CS 133 : Automata Theory and Computability

Lecture Slides

Regular Languages and Finite Automata

Francis George C. Cabarle

Algorithms and Complexity Laboratory
Department of Computer Science
University of the Philippines Diliman
fccabarle@up.edu.ph

Day 2

Regular Languages and Finite Automata

Regular Languages and Finite Automata

Finite Automata

Finite Automata

- also known as finite state machines,

Finite Automata

- also known as finite state machines,
- are one of the simplest models of computation,

Finite Automata

- also known as finite state machines,
- are one of the simplest models of computation,
- are excellent models for computers with an extremely limited amount of memory,

Finite Automata

- also known as finite state machines,
- are one of the simplest models of computation,
- are excellent models for computers with an extremely limited amount of memory,

Examples:

- game configurations/ states (Remember the man-wolf-goat-cabbage finite (game) system?)
- automatic door
- elevator control
- vending machine
- calculator

Finite Automata

- also known as finite state machines,
- are one of the simplest models of computation,
- are excellent models for computers with an extremely limited amount of memory,
- are also the basis for programs that perform spell checking, grammar checking, indexing,

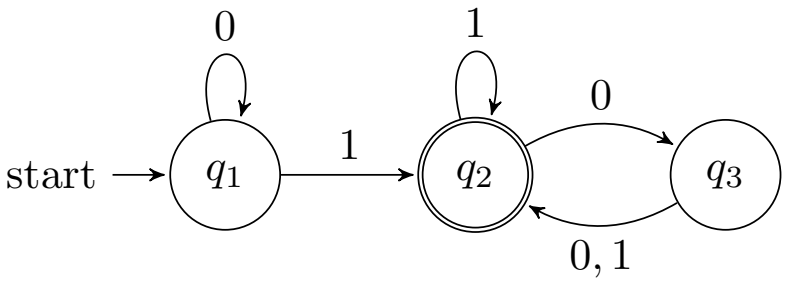
Finite Automata

- also known as finite state machines,
- are one of the simplest models of computation,
- are excellent models for computers with an extremely limited amount of memory,
- are also the basis for programs that perform spell checking, grammar checking, indexing,
- are useful tools when attempting to recognize patterns in data (also, *Markov chains*),

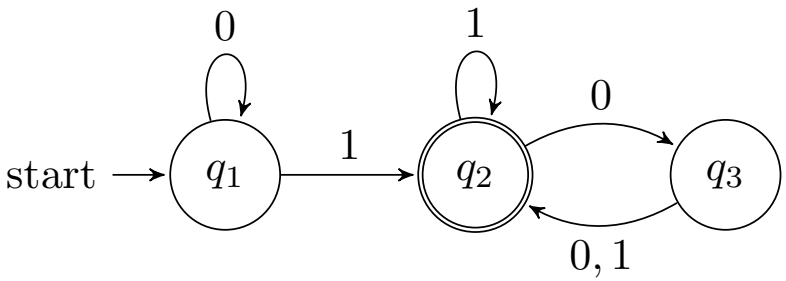
Finite Automata

- also known as finite state machines,
- are one of the simplest models of computation,
- are excellent models for computers with an extremely limited amount of memory,
- are also the basis for programs that perform spell checking, grammar checking, indexing,
- are useful tools when attempting to recognize patterns in data (also, *Markov chains*),
- fundamental in the study of programming languages and compilers.

A Finite Automaton that has three states, M_1



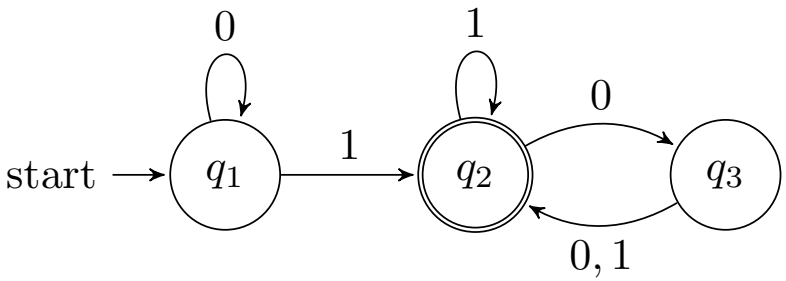
A Finite Automaton that has three states, M_1



Terms:

- state diagram,
- states,
- start state,
- accept state,
- transitions,
- *accept* or *reject* states.

