Foundations of Computing

Lecture 2

Arkady Yerukhimovich

January 16, 2025

Outline

1 Academic Integrity Policies

2 Lecture 1 Review

3 Language accepted by M

4 Quiz Solutions

5 Building DFAs

6 Proving Correctness of a DFA

Homework Policies

Important

Any work you submit MUST be your own!

Homework Policies

Important

Any work you submit MUST be your own!

You may do the following:

- discuss general concepts/questions with others
- discuss similar problems not in homework (e.g., from the book(s))

Homework Policies

Important

Any work you submit MUST be your own!

You may do the following:

- discuss general concepts/questions with others
- discuss similar problems not in homework (e.g., from the book(s))

You may NOT do the following:

- Copy or provide answers to any hw problems to others
- Use ChatGPT or any other LLM to produce your answers
- Search the web for solutions or use services like chegg.com or StackExchange

Outline

1. Academic Integrity Policies
2. Lecture 1 Review
3. Language accepted by M
4. Quiz Solutions
5. Building DFAs
6. Proving Correctness of a DFA

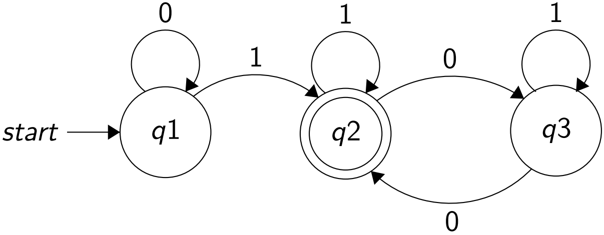
Lecture 1 Review

- Syllabus review and course details
- Strings, languages, and functions
- Finite automata

Outline

1. Academic Integrity Policies
2. Lecture 1 Review
3. Language accepted by M
4. Quiz Solutions
5. Building DFAs
6. Proving Correctness of a DFA

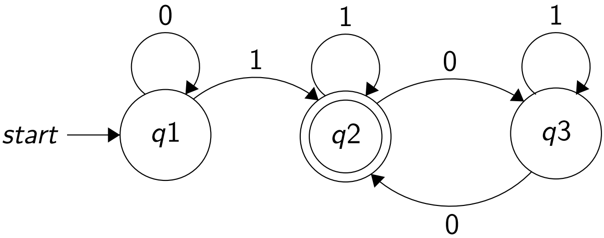
Language accepted by M



Accepting a string

- M accepts a string x (over Σ) if $M(x)$ stops in an accept state
- What strings does M accept?

Language accepted by M



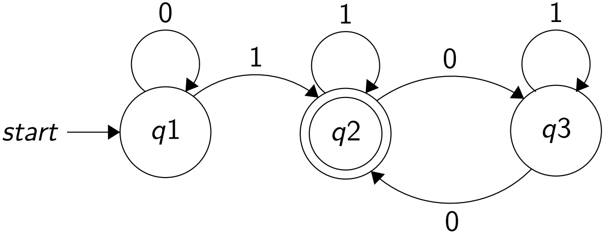
Accepting a string

- M accepts a string x (over Σ) if $M(x)$ stops in an accept state
- What strings does M accept?

Deciding a language

- M <u>decides</u> a language L if it accepts:
  - ALL strings in L, and
  - NO strings not in L

Language accepted by M



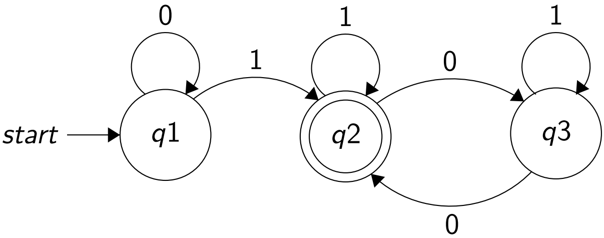
Accepting a string

- M accepts a string x (over Σ) if $M(x)$ stops in an accept state
- What strings does M accept?

Deciding a language

- M decides a language L if it accepts:
  - ALL strings in L, and
  - NO strings not in L
- Every M accepts exactly one language $L(M)$

What language does M accept?



