Computación y Estructuras Discretas III

Andrés A. Aristizábal P.
aaaristizabal@icesi.edu.co
Ángela Villota
apvillota@icesi.edu.co

Departamento de Computación y Sistemas Inteligentes

UNIVERSIDAD ICESI

2024-2

Agenda del día

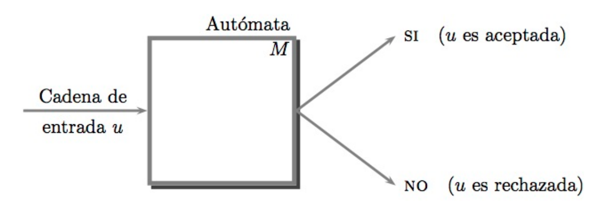
1. **Regular languages and Automata theory**
   - Topic presentation
   - Exercises

Agenda del día

1. **Regular languages and Automata theory**
   - Topic presentation
   - Exercises

What is a finite automaton?

Finite automata are abstract machines that process input strings, which are either accepted or rejected.



What is the formal definition of a finite automaton?

What is the formal definition of a finite automaton?

Definition

A finite automaton M is defined as a 5-tuple, such that $M = (\Sigma, Q, q_0, F, \delta)$.

What is the formal definition of a finite automaton?

Definition

A finite automaton M is defined as a 5-tuple, such that $M = (\Sigma, Q, q_0, F, \delta)$.

1. *An alphabet* Σ*, called the tape alphabet. All strings processed by M belong to* Σ^*.

What is the formal definition of a finite automaton?

Definition

A finite automaton M is defined as a 5-tuple, such that $M = (\Sigma, Q, q_0, F, \delta)$.

1. *An alphabet* Σ, *called the tape alphabet. All strings processed by M belong to* Σ^*.
2. $Q = q_0, q_1, ..., q_n$, *the set of internal states of the automaton.*

What is the formal definition of a finite automaton?

Definition

A finite automaton M is defined as a 5-tuple, such that $M = (\Sigma, Q, q_0, F, \delta)$*.*

1. *An alphabet* Σ*, called the tape alphabet. All strings processed by M belong to* Σ^**.*
2. $Q = q_0, q_1, ..., q_n$*, the set of internal states of the automaton.*
3. $q_0 \in Q$*, the initial state.*

What is the formal definition of a finite automaton?

Definition

A finite automaton M is defined as a 5-tuple, such that $M = (\Sigma, Q, q_0, F, \delta)$.

1. *An alphabet* Σ*, called the tape alphabet. All strings processed by M belong to* Σ^*.
2. $Q = q_0, q_1, ..., q_n$*, the set of internal states of the automaton.*
3. $q_0 \in Q$*, the initial state.*
4. $F \subseteq Q$*, the set of final or accepting states.* $F \neq \emptyset$.

What is the formal definition of a finite automaton?

Definition

A finite automaton M is defined as a 5-tuple, such that $M = (\Sigma, Q, q_0, F, \delta)$.

1. *An alphabet* Σ*, called the tape alphabet. All strings processed by M belong to* Σ^*.
2. $Q = q_0, q_1, ..., q_n$*, the set of internal states of the automaton.*
3. $q_0 \in Q$*, the initial state.*
4. $F \subseteq Q$*, the set of final or accepting states.* $F \neq \emptyset$.
5. *The transition function of the automaton:*

$$\begin{array}{rcc} \delta : Q \times \Sigma & \longrightarrow & Q \\ (q, s) & \longmapsto & \delta(q, s) \end{array}$$

Example

Let's consider the automaton defined by the following 5 *components:*

$\Sigma = a, b$

$Q = q_0, q_1, q_2$

q_0 *: initial state.*

$F = q_0, q_2$*, accepting states.*

Transition function δ :

Example

Let's consider the automaton defined by the following 5 *components:*

$\Sigma = a, b$

$Q = q_0, q_1, q_2$

q_0 *: initial state.*

$F = q_0, q_2$*, accepting states.*

Transition function δ :

