Finite-State Machines (Automata) lecture 12

• a simple form of computation
• used widely
• one way to find patterns
current
A B C D
next
B C D A
Application Fields

Industry
- real-time control, vending machines, cash dispensers, etc.

Electronic circuits
- data path / control path
- memory / cache handling
- protocols, USB, etc.

Communication protocols
- initiation and maintenance of communication links
- error detection and handling, packet retransmission

Language analysis
- natural languages
- programming languages
- search engines
Finite State Machines

• A conceptual tool for modelling reactive systems.
• Not limited to software systems.
• Used to specify required system behaviour in a precise way.
• Then implement as software/hardware (and perhaps verify behaviour against FSM).

• Finite state machines can also be viewed as rules for generating sets of strings ( = sets of finite sequences of symbols)
A Decimal Number

Each path from start to end, with a choice of one of the labels for each edge traversal, gives a decimal number.

\[ d = \{"0","1","2","3","4","5","6","7","8","9"\} \]

\[ +|- = \{"+","-"\} \quad \varepsilon = \"\" \quad \cdot = "." \]
This gives us a new way of defining sets by rules.

A whole number is a digit followed by any number of digits, or a sign followed by a digit followed by any number of digits.
This gives us a new way of defining sets by rules.

A whole number is a digit followed by any number of digits, or a sign followed by a digit followed by any number of digits.

A number is a whole number or a whole number followed by a decimal point followed by a digit followed by any number of digits.
finite state machines

A labelled directed graph

nodes

edges

with

a set of start nodes

and

a set of end nodes

Each edge is labelled with

a set of strings
finite state machines

A labelled directed graph

- nodes
- edges

with

- a set of start nodes
- a set of end nodes

Each edge is labelled with

- a set of strings

Each path from a start node to an end node, with a choice of one of the labels for each edge traversal, gives, by concatenation, a string.

The language defined by the machine is the set of all such strings.
The language defined by this machine is the set of all decimal numerals.
finite state machines

What happens when we link two machines together?
sequence

RS

RS = \{ r ++ s \mid r \in R \land s \in S \}
alternation \quad R \mid S

R \mid S = R \cup S
iteration $R^*$

""" $\in R^*$

$x \in R^*, r \in R \Rightarrow x ++ r \in R^*$
finite state spaghetti

A language is a set of finite sequences of words.

A language is a set of finite sequences of symbols.

A language is **regular** iff it is the language recognised by some Finite State Machine.
each transition is labelled with a finite sequence of tokens, the tokens, depend on the application, they may be letters, symbols, words, actions, ...
Normally we only use sequences of length 0 or 1 as labels. Then we write $a$ for $<a>$ and $\varepsilon$ for $\langle\rangle$.

We generate strings of tokens by tracing a path from a start state to an accepting state.

Which strings are accepted by this automaton?
Normally we only use sequences of length 0 or 1 as labels. Then we write $a$ for $\langle a \rangle$ and $\varepsilon$ for $\langle \rangle$.

Strings that
begin with $b$ and are of even length or
begin with $a$ and are of odd length
This machine is non-deterministic
some strings have more than one trace
some strings have no traces

Which of these strings are accepted?
abba abab baba aba bab ab ba a b
This machine is non-deterministic.
Some strings have more than one trace.
Some strings have no traces.

Which of these strings are accepted?
abba abab **baba aba** bab ab ba a b
This machine is non-deterministic
some strings have more than one trace
some strings have no traces

Which of these strings are accepted?
abba abab baba aba bab ab ba a b

\[
a \in L \\
ba \in L \\
x \in L \rightarrow x ba \in L
\]
KISS – DFA

Deterministic Finite Automaton

Exactly one start state, and from each state, \( q \), for each token, \( t \), there is exactly one transition from \( s \) with label \( t \).
This machine has four states,\
\[ Q = \{ s0, s1, s2, s3 \} \]
Its alphabet has three symbols\
\[ \Sigma = \{ 0, 1, 2 \} \]
For each\n\[ (\text{state}, \text{symbol}) \text{ pair, } (q,s) \]\nthere is exactly one transition\nfrom \( q \) with label \( s \)\n\[
\begin{array}{c}
q \xrightarrow{S} q' \\
\end{array}
\]
This means that\n\( q' \) is determined by \( (q,s) \)
For each (state, symbol) pair, \((q,s)\) there is exactly one transition from \(q\) with label \(s\)

\[ q \xrightarrow{s} q' \]

This means that \(q'\) is determined by \((q,s)\)

It has only one start state, this means that any input string determines a sequence of states.

