CS 611: Theory of Computation

Hongmin Li

Department of Computer Science
California State University, East Bay

Defining an Automaton

To describe an automaton, we to need to specify

- What the alphabet is,
- What the states are,
- What the initial state is,
- What states are accepting/final, and
- What the transition from each state and input symbol is.

Thus, the above 5 things are part of the formal definition.

Finite Automata

Formal Definition

Definition

A finite automaton is $M = (Q, \Sigma, \delta, q_0, F)$, where

- Q is the finite set of states
- Σ is the finite alphabet
- $\delta : Q \times \Sigma \to Q$ "Next-state" transition function
- $q_0 \in Q$ initial state
- $F \subseteq Q$ final/accepting states

Deterministic Finite Automata

Formal Definition

Definition

A deterministic finite automaton (DFA) is $M = (Q, \Sigma, \delta, q_0, F)$, where

- Q is the finite set of states
- Σ is the finite alphabet
- $\delta : Q \times \Sigma \to Q$ "Next-state" transition function
- $q_0 \in Q$ initial state
- $F \subseteq Q$ final/accepting states

Given a state and a symbol, the next state is "determined".

Computation

Definition

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$, let us define a function $\hat{\delta} : Q \times \Sigma^* \to Q$ such that $\hat{\delta}(q, w)$ is M's state after reading w from state q.

Computation

Definition

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$, let us define a function $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$ such that $\hat{\delta}(q, w)$ is M's state after reading w from state q. Formally,

$$\hat{\delta}(q, w) = \begin{cases} & \text{if } w = \epsilon \\ & \text{if } w = ua \end{cases}$$

Computation

Definition

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$, let us define a function $\hat{\delta} : Q \times \Sigma^* \to Q$ such that $\hat{\delta}(q, w)$ is M's state after reading w from state q. Formally,

$$\hat{\delta}(q, w) = \begin{cases} q & \text{if } w = \epsilon \\ & \text{if } w = ua \end{cases}$$

Computation

Definition

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$, let us define a function $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$ such that $\hat{\delta}(q, w)$ is M's state after reading w from state q. Formally,

$$\hat{\delta}(q, w) = \begin{cases} q & \text{if } w = \epsilon \\ \delta(\hat{\delta}(q, u), a) & \text{if } w = ua \end{cases}$$

Computation

Definition

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$, let us define a function $\hat{\delta} : Q \times \Sigma^* \to Q$ such that $\hat{\delta}(q, w)$ is M's state after reading w from state q. Formally,

$$\hat{\delta}(q, w) = \begin{cases} q & \text{if } w = \epsilon \\ \delta(\hat{\delta}(q, u), a) & \text{if } w = ua \end{cases}$$

Definition

We say a DFA $M = (Q, \Sigma, \delta, q_0, F)$ accepts string $w \in \Sigma^*$ iff $\hat{\delta}(q_0, w) \in F$.

Acceptance/Recognition

Definition

The language accepted or recognized by a DFA M over alphabet Σ is $L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$.

Acceptance/Recognition

Definition

The language accepted or recognized by a DFA M over alphabet Σ is $L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$. A language L is said to be accepted/recognized by M if $L = L(M)$.

Acceptance/Recognition and Regular Languages

Definition

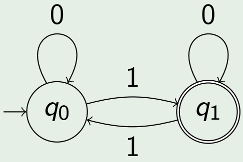
The language accepted or recognized by a DFA M over alphabet Σ is $L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$. A language L is said to be accepted/recognized by M if $L = L(M)$.

Definition

A language L is regular if there is some DFA M such that $L = L(M)$.

Formal Example of DFA

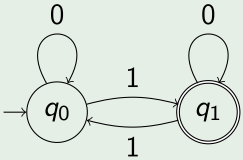
Example



Transition Diagram of DFA

Formal Example of DFA

Example



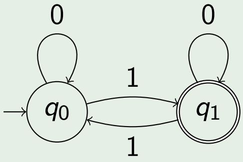
Transition Diagram of DFA

	0	1
q_0	q_0	q_1
q_1	q_1	q_0

Transition Table representation

Formal Example of DFA

Example



Transition Diagram of DFA

	0	1
q_0	q_0	q_1
q_1	q_1	q_0

Transition Table representation

Formally the automaton is $M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$ where

$$\delta(q_0, 0) = q_0 \qquad \delta(q_0, 1) = q_1$$
$$\delta(q_1, 0) = q_1 \qquad \delta(q_1, 1) = q_0$$

A Simple Observation about DFAs

Proposition

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$, and any strings $u, v \in \Sigma^*$ and state $q \in Q$, $\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v)$.

