Course: Automata Theory

Lecture 7: Models of Computation – Finite State Machines

Lecturer: Martha Gichuki

Course description

- The course begins with an introduction to logic and formal grammar where learners will do a recap on sets, logic and truth tables, sequences, relations and functions
- A coverage of finite state machines, Push Down automata and Turing Machines (The Church's thesis) will culminate the study of various models of computation.
- Formal language and grammar will then follow to enable learners differentiate regular and context free languages.
- An evaluation of the computability and complexity of practical computational problems which are the foundations of automata theory will then be done and the outcome will be problem description.

Learning outcomes:

Lecture 7: Models of Computation – Finite State Machines

At the end of the lecture the learner will be able to:

- Define Finite State Machine.
- Describe Finite Automata using state diagrams
- Formally describe various NFA's and DFA's

7.1 Finite Automata

- A finite-state machine (finite automaton) is a simple, limited model of computation with a number of states and transitions.
- Finite Automata are good models for computer with a **finite number of states**, which means a **finite memory,** translating into **limited storage capacity**.
- What can a computer do with such a small memory? Many useful things!! We interact with such computers all the time as they lie at the heart of various electromechanical devices.

- An automaton is a mathematical model for a Finite State Machine (FSM).
- A finite State Machine is a machine that, given an **input of symbols**, "**jumps**" or transitions through a series of states according to a transition function (which can be expressed as a table).
- This **transition function** tells the automaton **which state to go to next** given a **current state** and a **current symbol**.

- The input is read **symbol by symbol**, until it is consumed completely (think of it as a tape with a word written on it, that is read by a reading head of the automaton; the head moves forward over the tape, reading one symbol at a time). Once the input is depleted, the automaton is said to have stopped.
- Depending on the **state in which the automaton stops**, it's said that the automaton either **accepts or rejects the input**.
- The **set of all the words accepted by an automaton** is called the **language accepted by the automaton.** A language is called **regular language if some finite automaton recognizes it**.

FORMAL DEFINITION OF A FINITE AUTOMATON

- *Other than using state diagrams,* which are easier to grasp we need to define finite automata formally for two main reasons: -

(i) Precision - Formal definition resolves any uncertainties about what is allowed in a finite automaton.

(ii) Clarity - Good notation helps us think and express our thoughts **clearly**.

- Putting it all together we arrive at the formal definition of finite automata.

- In mathematical language a list of **five elements is called a 5-tuple** and a **finite automaton** is a **5-tuple** consisting of five parts:

i. A set of states
ii. Rules for going from one state to the other, depending on the input symbol.
iii. An input alphabet that indicates the allowed input symbols.
iv. A start state and
v. A set of accept states also known as final states.

The Transition Function

- The transition function is denoted as **δ** ***(delta sign)*** to define the **rules for m**oving.
- If the finite automaton has an arrow **from a state X to a state Y labeled with the input symbol 1**, that means that, if the automaton is in **state X when it reads a 1, it then moves to state Y.**
- We can indicate the same thing with the transition function by saying **that δ(X, 1) = Y.**
- This notation is a kind of mathematical shorthand.

Types of Automatons

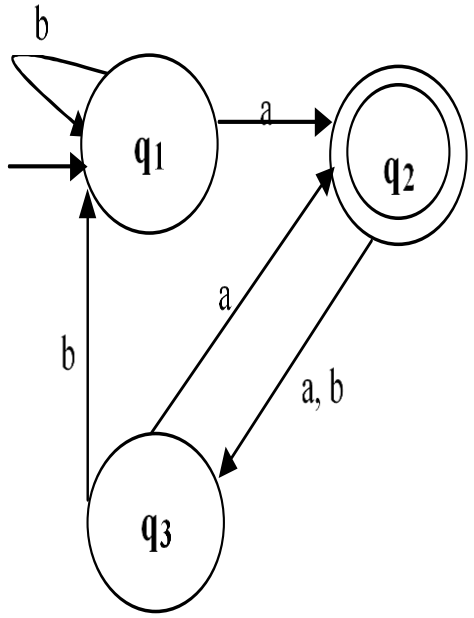
1. Deterministic Finite Automata (DFA)