This is a deterministic finite-state automaton (DFA)
For each (state, symbol) pair, \((q, s)\), there is exactly one transition from \(q\) with label \(s\).

\[ q' \text{ is determined by } (q, s) \]

It has only one start state.

Every DFA has a next-state function:

\[ q' = F(q, s) \]
DFA C NFA

A DFA has only one start state
For each (state, symbol) pair, \((q,s)\)
there is exactly one transition from \(q\) with label \(s\)
\(q'\) is determined by \(q\) and \(s\)

Every DFA has a next-state function
\(q' = F(q,s)\)
any input string determines a sequence of states.

We will focus now on DFA and return later to NFA.
Then we will show that for any NFA there isa a DFA that defines the same language.
Every DFA has a next-state function:

\[ q' = F(q,s) \]

Since we have finitely many symbols and states, this can be given by a table:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>s0</strong></td>
<td>s0</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>s1</strong></td>
<td></td>
<td>s2</td>
<td></td>
</tr>
<tr>
<td><strong>s2</strong></td>
<td></td>
<td></td>
<td>s3</td>
</tr>
<tr>
<td><strong>s3</strong></td>
<td>s1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This is a deterministic finite-state automaton, DFA.
Every DFA has a next-state function

\[ q' = F(q, s) \]

Since we have finitely many symbols and states
This can be given by a next-state table

<table>
<thead>
<tr>
<th>q \ s</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>s0</td>
<td>s0</td>
<td>s1</td>
<td>s2</td>
</tr>
<tr>
<td>s1</td>
<td>s3</td>
<td>s0</td>
<td>s1</td>
</tr>
<tr>
<td>s2</td>
<td>s2</td>
<td>s3</td>
<td>s0</td>
</tr>
<tr>
<td>s3</td>
<td>s1</td>
<td>s2</td>
<td>s3</td>
</tr>
</tbody>
</table>

This is a deterministic finite-state automaton, DFA
Every DFA has a next-state function

\[ q' = F(q,s) \]

We can also represent the transitions by a relation. Which symbol(s) take us from state \( q \) to state \( r \)?

<table>
<thead>
<tr>
<th>( q' )</th>
<th>( r )</th>
</tr>
</thead>
<tbody>
<tr>
<td>s0</td>
<td>s0</td>
</tr>
<tr>
<td>s1</td>
<td>s1</td>
</tr>
<tr>
<td>s2</td>
<td>s2</td>
</tr>
<tr>
<td>s3</td>
<td>s3</td>
</tr>
</tbody>
</table>

This is a deterministic finite-state automaton, DFA.
Every DFA has a next-state function

\[ q' = F(q, s) \]

\[ \begin{array}{c}
q \\
\downarrow \quad S \\
q'
\end{array} \]

We can also represent the transitions by a relation. Which symbol(s) take us from state \( q \) to state \( r \)?

<table>
<thead>
<tr>
<th>( q )</th>
<th>( s_0 )</th>
<th>( s_1 )</th>
<th>( s_2 )</th>
<th>( s_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_0 )</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>( s_1 )</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>( s_2 )</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>( s_3 )</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

This is a deterministic finite-state automaton, DFA.
For every DFA, any input string determines a path or **trace**