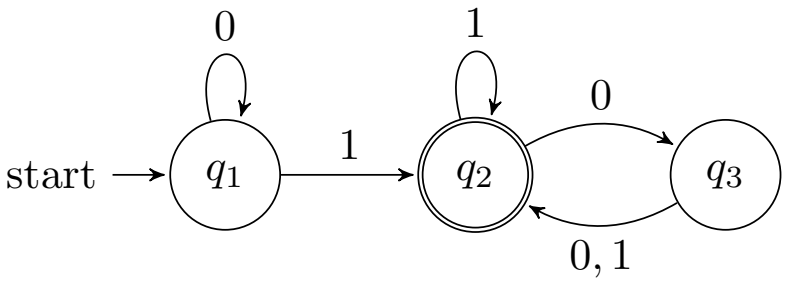
A Finite Automaton that has three states, M_1



Terms:

- state diagram,
- states,
- start state,
- accept state,
- transitions,
- *accept* or *reject* states.

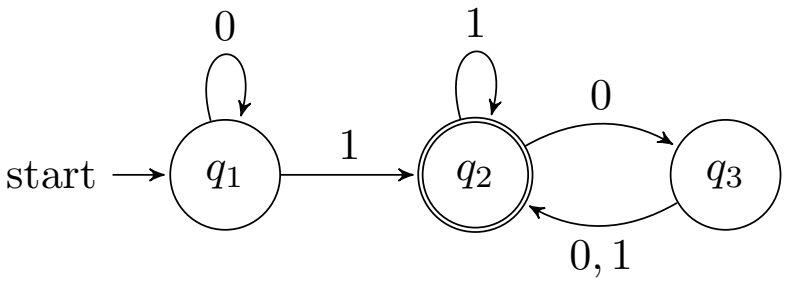
A Finite Automaton that has three states, M_1



Terms:

- state diagram,
- states,
- start state,
- accept state,
- transitions,
- *accept* or *reject* states.

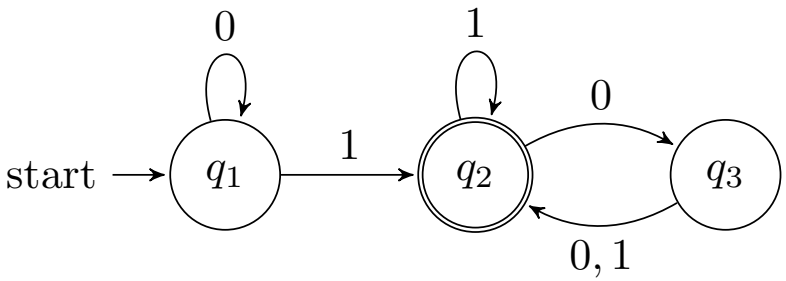
A Finite Automaton that has three states, M_1



Terms:

- state diagram,
- states,
- start state,
- accept state,
- transitions,
- *accept* or *reject* states.

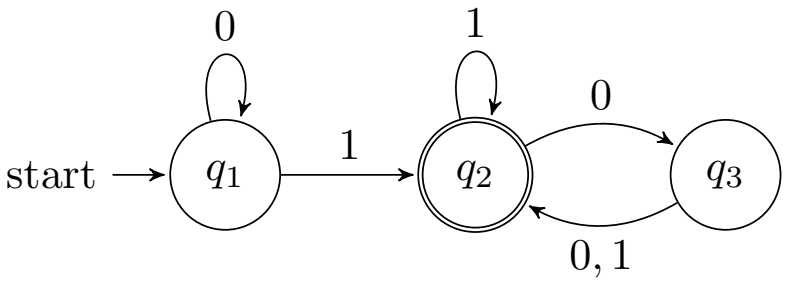
A Finite Automaton that has three states, M_1



Terms:

- state diagram,
- states,
- start state,
- accept state,
- transitions,
- *accept* or *reject* states.

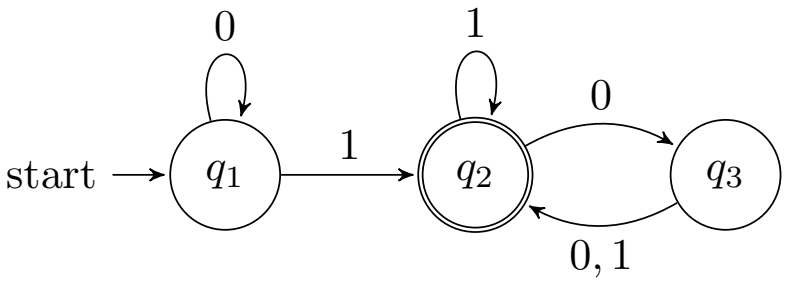
A Finite Automaton that has three states, M_1



Terms:

- state diagram,
- states,
- start state,
- accept state,
- transitions,
- *accept* or *reject* states.