Proof.

By induction! Let's see . . . $\square$

Induction Proofs

An Example

Proposition

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$, and any strings $u, v \in \Sigma^*$ and state $q \in Q$, $\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v)$.

Proof.

We will prove this by induction.

- Let S_i be "$\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v)$ when $|v| = i$"
  - Observe that if S_i is true for all i then $\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v)$ for every u and v

Example Inductive Proof

Base Case

Proof (contd).

To establish S_0, i.e., "$\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v)$ when $|v| = 0$"

Example Inductive Proof

Base Case

Proof (contd).

To establish S_0, i.e., "$\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v)$ when $|v| = 0$"

- If $|v| = 0$ then $v = \epsilon$
- Observe $u\epsilon = u$
- Thus, LHS $= \hat{\delta}(q, u\epsilon) = \hat{\delta}(q, u)$
- Observe by definition of $\hat{\delta}(\cdot, \cdot)$, for any q', $\hat{\delta}(q', \epsilon) = q'$
- Thus, RHS $= \hat{\delta}(\hat{\delta}(q, u), \epsilon) = \hat{\delta}(q, u)$ $\cdot\cdot\longrightarrow$

Example Inductive Proof

Induction Step

Proof (contd).

Assume S_i, i.e., "$\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v)$ when $|v| = i$". Need to establish S_{i+1}.

$\square$

Example Inductive Proof

Induction Step

Proof (contd).

Assume S_i, i.e., "$\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v)$ when $|v| = i$". Need to establish S_{i+1}.

- Consider v such that $|v| = i + 1$.

$\square$

Example Inductive Proof

Induction Step

Proof (contd).

Assume S_i, i.e., "$\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v)$ when $|v| = i$". Need to establish S_{i+1}.

- Consider v such that $|v| = i + 1$. WLOG, $v = wa$, where $w \in \Sigma^*$ with $|w| = n$ and $a \in \Sigma$

$\square$

Example Inductive Proof

Induction Step

Proof (contd).

Assume S_i, i.e., "$\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v)$ when $|v| = i$". Need to establish S_{i+1}.

- Consider v such that $|v| = i + 1$. WLOG, $v = wa$, where $w \in \Sigma^*$ with $|w| = n$ and $a \in \Sigma$

$$\hat{\delta}(q, uwa) = \delta(\hat{\delta}(q, uw), a) \qquad \text{defn. of } \hat{\delta}$$

□

Example Inductive Proof

Induction Step

Proof (contd).

Assume S_i, i.e., "$\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v)$ when $|v| = i$". Need to establish S_{i+1}.

- Consider v such that $|v| = i + 1$. WLOG, $v = wa$, where $w \in \Sigma^*$ with $|w| = n$ and $a \in \Sigma$

$$
\begin{aligned}
\hat{\delta}(q, uwa) &= \delta(\hat{\delta}(q, uw), a) && \text{defn. of } \hat{\delta} \\
&= \delta(\hat{\delta}(\hat{\delta}(q, u), w), a) && \text{ind. hyp.}
\end{aligned}
$$

□

Example Inductive Proof

Induction Step

Proof (contd).

Assume S_i, i.e., "$\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v)$ when $|v| = i$". Need to establish S_{i+1}.

- Consider v such that $|v| = i + 1$. WLOG, $v = wa$, where $w \in \Sigma^*$ with $|w| = n$ and $a \in \Sigma$

$$
\begin{aligned}
\hat{\delta}(q, uwa) &= \delta(\hat{\delta}(q, uw), a) && \text{defn. of } \hat{\delta} \\
&= \delta(\hat{\delta}(\hat{\delta}(q, u), w), a) && \text{ind. hyp.} \\
&= \hat{\delta}(\hat{\delta}(q, u), wa) && \text{defn. of } \hat{\delta}
\end{aligned}
$$

□

Conventions in Inductive Proofs

Proposition

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$, and any strings $u, v \in \Sigma^*$ and state $q \in Q$, $\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v)$.