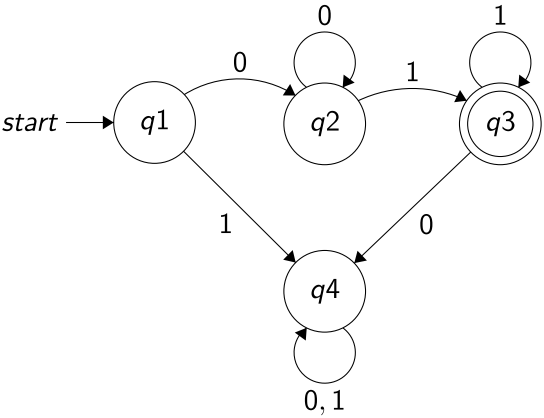
$L(M)$:

- String must contain at least one 1
- After the first string of 1's, there must be an even number of 0's or no 0's

Outline

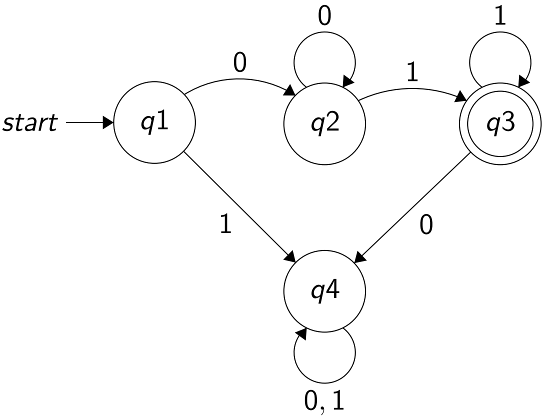
1. Academic Integrity Policies
2. Lecture 1 Review
3. Language accepted by M
4. Quiz Solutions
5. Building DFAs
6. Proving Correctness of a DFA

Quiz Solutions



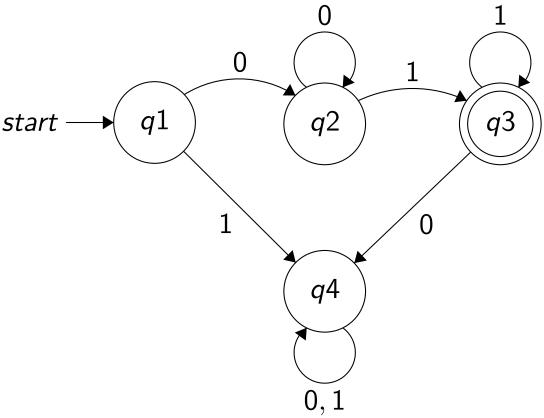
- Does M accept 00011?:

Quiz Solutions



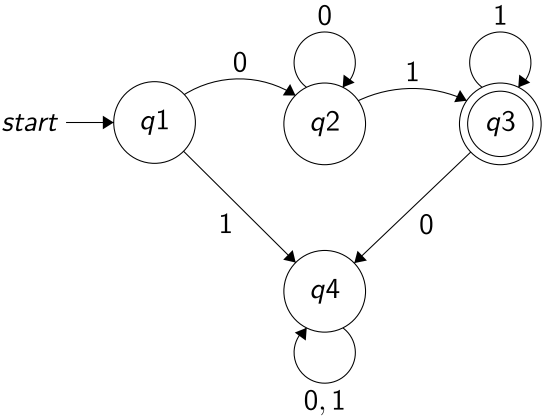
- Does M accept 00011?:
- Does M accept 01100?

Quiz Solutions



- Does M accept 00011?:
- Does M accept 01100?
- Describe the language $L(M)$:

Quiz Solutions



- Does M accept 00011?:
- Does M accept 01100?
- Describe the language $L(M)$: all strings with one or more 0s followed by one or more 1s

Outline

1 Academic Integrity Policies

2 Lecture 1 Review

3 Language accepted by M

4 Quiz Solutions

5 Building DFAs

6 Proving Correctness of a DFA

Important Rules of Deterministic Finite Automata

Deterministic Finite Automata

- Transition function must be fully defined:
  - For every state in Q, for every symbol in Σ, δ must specify a next state

Important Rules of Deterministic Finite Automata

Deterministic Finite Automata

- Transition function must be fully defined:
  - For every state in Q, for every symbol in Σ, δ must specify a next state
- Transition function must be a function:
  - For every state in Q, for every symbol in Σ, δ must specify exactly one next state

Important Rules of Deterministic Finite Automata

Deterministic Finite Automata

- Transition function must be fully defined:
  - For every state in Q, for every symbol in Σ, δ must specify a next state
- Transition function must be a function:
  - For every state in Q, for every symbol in Σ, δ must specify exactly one next state

Important: Deterministic means that the execution of M on any input in Σ^* must be fully specified.

