ➡
    - $q \in F$ is notated by * or a double circle

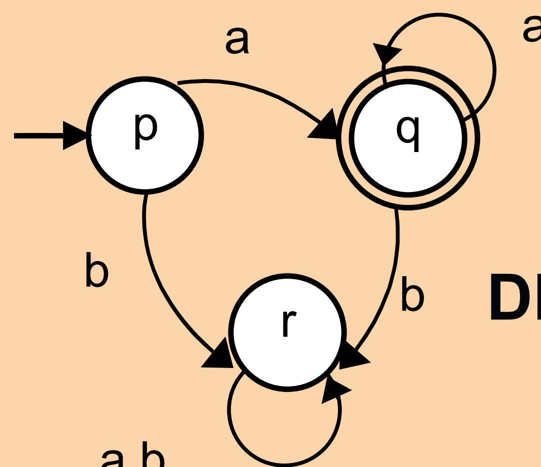
DFA: Representation

- **Example**: Given the DFA_1 = ({a,b}, {p,q,r}, f, p, {q}) where f is defined by:

  f(p,a) = q f(p,b) = r
  f(q,a) = q f(q,b) = r
  f(r,a) = r f(r,b) = r

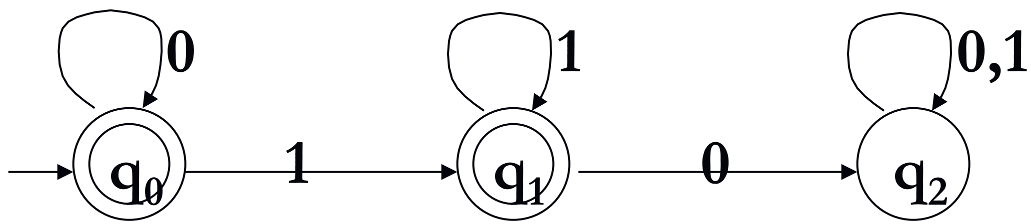
  Write the transition table and the transition diagram.

	a	b
→ **p**	**q**	**r**
***q**	**q**	**r**
r	**r**	**r**



DFA_1

DFA: Example of Representation



alphabet $\Sigma = \{0, 1\}$
start state $\mathcal{Q} = \{q_0, q_1, q_2\}$
initial state q_0
accepting states $F = \{q_0, q_1\}$

transition function δ**:**

inputs

states		0	1
	q_0	q_0	q_1
	q_1	q_2	q_1
	q_2	q_2	q_2

DFA: Basic Concepts

- **Configuration:** ordered pair (q,w) where:
  - q: current state of the DFA.
  - w: string that it is still to be read, $w \in \Sigma^*$
  - Initial configuration: (q_0, t)
    - q_0: initial state
    - t: string to be recognized by the DFA $\in \Sigma^*$
  - Final configuration: (q_i,λ)
    - q_i: final state
    - λ: the input string has been completely read
- **Movement:** it is the transit between two configurations.

$$(q,aw) \xrightarrow[f(\,q\,,\,a\,)\,=\,q']{} (q',w)$$

$a,w \in \Sigma^*$

DFA: Basic Concepts

- **DFA as a language recognizer:**
  - When a DFA transits from q_0 to a final state in several movements➡ **RECOGNITION** or **ACCEPTANCE** of the input string.
  - When a DFA is not able to reach a final state, the AF **DOES NOT RECOGNICE** the input string and this is **NOT INCLUDED** in the language recognized by the FA.

DFA: Basic Concepts

Next, we are going to study how to formalize:

- Movement: extension of the transition function to the case of words.

- Language recognized by a DFA.

DFA: Basic Concepts

- **Extension to a word of the transition function f:**
  - Expand its definition to words in Σ^*

    $f': Q \times \Sigma^* \rightarrow Q$

    - From f, which only considers words of length 1, it is necessary to add:

      $f'(q,\lambda) = q \quad \forall\, q \in Q$

      $f'(q, ax) = f'(f(q,a),x)\ \forall\, q \in Q,\ \boxed{a \in \Sigma \text{ and } x \in \Sigma^*}$

DFA: Basic Concepts

- **In the DFA_1, indicate the result of the following expressions:**

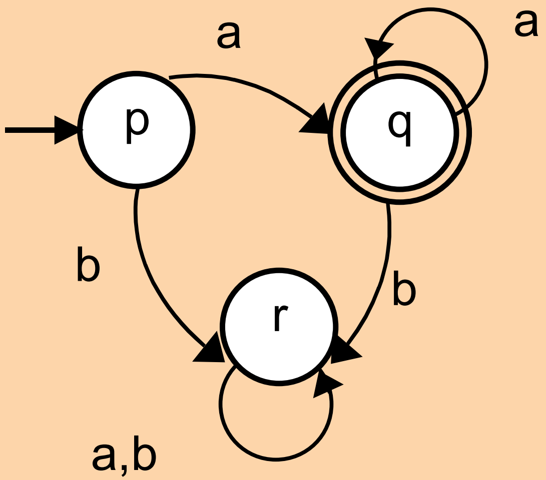
  $f'(p,\lambda)$

  $f'(p, a^n)$

  $f'(p,bb)$

  $f'(p,aabbaba)$

  $f'(p,baa)$



DFA_1

DFA: Basic Concepts

- **Language associated to a DFA:**
  - Given a DFA = (Σ, Q, f, q_0, F), a word **x** is **accepted** or **recognized** by the DFA if **$f'(q_0,x) \in F$**
  - The language associated to a DFA is the set of all the words accepted by it:

    $$\underline{\mathbf{L = \{ x \,/\, x \in \Sigma^* \text{ and } f'(q0,x) \in F \}}}$$

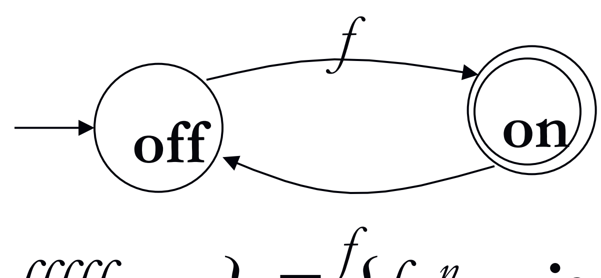
    - If $F = \{\} = \emptyset \Rightarrow L = \phi$
    - If $L = \Sigma^* \Rightarrow F = Q$
    - Another definition:

    $$\underline{\mathbf{L = \{ x \,/\, x \in \Sigma^* \text{ and } (q_0, x) \rightarrow (q,\lambda) \text{ and } q \in F\}}}$$

DFA: Basic Concepts

The language of a DFA (Q, Σ, δ, q_0, F) is the set of all strings over Σ that, starting from q_0 and following the transitions as the string is read left to right, will reach some final state.

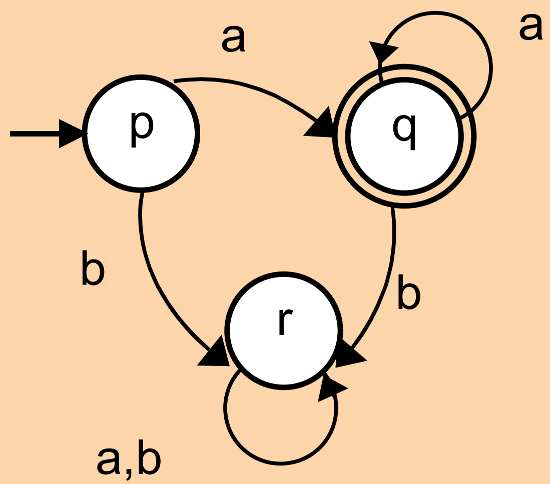
***M*:**



- Language of M is $\{f, fff, fffff, \ldots\} = \{f^{\,n}$: n is odd$\}$

DFA: Basic Concepts

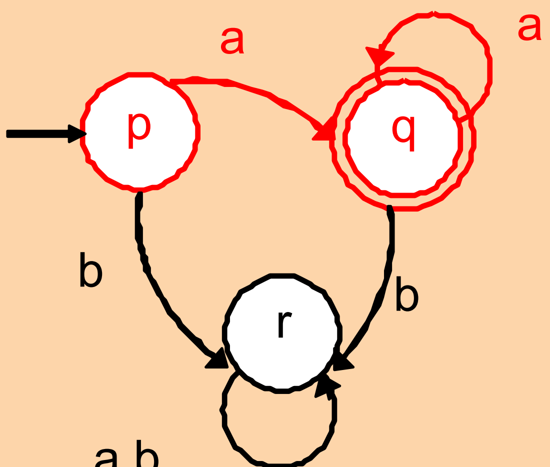
- In the $DFA_{1,}$, verify that $L(DFA_1) = \{a^n \,/\, n>0\}$. Verify also that doing $F = \{r\}$, then $L(DFA_1) = \{a^n bx \mid n \geq 0,\ x \in \Sigma^*\}$.