Proof.

"We will prove by induction on $|v|$" is a short-hand for

Conventions in Inductive Proofs

Proposition

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$*, and any strings* $u, v \in \Sigma^*$ *and state* $q \in Q$*,* $\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v)$*.*

Proof.

"We will prove by induction on $|v|$" is a short-hand for "We will prove the proposition by induction. Take S_i to be statement of the proposition restricted to strings v where $|v| = i$." $\square$

Properties of $\hat{\delta}$

Corollary

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$, and any string $v \in \Sigma^*$, $a \in \Sigma$ and state $q \in Q$, $\hat{\delta}(q, av) = \hat{\delta}(\delta(q, a), v)$.

Properties of $\hat{\delta}$

Corollary

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$*, and any string* $v \in \Sigma^*$*,* $a \in \Sigma$ *and state* $q \in Q$*,* $\hat{\delta}(q, av) = \hat{\delta}(\delta(q, a), v)$*.*

Proof.

From previous proposition we have, $\hat{\delta}(q, av) = \hat{\delta}(\hat{\delta}(q, a), v)$ (taking $u = a$).

$\square$

Properties of $\hat{\delta}$

Corollary

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$, and any string $v \in \Sigma^*$, $a \in \Sigma$ and state $q \in Q$, $\hat{\delta}(q, av) = \hat{\delta}(\delta(q, a), v)$.

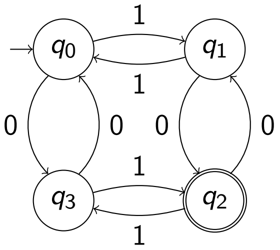
Proof.

From previous proposition we have, $\hat{\delta}(q, av) = \hat{\delta}(\hat{\delta}(q, a), v)$ (taking $u = a$). Next,

$$\begin{aligned} \hat{\delta}(q, a) &= \delta(\hat{\delta}(q, \epsilon), a) && \text{defn. of } \hat{\delta} \\ &= \delta(q, a) && \text{as } \hat{\delta}(q, \epsilon) = q \end{aligned}$$

□

Language of M_{odd}

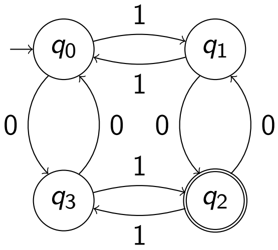


Transition Diagram of M_{odd}

Language of M_{odd}

Proposition

$L(M_{\mathrm{odd}}) = \{w \in \{0,1\}^* \mid w$ *has an odd number of 0s and an odd number of 1s*$\}$.



Transition Diagram of M_{odd}

Proof about the language of M_{odd}

Proof.

We will prove by induction on $|w|$ that $\hat{\delta}(q_0, w) \in F = \{q_2\}$ iff w has an odd number of 0s and an odd number of 1s.

Proof about the language of M_{odd}

Proof.

We will prove by induction on $|w|$ that $\hat{\delta}(q_0, w) \in F = \{q_2\}$ iff w has an odd number of 0s and an odd number of 1s.

- Base Case: When $w = \epsilon$, w has an even number of 0s and an even number of 1s and $\hat{\delta}(q_0, \epsilon) = q_0$ so the observation holds.

Proof about the language of M_{odd}

Proof.

We will prove by induction on $|w|$ that $\hat{\delta}(q_0, w) \in F = \{q_2\}$ iff w has an odd number of 0s and an odd number of 1s.

- Base Case: When $w = \epsilon$, w has an even number of 0s and an even number of 1s and $\hat{\delta}(q_0, \epsilon) = q_0$ so the observation holds.
- Induction Step $w = 0u$: The parity of the number of 1s in u and w is the same, and the parity of the number of 0s is opposite. And $\hat{\delta}(q_0, w) = \hat{\delta}(\delta(q_0, 0), u) = \hat{\delta}(q_3, u)$

Proof about the language of M_{odd}

It fails!