- This is an automaton in which **each move (transition from one state to another) is uniquely determined by the current configuration**.
- If the **internal state, input and contents of the storage** are known, **it is possible to predict the future behaviour of the automaton**.
- For a DFA, **exactly one transition arrow exits every state for each possible input alphabet.**
- An automaton whose output response is “Yes” or “No” is called an **acceptor.**

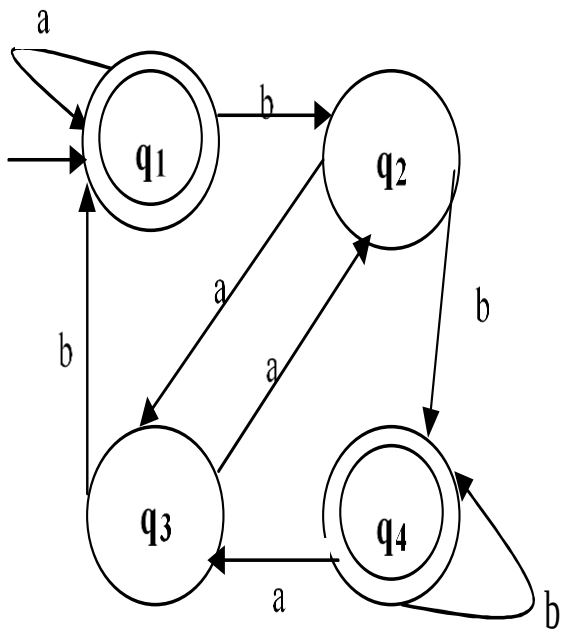
The **formal description of a DFA is a <u>five-</u> tuple M= (Q,∑** *(sigma sign)* **, δ** *(delta sign)* **, q_0, F);** where: -

i. **Q** = Finite set of Internal States,
ii. **∑**= Finite Set of Symbols "Input Alphabet" ,
iii. **δ** : Q X ∑→Q is the Transition Function
iv. **q_0∈Q** is the Initial State
v. **F ⊆ Q** is the **<u>Set</u>** of Final States

Example:The following are state diagrams of two Deterministic Finite Automata (DFAs). Answer the questions about these machines: -



M_3



M_4

a)What are the start states of M_3 and M_4?

$\{q_1\}$ and $\{q_1\}$ respectively

b) What the sets are of accept /Final states of M_3 and M_4?

$\{q_2\}$ and $\{q_1, q_4\}$ respectively.

c) What sequence of states does M_4 go through on input aabb?

