DFA Product Construction

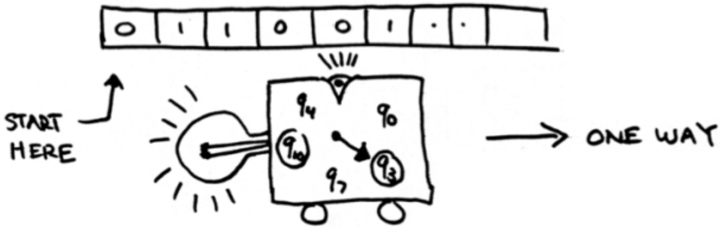
Lecture 4

Jan 30, 2020

Part I

DFA Recall

Machine View



- Machine with a fixed memory encoded in states.
- Start in specified start state
- Read input from a read-only tape: scan symbol, change state and move right
- Circled states are *accepting*
- Machine *accepts* input string if it is in an accepting state after scanning the last symbol.

Formal Tuple Notation

Definition

A deterministic finite automata (DFA) $M = (Q, \Sigma, \delta, s, A)$ is a five tuple where

Formal Tuple Notation

Definition

A deterministic finite automata (DFA) $M = (Q, \Sigma, \delta, s, A)$ is a five tuple where

- Q is a finite set whose elements are called states,

Common alternate notation: q_0 for start state, F for final states.

Formal Tuple Notation

Definition

A deterministic finite automata (DFA) $M = (Q, \Sigma, \delta, s, A)$ is a five tuple where

- Q is a finite set whose elements are called states,
- Σ is a finite set called the input alphabet,

Common alternate notation: q_0 for start state, F for final states.

Formal Tuple Notation

Definition

A deterministic finite automata (DFA) $M = (Q, \Sigma, \delta, s, A)$ is a five tuple where

- Q is a finite set whose elements are called states,
- Σ is a finite set called the input alphabet,
- $\delta : Q \times \Sigma \to Q$ is the transition function,

Common alternate notation: q_0 for start state, F for final states.

Formal Tuple Notation

Definition

A deterministic finite automata (DFA) $M = (Q, \Sigma, \delta, s, A)$ is a five tuple where

- Q is a finite set whose elements are called states,
- Σ is a finite set called the input alphabet,
- $\delta : Q \times \Sigma \rightarrow Q$ is the transition function,
- $s \in Q$ is the start state,

Common alternate notation: q_0 for start state, F for final states.

Formal Tuple Notation

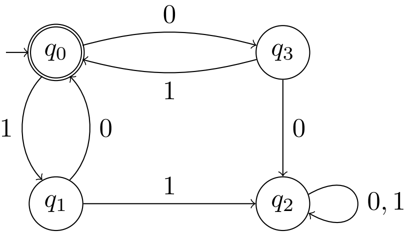
Definition

A deterministic finite automata (DFA) $M = (Q, \Sigma, \delta, s, A)$ is a five tuple where

- Q is a finite set whose elements are called states,
- Σ is a finite set called the input alphabet,
- $\delta : Q \times \Sigma \rightarrow Q$ is the transition function,
- $s \in Q$ is the start state,
- $A \subseteq Q$ is the set of accepting/final states.

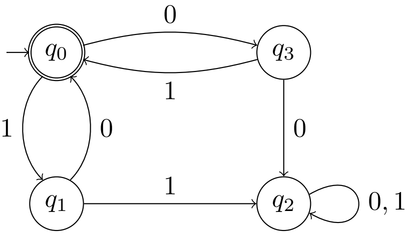
Common alternate notation: q_0 for start state, F for final states.