Proof.

We will prove by induction on $|w|$ that $\hat{\delta}(q_0, w) \in F = \{q_2\}$ iff w has an odd number of 0s and an odd number of 1s.

- Base Case: When $w = \epsilon$, w has an even number of 0s and an even number of 1s and $\hat{\delta}(q_0, \epsilon) = q_0$ so the observation holds.
- Induction Step $w = 0u$: The parity of the number of 1s in u and w is the same, and the parity of the number of 0s is opposite. And $\hat{\delta}(q_0, w) = \hat{\delta}(\delta(q_0, 0), u) = \hat{\delta}(q_3, u)$
- Need to know what strings are accepted from q_3! Need to prove a stronger statement. $\square$

Corrected Proof

Proof.

We need to a stronger statement that asserts what strings are accepted from each state of the DFA. We will prove by induction on $|w|$ that

(a) $\hat{\delta}(q_0, w) \in F$ iff w has odd number of 0s & odd number of 1s
(b) $\hat{\delta}(q_1, w) \in F$ iff
(c) $\hat{\delta}(q_2, w) \in F$ iff
(d) $\hat{\delta}(q_3, w) \in F$ iff

$\cdot\cdot\longrightarrow$

Corrected Proof

Proof.

We need to a stronger statement that asserts what strings are accepted from each state of the DFA. We will prove by induction on $|w|$ that

(a) $\hat{\delta}(q_0, w) \in F$ iff w has odd number of 0s & odd number of 1s

(b) $\hat{\delta}(q_1, w) \in F$ iff w has odd number of 0s & even number of 1s

(c) $\hat{\delta}(q_2, w) \in F$ iff

(d) $\hat{\delta}(q_3, w) \in F$ iff

Corrected Proof

Proof.

We need to a stronger statement that asserts what strings are accepted from each state of the DFA. We will prove by induction on $|w|$ that

(a) $\hat{\delta}(q_0, w) \in F$ iff w has odd number of 0s & odd number of 1s
(b) $\hat{\delta}(q_1, w) \in F$ iff w has odd number of 0s & even number of 1s
(c) $\hat{\delta}(q_2, w) \in F$ iff w has even number of 0s & even number of 1s
(d) $\hat{\delta}(q_3, w) \in F$ iff

Corrected Proof

Proof.

We need to a stronger statement that asserts what strings are accepted from each state of the DFA. We will prove by induction on $|w|$ that

(a) $\hat{\delta}(q_0, w) \in F$ iff w has odd number of 0s & odd number of 1s
(b) $\hat{\delta}(q_1, w) \in F$ iff w has odd number of 0s & even number of 1s
(c) $\hat{\delta}(q_2, w) \in F$ iff w has even number of 0s & even number of 1s
(d) $\hat{\delta}(q_3, w) \in F$ iff w has even number of 0s & odd number of 1s

$\cdot\cdot\longrightarrow$

Corrected Proof

Base Case

Proof (contd).

Consider w such that $|w| = 0$. Then $w = \epsilon$.

- w has even number of 0s and even number of 1s
- For any $q \in Q$, $\hat{\delta}(q, w) = q$
- Thus, $\hat{\delta}(q, w) \in F$ iff $q = q_3$, and statements (a),(b),(c), and (d) hold in the base case.

Corrected Proof

Induction Step: part (a)

Proof (contd).

Suppose (a),(b),(c), and (d) hold for strings w of length n.
Consider $w = au$, where $a \in \{0, 1\}$ and $u \in \Sigma^*$ of length n.

Corrected Proof

Induction Step: part (a)

Proof (contd).

Suppose (a),(b),(c), and (d) hold for strings w of length n. Consider $w = au$, where $a \in \{0, 1\}$ and $u \in \Sigma^*$ of length n. Recall that $\hat{\delta}(q, au) = \hat{\delta}(\delta(q, a), u)$.

Corrected Proof

Induction Step: part (a)

Proof (contd).

Suppose (a),(b),(c), and (d) hold for strings w of length n. Consider $w = au$, where $a \in \{0, 1\}$ and $u \in \Sigma^*$ of length n. Recall that $\hat{\delta}(q, au) = \hat{\delta}(\delta(q, a), u)$.