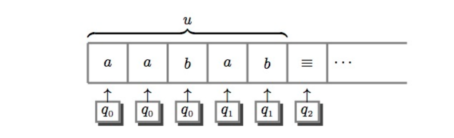
δ	a	b
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_1	q_1

$$\delta(q_0, a) = q_0 \qquad \delta(q_0, b) = q_1$$
$$\delta(q_1, a) = q_1 \qquad \delta(q_1, b) = q_2$$
$$\delta(q_2, a) = q_1 \qquad \delta(q_2, b) = q_1.$$

Example

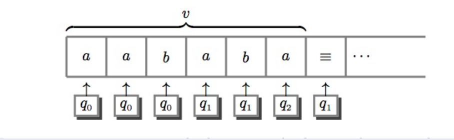
We illustrate the processing of two input strings.

1. $u = aabab$



Since q_2 is an accepting state, the input string u is accepted.

2. $v = aababa$



Since q_1 is not an accepting state, the string v is rejected.

How do we call these previously defined automata?

They are called Deterministic Finite Automata (DFA) because for each state q and for each symbol $a \in \Sigma$, the transition function $\delta(q, a)$ is always defined.

How do we call the language accepted by a finite automaton?

Let M be an automaton, the language accepted by this automaton is denoted as $L(M)$, and it is defined as follows:

$$L(M) := \{u \in \Sigma^* \mid M \text{ terminates processing the input string } u \text{ in a state } q \in F\}.$$

What is a transition diagram of an automaton?

What is a transition diagram of an automaton?

It is a directed and labeled graph that represents the finite automaton.

What is a transition diagram of an automaton?

It is a directed and labeled graph that represents the finite automaton.

How is the transition diagram of an automaton obtained?

Regular languages and Automata theory

What is a transition diagram of an automaton?

It is a directed and labeled graph that represents the finite automaton.

How is the transition diagram of an automaton obtained?

- *The vertices or nodes are the automaton's states*

What is a transition diagram of an automaton?

It is a directed and labeled graph that represents the finite automaton.

How is the transition diagram of an automaton obtained?

- *The vertices or nodes are the automaton's states*
- *State q is represented by:* q

What is a transition diagram of an automaton?

It is a directed and labeled graph that represents the finite automaton.

How is the transition diagram of an automaton obtained?

- *The vertices or nodes are the automaton's states*
- *State q is represented by:* q
- *The initial state* q_0 *is represented by:* $>q_0$

What is a transition diagram of an automaton?

It is a directed and labeled graph that represents the finite automaton.

How is the transition diagram of an automaton obtained?

- *The vertices or nodes are the automaton's states*
- *State q is represented by:* q
- *The initial state* q_0 *is represented by:* $>q_0$
- *The final state q is represented by:* q

What is a transition diagram of an automaton?

It is a directed and labeled graph that represents the finite automaton.

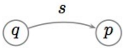
How is the transition diagram of an automaton obtained?

- *The vertices or nodes are the automaton's states*
- *State q is represented by:* 

- *The initial state* q_0 *is represented by:* 

- *The final state q is represented by:* 

- *Transition* $\delta = (q, s) = p$ *is represented as:*



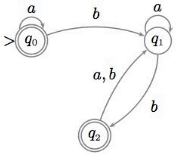
Regular languages and Automata theory

Example

$\Sigma = \{a, b\}$.
$Q = \{q_0, q_1, q_2\}$.
q_0 : *initial state*
$F = \{q_0, q_2\}$, *accepting states*
Transition function δ:

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_1	q_1

$\delta(q_0, a) = q_0$ $\quad$ $\delta(q_0, b) = q_1$
$\delta(q_1, a) = q_1$ $\quad$ $\delta(q_1, b) = q_2$
$\delta(q_2, a) = q_1$ $\quad$ $\delta(q_2, b) = q_1$



Find a set of strings accepted by this automata

Agenda del día

1. **Regular languages and Automata theory**
   - Topic presentation
   - Exercises

1. **Regular languages and Automaton theory**
   - Non-deterministic Automaton (NFA)
   - NFA with λ transitions
   - Exercises

1. **Regular languages and Automaton theory**
   - Non-deterministic Automaton (NFA)
   - NFA with λ transitions
   - Exercises

DFA vs NFA

Deterministic Finite Automaton (DFA)
$\rightarrow$ For every state in the automaton, there is exactly one transition for each possible input symbol in the alphabet.
$\rightarrow$ If the DFA reaches the end of the input string, it accepts the input if it is in a designated accepting state, otherwise, it rejects the input.
s DFA

Non-Deterministic Finite Automaton (NFA)
$\rightarrow$ For a given state and input symbol, there can be multiple possible transitions or no transition at all.
$\rightarrow$ NFA accepts an input string if there exists at least one path through the states that leads to an accepting state, regardless of whether other paths lead to rejection.