A Finite Automaton that has three states, M_1



Terms:

- state diagram,
- states,
- start state,
- accept state,
- transitions,
- *accept* or *reject* states.

Regular Languages and Finite Automata

Definition

A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$,

Definition

A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the states,

Definition

A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the states,
2. Σ is a finite set called the alphabet,

Definition

A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the states,
2. Σ is a finite set called the alphabet,
3. $\delta : Q \times \Sigma \to Q$ is the transition function,

Definition

A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the states,
2. Σ is a finite set called the alphabet,
3. $\delta : Q \times \Sigma \to Q$ is the transition function,
4. $q_0 \in Q$ is the initial or start state,

Definition

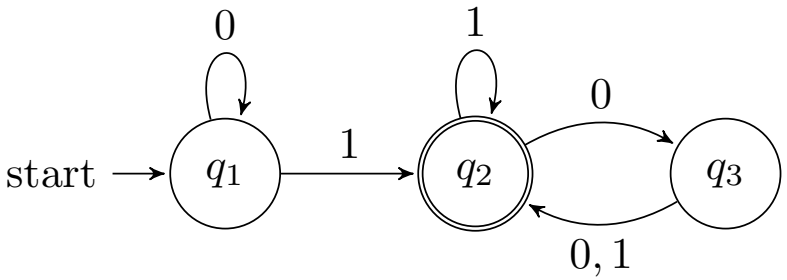
A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the states,
2. Σ is a finite set called the alphabet,
3. $\delta : Q \times \Sigma \to Q$ is the transition function,
4. $q_0 \in Q$ is the initial or start state, and
5. $F \subseteq Q$ is the set of accept (final) states.

Definition

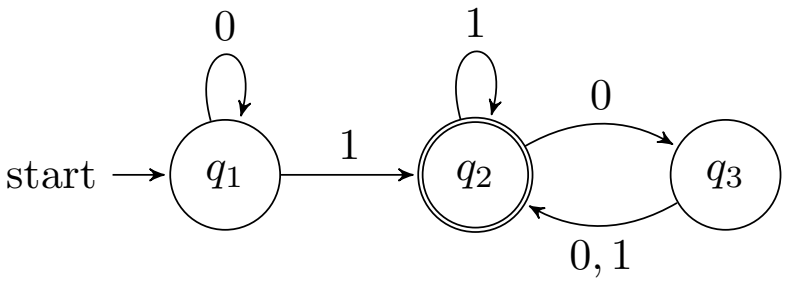
A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the states,
2. Σ is a finite set called the alphabet,
3. $\delta : Q \times \Sigma \to Q$ is the transition function,
4. $q_0 \in Q$ is the initial or start state, and
5. $F \subseteq Q$ is the set of accept (final) states.



Exercise: Describe M_1 formally.

State Diagrams for Finite Automata



State diagrams can be used to represent finite automata as follows:

- The states are represented as vertices
- Directed edges labeled with the input represent the transitions
- Final states are drawn with double circles
- A special arrow pointing from nowhere indicates the start state

Finite Automata Terminology

Definition

We say that an input string ω is **accepted** by a finite automaton M if M begins from the start state q_0 and ends at an accept state.

Finite Automata Terminology

Definition

We say that an input string ω is **accepted** by a finite automaton M if M begins from the start state q_0 and ends at an accept state.

The language **recognized** by the finite automaton M, denoted by $L(M)$, is the set of all strings that are accepted by M.

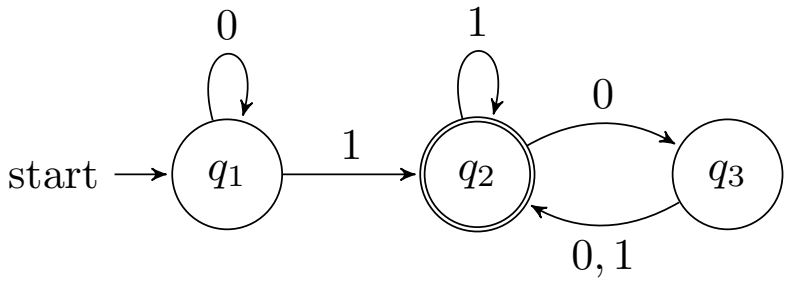
Finite Automata Terminology

Definition

We say that an input string ω is **accepted** by a finite automaton M if M begins from the start state q_0 and ends at an accept state.