- Case $q = q_0$, $a = 0$: $\hat{\delta}(q_0, w) \in F$ iff

Corrected Proof

Induction Step: part (a)

Proof (contd).

Suppose (a),(b),(c), and (d) hold for strings w of length n. Consider $w = au$, where $a \in \{0, 1\}$ and $u \in \Sigma^*$ of length n. Recall that $\hat{\delta}(q, au) = \hat{\delta}(\delta(q, a), u)$.

- Case $q = q_0$, $a = 0$: $\hat{\delta}(q_0, w) \in F$ iff $\hat{\delta}(q_3, u) \in F$ iff

Corrected Proof

Induction Step: part (a)

Proof (contd).

Suppose (a),(b),(c), and (d) hold for strings w of length n. Consider $w = au$, where $a \in \{0, 1\}$ and $u \in \Sigma^*$ of length n. Recall that $\hat{\delta}(q, au) = \hat{\delta}(\delta(q, a), u)$.

- Case $q = q_0$, $a = 0$: $\hat{\delta}(q_0, w) \in F$ iff $\hat{\delta}(q_3, u) \in F$ iff u has even number of 0s and odd number of 1s (by ind. hyp. (d)) iff

Corrected Proof

Induction Step: part (a)

Proof (contd).

Suppose (a),(b),(c), and (d) hold for strings w of length n. Consider $w = au$, where $a \in \{0, 1\}$ and $u \in \Sigma^*$ of length n. Recall that $\hat{\delta}(q, au) = \hat{\delta}(\delta(q, a), u)$.

- Case $q = q_0$, $a = 0$: $\hat{\delta}(q_0, w) \in F$ iff $\hat{\delta}(q_3, u) \in F$ iff u has even number of 0s and odd number of 1s (by ind. hyp. (d)) iff w has odd number of 0s and odd number of 1s

Corrected Proof

Induction Step: part (a)

Proof (contd).

Suppose (a),(b),(c), and (d) hold for strings w of length n. Consider $w = au$, where $a \in \{0,1\}$ and $u \in \Sigma^*$ of length n. Recall that $\hat{\delta}(q, au) = \hat{\delta}(\delta(q, a), u)$.

- Case $q = q_0$, $a = 0$: $\hat{\delta}(q_0, w) \in F$ iff $\hat{\delta}(q_3, u) \in F$ iff u has even number of 0s and odd number of 1s (by ind. hyp. (d)) iff w has odd number of 0s and odd number of 1s
- Case $q = q_0$, $a = 1$: $\hat{\delta}(q_0, w) \in F$ iff

Corrected Proof

Induction Step: part (a)

Proof (contd).

Suppose (a),(b),(c), and (d) hold for strings w of length n. Consider $w = au$, where $a \in \{0, 1\}$ and $u \in \Sigma^*$ of length n. Recall that $\hat{\delta}(q, au) = \hat{\delta}(\delta(q, a), u)$.

- Case $q = q_0$, $a = 0$: $\hat{\delta}(q_0, w) \in F$ iff $\hat{\delta}(q_3, u) \in F$ iff u has even number of 0s and odd number of 1s (by ind. hyp. (d)) iff w has odd number of 0s and odd number of 1s
- Case $q = q_0$, $a = 1$: $\hat{\delta}(q_0, w) \in F$ iff $\hat{\delta}(q_1, u) \in F$ iff

Corrected Proof

Induction Step: part (a)

Proof (contd).

Suppose (a),(b),(c), and (d) hold for strings w of length n. Consider $w = au$, where $a \in \{0,1\}$ and $u \in \Sigma^*$ of length n. Recall that $\hat{\delta}(q, au) = \hat{\delta}(\delta(q,a), u)$.

- Case $q = q_0$, $a = 0$: $\hat{\delta}(q_0, w) \in F$ iff $\hat{\delta}(q_3, u) \in F$ iff u has even number of 0s and odd number of 1s (by ind. hyp. (d)) iff w has odd number of 0s and odd number of 1s
- Case $q = q_0$, $a = 1$: $\hat{\delta}(q_0, w) \in F$ iff $\hat{\delta}(q_1, u) \in F$ iff u has odd number of 0s and even number of 1s (by ind. hyp. (b)) iff

Corrected Proof

Induction Step: part (a)

Proof (contd).