DFA as an Algorithm

DFA Execution

1. Read next input symbol and use transition function to determine next step until run out of input symbols
2. If stop in accept state, then output 1

DFA as an Algorithm

DFA Execution

1. Read next input symbol and use transition function to determine next step until run out of input symbols
2. If stop in accept state, then output 1

Memory in a DFA:

- Each state stores a summary of the input seen so far
- Next state depends on the current state and the next symbol
- Think of this as an "if" statement

DFA as an Algorithm

DFA Execution

1. Read next input symbol and use transition function to determine next step until run out of input symbols
2. If stop in accept state, then output 1

Memory in a DFA:

- Each state stores a summary of the input seen so far
- Next state depends on the current state and the next symbol
- Think of this as an "if" statement

Important

Since $|Q|$ is finite, need to be able to take in inputs longer than the number of states

- Cannot just store the entire string!

Example 1

Problem

Build a DFA that decides

$$L = \{w | w \in \{0,1\}^* \text{ and } w \text{ contains the substring } 101\}$$

Example 1

Problem

Build a DFA that decides

$$L = \{w | w \in \{0,1\}^* \text{ and } w \text{ contains the substring 101}\}$$

Building the DFA:

- Idea: State should store the part of 101 seen so far

Example 1

Problem

Build a DFA that decides

$$L = \{w | w \in \{0,1\}^* \text{ and } w \text{ contains the substring } 101\}$$

Building the DFA:

- Idea: State should store the part of 101 seen so far
- Transition function should change state depending on whether next symbol fits pattern

Example 1

Problem

Build a DFA that decides

$$L = \{w | w \in \{0, 1\}^* \text{ and } w \text{ contains the substring } 101\}$$

Building the DFA:

- Idea: State should store the part of 101 seen so far
- Transition function should change state depending on whether next symbol fits pattern

Observations:

- If see a 0:
  - this cannot be the first symbol of 101
  - but can be second character if previous symbol was a 1

Example 1

Problem

Build a DFA that decides

$$L = \{w | w \in \{0,1\}^* \text{ and } w \text{ contains the substring 101}\}$$

Building the DFA:

- Idea: State should store the part of 101 seen so far
- Transition function should change state depending on whether next symbol fits pattern

Observations:

- If see a 0:
  - this cannot be the first symbol of 101
  - but can be second character if previous symbol was a 1
- If see a 1:
  - this can be the first character of 101
  - or, it can be the last character if we previously saw 10 – in this case, we should accept

Example 1 – The Algorithm

Problem

Build a DFA that decides

$$L = \{w | w \in \{0,1\}^* \text{ and } w \text{ contains the substring } 101\}$$

Algorithm:

1. Start:
   - If read a 0, stay in step 1 – first symbol cannot be a 0
   - If read a 1, goto step 2 – record that we saw a 1

Example 1 – The Algorithm

Problem

Build a DFA that decides

$$L = \{w|w \in \{0,1\}^* \text{ and } w \text{ contains the substring } 101\}$$

Algorithm:

1. Start:
   - If read a 0, stay in step 1 – first symbol cannot be a 0
   - If read a 1, goto step 2 – record that we saw a 1
2. Step 2:
   - If read a 0, goto step 3 – record that we saw 10
   - If read a 1, stay in step 2 – may be first 1 of 101

Example 1 – The Algorithm

Problem

Build a DFA that decides

$$L = \{w | w \in \{0,1\}^* \text{ and } w \text{ contains the substring } 101\}$$

Algorithm:

1. Start:
   - If read a 0, stay in step 1 – first symbol cannot be a 0
   - If read a 1, goto step 2 – record that we saw a 1
2. Step 2:
   - If read a 0, goto step 3 – record that we saw 10
   - If read a 1, stay in step 2 – may be first 1 of 101
3. Step 3:
   - If read a 0, goto step 1 – this is not 101, time to start over
   - If read a 1, goto step 4 – we have seen 101

Example 1 – The Algorithm

Problem

Build a DFA that decides

$$L = \{w | w \in \{0, 1\}^* \text{ and } w \text{ contains the substring 101}\}$$

Algorithm:

1. Start:
   - If read a 0, stay in step 1 – first symbol cannot be a 0
   - If read a 1, goto step 2 – record that we saw a 1
2. Step 2:
   - If read a 0, goto step 3 – record that we saw 10
   - If read a 1, stay in step 2 – may be first 1 of 101
3. Step 3:
   - If read a 0, goto step 1 – this is not 101, time to start over
   - If read a 1, goto step 4 – we have seen 101
4. Step 4:
   - On any input, stay in step 4 and accept