Example



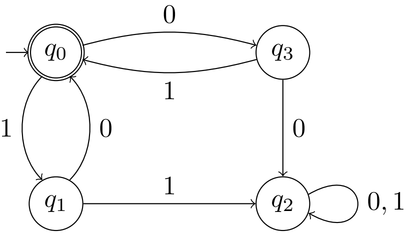
- $Q =$

Example



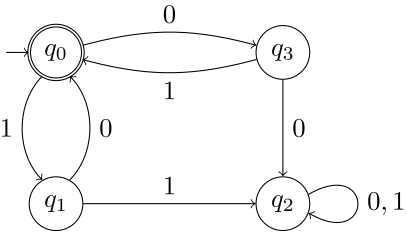
- $Q = \{q_0, q_1, q_1, q_3\}$

Example



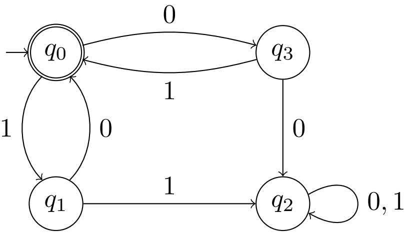
- $Q = \{q_0, q_1, q_1, q_3\}$
- $\Sigma =$

Example



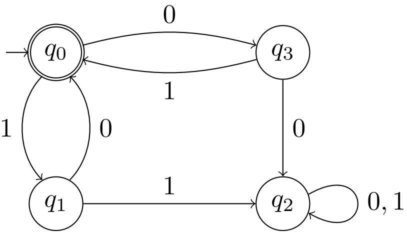
- $Q = \{q_0, q_1, q_1, q_3\}$
- $\Sigma = \{0, 1\}$

Example



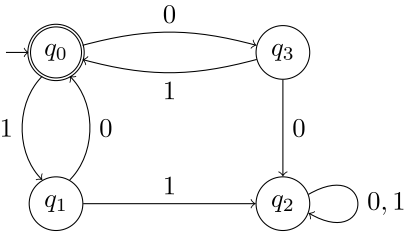
- $Q = \{q_0, q_1, q_1, q_3\}$
- $\Sigma = \{0, 1\}$
- δ

Example



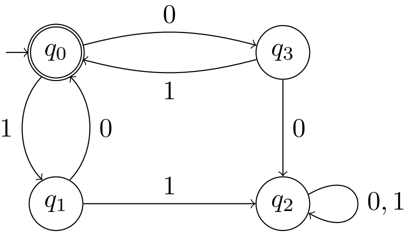
- $Q = \{q_0, q_1, q_1, q_3\}$
- $\Sigma = \{0, 1\}$
- δ
- $s =$

Example



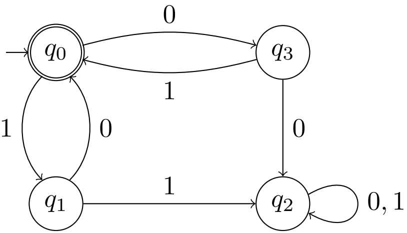
- $Q = \{q_0, q_1, q_1, q_3\}$
- $\Sigma = \{0, 1\}$
- δ
- $s = q_0$

Example



- $Q = \{q_0, q_1, q_1, q_3\}$
- $\Sigma = \{0, 1\}$
- δ
- $s = q_0$
- $A =$

Example



- $Q = \{q_0, q_1, q_1, q_3\}$
- $\Sigma = \{0, 1\}$
- δ
- $s = q_0$
- $A = \{q_0\}$

$(01 + 10)^*$

Extending the transition function to strings

Given DFA $M = (Q, \Sigma, \delta, s, A)$, $\delta(q, a)$ is the state that M goes to from q on reading letter a

Useful to have notation to specify the unique state that M will reach from q on reading *string* w

Extending the transition function to strings

Given DFA $M = (Q, \Sigma, \delta, s, A)$, $\delta(q, a)$ is the state that M goes to from q on reading letter a

Useful to have notation to specify the unique state that M will reach from q on reading *string* w

Transition function $\delta^* : Q \times \Sigma^* \to Q$ defined inductively as follows:

- $\delta^*(q, w) = q$ if $w = \epsilon$
- $\delta^*(q, w) = \delta^*(\delta(q, a), x)$ if $w = ax$.