Suppose (a),(b),(c), and (d) hold for strings w of length n. Consider $w = au$, where $a \in \{0,1\}$ and $u \in \Sigma^*$ of length n. Recall that $\hat{\delta}(q, au) = \hat{\delta}(\delta(q,a), u)$.

- Case $q = q_0$, $a = 0$: $\hat{\delta}(q_0, w) \in F$ iff $\hat{\delta}(q_3, u) \in F$ iff u has even number of 0s and odd number of 1s (by ind. hyp. (d)) iff w has odd number of 0s and odd number of 1s
- Case $q = q_0$, $a = 1$: $\hat{\delta}(q_0, w) \in F$ iff $\hat{\delta}(q_1, u) \in F$ iff u has odd number of 0s and even number of 1s (by ind. hyp. (b)) iff w has odd number of 0s and odd number of 1s

Corrected Proof

Induction Step: other parts

Proof (contd).

- Case $q = q_1$, $a = 0$: $\hat{\delta}(q_1, w) \in F$ iff $\hat{\delta}(q_2, u) \in F$ iff u has even number of 0s and even number of 1s (by ind. hyp. (c)) iff w has odd number of 0s and even number of 1s

Corrected Proof

Induction Step: other parts

Proof (contd).

- Case $q = q_1$, $a = 0$: $\hat{\delta}(q_1, w) \in F$ iff $\hat{\delta}(q_2, u) \in F$ iff u has even number of 0s and even number of 1s (by ind. hyp. (c)) iff w has odd number of 0s and even number of 1s
- …And so on for the other cases of $q = q_1$ and $a = 1$, $q = q_2$ and $a = 0$, $q = q_2$ and $a = 1$, $q = q_3$ and $a = 0$, and finally $q = q_3$ and $a = 1$. $\square$

Proving Correctness of a DFA

Proof Template

Given a DFA M having n states $\{q_0, q_1, \dots q_{n-1}\}$ with initial state q_0, and final states F, to prove that $L(M) = L$, we do the following.

1. Come up with languages $L_0, L_1, \dots L_{n-1}$ such that $L_0 = L$
2. Prove by induction on $|w|$, $\hat{\delta}(q_i, w) \in F$ if and only if $w \in L_i$

Typical Problem

Problem

Given a language L, design a DFA M that accepts L, i.e., $L(M) = L$.
How does one go about it?

Methodology

- Imagine yourself in the place of the machine, reading symbols of the input, and trying to determine if it should be accepted.
- Remember at any point you have only seen a part of the input, and you don't know when it ends.

Methodology

- Imagine yourself in the place of the machine, reading symbols of the input, and trying to determine if it should be accepted.
- Remember at any point you have only seen a part of the input, and you don't know when it ends.
- Figure out what to keep in memory. It cannot be all the symbols seen so far: it must fit into a finite number of bits.

Strings containing 0

Problem

Design an automaton that accepts all strings over $\{0, 1\}$ that contain at least one 0.

Solution

What do you need to remember?

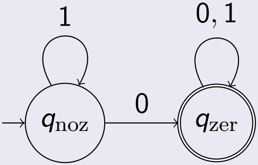
Strings containing 0

Problem

Design an automaton that accepts all strings over $\{0, 1\}$ that contain at least one 0.

Solution

What do you need to remember? Whether you have seen a 0 so far or not!



Automaton accepting strings with at least one 0.

Even length strings

Problem

Design an automaton that accepts all strings over $\{0, 1\}$ that have an even length.

Solution

What do you need to remember?

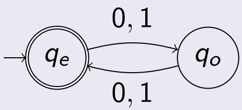
Even length strings

Problem

Design an automaton that accepts all strings over $\{0, 1\}$ that have an even length.

Solution

What do you need to remember? Whether you have seen an odd or an even number of symbols.



Automaton accepting strings of even length.

