Deterministic Finite Automata (DFAs)

Lecture 3
Tuesday, September 5, 2017

DFAs also called Finite State Machines (FSMs)
- The "simplest" model for computers?
- State machines that are common in practice.
  - Vending machines
  - Elevators
  - Digital watches
  - Simple network protocols
- Programs with fixed memory

A simple program
Program to check if a given input string $w$ has odd length

```
int n = 0
While input is not finished
  read next character c
  n ← n + 1
endWhile
If (n is odd) output YES
Else output NO
```

```
bit x = 0
While input is not finished
  read next character c
  x ← flip(x)
endWhile
If (x = 1) output YES
Else output NO
```
Definition 4. A deterministic finite automaton (DFA) is \( M = (Q, \Sigma, \delta, s, A) \) where
- \( Q \) is a finite set whose elements are the states
- \( \Sigma \) is the input alphabet
- \( s \) is the initial state
- \( A \) is the set of final states

5. What is the language recognized if we change the set of final states to be \( \{B\} \) (with initial state \( A \))?

Graphical Representation

- Directed graph with nodes representing states and edge/arc labels representing transitions labeled by symbols in \( \Sigma \)
- For each state (vertex) \( q \) and symbol \( a \in \Sigma \) there is exactly one outgoing edge labeled by \( a \)
- Initial/start state has a pointer (or labeled as \( s, q_0 \) or “start”)
- Some states with double circles labeled as accepting/final states

Another view

- Machine has input written on a read-only tape
- Start in specified start state
- Start at left, 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.

Graphical Representation/State Machine

Graphical Representation

- Where does \( 001 \) lead? \( 10010 \)?
- Which strings end up in accepting state?
- Can you prove it?
- Every string \( w \) has a unique walk that it follows from a given state \( q \) by reading one letter of \( w \) from left to right.

Definition

A DFA \( M \) accepts a string \( w \) iff the unique walk starting at the start state and spelling out \( w \) ends in an accepting state.

Definition

The language accepted (or recognized) by a DFA \( M \) is denoted by \( L(M) \) and defined as: \( L(M) = \{ w \mid M \text{ accepts } w \} \).
Warning

"M accepts language L" does not mean simply that that M accepts each string in L.

It means that M accepts each string in L and no others. Equivalently M accepts each string in L and does not accept/rejects strings in \( \Sigma^* \setminus L \).

M "recognizes" L is a better term but "accepts" is widely accepted (and recognized) (joke attributed to Lenny Pitt)

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,
- \( \Sigma \) is a finite set called the input alphabet,
- \( \delta : Q \times \Sigma \to Q \) 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.

DFA Notation

\[ M = \left( \hat{Q}, \Sigma, \delta, s, \hat{A} \right) \]

set of all states \( Q \)

transition func \( \delta \)

alphabet \( \Sigma \)

set of all accept states \( A \)

start state \( s \)

Example

\[ q_0 \]
0
\[ q_1 \]
1
0
\[ q_2 \]
1
0
\[ q_3 \]
0, 1

- \( Q = \{ q_0, q_1, q_1, q_3 \} \)
- \( \Sigma = \{ 0, 1 \} \)
- \( \delta \)
- \( s = q_0 \)
- \( A = \{ q_0 \} \)
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:
- $\delta^*(q, \epsilon) = q$ if $w = \epsilon$
- $\delta^*(q, ax) = \delta^*(\delta(q, a), x)$ 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

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

Example

![DFA Diagram]

What is:
- $\delta^*(q_1, \epsilon)$
- $\delta^*(q_0, 1011)$
- $\delta^*(q_1, 010)$
- $\delta^*(q_4, 10)$

Example continued

![DFA Diagram]

- What is $L(M)$ if start state is changed to $q_1$?
- What is $L(M)$ if final/accept states are set to $\{q_2, q_3\}$ instead of $\{q_0\}$?
Advantages of formal specification

- Necessary for proofs
- Necessary to specify abstractly for class of languages

**Exercise:** Prove by induction that for any two strings $u, v$, any state $q, \delta^*(q, uv) = \delta^*(\delta^*(q, u), v)$.

---

**Part II**

**Constructing DFAs**

---

**DFAs: State = Memory**

How do we design a DFA $M$ for a given language $L$? That is $L(M) = L$.

- DFA is a like a program that has fixed amount of memory independent of input size.
- The memory of a DFA is encoded in its states
- The state/memory must capture enough information from the input seen so far that it is sufficient for the suffix that is yet to be seen (note that DFA cannot go back)

---

**DFA Construction: Example**

Assume $\Sigma = \{0, 1\}$

- $L = \emptyset, L = \Sigma^*, L = \{\epsilon\}, L = \{0\}$.
- $L = \{w \in \{0, 1\}^* \mid |w| \text{ is divisible by } 5\}$
- $L = \{w \in \{0, 1\}^* \mid w \text{ ends with } 01\}$
- $L = \{w \in \{0, 1\}^* \mid w \text{ contains } 001 \text{ as substring}\}$
- $L = \{w \in \{0, 1\}^* \mid w \text{ contains } 001 \text{ or } 010 \text{ as substring}\}$
- $L = \{w \mid w \text{ has a } 1 \text{ } k \text{ positions from the end}\}$
DFA Construction: Example

$L = \{\text{Binary numbers congruent to } 0 \mod 5\}$

Example: $1101011 = 107 = 2 \mod 5$, $1010 = 10 = 0 \mod 5$

Key observation:

- $w_0 \mod 5 = a$ implies $w_0 \mod 5 = 2a \mod 5$
- $w_1 \mod 5 = (2a + 1) \mod 5$

Part III

Product Construction and Closure Properties

Part IV

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?
Complement Example...

Just flip the state of the states!

Part V
Product Construction

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: $L(M') = \overline{L}$. Why?
$\delta'_M = \delta_M$. Thus, for every string $w$, $\delta'_M(s, w) = \delta_M(s, w)$.
$\delta'_M(s, w) \in A \Rightarrow \delta'_M(s, w) \not\in Q \setminus A$.
$\delta'_M(s, w) \not\in A \Rightarrow \delta'_M(s, w) \in Q \setminus A$.

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 $L(M_1) \cup L(M_2)$?
How about intersection $L(M_1) \cap L(M_2)$?

Idea from programming: on input string $w$
- Simulate $M_1$ on $w$
- Simulate $M_2$ on $w$
- If both accept than $w \in L(M_1) \cap L(M_2)$. If at least one accepts then $w \in L(M_1) \cup L(M_2)$.
- **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

\[
M_2 \text{ accepts } #1 = \text{odd}
\]

\[
M_1 \text{ accepts } #0 = \text{odd}
\]

Example II

Accept all binary strings of length divisible by 3 and 5

Assume all edges are labeled by 0, 1.

Product construction for intersection

\[ M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1) \text{ and } M_2 = (Q_2, \Sigma, \delta_2, s_2, A_2) \]

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 \( w \), \( \delta^*(s, w) = (\delta_1^*(s_1, w), \delta_2^*(s_2, w)) \).

**Exercise:** Assuming lemma prove the theorem in previous slide. Proof of lemma by induction on \( |w| \)

---

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

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

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

---

Set Difference

**Theorem**

\( M_1, M_2 \) DFAs. There is a DFA \( M \) such that \( L(M) = L(M_1) \setminus L(M_2) \).

**Exercise:** Prove the above using two methods.

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

---

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