DFA_1

DFA: Basic Concepts

- **In the $DFA_{1,}$, verify that $L(DFA_1) = \{a^n / n>0\}$.** Verify also that doing $F = \{r\}$, then $L(DFA_1) = \{a^n bx \mid n \geq 0, x \in \Sigma^*\}$.

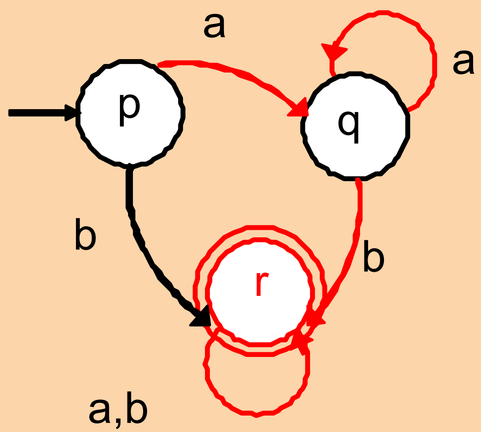


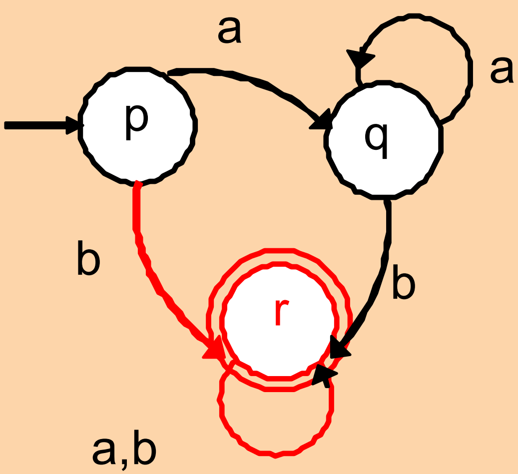
DFA_1

From the state p, with any number of a's, but always at least one, the final state q is reached.

DFA: Basic Concepts

- **In the DFA$_{1,}$**, verify that L(DFA$_1$) = {a^n / n>0}. **Verify also that doing F = {r}, then L(AFD$_1$) = {a^nbx / n≥0, x$\in\Sigma$ *}.**



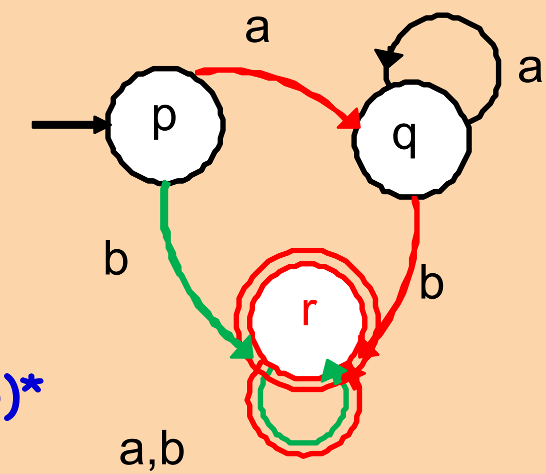


From p, with only one a the state q is reached. There, any number of a's can be accepted, and using a b the final state can be reached. There, the iteration ends or every string with a's and b's can be recognized → **L= a* b (a | b)***

From p, given a b, it is possible to reach the final state. There, the iteration ends or every string with a's and b's can be recognized → **L= b (a | b)***

DFA: Basic Concepts

- **In the DFA$_{1}$,** verify that L(DFA$_1$) = $\{a^n$ / $n>0\}$. **Verify also that doing F = {r}, then L(AFD$_1$) = $\{a^n bx$ / $n \geq 0$, $x \in \Sigma^*\}$.**



$L_B = b(a|b)^*$

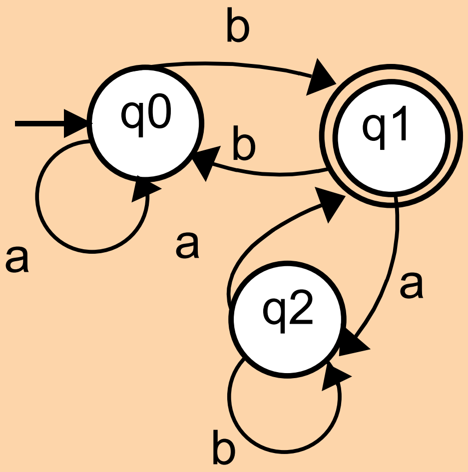
$L_A = a^+b\ (a|b)^*$

$L = L_A \cup L_B = a^*b(a|b)^*$

DFA: Basic Concepts

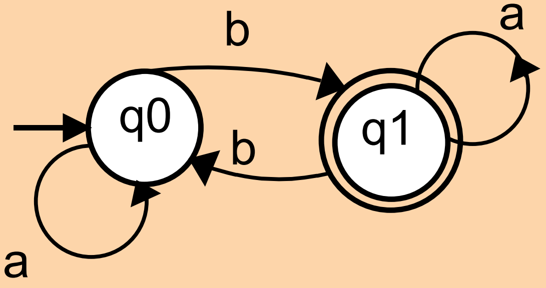
- In the $DFA_{2,}$, verify that the language accepted is

  $L(DFA_2) = \{a^*b\ ((b\ a^*b)^*\ (a\ b^*a)^*)^*\}$.