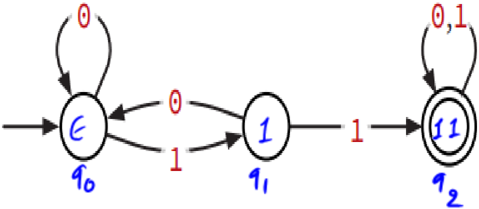
Formal definition of language accepted by M

Definition

The language $L(M)$ accepted by a DFA $M = (Q, \Sigma, \delta, s, A)$ is

$$\{w \in \Sigma^* \mid \delta^*(s, w) \in A\}.$$

Example



- What is $L(M)$?
- What is $L(M)$ if start state is changed to q_1?

Part II

Product Construction and Closure Properties

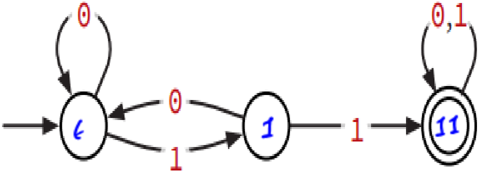
Part III

Complement

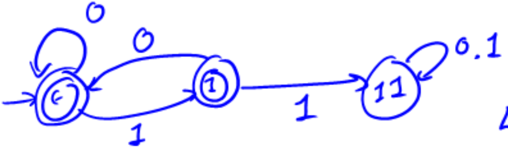
Complement

Question: If M is a DFA, is there a DFA M' such that $L(M') = \Sigma^* \setminus L(M)$? That is, are languages recognized by DFAs closed under complement?

$(0+1)^* 11 (0+1)^*$



All strings w/ "11" as a sub-string



Does not have "11" as a sub-string

Complement

Theorem

Languages accepted by DFAs are closed under complement.

Complement

Theorem

Languages accepted by DFAs are closed under complement.

Proof.

Let $M = (Q, \Sigma, \delta, s, A)$ such that $L = L(M)$. $\Sigma^* \setminus L$
Let $M' = (Q, \Sigma, \delta, s, Q \setminus A)$. Claim: $L(M') = \bar{L}$. Why?

Complement

Theorem

Languages accepted by DFAs are closed under complement.

Proof.

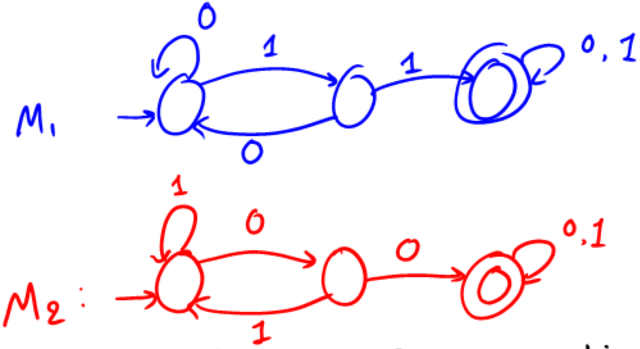
Let $M = (Q, \Sigma, \delta, s, A)$ such that $L = L(M)$.
Let $M' = (Q, \Sigma, \delta, s, Q \setminus A)$. Claim: $L(M') = \bar{L}$. Why?
$\delta^*_M = \delta^*_{M'}$. Thus, for every string w, $\delta^*_M(s, w) = \delta^*_{M'}(s, w)$.
$\delta^*_M(s, w) \in A \Rightarrow \delta^*_{M'}(s, w) \notin Q \setminus A$.
$\delta^*_M(s, w) \notin A \Rightarrow \delta^*_{M'}(s, w) \in Q \setminus A$. □

Part IV

Product Construction

Union and Intersection

Question: Are languages accepted by DFAs closed under union?
That is, given DFAs M_1 and M_2 is there a DFA that accepts $L(M_1) \cap L(M_2)$?



"11"

Has sub-string "00"

$L(M_1) \cap L(M_2)$: w/ substring "11" & "00"

Union and Intersection

Question: Are languages accepted by DFAs closed under union? That is, given DFAs M_1 and M_2 is there a DFA that accepts $L(M_1) \cap L(M_2)$?

Idea from programming: on input string w

- Simulate M_1 on w
- Simulate M_2 on w
- If both accept than $w \in L(M_1) \cap L(M_2)$.