q_1, q_1, q_2, q_4

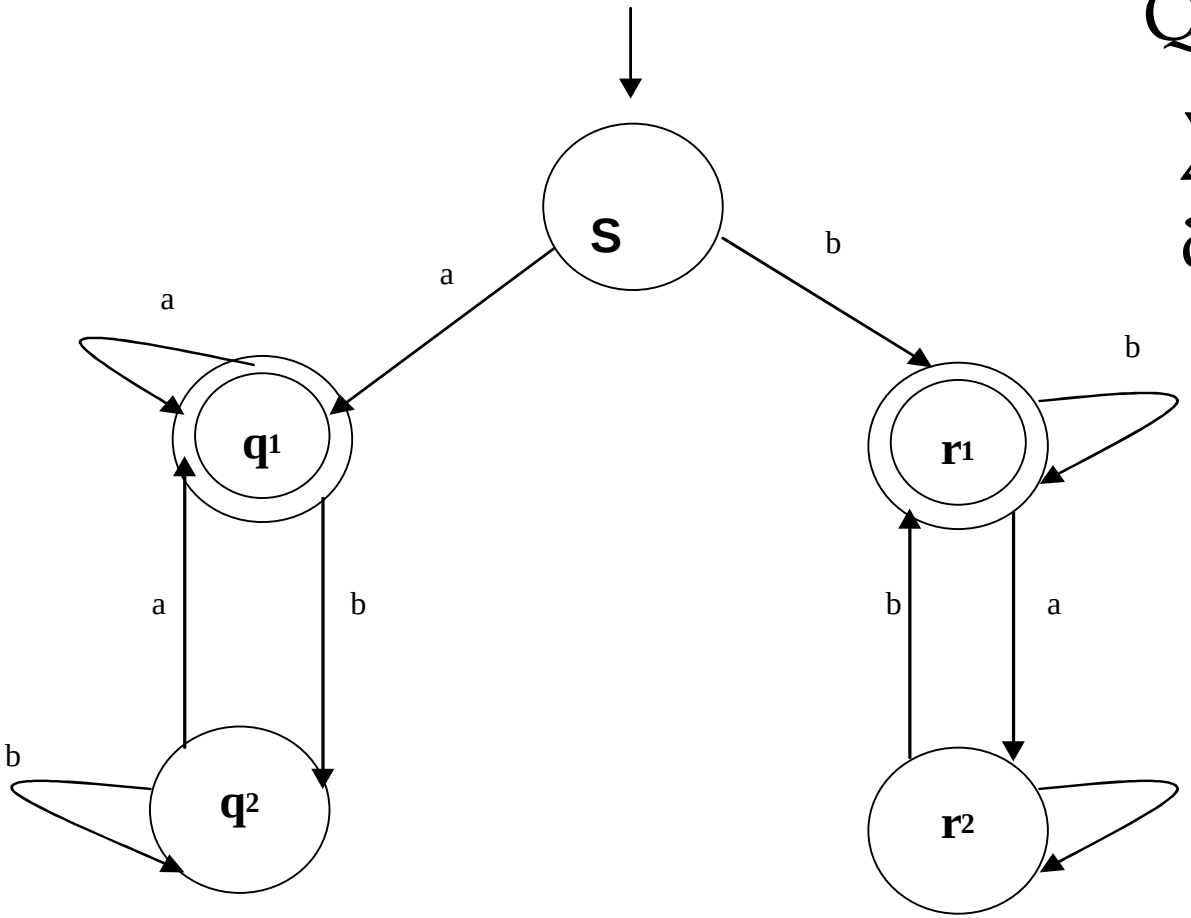
d) Does M_4 accept the string aabb? Give a reason for your answer

- **Yes it does because the final state q_4 is an accept stat**e

e) Does M_4 accept the string string { } ? Give a reason for your answer

- **Yes it does, even without any input, we are already in an accept state**

The figure below shows the state diagram of a finite automaton M_1. Give the formal description of this automaton.