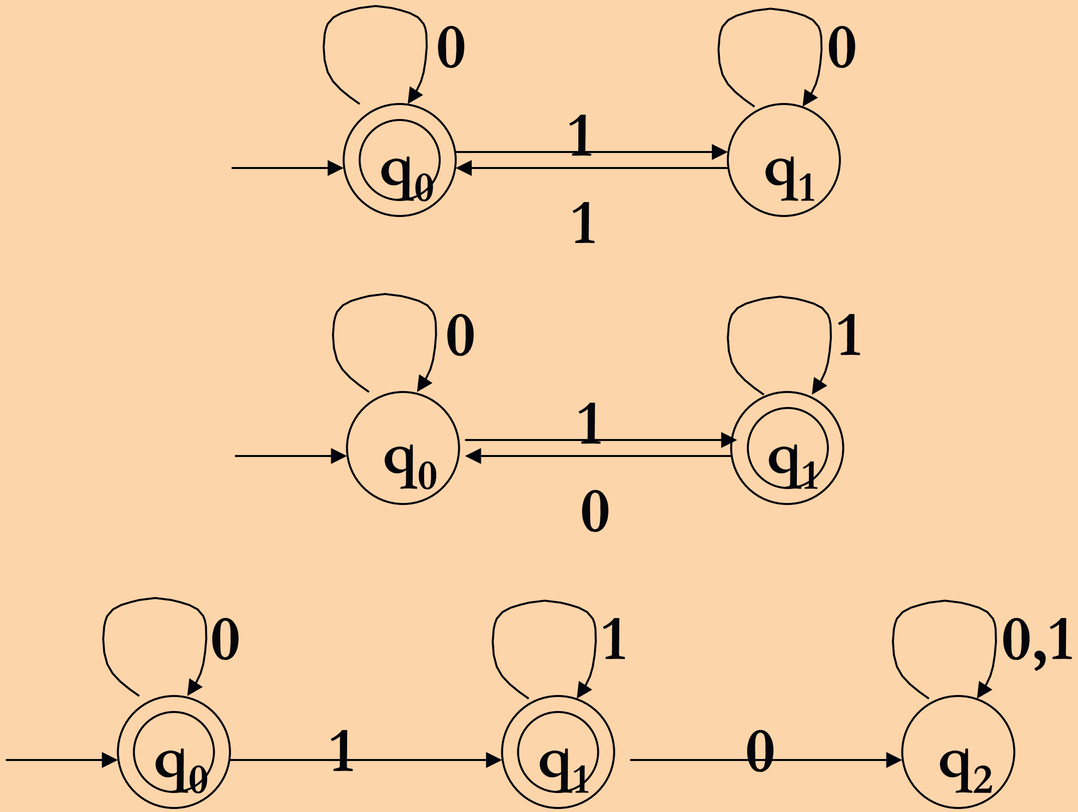


DFA: Basic Concepts

- In the DFA_3, indicate $L(AFD_3)$



DFA: Basic Concepts



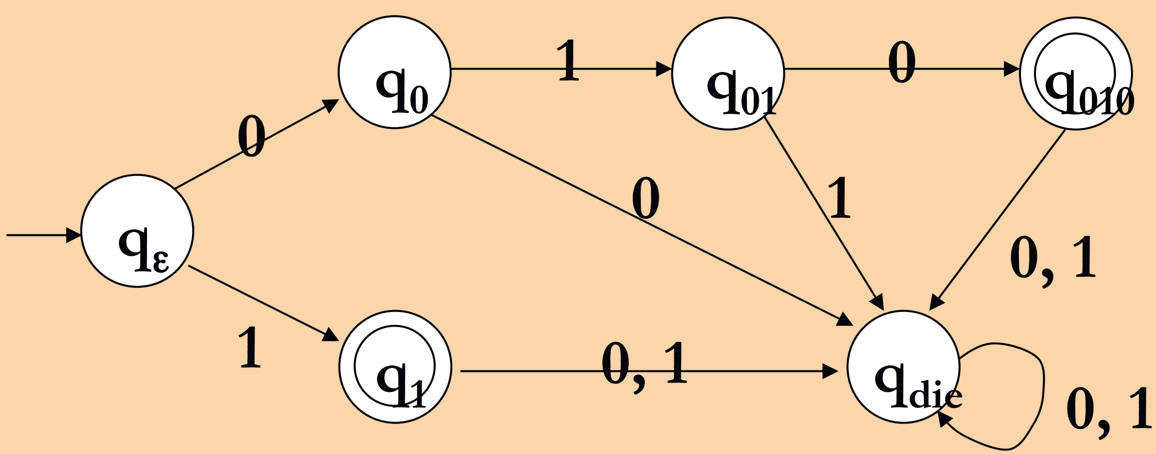
What are the languages of these DFAs?

DFA: Basic Concepts

- Construct a DFA that accepts the language

$$L = \{010, 1\} \qquad (\, \Sigma = \{0, 1\} \,)$$

- Answer



DFA: Basic Concepts

- Construct a DFA over alphabet $\{0, 1\}$ that accepts all strings that end in 101

- Hint: The DFA must "remember" the last 3 bits of the string it is reading

DFA: Basic Concepts

- **Reachable states**
  - Given a DFA = (Σ, Q, f, q_0, F)

    The state $p \in Q$ is **reachable** from $q \in Q$ if $\exists\ x \in \Sigma^*$ $f'(q,x) = p$. (Any other state is **unreachable**)

    Every state is reachable from itself given that

    $$f'(p,\lambda) = p$$

- **<u>Theorem</u>**: Given a DFA, $|Q| = n$, $\forall\ p, q \in Q$ p is reachable from q iff $\exists\ x \in \Sigma^*$, $|x| < n$ / $f'(p,x) = q$
- **<u>Theorem</u>**: Given a DFA, $|Q| = n$, then $L_{DFA} \neq \phi$ iff the DFA accepts at least one word $x \in \Sigma^*$, $|x| < n$

DFA: Basic Concepts

- **Connected Automata:**

Given a DFA = (Σ, Q, f, q_0, F), it is connected if:

- Every state is reachable from q_0.
- Given a non-connected automaton, we can get from it another automaton that is connected by eliminating all the states that are not reachable from q_0.
- It is clear that both automata recognize the same language.

DFA: Basic Concepts

- Calculate an equivalent connected DFA for the following automaton:
  FA= ({a,b}, {p,q,r,s}, p, f, {q,r,s}), where f is given by the folllowing table:

	a	b
→ p	r	p
*q	r	p
*r	r	p
*s	s	s

Indicate which is the language recognized for both DFA's.

DFA: Basic Concepts

- Given DFA1= ({a,b}, {q1,q2,q3,q4}, q1, {q3}), where f is given by the following table. Show the language that recognizes.

	a	b
→ q_1	q_2	q_4
q_2	q_2	q_3
*q_3	q_4	q_3
q_4	q_4	q_4

- Given DFA2= ({0,1}, {q1,q2,q3,q4}, q1, {q2}), where f is given by the following table. Show the language that recognizes.

	0	1
→ q_1	q_4	q_2
*q_2	q_3	q_4
q_3	q_4	q_2
q_4	q_4	q_4

DFA: Basic Concepts

- Given AFD3= ({a,b,c}, {q1,q2,q3,q4,q5}, q1, {q2,q4}), where f is given by the following table. Show the language that recognizes.

	a	b	c
→ q_1	q_2	q_3	q_5
* q_2	q_5	q_5	q_5
q_3	q_5	q_5	q_4
* q_4	q_5	q_5	q_5
q_5	q_5	q_5	q_5

- Given AFD4= ({1,2,3}, {q1,q2,q3}, q1, {q2}), where f is given by the following table. Show the language that recognizes.

	1	2	3
→ q_1	q_1	q_1	q_2
* q_2	q_3	q_3	q_3
q_3	q_3	q_3	q_3

OUTLINE

- Sequential machines
- Finite Automata
- **Deterministic Finite Automata (DFA)**
  - Representation and Basic Concepts
  - **Equivalence and Minimization**
- Nondeterministic Finite Automata (NFA)
- DFA equivalent to a NFA (NFA → DFA)

DFA. Equivalence and Minimization

Next, we are going to study:

- It is possible to have several automata which recognize the same language.
- For every automaton, it is possible to find an equivalent automaton (i.e. both recognize the same language) with the minimal number of states.

Why minimal DFAs?

- A descriptor of the language is available (regular language): Type-3 grammar, DFA, NFA, regular expression.
- Decision problems:
  - Is the described language an empty language? **EASY**
  - Is the string w in the language that is generated? **EASY**
  - Do two different descriptors really recognize the same language? **NOT AS EASY** (infinite languages) → **Solution: Obtain the minimal DFA and then verify it.**

DFA. Equivalence and Minimization

- **Equivalence and Minimization of DFA's**

A DFA is a Moore sequential machine, so same theorems:

- Equivalence of states:

  p E q, where $p,q \in Q$, if $\forall\ x \in \Sigma^* \Rightarrow$ **f'(p,x)** $\in$ **F** $\Leftrightarrow$ **f'(q,x)** $\in$ **F**

- Equivalence of order/length "n":

  p E_n q, $\forall\ p,q \in Q$, if $\forall\ x \in \Sigma^* \ /\ |x| \leq n \Rightarrow$ **f'(p,x)** $\in$ **F** $\Leftrightarrow$ **f'(q,x)** $\in$ **F**

- E and E_n are equivalence relations.

DFA. Equivalence and Minimization

- Equivalence of states. Particular cases.
- E_0 , x word $|x| \leq 0 \Rightarrow x = \lambda$ It can be verified:

  p E_0 q, $\forall$ p,q $\in$ Q, if $\forall$ x$\in \Sigma^*$ / $|x| \leq 0$ then:

  $$f'(p,x) \in F \Leftrightarrow f'(q,x) \in F$$

  x is λ

  $f'(p,x) = f'(p,\lambda) = p$ (given the definition of f')

  $f(p, \lambda) \in F \Leftrightarrow f(q, \lambda) \in F$ -> $p \in F \Leftrightarrow q \in F$

  All the final states are E_0 equivalent.

  $\forall$ p,q $\in$ F it is fulfilled p E_0 q

  $\forall$ p,q $\in$ Q - F it is fulfilled p E_0 q

DFA. Equivalence and Minimization

- Equivalence of states. Particular cases.
- E_1 , x word $|x| \leq 1$ It can be verified:

  p E_1 q, $\forall$ p, q $\in$ Q, if $\forall$ x$\in \Sigma^*$ / $|x| \leq 1$ then:

  $$f'(p,x) \in F \Leftrightarrow f'(q,x) \in F$$

  x is λ or a symbol of the alphabet.

