Nondeterministic Finite Automata

- 2017/08/29
- Chapter 1.2 in Sipser

- Announcement:
  - Piazza registration: http://piazza.com/asu/fall2017/cse355
  - First poll will be posted on Piazza soon
  - Slides for this lecture are here: http://www.public.asu.edu/~yzhan442/teaching/CSE355/Lectures/NFA.pdf
Last time

• Finite automata
  • Definition of FA
  • Computation of FA
  • Regular language
    • Regular language and FA
    • Regular operations
  • Design an FA
Limitations of FA discussed so far?

• Limitations of deterministic FA
  - A single sequence of steps – sequential computation

  What if we need to express parallelism
  - e.g., threads and processes
Nondeterministic Computation

• More than one way to perform computation at a step

\[(Q, \Sigma, \delta, q0, F)\]

• For FA, more than one state to transition to in a state!
The class of regular languages is closed under the union operation.

1. Given any RLs A1 and A2
2. Based on the definition, we can construct M1 for A1 and M2 for A2

\[ M_1 = (Q_1, \Sigma, \delta_1, q_0, F_1) \]
\[ M_2 = (Q_2, \Sigma, \delta_2, s_0, F_2) \]

1. Prove that A1 \( \cup \) A2 is also a RL
2. Construct a machine M to simulate both M1 and M2 at the same time and accept if either one accepts

- Keep a copy of both M1 and M2; for every step in M, run a step from M1 and then a step in M2; M accepts if either M1 or M2 accepts; otherwise, reject
Outline for today

• Nondeterministic finite automata
  o Definition of NFA
  o Computation of NFA
  o Equivalence of DFA and NFA
Definition of NFA

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

1. \(Q\) is a finite set of states,
2. \(\Sigma\) is a finite alphabet,
3. \(\delta : Q \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q)\) is the transition function,
4. \(q_0 \in Q\) is the start state, and
5. \(F \subseteq Q\) is the set of accept states.

\[\delta(q, a) = \{q_2, q_3\}\]

\[\delta(q, b) = \{q_1\}\]

\[\delta(q_1, \varepsilon) = \{q_2\}\]

\[\delta(q_2, \varepsilon) = \{q_3\}\]

\[\delta(q_3, \varepsilon) = \{q_1\}\]

\[\Sigma_\varepsilon = \{\varepsilon\} \cup \Sigma\]

\[\mathcal{P}(Q)\], or the power set of \(Q\): the set of all possible subsets of \(Q\).

<table>
<thead>
<tr>
<th>State</th>
<th>(a)</th>
<th>(b)</th>
<th>(\varepsilon)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(q_1)</td>
<td>{}</td>
<td>{q_2}</td>
<td>{q_3}</td>
</tr>
<tr>
<td>(q_2)</td>
<td>{q_2, q_3}</td>
<td>{q_3}</td>
<td>{}</td>
</tr>
<tr>
<td>(q_3)</td>
<td>{q_1}</td>
<td>{}</td>
<td>{}</td>
</tr>
</tbody>
</table>
Definition of NFA

- Notes about NFA
  - Multiple transitions out of a state on the same symbol
  - Transitions on the empty symbol are allowed
  - Not required to have a transition on every symbol

<table>
<thead>
<tr>
<th>State</th>
<th>Transition on a</th>
<th>Transition on b</th>
<th>Transition on ε</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1</td>
<td>{}</td>
<td>{q2}</td>
<td>{q3}</td>
</tr>
<tr>
<td>q2</td>
<td>{q2, q3}</td>
<td>{q3}</td>
<td>{}</td>
</tr>
<tr>
<td>q3</td>
<td>{q1}</td>
<td>{}</td>
<td>{}</td>
</tr>
</tbody>
</table>

\[ (Q, \delta, q_0, F) \]

\[ \text{nondeterminism} \]
Outline for today

• Nondeterministic finite automata
  o Definition of NFA
  o Computation of NFA
  o Equivalence of DFA and NFA
Computation of NFA

<table>
<thead>
<tr>
<th>State</th>
<th>a</th>
<th>b</th>
<th>$\varepsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1</td>
<td>$\emptyset$</td>
<td>{q2}</td>
<td>{q3}</td>
</tr>
<tr>
<td>q2</td>
<td>{q2, q3}</td>
<td>{q3}</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>q3</td>
<td>{q1}</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>
Computation of NFA

| a | b | a | b |

reject?
discard!
Computation of NFA

Ira A. Fulton
Schools of Engineering

Computation of NFA

discard!
Computation of NFA

- Transition from q2 to q1 on symbol a
- Transition from q1 to q3 on symbol b
- Transition from q1 to q2 on symbol a

Input sequence: abab
Computation of NFA

Discard!
Computation of NFA

The diagram shows a non-deterministic finite automaton (NFA) with states q1, q2, and q3. The input sequence is "aba". The transition labels are:
- From q2 to q1 on input 'a'
- From q1 to q3 on input 'b'
- From q3 to q1 on input 'a'

The diagram indicates that the path through the NFA does not lead to an accepting state, hence the output is "Reject".
Computation of NFA

- **All** computation stops in an non-accept state: **reject**
- A single computation stops in an accept state: **accept**

The formal definition of computation for an NFA is similar to that for a DFA. Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and $w$ a string over the alphabet $\Sigma$. Then we say that $N$ *accepts* $w$ if we can write $w = y_1y_2\cdots y_m$, where each $y_i$ is a member of $\Sigma_e$ and a sequence of states $r_0, r_1, \ldots, r_m$ exists in $Q$ with three conditions:

1. $r_0 = q_0$,
2. $r_{i+1} \in \delta(r_i, y_{i+1})$, for $i = 0, \ldots, m - 1$, and
3. $r_m \in F$.  