M_1

$Q = \{S, q1, q2, r1, r2\}$

$\sum = \{a, b\}$

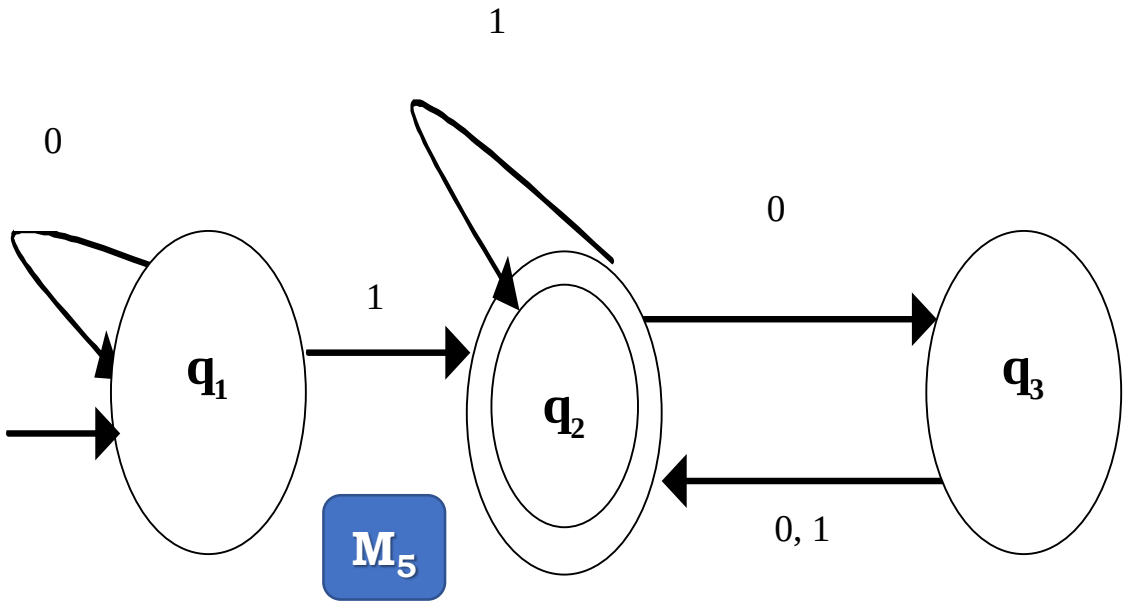
δ = Transition Symbol

	Input Alphabet	
States	a	b
S	q1	r1
q1	q1	q2
q2	q1	q2
r1	r2	r1
r2	r2	r1

$q_0 = \{S\}$

$F = \{q1, r1\}$

Given the state diagram below for a finite automaton M_5 describe the machine formally:-



- **This machine** recognizes language A where: - **A= {w ∈ w contains at least one 1 and an even number of 0s follow the last 1**}. Therefore **L (M5) = A** or equivalently, **M5 recognizes A**.

Solution:

M= (Q, ∑, δ , q_0, F)

Where

Q= { q_1, q_2, ,q_3}

∑ = {0,1}

δ is the transition function

	Input Alphabet	
States	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

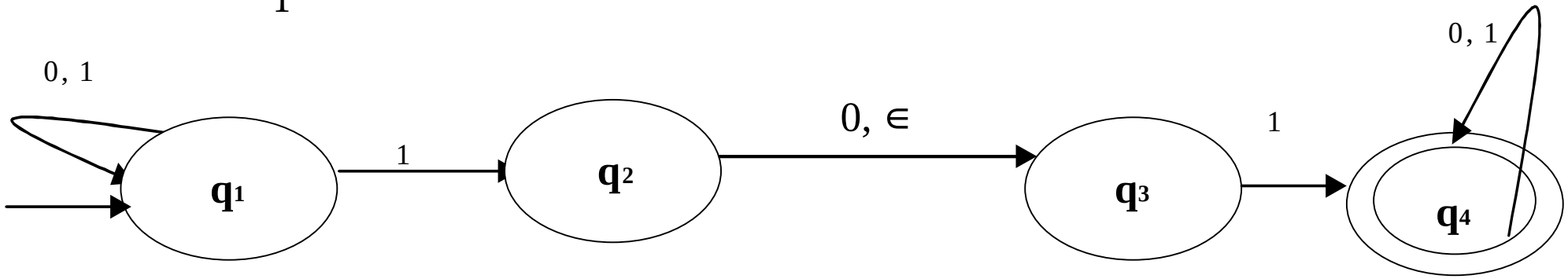
$q_0 = \{q_1\}$

$F = \{q_2\}$

Types of Automatons
2. Non-deterministic Finite Automata

- Non-determinism is a useful concept with a great impact on the theory of computation.
- So far in our discussion, every step of a computation follows in a **unique way from the preceding step**.
- When the machine is in a **given state and reads the next input symbol**, we **know what the next state will be** – it is **determined**. We call this **deterministic computation**.

- For NFAs, **several choices may exist for the next state at any point** and the machine therefore may have **additional features**.
- **Example: -** Consider the state diagram for an NFA machine N_1 below: -



- A good way to view the computation of this NFA is to say that:-
- The machine stays in the **start state q_1** until it "**guesses**" that it is **three places from the end**.
- At that point, **if the input symbol is a 1**, it **branches to state q_2 ,uses q_3 and q_4** to "**check**" whether ***its guess was correct***.