Union and Intersection

Question: Are languages accepted by DFAs closed under union? That is, given DFAs M_1 and M_2 is there a DFA that accepts $L(M_1) \cap L(M_2)$?

Idea from programming: on input string w

- Simulate M_1 on w
- Simulate M_2 on w
- If both accept than $w \in L(M_1) \cap L(M_2)$.
- Catch: We want a single DFA M that can only read w once.

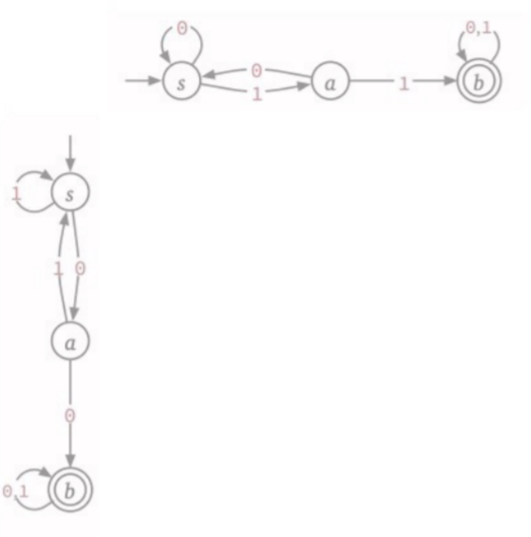
Union and Intersection

Question: Are languages accepted by DFAs closed under ~~union~~ intersection? That is, given DFAs M_1 and M_2 is there a DFA that accepts $L(M_1) \cap L(M_2)$?

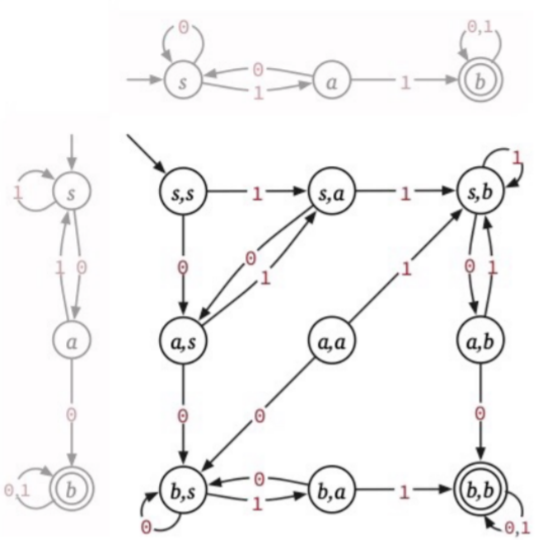
Idea from programming: on input string w

- Simulate M_1 on w
- Simulate M_2 on w
- If both accept than $w \in L(M_1) \cap L(M_2)$.
- Catch: We want a single DFA M that can only read w once.
- Solution: Simulate M_1 and M_2 in parallel by keeping track of states of *both* machines

Example



Example



Product construction for intersection

$M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2)$

Create $M = (Q, \Sigma, \delta, s, A)$ where

Product construction for intersection

$M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2)$

Create $M = (Q, \Sigma, \delta, s, A)$ where

- $Q =$

Product construction for intersection

$M_1 = (Q_1, \mathbf{\Sigma}, \delta_1, s_1, A_1)$ and $M_2 = (Q_1, \mathbf{\Sigma}, \delta_2, s_2, A_2)$

Create $M = (Q, \mathbf{\Sigma}, \delta, s, A)$ where

- $Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\}$

Product construction for intersection

$M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2)$

Create $M = (Q, \Sigma, \delta, s, A)$ where

- $Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\}$
- $s =$

Product construction for intersection

$M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2)$

Create $M = (Q, \Sigma, \delta, s, A)$ where

- $Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\}$
- $s = (s_1, s_2)$

Product construction for intersection

$M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2)$

Create $M = (Q, \Sigma, \delta, s, A)$ where

- $Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\}$
- $s = (s_1, s_2)$
- $\delta : Q \times \Sigma \to Q$ where

$$\delta((q_1, q_2), a) =$$

Product construction for intersection

$M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2)$

Create $M = (Q, \Sigma, \delta, s, A)$ where

- $Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\}$
- $s = (s_1, s_2)$
- $\delta : Q \times \Sigma \to Q$ where