The language **recognized** by the finite automaton M, denoted by $L(M)$, is the set of all strings that are accepted by M.

Exercise: Describe $L(M_1)$, i.e. the collection of strings accepted by automaton M_1.



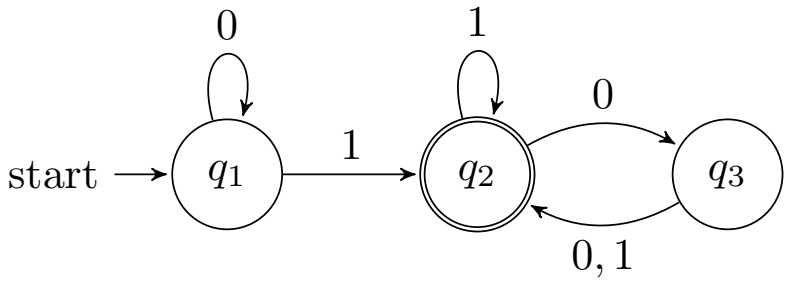
Finite Automata Terminology

Definition

We say that an input string ω is **accepted** by a finite automaton M if M begins from the start state q_0 and ends at an accept state.

The language **recognized** by the finite automaton M, denoted by $L(M)$, is the set of all strings that are accepted by M.

Exercise: Describe $L(M_1)$, i.e. the collection of strings accepted by automaton M_1.



Let $A = \{\omega \mid \omega$ contains at least one 1 and an even number of 0s follow the last 1$\}$.

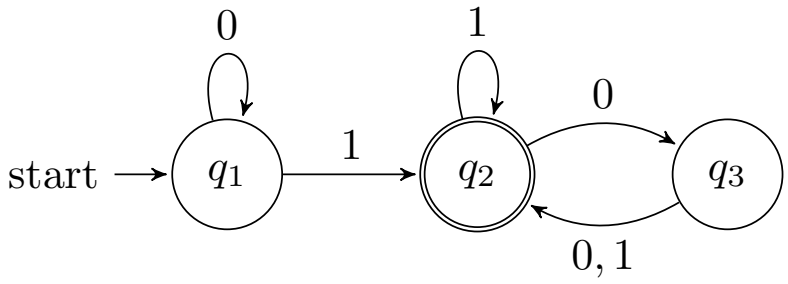
Finite Automata Terminology

Definition

We say that an input string ω is **accepted** by a finite automaton M if M begins from the start state q_0 and ends at an accept state.

The language **recognized** by the finite automaton M, denoted by $L(M)$, is the set of all strings that are accepted by M.

Exercise: Describe $L(M_1)$, i.e. the collection of strings accepted by automaton M_1.



Let $A = \{\omega \mid \omega$ contains at least one 1 and an even number of 0s follow the last 1$\}$.
Then $L(M_1) = A$,

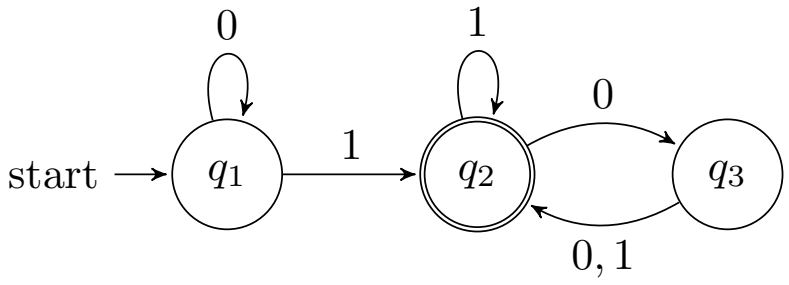
Finite Automata Terminology

Definition

We say that an input string ω is **accepted** by a finite automaton M if M begins from the start state q_0 and ends at an accept state.

The language **recognized** by the finite automaton M, denoted by $L(M)$, is the set of all strings that are accepted by M.

Exercise: Describe $L(M_1)$, i.e. the collection of strings accepted by automaton M_1.