  $f'(p,x) = f'(p,a) = f(p,a)$ ó $f'(p,x) = f'(p,\lambda) = p$ (given the definition of f')

  $f(p,a) \in F \Leftrightarrow f(q,a) \in F$

  From p and q, with only one transition, a final state or a nonfinal state must be reached in both cases.

DFA. Equivalence and Minimization

- Properties:
  - Lemma: $p \; E \; q \Rightarrow p \; E_n \; q, \;\; \forall \; n, \;\; p, q \in Q$
  - Lemma: $p \; E_n \; q \Rightarrow p \; E_k \; q, \;\; \forall \; n > k$
  - Lemma: $p \; E_{n+1} \; q \Leftrightarrow p \; E_n \; q$ and $f(p,a) \; E_n \; f(q,a) \; \forall \; a \in \Sigma$

DFA. Equivalence and Minimization

- Properties:
  - Lemma: $p \; E \; q \Rightarrow p \; E_n \; q, \; \forall \; n, \; p, q \in Q$
  - Lemma: $p \; E_n \; q \Rightarrow p \; E_k \; q, \; \forall \; n > k$
  - Lemma: $p \; E_{n+1} \; q \Leftrightarrow p \; E_n \; q$ and $f(p,a) \; E_n \; f(q,a) \; \forall \; a \in \Sigma$
- Theorem: $p \; E \; q \Leftrightarrow p \; E_m \; q \quad |Q| = n > 1$

  $p \; E \; q$ iff $\forall \; x \in \Sigma^*, \; |x| = m \leq n-2$ it is fulfilled

  $$f(p,x) \in F \Leftrightarrow f(q,x) \in F$$

  $m = n-2$ is the lowest value which fulfills this theorem

  (n-1 is valid, but n-3 is not guaranteed)

DFA. Equivalence and Minimization

- E is an equivalence relation. Meaning of Q/E?
  - Q/E is a partition of Q,
  - $Q/E = \{C1, C2, \ldots, Cm\}$, where $Ci \cap Cj = \emptyset$
  - $p \; E \; q \Leftrightarrow (p,q \in Ci;)$
  - Therefore $\forall \; x \in \Sigma^*$ it is fulfilled

$$f'(p,x) \in Ci \Leftrightarrow f'(q,x) \in Ci$$

- For the relation of order n:
  - E_n: $Q/En = \{C1, C2, \ldots, Cm\}$, Ci intersection $Cj = \emptyset$
  - $p \; En \; q \Leftrightarrow p,q \in Ci;$
  - Therefore $\forall \; x \in \Sigma^*$, $\underline{|x| \leq n}$ it is fulfilled

$$f'(p,x) \in Ci \Leftrightarrow f'(q,x) \in Ci$$

DFA. Equivalence and Minimization

Particular case: E_0

$Q/E0 = \{C1, C2, \ldots, Cm\}$, Ci intersection Cj $= \emptyset$

$p\ E_0\ q \Leftrightarrow p,q \in Ci$; therefore:

$\forall\ x \in \Sigma^*,\ |x| \leq 0 \Rightarrow x = \lambda$ it is fulfilled:

$$f'(p,\underline{\lambda}) \in Ci \Leftrightarrow f'(q,\underline{\lambda}) \in Ci$$

Given $p\ E_0\ q$ $\quad f'(p,\underline{\lambda}) \in F \Leftrightarrow f'(q,\underline{\lambda}) \in F$

$$f'(p,\underline{\lambda}) = p \in F \Leftrightarrow f'(q,\underline{\lambda}) = q \in F$$

$$p \in F \Leftrightarrow q \in F$$

(Interpretation: For $Q/E_{0,}$ Ci is F or Q-F, i.e. for Q/E0 there are only two classes).

$Q/E0 = \{F, Q\text{-}F\}$, and therefore: $\forall$ p,q $x \in Q$, ifp E_0 q then $p \in F \Leftrightarrow q \in F$

DFA. Equivalence and Minimization

Properties (Lemmas)

- $Q/E_n = Q/E_{n+1} \Rightarrow Q/E_n = Q/E_{n+i} \ \forall\, i = 0, 1, \ldots$
- $Q/E_n = Q/E_{n+1} \Rightarrow Q/E_n = Q/E$ Quotient Set
- $|\, Q/E_0 \,| = 1 \Rightarrow Q/E_0 = Q/E_1$
- $n = |Q| > 1 \Rightarrow Q/E_{n-2} = Q/E_{n-1}$
- $p\ E_{n+1}\ q \Leftrightarrow (\ p\ E_n\ q$ and $f(p,a)\ E_n\ f(q,a)\ \forall\ a \in \Sigma\ )$

DFA. Equivalence and Minimization

Properties (Lemmas)

- $Q/E_n = Q/E_{n+1} \Rightarrow Q/E_n = Q/E_{n+i}\ \forall\ i = 0, 1, \dots$
- $Q/E_n = Q/E_{n+1} \Rightarrow Q/E_n = Q/E$ Quotient Set
- $|Q/E_0| = 1 \Rightarrow Q/E_0 = Q/E_1$
- $n = |Q| > 1 \Rightarrow Q/E_{n-2} = Q/E_{n-1}$
- $p\ E_{n+1}\ q \Leftrightarrow (\ p\ E_n\ q$ and $f(p,a)\ E_n\ f(q,a)\ \forall\ a \in \Sigma\ )$

Interpretation:

The objective is to obtain the partition Q/E (minimal automaton).

- We stop when $Q/E_k = Q/E_{k+1}$.
- To obtain Q/E, we have to start calculating Q/E_0, Q/E_1, etc.
- To obtain Q/E, we have to obtain Q/E_{n-2} in the worst case.
- The lemma $p\ E_{n+1}\ q \Leftrightarrow p\ E_n\ q$ and $f(p,a)$ In $f(q,a)\ \forall\ a \in \Sigma$, allows to extend the equivalence of order n from E_0 and E_1

DFA. Equivalence and Minimization

- Equivalence and Minimization of DFA's

Theorem: $pEq \Leftrightarrow pE_{n-2}\, q \quad |Q| = n > 1$

That is to say:

pEq iff $\forall\, x \in \Sigma^*,\ |x| \leq n-2,\ \ f(p,x) \in F \Leftrightarrow f(q,x) \in F$

n-2 is the lowest value that meets this theorem

DFA. Equivalence and Minimization

- **Formal Algorithm to calculate Q/E in DFA's**

1. $\mathbf{Q/E_0 = \{ F, not\ F\}}$

First division taking into account if the states are final or not.

2. $\mathbf{Q/E_{i+1}}$:

   From $\mathbf{Q/E_i}$ = {C1,C2,...Cn}, build $\mathbf{Q/E_{i+1}}$:

   p and q are in the same class in $\mathbf{Q/E_{i+1}}$ if:

   $p, q \in C_k \in Q/E_i\ \forall\ a \in \Sigma \Rightarrow f(p,a)$ and $f(q,a) \in C_m \in Q/Ei$