<table>
<thead>
<tr>
<th>input</th>
<th>path</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>s0 s0 s0 s0</td>
</tr>
<tr>
<td>021</td>
<td>s0 s0 s2 s3</td>
</tr>
<tr>
<td>120</td>
<td>s0 s1 s1 s3</td>
</tr>
<tr>
<td>11</td>
<td>s0 s1 s1 s3</td>
</tr>
<tr>
<td>12</td>
<td>s0 s1 s1 s3</td>
</tr>
<tr>
<td>21</td>
<td>s0 s1 s1 s3</td>
</tr>
<tr>
<td>22</td>
<td>s0 s1 s1 s3</td>
</tr>
<tr>
<td>111</td>
<td>s0 s1 s1 s3</td>
</tr>
<tr>
<td>110</td>
<td>s0 s1 s1 s3</td>
</tr>
<tr>
<td>212</td>
<td>s0 s1 s1 s3</td>
</tr>
</tbody>
</table>

This is a deterministic finite-state automaton, DFA
For every DFA, any input string determines a path.

<table>
<thead>
<tr>
<th>input</th>
<th>path</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>s0 s0 s0 s0</td>
</tr>
<tr>
<td>021</td>
<td>s0 s0 s2 s3</td>
</tr>
<tr>
<td>120</td>
<td>s0 s1 s1 s3</td>
</tr>
<tr>
<td>11</td>
<td>s0 s1 s0</td>
</tr>
<tr>
<td>12</td>
<td>s0 s1 s1</td>
</tr>
<tr>
<td>21</td>
<td>s0 s2 s3</td>
</tr>
<tr>
<td>22</td>
<td>s0 s2 s0</td>
</tr>
<tr>
<td>111</td>
<td>s0 s1 s0 s1</td>
</tr>
<tr>
<td>110</td>
<td>s0 s1 s0 s0</td>
</tr>
<tr>
<td>212</td>
<td>s0 s2 s3 s3</td>
</tr>
</tbody>
</table>

This is a deterministic finite-state automaton, DFA.
Which paths end in an accepting state?

<table>
<thead>
<tr>
<th>input</th>
<th>path</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>s0 s0 s0 s0</td>
</tr>
<tr>
<td>021</td>
<td>s0 s0 s2 s3</td>
</tr>
<tr>
<td>120</td>
<td>s0 s1 s1 s3</td>
</tr>
<tr>
<td>11</td>
<td>s0 s1 s0</td>
</tr>
<tr>
<td>12</td>
<td>s0 s1 s1</td>
</tr>
<tr>
<td>21</td>
<td>s0 s2 s3</td>
</tr>
<tr>
<td>22</td>
<td>s0 s2 s0</td>
</tr>
<tr>
<td>111</td>
<td>s0 s1 s0 s1</td>
</tr>
<tr>
<td>110</td>
<td>s0 s1 s0 s0</td>
</tr>
<tr>
<td>212</td>
<td>s0 s2 s3 s3</td>
</tr>
</tbody>
</table>
Which paths end in an accepting state?

<table>
<thead>
<tr>
<th>input</th>
<th>path</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>s0 s0 s0 s0</td>
</tr>
<tr>
<td>7</td>
<td>s0 s0 s2 s3</td>
</tr>
<tr>
<td>15</td>
<td>s0 s1 s1 s3</td>
</tr>
<tr>
<td>4</td>
<td>s0 s1 s0</td>
</tr>
<tr>
<td>5</td>
<td>s0 s1 s1</td>
</tr>
<tr>
<td>7</td>
<td>s0 s2 s3</td>
</tr>
<tr>
<td>8</td>
<td>s0 s2 s0</td>
</tr>
<tr>
<td>13</td>
<td>s0 s1 s0 s1</td>
</tr>
<tr>
<td>12</td>
<td>s0 s1 s0 s0</td>
</tr>
<tr>
<td>23</td>
<td>s0 s2 s3 s3</td>
</tr>
</tbody>
</table>
What is the meaning of each state?

Express each of these decimal numbers in ternary notation, and find out what state is reached by the trace they determine.

This is a deterministic finite-state automaton, DFA.
This is a deterministic finite-state automaton, DFA

What is the meaning of each state?

This machine counts ternary mod 4

\[ F(q, s) = 3 \times q + s \pmod{4} \]