Pattern Recognition

Problem

Design an automaton that accepts all strings over $\{0, 1\}$ that have 001 as a substring, where u is a substring of w if there are w_1 and w_2 such that $w = w_1 u w_2$.

Solution

What do you need to remember?

Pattern Recognition

Problem

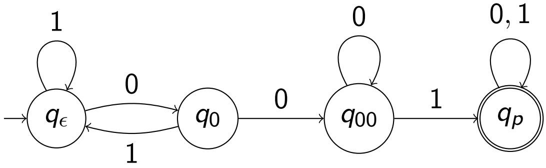
Design an automaton that accepts all strings over $\{0, 1\}$ that have 001 as a substring, where u is a substring of w if there are w_1 and w_2 such that $w = w_1 u w_2$.

Solution

What do you need to remember? Whether you

- haven't seen any symbols of the pattern
- have just seen 0
- have just seen 00
- have seen the entire pattern 001

Pattern Recognition Automaton



Automaton accepting strings having 001 as substring.

`grep` Problem

Problem

Given text T and string s, does s appear in T?

grep Problem

Problem

Given text T and string s, does s appear in T?

Solution

$$\overbrace{T_1\ T_2\ T_3\ \ldots\ T_n}^{=s?}\ T_{n+1}\ \ldots\ T_t$$

grep Problem

Problem

Given text T and string s, does s appear in T?

Naïve Solution

$$\overbrace{T_1\ T_2\ T_3\ \ldots\ T_n}^{=s?}\ T_{n+1}\ \ldots\ T_t$$

Running time $= O(nt)$, where $|T| = t$ and $|s| = n$.

grep Problem

Smarter Solution

Solution

- Build DFA M for $L = \{w \mid \text{there are } u, v \ s.t. \ w = usv\}$
- Run M on text T

grep Problem
Smarter Solution

Solution

- Build DFA M for $L = \{w \mid \text{there are } u, v \textit{ s.t. } w = usv\}$
- Run M on text T

Time = time to build $M + O(t)$!

grep Problem

Smarter Solution

Solution

- Build DFA M for $L = \{w \mid \text{there are } u, v \text{ s.t. } w = usv\}$
- Run M on text T

Time = time to build M + $O(t)$!

Questions

- Is L regular no matter what s is?
- If yes, can M be built "efficiently"?

grep Problem

Smarter Solution

Solution

- Build DFA M for $L = \{w \mid \text{there are } u, v \ s.t. \ w = usv\}$
- Run M on text T

Time = time to build M + $O(t)$!

Questions

- Is L regular no matter what s is?
- If yes, can M be built "efficiently"?

Knuth-Morris-Pratt (1977): Yes to both the above questions.

Multiples

Problem

Design an automaton that accepts all strings w over $\{0, 1\}$ such that w is the binary representation of a number that is a multiple of 5.

Solution

What must be remembered?

Multiples

Problem

Design an automaton that accepts all strings w over $\{0, 1\}$ such that w is the binary representation of a number that is a multiple of 5.

Solution

What must be remembered? The remainder when divided by 5.

Multiples

Problem

Design an automaton that accepts all strings w over $\{0, 1\}$ such that w is the binary representation of a number that is a multiple of 5.

Solution

What must be remembered? The remainder when divided by 5.
How do you compute remainders?

Multiples

Problem

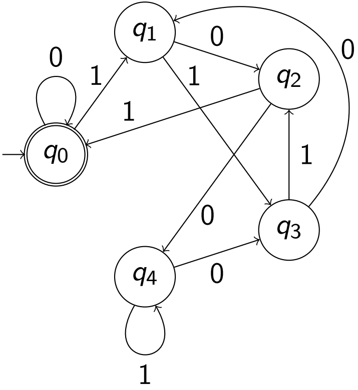
Design an automaton that accepts all strings w over $\{0, 1\}$ such that w is the binary representation of a number that is a multiple of 5.

Solution

What must be remembered? The remainder when divided by 5. How do you compute remainders?