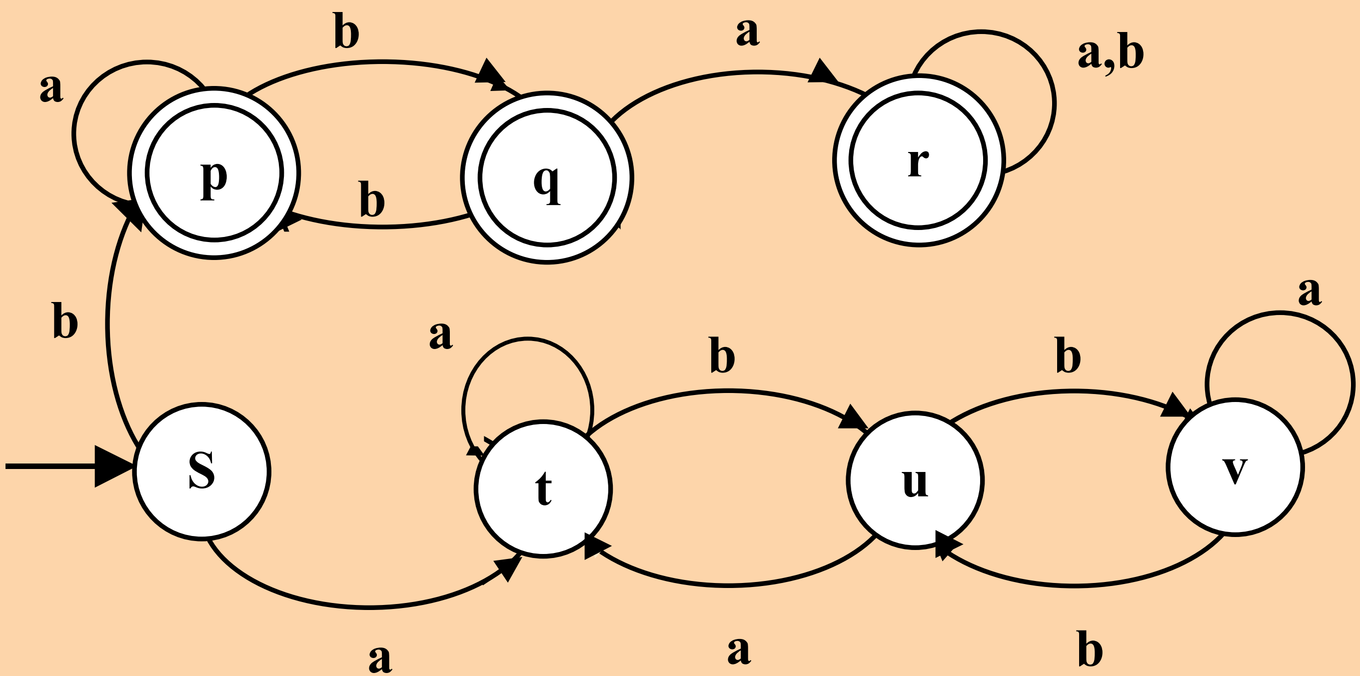
3. If $\mathbf{Q/E_i = Q/E_{i+1}}$ then $\mathbf{Q/E_i = Q/E}$

   also if $\mathbf{Q/E_i}$ and **i=n-2** then $\mathbf{Q/E_i = Q/E}$

   If not, repeat step 2 taking Q/E_{i+1}

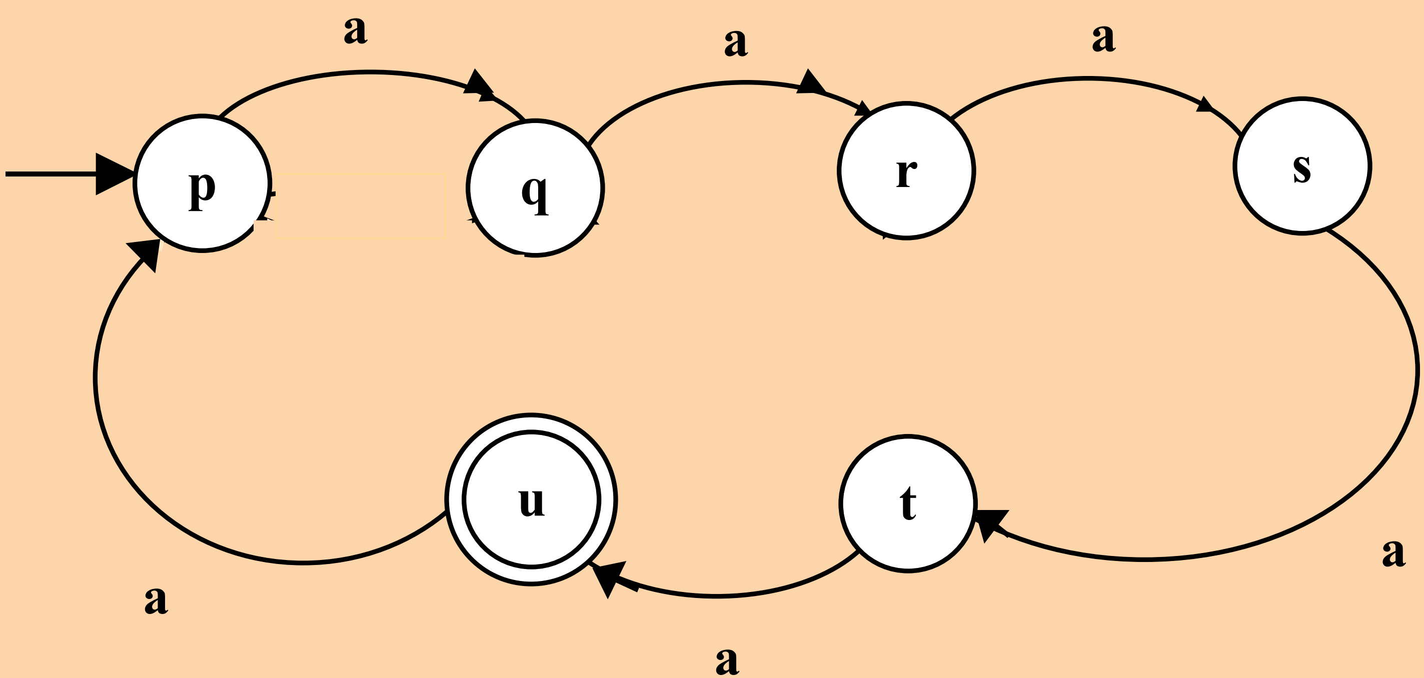
DFA. Equivalence and Minimization

Exercise 1. Calculate the equivalent minimum DFA.



DFA. Equivalence and Minimization

Exercise 2. Calculate the equivalent minimum DFA.



DFA. Equivalence and Minimization

- **Equivalent automata**
  - **Equivalent states in different DFAs**:
    - Given two DFA's: $(\Sigma,Q_1,f_1,q_{01},F_1)$ and $(\Sigma,Q_2,f_2,q_{02},F_2)$
    - the states p,q / $p \in Q_1$ and $q \in Q_2$ are equivalent (pEq) if

      $f_1(p,x) \in F_1 \Leftrightarrow f_2(q,x) \in F_2 \; \forall \; x \in \Sigma^*$
  - Two DFAs are equivalent if they recognize the same language: If $f(q_{01},x) \in F_1 \Leftrightarrow f(q_{02},x) \in F_2 \; \forall \; x \in \Sigma^*$
  - **Two DFA's are equivalent if their initial states are equivalent:**

$$\mathbf{q_{01} \; E \; q_{02}}$$

DFA. Equivalence and Minimization

- **Equivalent automata, verification**:

1. Direct sum of DFA’s.

2. Theorem.

3. Algorithm to prove the equivalence of DFAs

DFA. Equivalence and Minimization

- **<u>Equivalent automata, verification</u>:**

  1. **Direct sum of DFA's:**

     Given two DFA's:

     $A1 = (\Sigma, Q_1, f_1, q_{01}, F_1)$

     $A2 = (\Sigma, Q_2, f_2, q_{02}, F_2)$

     Where $Q_1 \cap Q_2 = \phi$

     The direct sum of A1 and A2 is a FA:

     $A = A1 + A2 = (\Sigma, Q_1 \cup Q_2, f, q_0, F_1 \cup F_2)$

     where:

     - q_0 is the initial state of one of the FA's
     - f: $f(p,a) = f_1(p,a)$ if $p \in Q1$

       $f(p,a) = f_2(p,a)$ if $p \in Q2$

DFA. Equivalence and Minimization

- **Equivalent automata, verification:**

  2. Theorem: Given A1, A2 / $Q_1 \cap Q_2 = \phi$, $|Q_1| = n_1$, $|Q_2| = n_2$

  $$\mathbf{A_1 \; E \; A_2 \text{ if } q_{01} \; E \; q_{02} \text{ in } A = A_1 + A_2}$$

  that it is to say, if A_1 and A_2 accepts the same words x / $|x| \leq n_1+n_2-2$

  In addition, n_1+n_2-2 is the minimum value that fulfills the theorem.