<table>
<thead>
<tr>
<th>State</th>
<th>a</th>
<th>b</th>
<th>$\varepsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1</td>
<td>{}</td>
<td>{q2}</td>
<td>{q3}</td>
</tr>
<tr>
<td>q2</td>
<td>{q2, q3}</td>
<td>{q3}</td>
<td>{}</td>
</tr>
<tr>
<td>q3</td>
<td>{q1}</td>
<td>{}</td>
<td>{}</td>
</tr>
</tbody>
</table>
Computation of NFA

Reading 'a':
- Move from q2 to q1.
- Discard.

Reading 'b':
- Move from q1 to q2.
- Discard.
- Move from q2 to q3.
- Discard & reject.

Input sequence: abab
What is the language of this NFA?

Reading ‘a’:
- q1
- q3
- Reading ‘b’:
- q2
- q3
- Read ‘b’:
- q3
- Discard & reject

The language of this NFA is the set of strings that start with ‘a’ and end with ‘b’.
NFA example

- Example: $L(M) = \{ w \mid w \in \{0, 1\}^* \text{ and } 001 \text{ is a suffix of } w \}$
  - Pretend that you are the machine reading the string
  - Identify how much memory you need or what are the states you need to remember?

- What are the states

- Is this correct now?
NFA example

- Example: $L(M) = \{ w | w \in \{0, 1\}^* \text{ and } 001 \text{ is a suffix of } w \}$
  
  ○ Pretend that you are the machine reading the string
  
  ○ Identify how much memory you need or what are the states you need to remember?

- What are the states
NFA example
Outline for today

• Nondeterministic finite automata
  • Definition of NFA
  • Computation of NFA
  • Equivalence of DFA and NFA
Equivalence of DFA and NFA

**Theorem 1.39**

Every nondeterministic finite automaton has an equivalent deterministic finite automaton.
Equivalence of DFA and NFA

**Theorem 1.39**

Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

\[ M = (Q', \Sigma, \delta', q'_0, F') \]

- What are the states for the DFA?
Equivalence of DFA and NFA

**Theorem 1.39**

Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

\[ M = (Q', \Sigma, \delta', q'_0, F') \]

- What are the states for the DFA?
- What is the transition function?
THEOREM 1.39

Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

$M = (Q', \Sigma, \delta', q'_0, F')$

• What are the states for the DFA?

• What is the transition function?

• What is the initial state?
Equivalence of DFA and NFA

**Theorem 1.39**

Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

\[ M = (Q', \Sigma, \delta', q'_0, F') \]

- What are the states for the DFA?
- What is the transition function?
- What is the initial state?
- What are the final states?
Equivalence of DFA and NFA

**Theorem 1.39**

Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

\[ M = (Q', \Sigma, \delta', q'_0, F') \]

- What are the states for the DFA?
- What is the transition function?
- What is the initial state?
- What are the final states?
Equivalence of DFA and NFA

**THEOREM 1.39**

Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

\[ M = (Q', \Sigma, \delta', q_0', F') \]

• What are the states for the DFA?
• What is the transition function?
• What is the initial state?
• What are the final states?

Remove unreachable states
Given an NFA $N = (Q, \Sigma, \delta, q_0, F)$ that recognizes $A$, we need to construct a DFA $M = (Q', \Sigma, \delta', q'_0, F')$ to recognize $A$:

- $Q' = \mathcal{P}(Q)$
- $\forall (R, a) \in Q' \times \Sigma \delta'(R, a) = E(\bigcup_{r \in R} \delta(r, a))$
- $q'_0 = E(\{q_0\})$
- $F' = \{R | R \in Q' \land \exists r (r \in R \land r \in F)\}$

$E(R) = \{q | q$ can be reached from a state in $R$ going alone zero or more $\varepsilon$ transitions$\}$
Equivalence of DFA and NFA

• Corollary: NFA and DFA are equivalent
  o Every NFA has an equivalent DFA (just proven)
  o Every DFA has an equivalent NFA
Equivalence of DFA and NFA

• Corollary: NFA and DFA are equivalent
  o Every NFA has an equivalent DFA (just proven)
  o Every DFA has an equivalent NFA

• Corollary: A language is regular if and only if some NFA recognizes it
Outline for today

• Nondeterministic finite automata
  o Definition of NFA
  o Computation of NFA
  o Equivalence of DFA and NFA

• Reading assignment for the next class:
  o Sipser Sec. 1.1, 1.2 (page 58 - 66, up to Equivalence with Finite Automata) – Quiz link will be sent out; due date is before the beginning of the next class