CS 301 - Lecture 3
NFA DFA Equivalence
Regular Expressions
Fall 2008

Review
• Languages and Grammars
  – Alphabets, strings, languages
• Regular Languages
  – Deterministic Finite Automata
  – Nondeterministic Finite Automata
• Today:
  – Equivalence of NFA and DFA
  – Regular Expressions
  – Equivalence to Regular Languages

Equivalence of Machines

Machine $M_1$ is equivalent to machine $M_2$
if $L(M_1) = L(M_2)$

Example of equivalent machines

NFA $M_1$
$L(M_1) = \{10\}^*$

DFA $M_2$
$L(M_2) = \{10\}^*$
Convert NFA to DFA

Begin with a set of start states
DFA start state = Union of NFA Start States

From the NFA start states, where could we get on a? List the set of all possible states you can reach using just an a.

How did we get from q0 to q2 using only a? (think λ)

From new set, where could we get on a?
Convert NFA to DFA

NFA

\[ M \]

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{\lambda} q_2 \]

DFA

\[ M' \]

\[ \{q_0\} \xrightarrow{a} \{q_1\} \xrightarrow{a} \{q_1, q_2\} \]

From new set, where could we get on b?

If a node has any NFA Final state, mark the node as Final in the DFA.

Could This Produce Infinite States?

If the NFA has states

\[ q_0, q_1, q_2, \ldots \]

There are a finite number of NFA states by definition

the DFA has states in the powerset

\[ \emptyset, \{q_0\}, \{q_1\}, \{q_1, q_2\}, \{q_3, q_4, q_7\}, \ldots \]

Powerset of finite set can be big, but it is not infinite!
Procedure NFA to DFA

1. Initial state of NFA: $q_0$
   
   Initial state of DFA: $\{q_0\}$

Procedure NFA to DFA

2. For every DFA’s state $\{q_i, q_j, \ldots, q_m\}$
   
   Compute in the NFA
   
   $\delta(q_i, a) = \{q_i, q_j, \ldots, q_m\}$
   
   $\delta(q_j, a) = \{q_i, q_j, \ldots, q_m\}$
   
   Add transition to DFA
   
   $\delta(q_i, q_j, \ldots, q_m, a) = \{q_i', q_j', \ldots, q_m'\}$

Procedure NFA to DFA

3. For any DFA state $\{q_i, q_j, \ldots, q_m\}$
   
   If some $q_j$ is a final state in the NFA
   
   Then, $\{q_i, q_j, \ldots, q_m\}$
   
   is a final state in the DFA
Theorem
Take NFA $M$
Apply procedure to obtain DFA $M'$
Then $M$ and $M'$ are equivalent:
$L(M) = L(M')$

Proof
$L(M) = L(M')$
$L(M) \subseteq L(M')$ AND $L(M) \supseteq L(M')$

First we show: $L(M) \subseteq L(M')$
Take arbitrary: $w \in L(M)$
We will prove: $w \in L(M')$

$w \in L(M)$
$M : \rightarrow q_0 \xrightarrow{\alpha_1} q_2 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_i} q_f$
$w = \sigma_1 \sigma_2 \cdots \sigma_k$
$M : \rightarrow q_0 \xrightarrow{\alpha_1} q_2 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_i} q_f$
We will show that if \( w \in L(M) \)

\[
M : \quad \begin{array}{c}
\xrightarrow{a_1} \quad \xrightarrow{a_2} \quad \cdots \quad \xrightarrow{a_k} \end{array}
\]

\( w = \sigma_1 \sigma_2 \cdots \sigma_k \)

\[
M' : \quad \begin{array}{c}
\xrightarrow{a_1} \quad \xrightarrow{a_2} \quad \cdots \quad \xrightarrow{a_k} \end{array}
\]

\( w \in L(M') \)

More generally, we will show that if in \( M \):

(an arbitrary string) \( v = a_1 a_2 \cdots a_n \)

\[
M : \quad \begin{array}{c}
\xrightarrow{a_1} \quad \xrightarrow{a_2} \quad \cdots \quad \xrightarrow{a_n} \end{array}
\]

\( M' : \quad \begin{array}{c}
\xrightarrow{a_1} \quad \xrightarrow{a_2} \quad \cdots \quad \xrightarrow{a_n} \end{array}
\]

Proof by induction on \( |v| \)

Induction Basis: \( v = a_1 \)

\[
M : \quad \begin{array}{c}
\xrightarrow{a_1} \quad \end{array}
\]

\[
M' : \quad \begin{array}{c}
\xrightarrow{a_1} \quad \end{array}
\]

Induction hypothesis: \( 1 \leq |v| \leq k \)