Let $A = \{\omega \mid \omega$ contains at least one 1 and an even number of 0s follow the last 1$\}$.
Then $L(M_1) = A$, or equivalently,

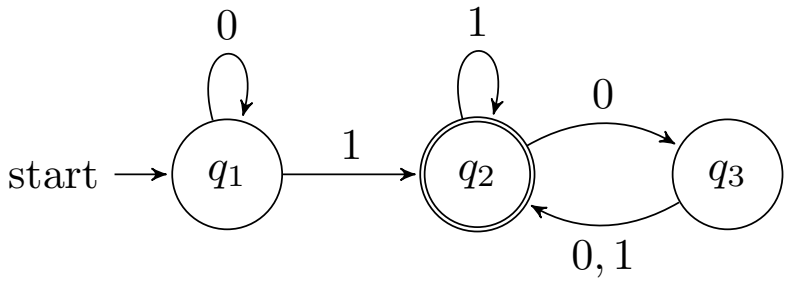
Finite Automata Terminology

Definition

We say that an input string ω is **accepted** by a finite automaton M if M begins from the start state q_0 and ends at an accept state.

The language **recognized** by the finite automaton M, denoted by $L(M)$, is the set of all strings that are accepted by M.

Exercise: Describe $L(M_1)$, i.e. the collection of strings accepted by automaton M_1.



Let $A = \{\omega \mid \omega$ contains at least one 1 and an even number of 0s follow the last 1$\}$.
Then $L(M_1) = A$, or equivalently, M_1 recognizes A.

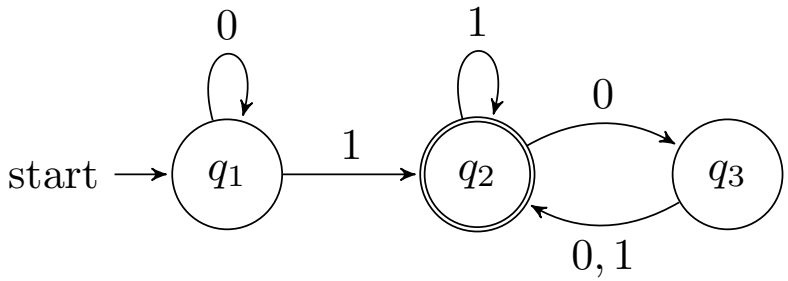
Finite Automata Terminology

Definition

We say that an input string ω is **accepted** by a finite automaton M if M begins from the start state q_0 and ends at an accept state.

The language **recognized** by the finite automaton M, denoted by $L(M)$, is the set of all strings that are accepted by M.

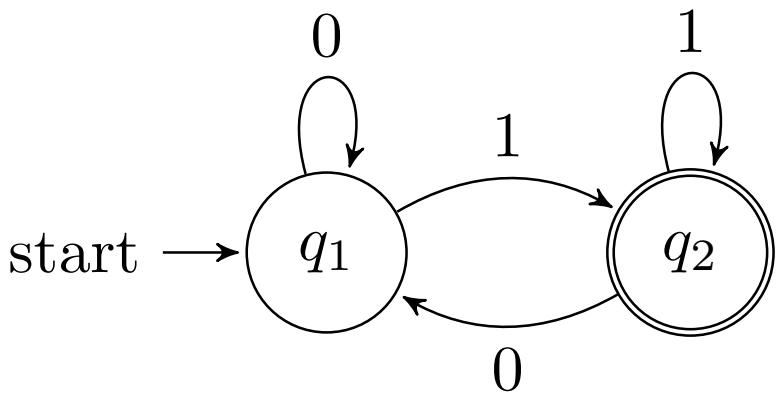
Exercise: Describe $L(M_1)$, i.e. the collection of strings accepted by automaton M_1.



Let $A = \{\omega \mid \omega$ contains at least one 1 and an even number of 0s follow the last 1$\}$.
Then $L(M_1) = A$, or equivalently, M_1 recognizes A.
Note: A machine may accept several strings, but it always recognizes only one language.