- If w is the number n then $w0$ is $2n$ and $w1$ is $2n + 1$.
- $(a.b + c) \bmod 5 = (a.(b \bmod 5) + c) \bmod 5$
- *e.g.* $1011 = 11$ (decimal) $\equiv 1 \bmod 5$
  $10110 = 22$ (decimal) $\equiv 2 \bmod 5$
  $10111 = 23$ (decimal) $\equiv 3 \bmod 5$

Automaton for recognizing Multiples



Automaton recognizing binary numbers that are multiples of 5.

A One k-positions from end

Problem

Design an automaton for the language $L_k = \{w \mid k\text{th character from end of } w \text{ is } 1\}$

Solution

What do you need to remember?

A One k-positions from end

Problem

Design an automaton for the language $L_k = \{w \mid k\text{th character from end of } w \text{ is } 1\}$

Solution

What do you need to remember? The last k characters seen so far!
Formally, $M_k = (Q, \{0,1\}, \delta, q_0, F)$

- States $= Q = \{\langle w \rangle \mid w \in \{0,1\}^* \text{ and } |w| \leq k\}$
- $\delta(\langle w \rangle, b) = \begin{cases} \langle wb \rangle & \text{if } |w| < k \\ \langle w_2 w_3 \ldots w_k b \rangle & \text{if } w = w_1 w_2 \ldots w_k \end{cases}$
- $q_0 = \langle \epsilon \rangle$
- $F = \{\langle 1 w_2 w_3 \ldots w_k \rangle \mid w_i \in \{0,1\}\}$

Lower Bound on DFA size

Proposition

Any DFA recognizing L_k has at least 2^k states.

Proof.

Let M, with initial state q_0, recognize L_k and assume (for contradiction) that M has $< 2^k$ states.

- Number of strings of length $k = 2^k$
- There must be two distinct string w_0 and w_1 of length k such that $\hat{\delta}(q_0, w_0) = \hat{\delta}(q_0, w_1)$.

Proof (contd)

Proof (contd).

Let i be the first position where w_0 and w_1 differ. Without loss of generality assume that w_0 has 0 in the *ith* position and w_1 has 1.

$$\begin{aligned} w_0 0^{i-1} &= \ldots \overbrace{0 \ldots 0^{i-1}}^{k} \\ w_1 0^{i-1} &= \underbrace{\ldots}_{i-1} 1 \underbrace{\ldots}_{k-i} \end{aligned}$$

$w_0 0^{i-1} \notin L_k$ and $w_1 0^{i-1} \in L_k$. Thus, M cannot accept both $w_0 0^{i-1}$ and $w_1 0^{i-1}$.

Proof (contd)

Proof (contd).

Let i be the first position where w_0 and w_1 differ. Without loss of generality assume that w_0 has 0 in the ith position and w_1 has 1.

$$\begin{aligned} w_0 0^{i-1} &= \ \ldots\ \overbrace{0\ \ldots\ 0^{i-1}}^{k} \\ w_1 0^{i-1} &= \underbrace{\ldots}_{i-1} 1 \underbrace{\ldots}_{k-i} 0^{i-1} \end{aligned}$$

$w_0 0^{i-1} \notin L_k$ and $w_1 0^{i-1} \in L_k$. Thus, M cannot accept both $w_0 0^{i-1}$ and $w_1 0^{i-1}$.

Proof (contd)

... Almost there

Proof (contd).

So far, $w_0 0^{i-1} \notin L_n$, $w_1 0^{i-1} \in L_n$, and $\hat{\delta}(q_0, w_0) = \hat{\delta}(q_0, w_1)$.

$$\begin{aligned}
\hat{\delta}(q_0, w_0 0^{i-1}) &= \hat{\delta}(\hat{\delta}(q_0, w_0), 0^{i-1}) && \text{by Proposition proved} \\
&= \hat{\delta}(\hat{\delta}(q_0, w_1), 0^{i-1}) && \text{by assump. on } w_0 \text{ and } w_1 \\
&= \hat{\delta}(q_0, w_1 0^{i-1}) && \text{by Proposition proved}
\end{aligned}$$

Thus, M accepts or rejects both $w_0 0^{i-1}$ and $w_1 0^{i-1}$.
Contradiction! □

Complement DFAs

problem

Design an automaton for the language $L = \{w|w$ does not contain the substring $01\}$