$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$$

Product construction for intersection

$M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2)$

Create $M = (Q, \Sigma, \delta, s, A)$ where

- $Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\}$
- $s = (s_1, s_2)$
- $\delta : Q \times \Sigma \to Q$ where

$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$$

- $A =$

Product construction for intersection

$M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2)$

Create $M = (Q, \Sigma, \delta, s, A)$ where

- $Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\}$
- $s = (s_1, s_2)$
- $\delta : Q \times \Sigma \to Q$ where

$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$$

- $A = A_1 \times A_2 = \{(q_1, q_2) \mid q_1 \in A_1, q_2 \in A_2\}$

Product construction for intersection

$M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2)$

Create $M = (Q, \Sigma, \delta, s, A)$ where

- $Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\}$
- $s = (s_1, s_2)$
- $\delta : Q \times \Sigma \to Q$ where

$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$$

- $A = A_1 \times A_2 = \{(q_1, q_2) \mid q_1 \in A_1, q_2 \in A_2\}$

Theorem

$L(M) = L(M_1) \cap L(M_2)$.

Correctness of construction

Lemma

For each string w, $\delta^*((p, q), w) = (\delta_1^*(p, w), \delta_2^*(q, w))$.

Correctness of construction

Lemma

For each string w, $\delta^*((p,q),w) = (\delta_1^*(p,w), \delta_2^*(q,w))$.

Exercise: Assuming lemma prove the theorem in previous slide.

Correctness of construction

Lemma

For each string w, $\delta^*((p,q),w) = (\delta_1^*(p,w), \delta_2^*(q,w))$.

Exercise: Assuming lemma prove the theorem in previous slide.

Proof of lemma by induction on $|w|$

Proof: Base case : $w = \epsilon$

$\delta^*((p,q),\epsilon) = (p,q) = (\delta_1^*(p,\epsilon), \delta_2^*(q,\epsilon))$

$k = |w|$ Inductive H. : Lemma holds for all strings w of size $< k$.

$w = a.x$

$\delta^*((p,q),w) = \delta^*(\underline{\delta((p,q),a)}, x)$

$= \delta^*((p',q'), x)$, $p' = \delta_1(p,a)$, $q' = \delta_2(q,a)$

$|x| = k-1$ $\stackrel{I.H.}{=} (\delta_1^*(\delta_1(p,a),x), \delta_2(\delta_2(q,a),x))$

$= (\delta_1^*(p,w), \delta_2^*(q,w))$

Product construction for union

$M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2)$

Create $M = (Q, \Sigma, \delta, s, A)$ where

- $Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\}$
- $s = (s_1, s_2)$
- $\delta : Q \times \Sigma \to Q$ where

$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$$

- $A =$

Product construction for union

$M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2)$

Create $M = (Q, \Sigma, \delta, s, A)$ where

- $Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\}$
- $s = (s_1, s_2)$
- $\delta : Q \times \Sigma \to Q$ where

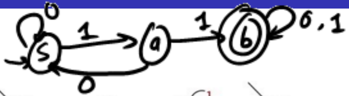
$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$$

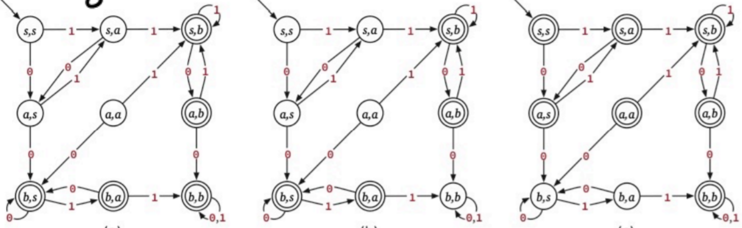
- $A = \{(q_1, q_2) \mid q_1 \in A_1 \text{ or } q_2 \in A_2\}$

Theorem

$L(M) = L(M_1) \cup L(M_2)$.