Non-Deterministic Finite Automaton

What are the changes on defining a DFA?

Mainly the way we define the transitions as for each state $q \in Q$ and each $a \in \Sigma$, the transition $\delta(q, a)$ can consist of **more than one state or may not be defined**.

Non-Deterministic Finite Automaton

What are the changes on defining a DFA?

Mainly the way we define the transitions as for each state $q \in Q$ and each $a \in \Sigma$, the transition $\delta(q, a)$ can consist of **more than one state or may not be defined**.

An NFA is defined by $M = (\Sigma, Q, q_0, F, \Delta)$ where:

Non-Deterministic Finite Automaton

What are the changes on defining a DFA?

Mainly the way we define the transitions as for each state $q \in Q$ and each $a \in \Sigma$, the transition $\delta(q, a)$ can consist of **more than one state or may not be defined**.

An NFA is defined by $M = (\Sigma, Q, q_0, F, \Delta)$ where:

1. Σ is the tape alphabet.
2. Q is the (finite) set of internal states.
3. $q_0 \in Q$ is the initial state.
4. $\emptyset \neq F \subseteq Q$ is the set of final or accepting states.
5. Let $\wp(Q)$ be the set of subsets of Q,

$$\begin{array}{rcl} \Delta : Q \times \Sigma & \longrightarrow & \wp(Q) \\ (q, s) & \mapsto & \Delta(q, s) = \{q_{i1}, q_{i2}, \ldots, q_{ik}\} \end{array}$$

Non-Deterministic Finite Automaton

What was the difference??

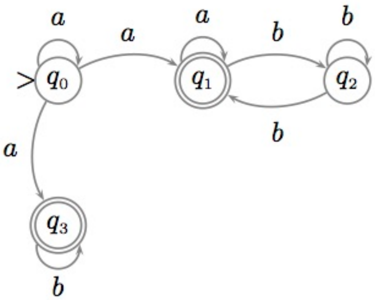
Non-Deterministic Finite Automaton

What was the difference??

$$\begin{array}{rcl} \Delta : Q \times \Sigma & \longrightarrow & \wp(Q) \\ (q, s) & \mapsto & \Delta(q, s) = \{q_{i1}, q_{i2}, ..., q_{ik}\} \end{array}$$

Example

Let M be the following NFA:



Δ	a	b
q_0	$\{q_0, q_1, q_3\}$	$\varnothing$
q_1	$\{q_1\}$	$\{q_2\}$
q_2	$\varnothing$	$\{q_1, q_2\}$
q_3	$\varnothing$	$\{q_3\}$

How does an FNA process an input string?

An NFA can process an input string $u \in \Sigma^*$ in different ways. Seen in the transition diagram, this means that there can be various paths from the initial state, labeled with the symbols of u.

How does an FNA process an input string?

An NFA can process an input string $u \in \Sigma^*$ in different ways. Seen in the transition diagram, this means that there can be various paths from the initial state, labeled with the symbols of u.

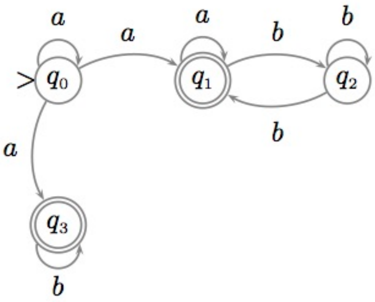
What is the notion of acceptance for nondeterministic finite automata?

$$
\begin{aligned}
L(M) &= \text{language accepted or recognized by } M \\
&= \{u \in \Sigma \mid \text{there exists at least one complete computation} \\
&\qquad \text{of } u \text{ that ends in a state } q \in F\}
\end{aligned}
$$

For a string u to be accepted, there must exist some computation in which u is processed completely and ends with M in an accepting state.