Build the DFA

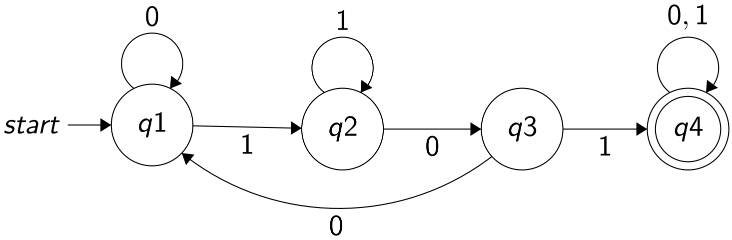
1. Start:
   - If read a 0, stay in step 1 – first symbol cannot be a 0
   - If read a 1, goto step 2 – record that we saw a 1
2. Step 2:
   - If read a 0, goto step 3 – record that we saw 10
   - If read a 1, stay in step 2 – may be first 1 of 101
3. Step 3:
   - If read a 0, goto step 1 – this is not 101, time to start over
   - If read a 1, goto step 4 – we have seen 101
4. Step 4:
   - On any input, stay in step 4 and accept

The DFA

Problem

Build a DFA that decides

$$L = \{w | w \in \{0,1\}^* \text{ and } w \text{ contains the substring } 101\}$$



1. $q1$ – not yet read first 1 in 101
2. $q2$ – last input was a 1, could be start of 101
3. $q3$ – have read 10
4. $q4$ – have read 101

Trap States

A useful tool for designing DFAs:

- Trap states allow you to “reject” as soon as you know that $w \notin L$

Trap States

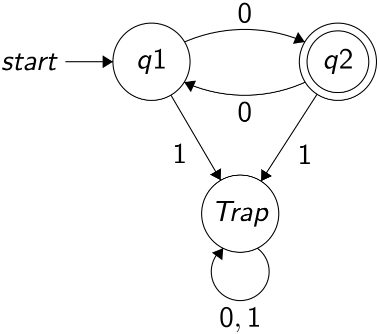
A useful tool for designing DFAs:

- Trap states allow you to "reject" as soon as you know that $w \notin L$
- Trap states have no out edges – no way to get to accept

Trap States

A useful tool for designing DFAs:

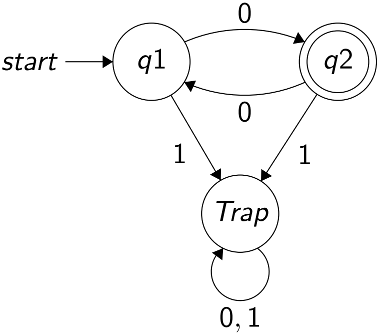
- Trap states allow you to "reject" as soon as you know that $w \notin L$
- Trap states have no out edges – no way to get to accept



Trap States

A useful tool for designing DFAs:

- Trap states allow you to “reject” as soon as you know that $w \notin L$
- Trap states have no out edges – no way to get to accept



For convenience

You can omit edges from transition diagram that point to the trap state

Exercise

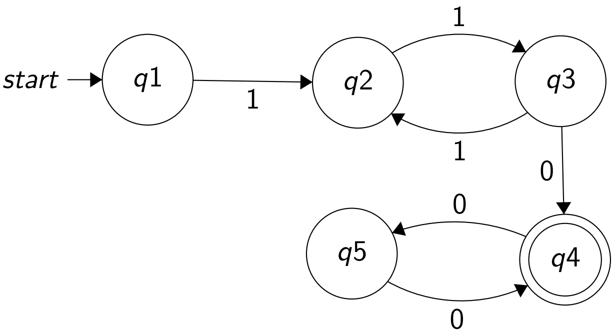
Problem

Build a DFA that decides:
$L = \{w | w \in \{0, 1\}^*$ that consists of an even number (≥ 2) 1's followed by an odd number (≥ 1) 0's$\}$

Exercise

Problem

Build a DFA that decides:
$L = \{w | w \in \{0,1\}^*$ that consists of an even number (≥ 2) 1's followed by an odd number (≥ 1) 0's$\}$

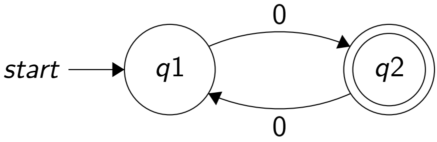


Outline

1. Academic Integrity Policies
2. Lecture 1 Review
3. Language accepted by M
4. Quiz Solutions
5. Building DFAs
6. Proving Correctness of a DFA