Example





(a) (b) (c)

$L(M_1) \cup L(M_2)$ $\qquad$ $L(M_1)$ XOR $L(M_2)$ $\qquad$ $L(M_1) \Rightarrow L(M_2)$

"00" => "11"

Set Difference

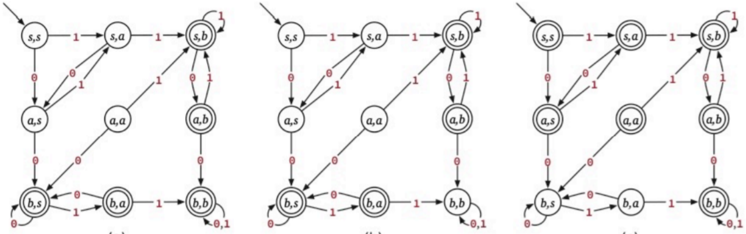
Theorem

M_1, M_2 DFAs. There is a DFA M such that
$L(M) = L(M_1) \setminus L(M_2)$. $= \overline{L(M_2)} \cap L(M_1)$

Exercise: Prove the above using two methods.

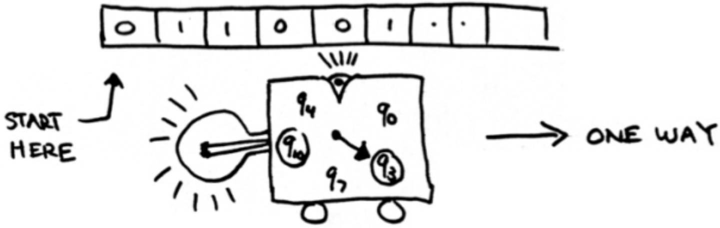
- Using a direct product construction
- Using closure under complement and intersection.

Any Boolean Function



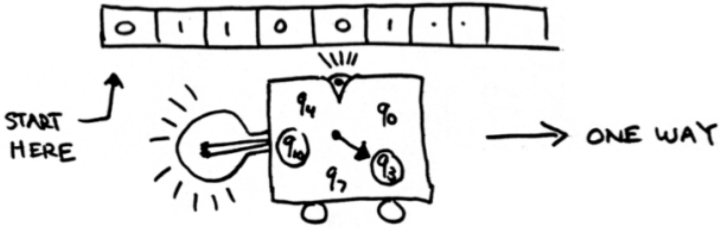
(a) (b) (c)

Things to know: 2-way DFA



Question: Why are DFAs required to only move right?
Can we allow DFA to scan back and forth? Caveat: Tape is read-only so only memory is in machine's state.

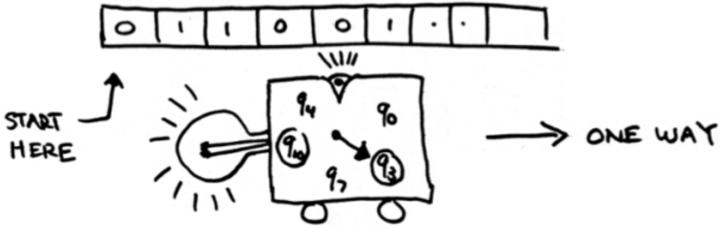
Things to know: 2-way DFA



Question: Why are DFAs required to only move right?
Can we allow DFA to scan back and forth? Caveat: Tape is read-only so only memory is in machine's state.

- Can define a formal notion of a "2-way" DFA
- Can show that any language recognized by a 2-way DFA can be recognized by a regular (1-way) DFA
- Proof is tricky simulation via NFAs

Things to know: 2-way DFA



Question: Why are DFAs required to only move right?
Can we allow DFA to scan back and forth? Caveat: Tape is read-only so only memory is in machine's state.

- Can define a formal notion of a "2-way" DFA
- Can show that any language recognized by a 2-way DFA can be recognized by a regular (1-way) DFA
- Proof is tricky simulation via NFAs

M_2: $(0+1)^*\ 1\ 1\ (0+1)^*$