Example

Let M be the following NFA:



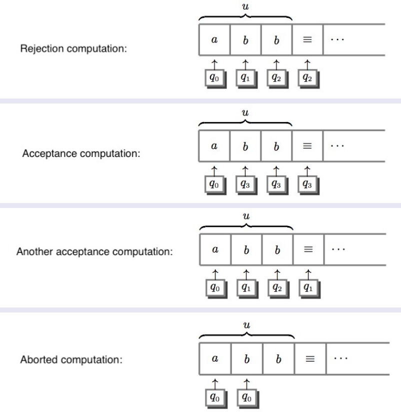
Δ	a	b
q_0	$\{q_0, q_1, q_3\}$	$\varnothing$
q_1	$\{q_1\}$	$\{q_2\}$
q_2	$\varnothing$	$\{q_1, q_2\}$
q_3	$\varnothing$	$\{q_3\}$

For the string $u = abb$, there are computations that lead to rejection, aborted computations, and computations that end in accepting states.

Regular languages and Automata theory

Example

According to the definition of accepted language, $u \in L(M)$, we have:



Are DFAs and NFAs computationally equivalent?

Are DFAs and NFAs computationally equivalent?

Tes they are!!
The DFAs and NFAs are computationally equivalent.

Are DFAs and NFAs computationally equivalent?

Tes they are!!
The DFAs and NFAs are computationally equivalent.

DFA $\rightarrow$ NFA
A DFA $M = (\Sigma, Q, q_0, F, \delta)$ can be considered as an NFA $M' = (\Sigma, Q, q_0, F, \Delta)$ by defining $\Delta(q, a) = \delta(q, a)$ for each $q \in Q$ and each $a \in \Sigma$

Are DFAs and NFAs computationally equivalent?

Tes they are!!
The DFAs and NFAs are computationally equivalent.

DFA $\rightarrow$ *NFA*
A DFA $M = (\Sigma, Q, q_0, F, \delta)$ can be considered as an NFA $M' = (\Sigma, Q, q_0, F, \Delta)$ by defining $\Delta(q, a) = \delta(q, a)$ for each $q \in Q$ and each $a \in \Sigma$

NFA $\rightarrow$ *DFA*??, we have the following theorem:

Are DFAs and NFAs computationally equivalent?

Tes they are!!
The DFAs and NFAs are computationally equivalent.

DFA $\rightarrow$ *NFA*
A DFA $M = (\Sigma, Q, q_0, F, \delta)$ can be considered as an NFA $M' = (\Sigma, Q, q_0, F, \Delta)$ by defining $\Delta(q, a) = \delta(q, a)$ for each $q \in Q$ and each $a \in \Sigma$

NFA $\rightarrow$ *DFA*??, we have the following theorem:

Theorem

Given an NFA $M = (\Sigma, Q, q_0, F, \Delta)$, we can construct an equivalent DFA M' to M, meaning that $L(M) = L(M')$

1. **Regular languages and Automaton theory**
   - Non-deterministic Automaton (NFA)
   - NFA with λ transitions
   - Exercises

A non-deterministic automaton with λ transitions is a tuple $M = (\Sigma, Q, q_0, F, \Delta)$ where the transition function is defined as $\Delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow \wp(Q)$.

Which means, it contains λ transitions.

A non-deterministic automaton with λ transitions is a tuple $M = (\Sigma, Q, q_0, F, \Delta)$ where the transition function is defined as $\Delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow \wp(Q)$.

Which means, it contains λ transitions.

The λ transition, $\Delta(q, \lambda) = \{p_{i_1}, ..., p_{i_n}\}$, also called null or spontaneous transition, allows transitioning between states without processing any symbol read on the tape.

In a transition diagram, these transitions result in arcs labeled with λ.

When is an input w accepted by an NFA-λ?

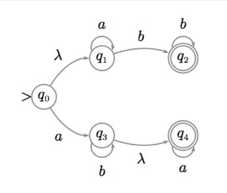
When is an input w accepted by an NFA-λ?