From this state diagram we can come up with a table showing **three differences between DFA and NFA** as follows: -

DFA	NFA
Every state of a DFA always has exactly one exiting transition arrow for each symbol in the alphabet.	A state may have zero, one or many exiting arrows for each alphabet symbol.
Labels on the transition arrows are symbols from the alphabet.	May have arrows labeled with members of the alphabet or ∈. Zero, one, or many arrows may exit from each state with the label ∈.
The transition function takes a state and an input symbol to produce the next state.	The transition function takes a state and input symbol or the empty string ∈, to produce the set of possible next states.

How does an NFA compute?

- Suppose we consider the above NFA (N_1). We realize when the NFA, reads an input string, there are **multiple ways to proceed**.
- **Example:** While in state q_1, if the next symbol is a 1, the machine splits into multiple loops and follows all the possibilities in parallel.

How does an NFA compute? *Cont…*

- If **a state with a ∈ symbol** on an exiting arrow is encountered, something similar happens. Without reading any input, the machine splits into multiple copies, one following each of the exiting ∈-labeled arrows and one staying at the current state.
- Then the machine **splits non-deterministically** as before.

- Non- determinism may be viewed as a kind of **parallel computation** wherein **several processes can be running concurrently**.
- When the NFA splits to follow several choices; this corresponds to **a process** known as "**forking**" into **several children each proceeding separately**.
- If **at least one of these processes leads to an accept state** then the **entire computation accepts**.

- Another way to think of a nondeterministic computation is as a **tree of possibilities**.
- The **root** of the tree corresponds to the **start of the computation**.
- Every **branching point** in the tree corresponds to a point in the computation at which the machine **has multiple choices**.
- The **machine accepts the computation if at least one of the computation branches ends in an accept state.**

Importance of NFA

- Every NFA can be converted into an equivalent DFA and constructing NFAs is sometimes easier than directly constructing DFAs.
- NFA may be much smaller than its deterministic counterpart, or its functioning may be easier to understand.
- Non-determinism in finite automata is also a good introduction to **non-determinism in more powerful computational models** because finite automata are especially easy to understand.

Formal Definition of NFA

- The formal definition of NFA is similar to that of a DFA.
- Both have **states, an input alphabet, a transition function, a start state and a collection of accept states**.
- However, they differ in the **type of transition function**.

- A set E with no elements is called an **empty set** and it is denoted by { } or ∅
- A machine **may accept several strings**, but it **always recognizes only one language**. If the machine **accepts no strings**, it still recognizes one language, namely, the **empty language**.
- For the NFA, the transition function produces **a set of possible next states.**
- For any **set Q** we write the power set of Q as **P(Q)** to be the collection of all subsets of Q. Here **P(Q)** is called the **power set of Q**.
- For any **input alphabet** $\sum$ we write $\sum_\epsilon$ to be $\sum \cup \{\epsilon\}$.
- Now we can easily write the formal description of the type of the transition function in an NFA. It is $\delta: Q \times \sum_\epsilon \rightarrow P(Q)$.

NFA Formal Definition

A Nondeterministic Finite Automaton is a **5-tuple** **$M= (Q,\Sigma, \delta , q_0, F)$;** where: -