DFA. Equivalence and Minimization

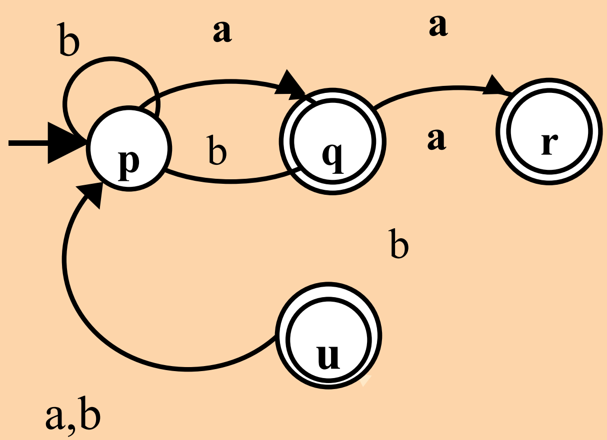
- **<u>Equivalent automata, verification</u>:**

  3. Algorithm 1 to verify the equivalence of DFAs

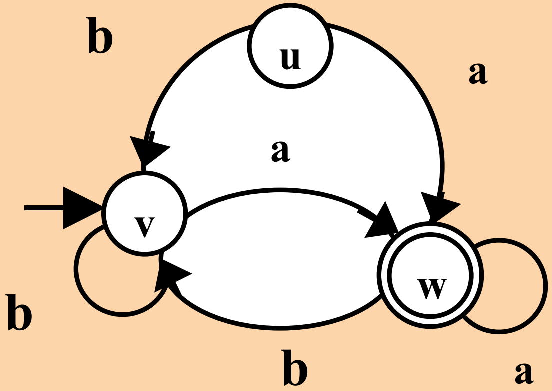
  1. Calculate the direct sum of the DFA's
  2. Calculate Q/E of the resulting AFD sum
  3. If the two initial states are in the same class of equivalence of Q/E $\Rightarrow$ the two DFA's are equivalent

DFA. Equivalence and Minimization

- **Exercise:**



A1



A2

Isomorphic DFA

Given two automata A1 = $(\Sigma, Q_1, f_1, q_{01}, F_1)$ and A2 = $(\Sigma', Q_2, f_2, q_{02}, F_2)$ which fulfill $|Q_1| = |Q_2|$

A1 and A2 are isomorphic, if exists a biyective application
$i : Q_1 \rightarrow Q_2$ that fulfills:

1. $i(q_{01}) = q_{02}$, i.e., the initial states are corresponding.
2. $q \in F_1 \Leftrightarrow i(q) \in F_2$ i.e., the final states are corresponding.
3. $i(f_1(q,a)) = f_2(i(q),a) \ \forall \ a \in \Sigma \ \ q \in Q_1$

In summary, each state is equivalent (both automata only differ in the name of its states)

Two isomorphic DFAs are also equivalent and recognize the same language.

Minimization of DFAs

Algorithm 2 to verify the equivalence of DFAs

Given the DFA, $A = (\Sigma, Q, f, q_0, F)$:

1. From the connected DFA: eliminate unreachable states from the initial state.
2. Calculate Q/E of the connected automata.
3. The minimum DFA, except isomorphisms, is:

   $A' = (\Sigma, Q', f', q_0', F')$

   where:

   $Q' = Q/E$

   f' is built: $f'(C_i,a) = C_j$ if $\exists q \in C_i$, $p \in C_j$ / $f(q,a) = p$

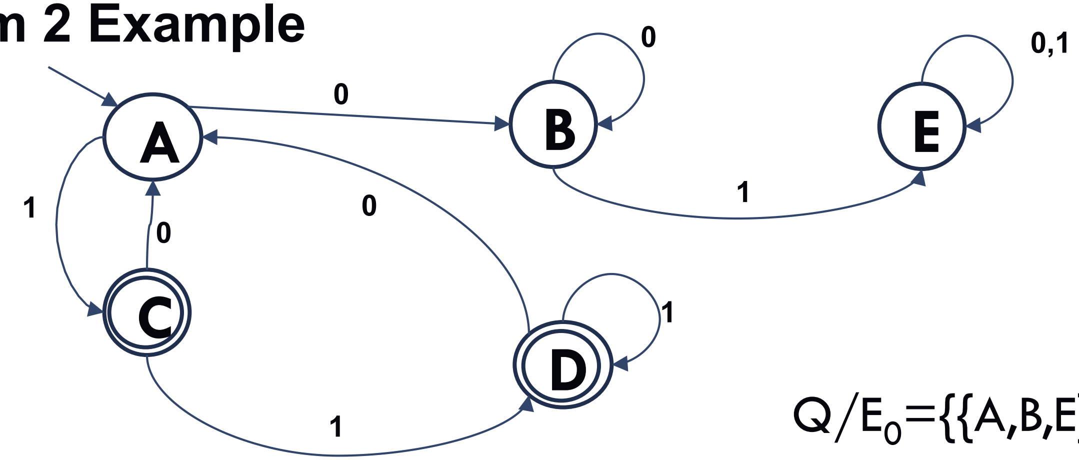
   $q_0' = C_0$ if $q0 \in C_0$, $C_0 \in Q/E$

   $F' = \{C$ / C contains at least one state of $F(\exists\ q \in F$ that fulfills $q \in C)\}$

<u>COROLLARY</u>: 2 DFA's are equivalent if their minimum FA are isomorphic.

Minimization of DFAs

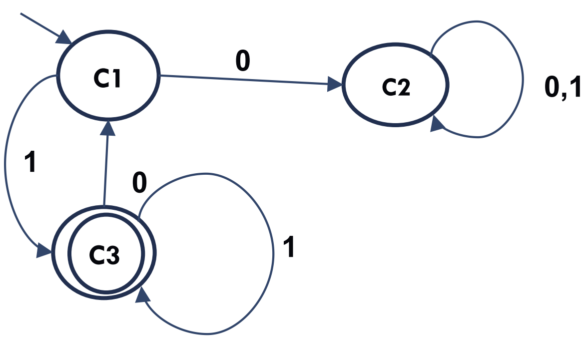
Algorithm 2 Example



	0	1	0	1	0	1
A	B	C	C_1	C_2	C_2	C_3
B	B	E	C_1	C_1	C_2	C_2
C	A	D	C_1	C_2	C_1	C_3
D	A	D	C_1	C_2	C_1	C_3
E	E	E	C_1	C_1	C_2	C_2

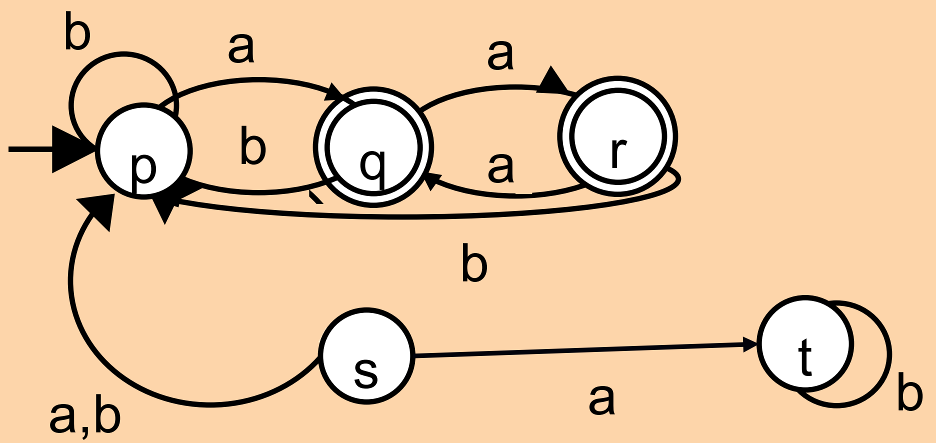
$$Q/E_0=\{\{A,B,E\}, \{C,D\}\}$$

$$= \begin{cases} Q/E_1=\{\{A\}, \{B,E\}, \{C,D\}\} \\ Q/E_2=\{\{A\}, \{B,E\}, \{C,D\}\} = Q/E \end{cases}$$



Minimization of DFAs

Exercise: Calculate the miminum equivalent DFA for:

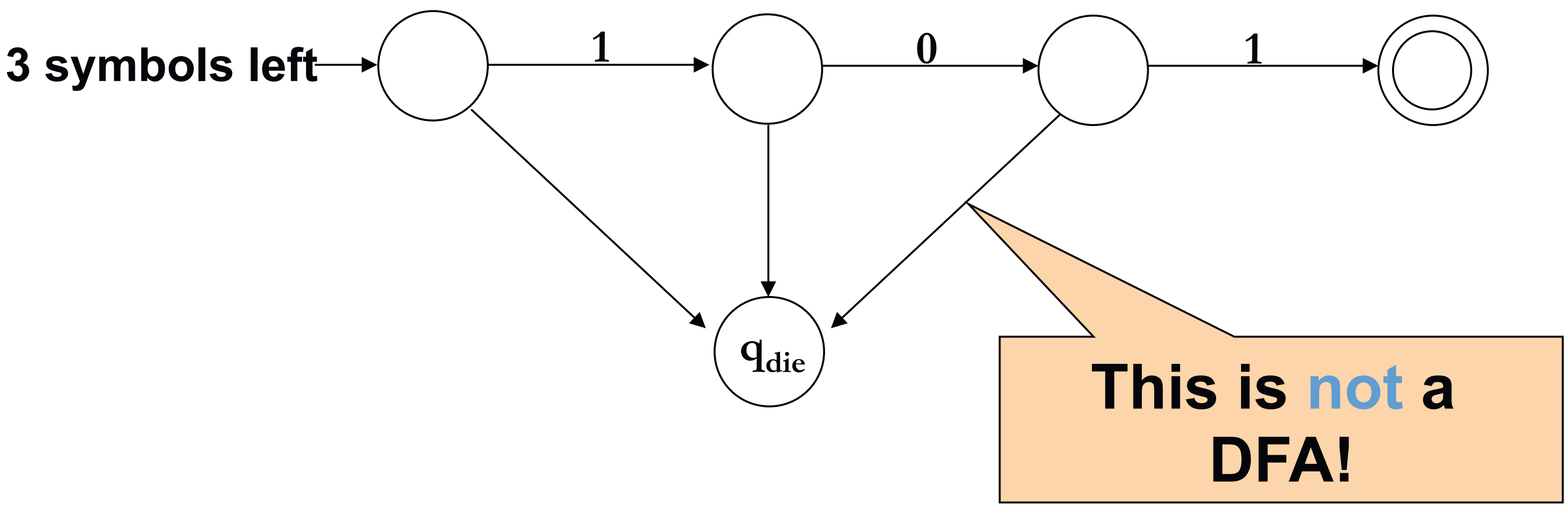


OUTLINE

- Sequential machines
- Finite Automata
- Deterministic Finite Automata (DFA)
  - Representation and Basic Concepts
  - Equivalence and Minimization
- **Nondeterministic Finite Automata (NFA)**
- DFA equivalent to a NFA (NFA → DFA)

Nondeterminism

- Suppose we could guess when the string we are reading has only 3 symbols left.
- Then we could simply look for the sequence 101 and accept if we see it.



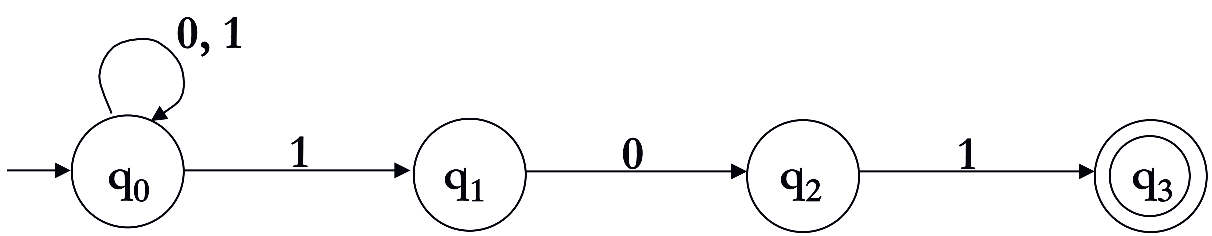
Nondeterminism

- Nondeterminism is the ability to make guesses, which we can later verify
- Informal nondeterministic algorithm for language of strings that end in 101:

1. **Guess if you are approaching end of input**
2. **If guess is yes, look for 101 and accept if you see it**
3. **If guess is no, read one more symbol and go to step 1**

Nondeterministic Finite Automata

- This is a kind of automaton that allows you to make guesses.



- Each state can have zero, one, or more transitions out labeled by the same symbol.

Nondeterministic Finite Automata

- Definition of a NFA:

1) NFA = (Σ, Q, f, q_0, F), where

$f: Q \times (\Sigma \cup \lambda) \rightarrow P(Q)$ (set of parts of Q)

is nondeterministic, for instance:

$f(p,a) = \{q,r\}$ or $f(p,\lambda) = \{q\}$ or $f(p,b) = \{\}$

We must define:

T : Relationship defined over pairs of elements of Q (Formal definition of the λ transition)

$pTq = (p,q) \in T$ if the transition $f(p, \lambda)=q$ is defined.

Nondeterministic Finite Automata

- **Example**: Given the following NFA:

A = ({a,b}, {p,q,r,s}, f,p, {p,s}, T= {(q,s), (r,s), (s,r)}) where f:

f(p,a) = {q} f(p,b) = {}

f(q,a) = {p,r,s} f(q,b) = {p,r}

f(r,a) = {} f(r,b) = {p,s}

f(s,a) = {} f(s,b) = {}

whose transition table is:

	a	**b**	**λ**
→*p	q		
q	p,r,s	p,r	s
r		p,s	s
*s			r

NFA. Extension of the transition function *f* to words

- From f it is defined a transition function f'' that acts over words in Σ^*

  f'' is the transition function over words.

- It is an application: $f'': Q \times \Sigma^* \rightarrow P(Q)$

  Considering:

  1) $f''(q,\lambda) = \{p \;/\; qT^*p \;\; \forall q \in Q\}$

  2) given $x = a_1a_2a_3...a_n$ $n>0$

     $f''(q,x) = \{p \;/\;$ p is reachable from q by means of the word $\underbrace{\lambda^*a_1\, \lambda^*a_2\, \lambda^*a_3\, \lambda^* ...\, \lambda^*a_n\, \lambda^*}_{\text{it is identical to } x}, \forall q \in Q\}$

NFA. Extension of the transition function *f* to words

Calculation of T*

- To calculatef'', it is required to extend transitions λ to λ*, i.e. , to calculate T* of the NFA

- Two possibilities to do this:
  - Formal method of boolean matrices.
  - Method of the matrix of pairs (state, state).

NFA. Extension of the transition function f to words

Calculation of T*

Method of the matrix of pairs (state, state).

1. A matrix is build with number of rows = number of states.
2. In the first column, we write the pair corresponding to the specific state, i.e. (p,p), given that each state is reachable from itself whith λ
3. In the following columns, we write the λ transitions defines in the NFA, considering if the fact of adding them allows to extend additional transitions.
   - E.g. If there is a transition (q,r) and we add the transition (r,s), we have to also add the transition (q,s).
4. When it is not possible to add additional transitions, we have T*

Calculation of T*. Example

- Given the NFA: A, previously defined, where T= {(q,s), (r,r), (r,s), (s,r)} , calculate T*

	a	b	λ
→ *p	q		
q	p,r,s	p,r	s
r		p,s	s
*s			r

$$T^* = \begin{pmatrix} (p,p) \\ (q,q)\boxed{(q,s)}\boxed{(q,r)} \\ (r,r)(r,s) \\ (s,s)\boxed{(s,r)} \end{pmatrix}$$

Produce (s,s), that was already included

(q,s) and (s,r) Produce (q,r)

Calculation of T*. Example

- The previous transition table is extended to contain T*, inserting a new col corresponding to λ*

	a	**b**	**λ**	**λ***
→*p	q			p
q	p,r,s	p,r	s	q,s,r
r		p,s	s	r,s
*s			r	r,s

Extended Table

Calculation of T*. Example

- And now, we calculate the transition table corresponding to f'', replacing transitions with a for λ*aλ* and those with b for λ*bλ*.

	a	b	λ	λ*
→*p	q			p
q	p,r,s	p,r	s	q,s,r
r		p,s	s	r,s
*s			r	r,s

Extended table

→

	λ*aλ*	λ*bλ*
→*p	q,r,s	Φ
q	p,r,s	p,r,s
r	Φ	p,r,s
*s	Φ	p,r,s

f'' table

Language accepted by a NFA

- The language recognized by a NFA can be defined in a similar way to the language recognized by a DFA, by means of the definition of the transition function over words (i.e., f' for the NFA).

