Deterministic Finite Automata

H. Geuvers and A. Kissinger

Institute for Computing and Information Sciences
Radboud University Nijmegen

Version: fall 2015
Outline

Finite Automata

Manipulating finite automata

Finite automata and regular languages
Regular languages

Definition

$L \subseteq A^*$ is regular if $L = \mathcal{L}(e)$ for some regular expression $e$.

Decidability Problems

Q1 Can we decide (algorithmically) if $w \in \mathcal{L}(e)$?
Q2 If yes, what is the complexity of this decision procedure?
Q3 Can we decide if $\mathcal{L}(e_1) = \mathcal{L}(e_2)$?

Some answers

A1 Yes, we can give a deterministic finite automaton for $e$ to decide $w \in \mathcal{L}(e)$. (This lecture)
A2 time is linear in $|w|$; memory is constant (independent of $w$).
A3 Yes: $\mathcal{L}(e_1) = \mathcal{L}(e_2)$ can be axiomatised, see Exercise 2.3.6 of the course notes. It is decidable whether $e_1 = e_2$ according to these axioms.
Intuition. Let $A = \{a, b\}$. Consider the DFA $M$:

Letters $a, b$ are the moves in the graph

$A w \in A^*$ is a sequence of moves

Start state is indicated by ‘start→’,
Accepting states by double circle
(there can be several accepting states)

The word $abba$ is accepted, but $baab$ is not accepted (rejected)

$M = (Q, q_0, \delta, F)$ with $Q = \{q_0, q_1, q_2\}$, $F = \{q_2\}$ and $\delta$ given by

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$q_0$</td>
<td>$q_1$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>$q_0$</td>
<td>$q_2$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$q_2$</td>
<td>$q_2$</td>
</tr>
</tbody>
</table>
Deterministic Finite Automata formally

\( M \) is a DFA over \( A \) if \( M = (Q, q_0, \delta, F) \) with
- \( A \) is a finite alphabet
- \( Q \) is a finite set of states
- \( q_0 \in Q \) is the initial state
- \( F \subseteq Q \) is a finite set of final states
- \( \delta : Q \times A \to Q \) is the transition function

The multi-step transition \( \delta^* : Q \times A^* \to Q \) is defined inductively by

\[
\begin{align*}
\delta^*(q, \lambda) & = q \\
\delta^*(q, aw) & = \delta^*(\delta(q, a), w)
\end{align*}
\]

The language accepted at state \( q \), notation \( L(q) \), is:

\[
L(q) = \{ w \in A^* \mid \delta^*(q, w) \in F \}
\]

The language accepted by \( M \), notation \( L(M) \), is \( L(q_0) \).
Reading words $w \in A^*$

Computation for $\delta^*(q_0, w)$ in the example DFA. Take $w = abba$:

$\begin{align*}
[q_0, abba] & \vdash [\delta(q_0, a), bba] = [q_0, bba] \\
& \vdash [\delta(q_0, b), ba] = [q_1, ba] \\
& \vdash [\delta(q_1, b), a] = [q_2, a] \\
& \vdash [\delta(q_2, a), \lambda] = [q_2, \lambda]
\end{align*}$

$\begin{align*}
[q_0, aba] & \vdash [\delta(q_0, a), ba] = [q_0, ba] \\
& \vdash [\delta(q_0, b), a] = [q_1, a] \\
& \vdash [\delta(q_1, a), \lambda] = [q_0, \lambda]
\end{align*}$

So $abba$ is accepted and $aba$ is not accepted.

The language accepted by $M$ (of the first slide) is regular. It is the language

$$\mathcal{L}((a + b)^* bb(a + b)^*).$$
Consider the automaton $M$ over $A = \{a, b\}$ with

- $Q = \{0, 1, 2, 3, 4\}$,
- $q_0 = 0$,
- $F = \{4\}$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

1. Which of the following words is accepted? $abba$, $baba$, $bba$
2. Is it the case that $\{w \mid |w|_b \text{ is even} \} \subseteq \mathcal{L}(M)$?
3. Is it the case that $\{w \mid w \text{ contains } aabbaa \} \subseteq \mathcal{L}(M)$?
Manipulating Finite Automata: products for intersection

\[ M \]

\[ L(M) \]

\[ L_1 = \{ w \mid |w|_a \text{ is even} \} \]

\[ L_2 = \{ w \mid |w|_b \geq 1 \} \]

\[ L_1 \cap L_2 = \{ w \mid |w|_a \text{ is even and } |w|_b \geq 1 \} \]
Product of two DFAs