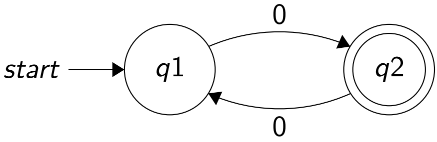
Another Example

Consider the following DFA M



Another Example

Consider the following DFA M

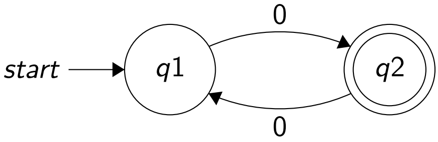


Theorem: This DFA recognizes

$$L = \{w \in \{0,1\}^* | w \text{ has odd number of 0s and no 1s}\}$$

Another Example

Consider the following DFA M



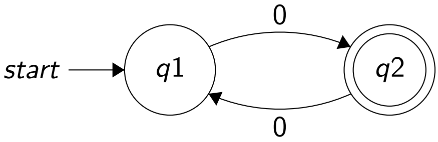
Theorem: This DFA recognizes

$$L = \{w \in \{0,1\}^* | w \text{ has odd number of 0s and no 1s}\}$$

Proof:

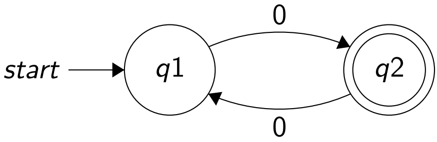
- Need to prove that $L = L(M)$
- Instead we prove the $L \subseteq L(M)$ and $L(M) \subseteq L$

$L \subseteq L(M)$



$$L = \{w \in \{0,1\}^* | w \text{ has odd number of 0s and no 1s}\}$$

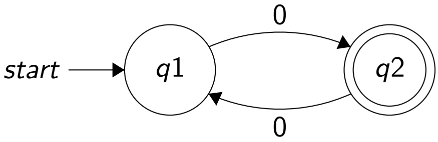
$L \subseteq L(M)$



$$L = \{w \in \{0,1\}^* | w \text{ has odd number of 0s and no 1s}\}$$

Claim: Every $w \in L$ will cause M to accept (i.e., stop in $q2$).

$L \subseteq L(M)$

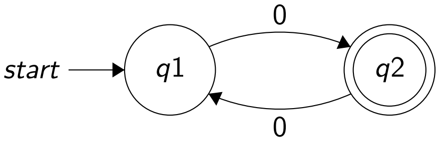


$$L = \{w \in \{0,1\}^* | w \text{ has odd number of 0s and no 1s}\}$$

Claim: Every $w \in L$ will cause M to accept (i.e., stop in $q2$).

Base Case:
If $|w| = 1$ and $w \in L$ then $w = 0$ and $M(w) = 1$

$L \subseteq L(M)$



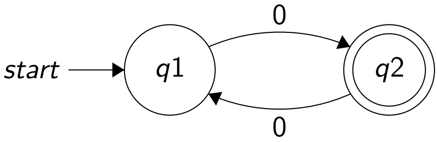
$$L = \{w \in \{0,1\}^* | w \text{ has odd number of 0s and no 1s}\}$$

Claim: Every $w \in L$ will cause M to accept (i.e., stop in $q2$).

Base Case:
If $|w| = 1$ and $w \in L$ then $w = 0$ and $M(w) = 1$

Inductive Hypothesis:
For any w of length k, if $w \in L$, $\delta^*(q1, w) = q2$

$L \subseteq L(M)$



$$L = \{w \in \{0,1\}^* | w \text{ has odd number of 0s and no 1s}\}$$

Claim: Every $w \in L$ will cause M to accept (i.e., stop in $q2$).

Base Case:
If $|w| = 1$ and $w \in L$ then $w = 0$ and $M(w) = 1$

Inductive Hypothesis:
For any w of length k, if $w \in L$, $\delta^*(q1, w) = q2$

Proof by Induction:
Consider $|w| = k + 2$ and let w' be the prefix of w of length k.
By hypothesis $\delta^*(q1, w') = q2$, and last two bits of w must be 0's
Hence $\delta^*(q1, w) = q2$

$L(M) \subseteq L$

Claim: Every w accepted by M is in L.

$L(M) \subseteq L$

Claim: Every w accepted by M is in L.

Proof by contradiction:
Assume there exists a string w accepted by M that is not in L

- i.e., has an even number of 0's or a 1

$L(M) \subseteq L$

Claim: Every w accepted by M is in L.

Proof by contradiction:
Assume there exists a string w accepted by M that is not in L

- i.e., has an even number of 0's or a 1

Proof:

1. w cannot have a 1, as any such input will not stop in $q2$
2. By similar proof to before, any w with even number of 0's must stop in $q1$
3. Contradiction!