\( v = a_1 a_2 \cdots a_k \)

\[
M : \quad \begin{array}{c}
\xrightarrow{a_1} \quad \xrightarrow{a_2} \quad \cdots \quad \xrightarrow{a_k} \end{array}
\]

\[
M' : \quad \begin{array}{c}
\xrightarrow{a_1} \quad \xrightarrow{a_2} \quad \cdots \quad \xrightarrow{a_k} \end{array}
\]
Induction Step: \(|v| = k + 1\)

\[ v = a_1a_2 \cdots a_k a_{k+1} = v'a_{k+1} \]

\[ M : q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_k} q_d \]

\[ M' : q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_{k+1}} q_d \]

Therefore if \(w \in L(M)\)

\[ w = \sigma_1\sigma_2 \cdots \sigma_k \]

\[ M : q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_k} \]

\[ M' : q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} \cdots \]

We have shown: \(L(M) \subseteq L(M')\)

We also need to show: \(L(M) \supseteq L(M')\)

(proof is similar)
NFAs Accept Regular Languages

We will prove:
\[
\left\{ \text{Languages accepted by NFAs} \right\} = \left\{ \text{Regular Languages} \right\}
\]

NFAs and DFAs have the same computation power

---

Step 1
\[
\left\{ \text{Languages accepted by NFAs} \right\} \subseteq \left\{ \text{Regular Languages} \right\}
\]

Proof: Every DFA is trivially an NFA

Any language \( L \) accepted by a DFA is also accepted by an NFA

---

Step 2
\[
\left\{ \text{Languages accepted by NFAs} \right\} \subseteq \left\{ \text{Regular Languages} \right\}
\]

Proof: Any NFA can be converted to an equivalent DFA

Any language \( L \) accepted by an NFA is also accepted by a DFA
Regular Expressions

- Regular expressions describe regular languages
- Example: 
  \[(a + b \cdot c)^*\]
  describes the language 
  \[\{a, bc\}^* = \{\lambda, a, bc, aa, abc, bca, \ldots\}\]

Recursive Definition

Primitive regular expressions: \(\emptyset, \lambda, \alpha\)
Given regular expressions \(\eta_1\) and \(\eta_2\)

\[
\begin{align*}
\eta_1 + \eta_2 \\
\eta_1 \cdot \eta_2 \\
\eta_1^* \\
(\eta_1)
\end{align*}
\]

Are regular expressions

Examples

A regular expression:

\[(a + b \cdot c)^* \cdot (c + \emptyset)\]

Not a regular expression: 

\[(a + b +)\]

Languages of Regular Expressions

- \(L(r)\): language of regular expression \(r\)
- Example
  \[L((a + b \cdot c)^*) = \{\lambda, a, bc, aa, abc, bca, \ldots\}\]
Formal Definition

- For primitive regular expressions:
  \[ L(\emptyset) = \emptyset \]
  \[ L(\lambda) = \{ \lambda \} \]
  \[ L(a) = \{ a \} \]

Definition (continued)

- For regular expressions \( r_1 \) and \( r_2 \)
  \[ L(r_1 + r_2) = L(r_1) \cup L(r_2) \]
  \[ L(r_1 \cdot r_2) = L(r_1) L(r_2) \]
  \[ L(r_1^*) = (L(r_1))^* \]
  \[ L((r_1)) = L(r_1) \]

Example

\[ L((a + b) \cdot a^*) = L((a + b)) L(a^*) \]
\[ = L(a + b) L(a^*) \]
\[ = (L(a) \cup L(b))(L(a))^* \]
\[ = \{a\} \cup \{b\} \{\{a\}\}^* \]
\[ = \{a, b\} \{\lambda, a, aa, aaa, ...\} \]
\[ = \{a, aa, aaa, ..., b, ba, baa, ...\} \]

Example

- Regular expression
  \[ r = (a + b)^* (a + bb) \]
  \[ L(r) = \{a, bb, aa, abb, ba, bbb, ...\} \]
What’s Next

• Read
  – Linz Chapter 1, 2.1, 2.2, 2.3, (skip 2.4), Chapter 3
  – JFLAP Startup, Chapter 1, 2.1, (skip 2.2) 3. 4
• Next Lecture Topics from Chapter 3.2 and 3.3
  – Regular Expressions and Regular Languages
  – Regular Grammars and Regular Languages
• Quiz 1 in Recitation on Wednesday 9/17
  – Covers Linz 1.1, 1.2, 2.1, 2.2, 2.3, and JFLAP 1, 2.1
  – Closed book, but you may bring one sheet of 8.5 x 11 inch paper
    with any notes you like.
  – Quiz will take the full hour
• Homework
  – Homework 2 is Due Thursday