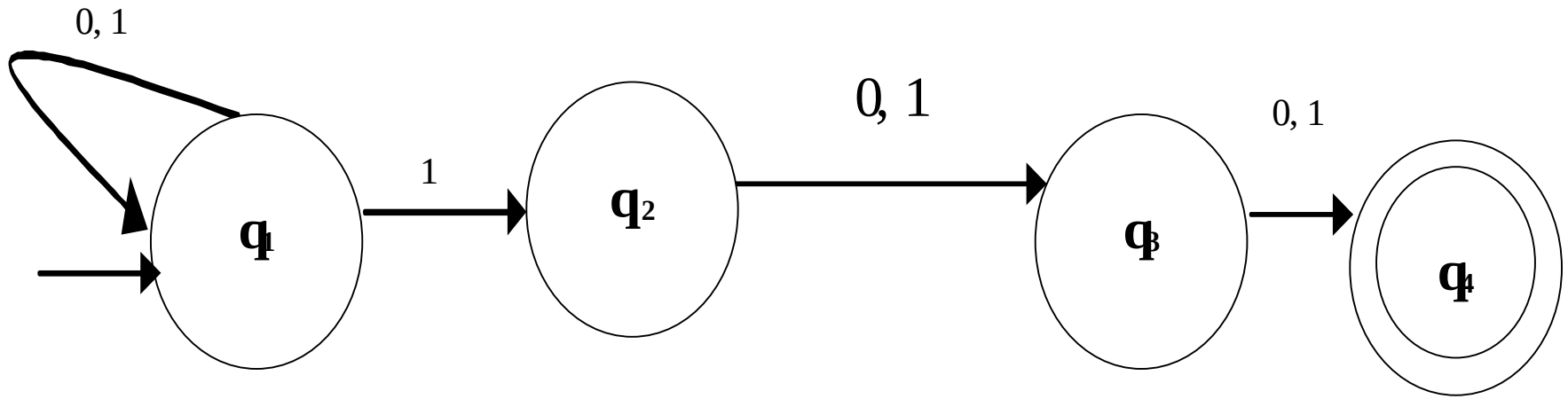
- **Q** is a **Finite set of Internal States**
- **Σ** is a **Finite Set of Symbols** "Input Alphabet"
- **δ: Q X $\Sigma_\epsilon \rightarrow$ P (Q)** is the **transition symbol**
- **$q_0 \in$ Q** is the **initial state**
- **F $\subseteq$Q** is the **Set of Final/accept States**

Recall that the formal definition precisely describes what we mean by a finite automaton.

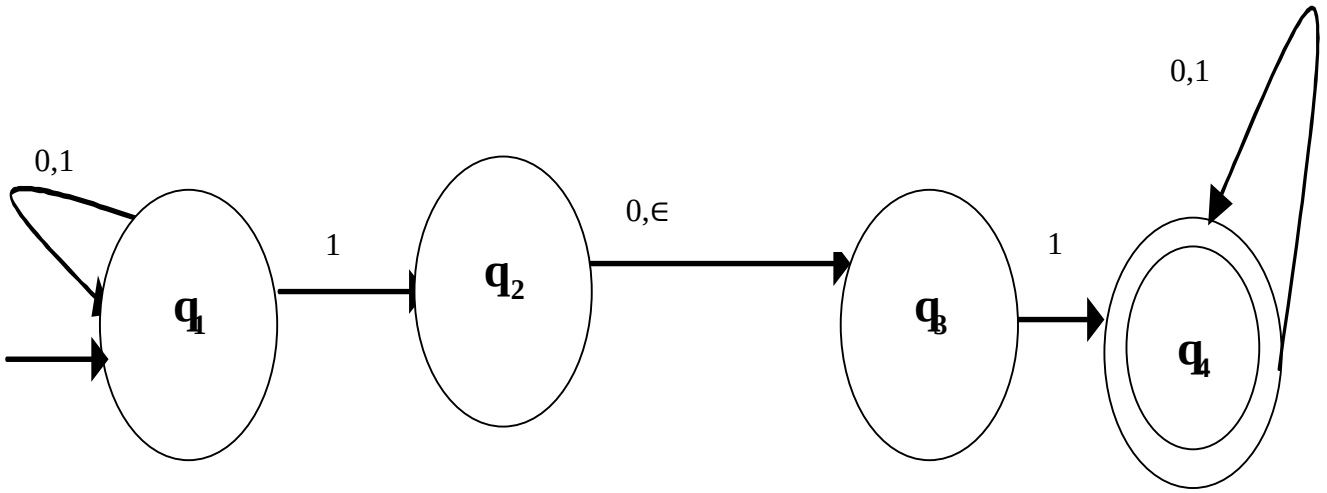
If **A** is the **set of all strings that machine M accepts**, we say that **A is the language of machine M** and write **L(M) = A**. We say that **M recognizes** A; or that **M accepts A**

Example: Let A be the language consisting of all strings over {0, 1} containing a 1 in the third position from the end (e.g. 000100 is in A but 0011 is not).