Given two DFAs over the same $A$

\[ M_1 = (Q_1, q_{01}, \delta_1, F_1) \]
\[ M_2 = (Q_2, q_{02}, \delta_2, F_2) \]

Define

\[ M_1 \times M_2 = (Q_1 \times Q_2, q_0, \delta, F) \]

with

\[ q_0 := (q_{01}, q_{02}) \]
\[ \delta(((q_1, q_2), a)) := (\delta_1(q_1, a), \delta_2(q_2, a)) \]

Then with

\[ F := F_1 \times F_2 := \{(q_1, q_2) \mid q_1 \in F_1 \text{ and } q_2 \in F_2\} \]

we have

\[ \mathcal{L}(M_1 \times M_2) = \mathcal{L}(M_1) \cap \mathcal{L}(M_2) \]
Closure Properties

Proposition Closure under complement
If $L$ is accepted by some DFA, then so is

$$\overline{L} = A^* - L.$$  

Proof. Suppose that $L$ is accepted by $M = (Q, q_0, \delta, F)$. Then $\overline{L}$ is accepted by $M = (Q, q_0, \delta, F')$.

Proposition Closure under intersection and union
If $L_1$ and $L_2$ are accepted by some DFA, then so are $L_1 \cap L_2$ and $L_1 \cup L_2$.

Proof. For the intersection, this follows from the product construction on the previous slide.
For the union, this can be seen by the product construction, taking a different $F$ (which one?) or by noticing that $L_1 \cup L_2 = \overline{\overline{L_1} \cap \overline{L_2}}$. 
Kleene’s Theorem

**Theorem** The languages accepted by DFAs are exactly the **regular languages**

We will prove this in this and the next lecture by

1. If \( L = \mathcal{L}(M) \), for some DFA \( M \), then there is a regular expression \( e \) such that \( L = \mathcal{L}(e) \) (this lecture).
2. If \( L = \mathcal{L}(e) \), for some regular expression \( e \), then there is a non-deterministic finite automaton (NFA) \( M \) such that \( L = \mathcal{L}(M) \). (next lecture).
3. For every NFA \( M \), there is a DFA \( M' \) such that \( \mathcal{L}(M) = \mathcal{L}(M') \) (next lecture).
From DFAs to regular expressions

Given the DFA $M = (Q, q_0, \delta, F)$, we construct a regular expression $e$ such that

$$\mathcal{L}(e) = \mathcal{L}(M).$$

Procedure:

- We remove states, replacing symbols from $A$ by regular expressions over $A$,
- until we end up with a “simple automaton” from which we can read off $e$. 
### Simple automata

<table>
<thead>
<tr>
<th>$M$</th>
<th>$e$ such that $\mathcal{L}(e) = \mathcal{L}(M)$</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="State 1" /></td>
<td>$w^*$</td>
</tr>
<tr>
<td><img src="#" alt="State 2" /></td>
<td>$0$</td>
</tr>
<tr>
<td><img src="#" alt="State 3" /></td>
<td>$(u + xv^<em>y)^</em>$</td>
</tr>
<tr>
<td><img src="#" alt="State 4" /></td>
<td>$u^*x(v + yu^<em>x)^</em>$</td>
</tr>
</tbody>
</table>
Eliminating states

- Remove a state $p$,
- while adding arrows $q \xrightarrow{w} q'$ between other pairs of states.

Before:

\[ q \xrightarrow{x} p \xrightarrow{v} \rightarrow y \rightarrow q' \]

After:

\[ q \xrightarrow{xv^*y} q' \]
Special cases

Join arrows using $+$

\[
\begin{array}{c}
q \xrightarrow{x} y \xrightarrow{} q' \\
q' \xrightarrow{} x + y \\
q \xrightarrow{x + y} q' \\
q \xrightarrow{} q'
\end{array}
\]

First create a new single final state

\[
\begin{array}{c}
q \xrightarrow{} q' \\
p \xrightarrow{} q, q' \\
q \xrightarrow{} q'
\end{array}
\]

Beware of loops!

\[
\begin{array}{c}
q \xrightarrow{x} p \xrightarrow{} q \\
q \xrightarrow{} p \xrightarrow{} q' \\
q \xrightarrow{} xv^*y \\
q \xrightarrow{} p
\end{array}
\]
The algorithm outlined in the previous slides produces for every deterministic finite automaton $M$, a regular expression $e_M$.

**Theorem** For $M$ a DFA we have

$$L(e_M) = L(M).$$