The same as with regular NFAs

A string w is accepted by an NFA-λ if there exists at least one path from the initial state q_0 to an accepting state, whose labels consist exactly of the symbols of w, **BUT** interspersed with zero or more λs.

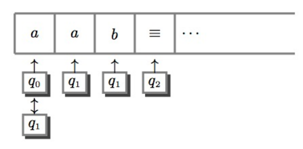
Example

Let M be the following NFA-λ:

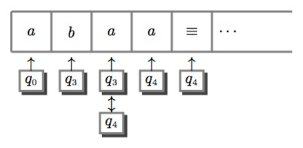


Example

The acceptance computation for $u = aab$ *is:*



The acceptance computation for $v = abaa$ *is:*



What advantages does an NFA-λ have?

Regular languages and Automata theory

What advantages does an NFA-λ have?

It allows even more freedom in the design of automata, especially when there are numerous concatenations.

What advantages does an NFA-λ have?

It allows even more freedom in the design of automata, especially when there are numerous concatenations.

Example

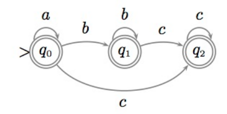
Let M be the following DFA that accepts the language $a^* b^* c^*$ *over* $\Sigma = \{a, b, c\}$*:*

What advantages does an NFA-λ have?

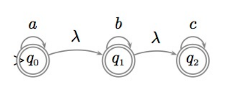
It allows even more freedom in the design of automata, especially when there are numerous concatenations.

Example

Let M be the following DFA that accepts the language $a^* b^* c^*$ *over* $\Sigma = \{a, b, c\}$:

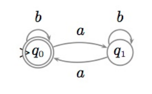


M' *is an NFA-*λ *that accepts* L :

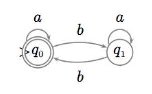


Example

Let M_1 be the following DFA that accepts the language of all strings with an even number of á's over $\Sigma = \{a, b\}$:

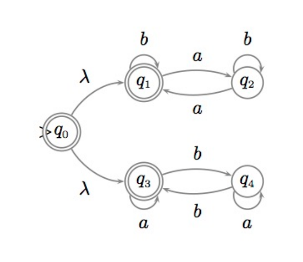


And let M_2 be the following DFA that accepts the language of all strings with an even number of 'b's over $\Sigma = \{a, b\}$:



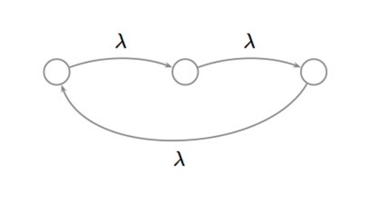
Example

M_3 is the following NFA-λ that accepts the language of all strings with an even number of á's or an even number of 'b's over $\Sigma = \{a, b\}$:



Can infinite loops occur in an NFA-λ?

Unlike DFAs and NFAs, in NFAs-λ, there can be computations that never terminate. This can happen if the automaton enters a state from which there are multiple chained λ transitions that return to the same state.



Are NFAs and NFAs-λ computationally equivalent?

Yes, NFA-λ are computationally equivalent to NFAs.
'Cause, λ transitions can be eliminated by adding transitions that simulate them, without altering the accepted language.
NFA $\rightarrow$ *NFA*λ An NFA $M = (\Sigma, Q, q_0, F, \Delta)$ can be considered as an NFA-λ where there are simply zero λ transitions.

Are NFAs and NFAs-λ computationally equivalent?

Yes, NFA-λ are computationally equivalent to NFAs.
'Cause, λ transitions can be eliminated by adding transitions that simulate them, without altering the accepted language.
NFA $\rightarrow$ *NFA*λ An NFA $M = (\Sigma, Q, q_0, F, \Delta)$ can be considered as an NFA-λ where there are simply zero λ transitions.

For *NFA*λ $\rightarrow$ *NFA*, there's the following theorem:

Theorem

Given an NFA-λ $M = (\Sigma, Q, q_0, F, \Delta)$, we can construct an equivalent NFA M' to M, meaning that $L(M) = L(M')$.

1. **Regular languages and Automaton theory**
   - Non-deterministic Automaton (NFA)
   - NFA with λ transitions
   - Exercises