The following four-state NFA N2 recognizes A.



Example: Given the state diagram of an NFA machine N_1 below, provide the formal description



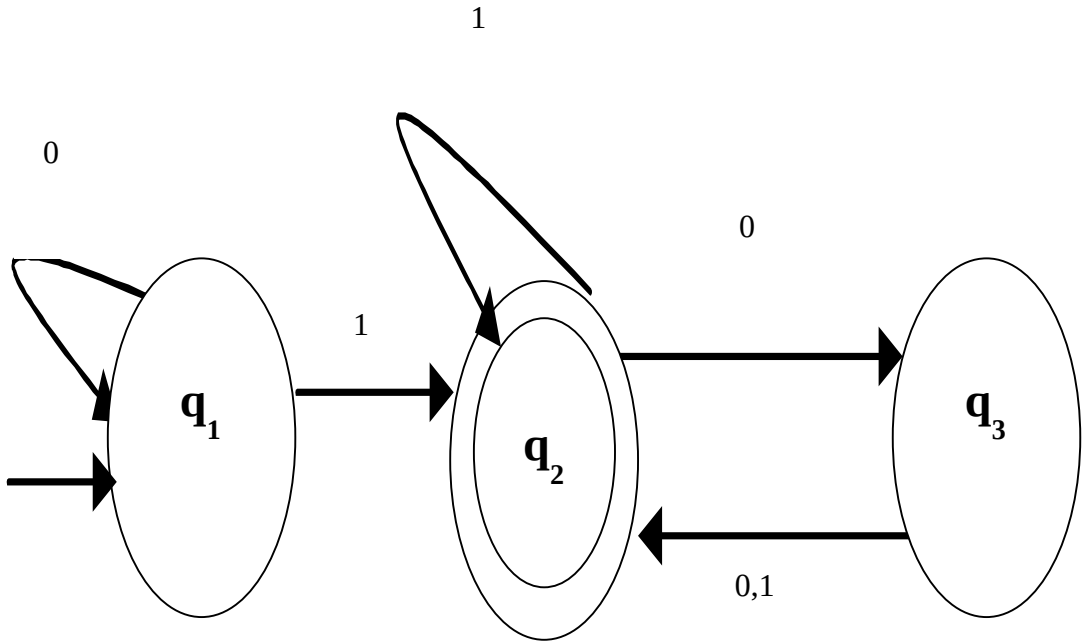
Solution:

- $Q = \{ q_1, q_2, , q_3, q_4\}$
- $\sum = \{0,1\}$
- δ is the transition function

	Input Alphabet		
States	0	1	∈
q1	{q1}	{q1, q2}	Ø
q2	{q3}	Ø	{q3}
q3	Ø	{q4}	Ø
q4	{q4}	{q4}	Ø

- q_0 the start state = $\{q_1\}$
- $F = \{q_4\}$

Example Two: Given the state diagram below for a finite automaton N_5, describe the machine formally:



Solution:

$N_5 = (Q, \sum, \delta, q_0, F)$

Where:

- $Q = \{ q_1, q_2, , q_3\}$
- $\sum = \{0,1\}$
- δ is the transition function

	Input Alphabet	
States	0	1
q1	q1	q2
q2	q3	q2
q3	q2	q2

- $q_0 = \{q_1\}$
- $F = \{q_2\}$

References

- Rowan G. & John T., (2009), *Discrete Mathematics: Proofs, Structures and Applications*, CRC Press, ISBN: 9781439812808.
- W. D. Wallis (2003), *A Beginners Guide to Discrete Mathematics*, Springer Science & Business Media, ISBN: 978-0817642693.
- Introduction to the theory of computation (3rd ed.), Michael, S. Boston, Cengage Learning. ISBN-13: 978-1133187790, (2012).
- Introduction to languages and the theory of computation (3rd ed.), Martin, J., New York: McGraw-Hill. ISBN-13: 978-0072322002, (2002)

Course: Automata Theory

Lecture 7: Models of Computation – Finite State Machines

Lecturer: Martha Gichuki