- We only must take into account that, in the case of the NFA, given that several sates are obtained from f', the condition of acceptance will be one of these states to be a final state of the automaton.

Language accepted by a NFA

- A word $\mathbf{x} \in \Sigma^*$ **is accepted by a NFA if:**
  - $f''(q_0,x)$ and F have at least one common element, i.e., $f''(q_0,x)$ contains at least one final state.
  - The set of all the words accepted by a NFA is the language accepted by the NFA.
  - Formally:

$$\mathbf{L_{NFA} = \{x \,/\, x \in \Sigma^* \; y \; \exists \; qo \rightarrow F\} = \{x \,/\, x \in \Sigma^* \; y \; f''(qo,x) \cap F \neq \Phi\}}$$

Language accepted by a NFA

- Given that it is a NFA, from q_o several paths can be valid for the word x, and x is accepted if at least one of the paths reaches a final state.

- In addition:

  $\lambda \in \mathbf{L_{NFA}}$ if:

  1 qo $\in$ F or

  2 $\exists$ a final state, q $\in$ F, that it is in the relation T* with qo (qo T* q)

OUTLINE

- Sequential machines
- Finite Automata
- Deterministic Finite Automata (DFA)
  - Representation and Basic Concepts
  - Equivalence and Minimization
- Nondeterministic Finite Automata (NFA)
- **DFA equivalent to a NFA (NFA → DFA)**

DFA equivalent to a NFA

- Given a NFA , it is always possible to find a DFA that recognizes the same language :
  - Set of L_{NFA} = set of L_{DFA}.
  - A NFA is not more powerful than a DFA, this is just a particular case of a NFA.
- From NFA to DFA:
  - Given the NFA N= (Σ, Q,f,q_o,F). The DFA D is defined by:

    D = (Σ, Q_D,f_D,q_{0D},F_D), where:

    Q_D= P(Q) set of of the parts of Q that includes Q and Φ.

    q_{0D} = f''(q_0,λ) (all the states which have relation T* with q0).

    $F_D = \{C \,/\, C \in Q_D \text{ and } \exists\, q \in C \,/\, q \in F\}$

    $f_D(C,a) = \{C' \,/\, C' = \bigcup_{q \in C} f''(q,a)\}$

DFA equivalent to a NFA. Cases

- **<u>Case 1</u>: NFA without λ transitions**
  - Given the NFA described by the following table, find the equivalent DFA.

	a	**b**
→p	q	
q	r,q	
*r		r

Verify that the DFA accepts the same language.

DFA equivalent to a NFA. Cases

- **<u>Case 2</u>: NFA with transitions λ**

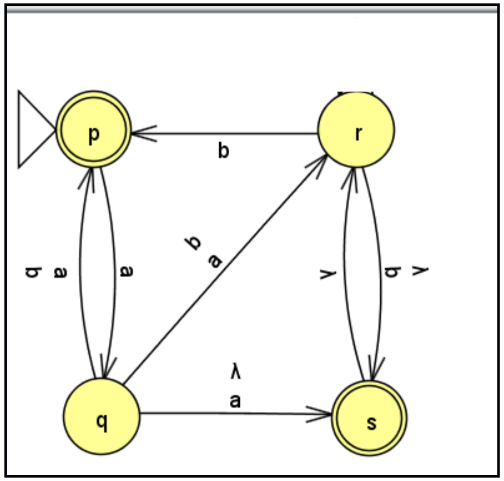
  Given the NFA described by the following table, find the equivalent DFA.

	a	**b**	**c**	**λ**
→p	p	q		q
q	q	p,r		r
r			s	p
*s	s			

Exercise

- Obtain the DFA corresponding to the following NFA

	a	b	λ
→*p	q		
q	p,r,s	p,r	s
r		p,s	s
*s			r



- Steps:
  1. Eliminate λ transitions
     1. Determine λ* (closure of transitions λ, T*)
     2. Obtain the table without λ transitions
  2. Apply algorithm for the creation of new states included in P(Q), adding their transitions.

Exercise

1. Eliminate λ transitions
   1. Determine λ * (closure of transitions λ , T*)

	a	b	λ
→*p	q		
q	p,r,s	p,r	s
r		p,s	s
*s			r

↓

	a	b	λ	λ*
→*p	q			***p***
q	p,r,s	p,r	s	***q***,s,r
r		p,s	s	***r***,s
*s			r	***s***,r

Exercise

1. Eliminate λ transitions
   1. Determine λ * (closure of transitions λ , T*)

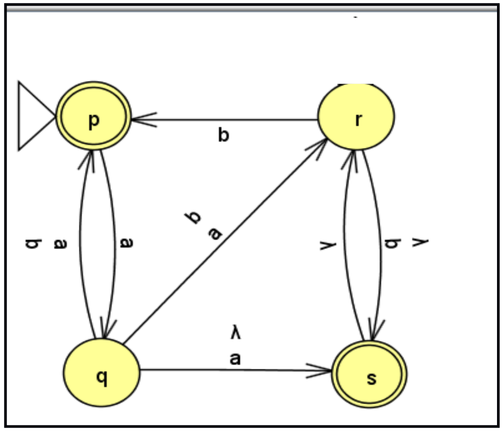
	a	b	λ	λ*
→*p	q			p
q	p,r,s	p,r	s	q,r,s
r		p,s	s	r,s
*s			r	r,s

2. Obtain the table f'' without λ transitions

   (transitions with input λ*a λ*, for each element, a, in the alphabet Σ)

	λ*aλ*	λ*bλ*
→*p	q,r,s	Ø
q	p,r,s	p,r,s
r	Ø	p,r,s
*s	Ø	p,r,s

CHECK →



Exercise

Detail of the obtention of the table f''

f''(p, λ *)= p

f(p, a)= q

f''(q, λ *)= {q,r,s}

f''(p, λ *a λ *)= {q,r,s}

f(p, b)= Ø

f''(q, λ *)= {q,r,s}

f(q,a)={p,r,s}

f''(p, λ *)= {p}

f''(r, λ *)= {r,s}

f''(s, λ *)= {r,s}

f(r,a)=Ø

f(s,a)=Ø

f''(q, λ *a λ *)={p} U {r,s} U {r,s} U Ø U Ø = {p,r,s}

	a	b	λ	λ*
→*p	q			p
q	p,r,s	p,r	s	q,r,s
r		p,s	s	r,s
*s			r	r,s

	λ*aλ*	λ*bλ*
→*p	q,r,s	Ø
q	p,r,s	p,r,s
r	Ø	p,r,s
*s	Ø	p,r,s

Exercise

Detail of the creation of new states

We start with the initial state in the DFA: $f''(p, \lambda^*) = \{p\} = \mathbf{A}$

$f''(p, \lambda^* a \lambda^*) = \{q,r,s\} = \mathbf{B}$

$f''(p, \lambda^* b \lambda^*) = \emptyset$

$f_D(A, a) = \mathbf{B}$

$f_D(A, b) = \emptyset$

Now we see the transitions from **B={q,r,s}**

$f''(q, \lambda^* a \lambda^*) = \{p,r,s\}$

$f''(r, \lambda^* a \lambda^*) = \emptyset$

$f''(s, \lambda^* a \lambda^*) = \emptyset$

$f_D(B, a) = \{p,r,s\} \cup \emptyset \cup \emptyset = \{p,r,s\} = \mathbf{C}$

Etc.

	a	**b**	λ	λ*
→*p	q			p
q	p,r,s	p,r	s	q,r,s
r		p,s	s	r,s
*s			r	r,s

	λ*aλ*	**λ*bλ***
→*p	q,r,s	∅
q	p,r,s	p,r,s
r	∅	p,r,s
*s	∅	p,r,s

	a	**b**
→*A	B	∅
*B	C	C
*C	B	C
∅	∅	∅

Exercise

RESUME

	λ*aλ*	λ*bλ*
→*p	q,r,s	∅
q	p,r,s	p,r,s
r	∅	p,r,s
*s	∅	p,r,s

	a	b	λ	λ*
→*p	q			p
q	p,r,s	p,r	s	q,r,s
r		p,s	r,s	r,s
*s			r	r,s

	a	b
→*p	{q,r,s}	∅
*{q,r,s}	{p,r,s}	{p,r,s}
*{p,r,s}	{q,r,s}	{p,r,s}
∅	∅	∅