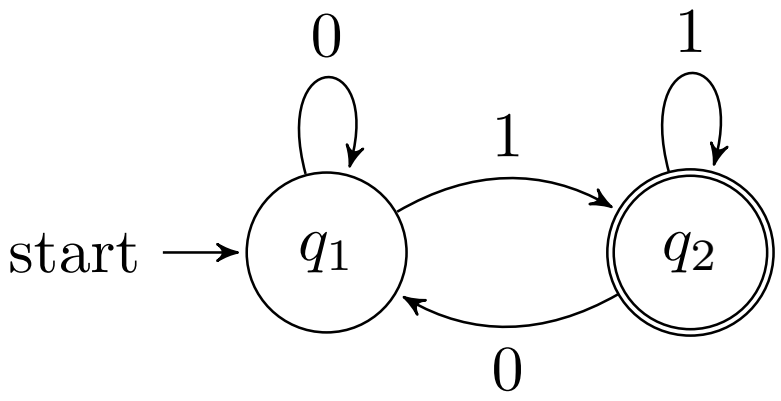
Examples

M2:

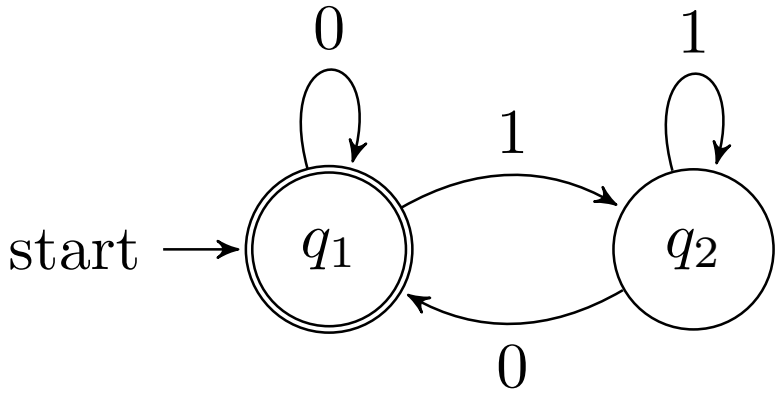


Examples

M2:

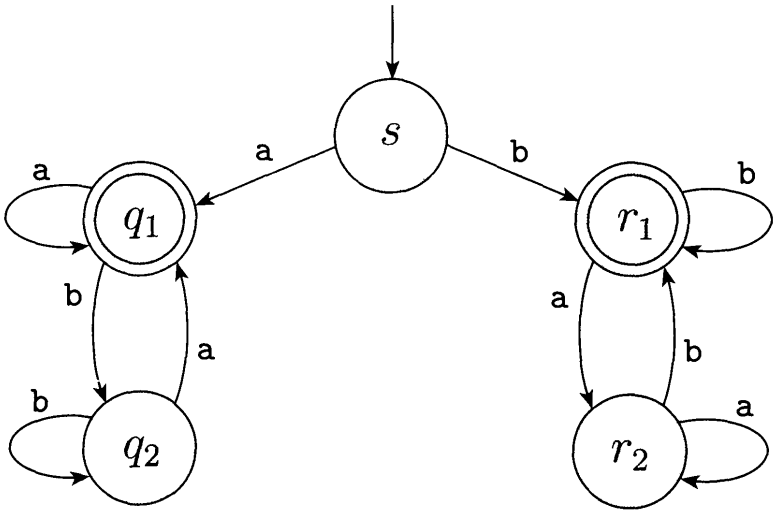


M3:



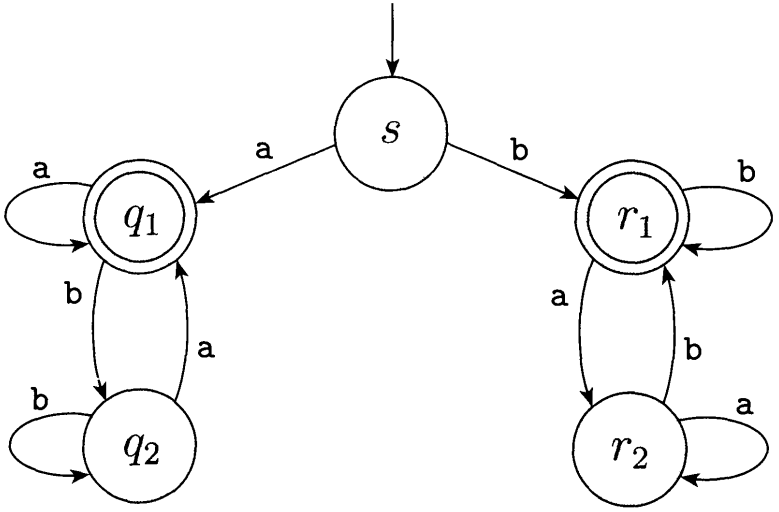
Examples

M4:

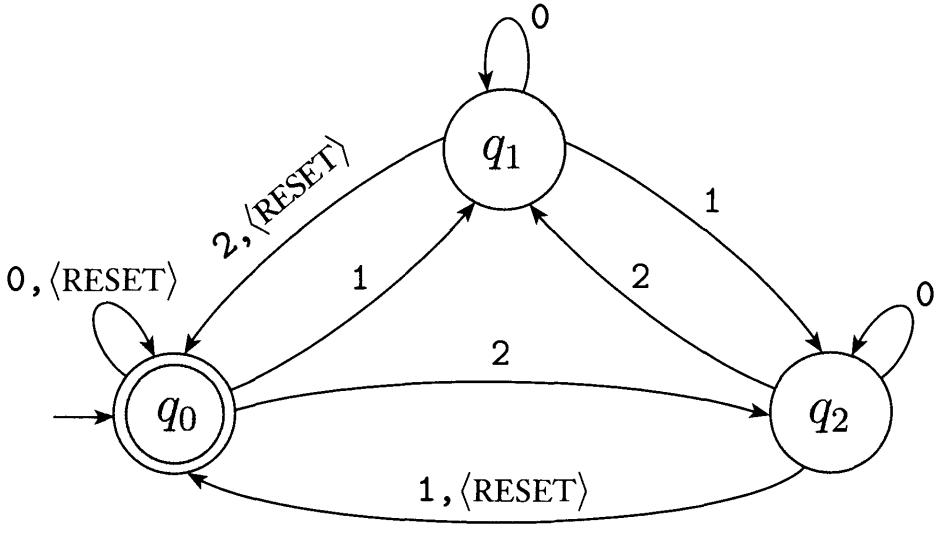


Examples

M4:



M5:



Formal Definition of Computation

Definition

Let M be a finite automaton $M = (Q, \Sigma, \delta, q_0, F)$, $w = w_1 w_2 \cdots w_n$ be a string where each $w_i \in \Sigma$ and $1 \leq i \leq n$.
M accepts string w if a sequence of states $r_0, r_1, \ldots, r_n$ exists, where each $r_i \in Q$, **satisfying** the following three conditions:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, 1, \ldots, n-1$,
3. $r_n \in F$.

We say that M recognizes A if $A = \{w \mid M \text{ accepts } w\}$.

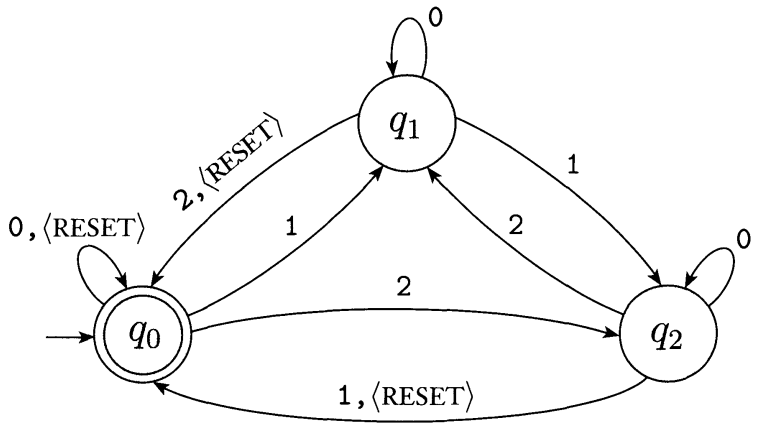
Formal Definition of Computation

Definition

Let M be a finite automaton $M = (Q, \Sigma, \delta, q_0, F)$, $w = w_1 w_2 \cdots w_n$ be a string where each $w_i \in \Sigma$ and $1 \leq i \leq n$.
M accepts string w if a sequence of states $r_0, r_1, \ldots, r_n$ exists, where each $r_i \in Q$, **satisfying** the following three conditions:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, 1, \ldots, n-1$,
3. $r_n \in F$.

We say that M recognizes A if $A = \{w \mid M \text{ accepts } w\}$.



Regular Languages and Finite Automata

Designing Finite Automata

Suppose you are given some language L. If possible, you want to design a finite automaton that will recognize it.

Designing Finite Automata

Suppose you are given some language L. If possible, you want to design a finite automaton that will recognize it.

- The input is received one element at a time. You never know when the input will end, so after each symbol, you must always be ready with a decision or answer: *accept* or *reject*?

Designing Finite Automata

Suppose you are given some language L. If possible, you want to design a finite automaton that will recognize it.

- The input is received one element at a time. You never know when the input will end, so after each symbol, you must always be ready with a decision or answer: *accept* or *reject*?

- Design a finite automaton that will recognize the following language over the alphabet $\{0, 1\}$: $L_1 = \{1, 00\}$

Designing Finite Automata

Suppose you are given some language L. If possible, you want to design a finite automaton that will recognize it.

- The input is received one element at a time. You never know when the input will end, so after each symbol, you must always be ready with a decision or answer: *accept* or *reject*?
- States are meant to be used as memory. Each state remembers some piece of information.

Designing Finite Automata

Suppose you are given some language L. If possible, you want to design a finite automaton that will recognize it.

- The input is received one element at a time. You never know when the input will end, so after each symbol, you must always be ready with a decision or answer: *accept* or *reject*?
- States are meant to be used as memory. Each state remembers some piece of information.

- Design a finite automaton that will recognize the following language over the alphabet $\{0, 1\}$: $L_2 = \{\omega \mid \omega \text{ has an odd number of } 1s\}$

Designing Finite Automata

Suppose you are given some language L. If possible, you want to design a finite automaton that will recognize it.

- The input is received one element at a time. You never know when the input will end, so after each symbol, you must always be ready with a decision or answer: *accept* or *reject*?
- States are meant to be used as memory. Each state remembers some piece of information.

- Design a finite automaton that will recognize the following language over the alphabet $\{0, 1\}$: $L_3 = \{\omega \mid \omega \text{ contains substring } 001\}$

Designing Finite Automata

Suppose you are given some language L. If possible, you want to design a finite automaton that will recognize it.

- The input is received one element at a time. You never know when the input will end, so after each symbol, you must always be ready with a decision or answer: *accept* or *reject*?
- States are meant to be used as memory. Each state remembers some piece of information.
- If language is finite, and longest string has length n, then we can build a trivial finite automaton that has one state for each node in a standard computation tree with height n. Each node corresponds to a finite string w of length less than n. If $w \in L$, we make that state (node) an accepting state. This is not true if n is not fixed – since we only have a finite number of states...

Designing Finite Automata

Suppose you are given some language L. If possible, you want to design a finite automaton that will recognize it.

- The input is received one element at a time. You never know when the input will end, so after each symbol, you must always be ready with a decision or answer: *accept* or *reject*?
- States are meant to be used as memory. Each state remembers some piece of information.
- If language is finite, and longest string has length n, then we can build a trivial finite automaton that has one state for each node in a standard computation tree with height n. Each node corresponds to a finite string w of length less than n. If $w \in L$, we make that state (node) an accepting state. This is not true if n is not fixed – since we only have a finite number of states...

- Design a finite automaton that will recognize the following language over the alphabet $\{0, 1\}$: $L_4 = \{\omega \mid \omega \text{ contains more } 0s \text{ than } 1s\}$

Designing Finite Automata

Suppose you are given some language L. If possible, you want to design a finite automaton that will recognize it.

- The input is received one element at a time. You never know when the input will end, so after each symbol, you must always be ready with a decision or answer: *accept* or *reject*?
- States are meant to be used as memory. Each state remembers some piece of information.
- If language is finite, and longest string has length n, then we can build a trivial finite automaton that has one state for each node in a standard computation tree with height n. Each node corresponds to a finite string w of length less than n. If $w \in L$, we make that state (node) an accepting state. This is not true if n is not fixed – since we only have a finite number of states...
- Only *some languages* can be recognized by finite automata, known as **regular languages**.

Regular Languages and Finite Automata

Definition: A language is called a **regular language**
if some finite automaton recognizes it.

Exercise: What are some specific examples of regular languages?

Fin (wakas)

Thanks for the attention.

Questions?

Reading assignment(s)

- Chapter 0, 1.1 of [Sipser 2005]

References:

[Sipser 2005] M. Sipser. *Introduction to the Theory of Computation*: 2ed. PWS Publishing Company, 2005.
[Hopcroft, Ullman 1979] J. Hopcroft, J. Ullman. *Introduction to Automata Theory, Languages, and Computation*: Addison-Wesley, 1979.
[Hernandez 2014] CS133 lecture slides of N.H.S. Hernandez, 2014