Deterministic Finite Automata (DFA)

Dr. Muhammad S Khan
(mskhan@liv.ac.uk)

Ashton Building, Room G22
http://www.csc.liv.ac.uk/~khan/comp218
Finite Automata

Some Applications

- Software for designing and checking the behavior of digital circuits

- Lexical analyzer of a typical compiler

- Software for scanning large bodies of text (e.g., web pages) for pattern finding

- Software for verifying systems of all types that have a finite number of states (e.g., stock market transaction, communication/network protocol)

- Also used in statistical models for analysing biological and textual sequences.
Finite Automata: Examples

- **On/Off switch**

- **Modeling recognition of the word “then”**
Structural expressions

- Grammars
- Regular expressions
  - E.g., unix style to capture city names such as “Palo Alto CA”:
  
  \[
  [A-Z][a-z]*([ ]?[A-Z][a-z]*)*[ ][A-Z][A-Z]
  \]

  - Start with a letter
  - A string of other letters (possibly empty)
  - Other space delimited words (part of city name)
  - Should end w/ 2-letter state code
Finite State Automata (FSA)

- **Deterministic**
  - On each input there is one and only one state to which the automaton can transition from its current state

- **Nondeterministic**
  - An automaton can be in several states at once
Lexical Analysis

Break a program up into "tokens" - these tokens can usually be described using deterministic finite automata

- pay
- =
- salary
- +
- ( 
  - overtimerate
  - *
  - overtime
- )
- ;
Deterministic Finite Automata

Simple mechanism for scanning a string

![Finite Automaton Diagram]

Figure: finite automaton

Gives method of recognising words belonging to certain languages (must accept or fail to accept at end of string)
Deterministic Finite Automata

We shall see that finite automata can be used to describe
- any finite set of strings
- various infinite sets of strings, e.g.
  - strings having exactly 2 occurrences of the letter \(a\)
  - strings having more than 6 letters
  - strings in which letter \(b\) never comes before letter \(a\)
Deterministic Finite Automata

1. A finite set of **states**, often denoted $Q$
2. A finite set of **input symbols**, often denoted $\Sigma$
3. A **transition function** that takes as arguments a state and an input symbol and returns a state.
   The transition function is commonly denoted $\delta$
   If $q$ is a state and $a$ is a symbol, then $\delta(q, a)$ is a state $p$ (and in the graph that
   represents the automaton there is an arc from $q$ to $p$ labeled $a$)
4. A **start state**, one of the states in $Q$
5. A set of **final or accepting** states $F$ ($F \subseteq Q$)

Notation: A DFA $A$ is a tuple

$$A = (Q, \Sigma, \delta, q_0, F)$$
A **deterministic finite automaton** (DFA) has 5 components:

1. $Q$ is a finite nonempty set whose members are called **states** of the automaton;

2. $A$ is a finite nonempty set called the **alphabet** of the automaton;

3. $\phi$ is a map from $Q \times A$ to $Q$ called the **transition function** of the automaton;

4. $i$ is a member of $Q$ and is called the **initial state**;

5. $T$ is a nonempty subset of $Q$ whose members are called **terminal states** or **accepting states**.
“quintuple” — any DFA can be divided into these 5 components

**state** of a machine tells you something about the prefix that has
been read so far. If the string is a member of the language of
interest, the state reached when the whole string has been scanned
will be an accepting state (a member of $T$).

There is only one empty string so there is only one initial state
(denoted $i$)

**Transition function** $\phi$ tells you how state should change when an
additional letter is read by the DFA

A DFA is often depicted as a labelled directed graph. (called
*transition diagram*)
Other notations for DFAs

Transition diagrams

- Each state is a node
- For each state \( q \in Q \) and each symbol \( a \in \Sigma \), let \( \delta(q, a) = p \), then the transition diagram has an arc from \( q \) to \( p \), labelled \( a \)
- There is an arrow to the start state \( q_0 \)
- Nodes corresponding to final states are marked with doubled circle or arrow going out of a node (not to another node)

Transition tables

- Tabular representation of a function
- The rows correspond to the states and the columns to the inputs
- The entry for the row corresponding to state \( q \) and the column corresponding to input \( a \) is the state \( \delta(q, a) \)
Example (transition diagram)

3 states, $i$, $r$ and $t$. Accepting state $t$ has outgoing arrow.
Example (transition diagram)

Alternative notation for accepting state is 2 concentric circles.
Example computation

Input word 110100

symbol 1 1 0 1 0 0
state i t t t t t t
Symbolic description of the example DFA

Automaton \( A = (Q, A, \phi, i, T) \)
Set of states \( Q = \{i, t, r\} \), \( A = \{0, 1\} \), \( T = \{t\} \) and the transition function \( \phi \) is given by

\[
\begin{align*}
\phi(i, 0) &= r, & \phi(i, 1) &= t, \\
\phi(t, 0) &= t, & \phi(t, 1) &= t, \\
\phi(r, 0) &= r, & \phi(r, 1) &= r.
\end{align*}
\]

It is simpler to describe a transition function by a table of values. In this example we have:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)</td>
<td>(r)</td>
<td>(t)</td>
</tr>
<tr>
<td>(t)</td>
<td>(t)</td>
<td>(t)</td>
</tr>
<tr>
<td>(r)</td>
<td>(r)</td>
<td>(r)</td>
</tr>
</tbody>
</table>
If $\phi$ is a partial function (not defined for some state/letter pairs), then the DFA rejects an input if it ever encounters such a pair. This convention often simplifies the definition of a DFA. In the previous example we could use transition table

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>$t$</td>
<td></td>
</tr>
<tr>
<td>$t$</td>
<td>$t$</td>
<td>$t$</td>
</tr>
</tbody>
</table>
Example

Give a DFA that accepts the words "cat" and "dog" (using the 6-letter alphabet $a, c, d, g, o, t$).
Any DFA that uses the convention that an undefined transition leads to a rejection, can be converted to a DFA that uses a total transition function (that is, one that is defined for all combinations of input symbols and states).

The convention is useful, but it does not add extra expressive power.
The DFA can be “stripped down” if we understand an undefined transition to mean: reject the whole string.
Exercises

Draw a diagram for a DFA that only accepts the words:
fun, fair, unfair, funfair

Note. this language can be represented by simple enumeration: in set-theoretic notation

\{fun, fair, unfair, funfair\}
How about the following infinite language. Can you give a DFA that accepts the words:
bad, baad, baaad, baaaad, ...?
Any finite language (and various infinite languages) have DFAs that recognise them.

Question: What sort of languages can be recognised by DFAs?

Note: although finite languages can be enumerated, it may be advantageous to describe them using a DFA (e.g. words of length 10)
Design a DFA to accept the language

\[ L = \{ w \mid w \text{ has both an even number of } 0 \text{ and an even number of } 1 \} \]
Construct a DFA that accepts words over the alphabet \{a, b\} which contain an odd number of a’s and an even number of b’s. State should keep track of parity of number of occurrences of each letter seen so far.

So: this suggests using 4 states, called \(E/E\), \(E/O\), \(O/E\), \(O/O\) where \(i = E/E\).

\[
\phi(E/E, a) = O/E \\
\phi(E/E, b) = E/O
\]

etc.

\[
T = \{O/E\}
\]
More examples

Write down DFAs which recognise the following languages over the alphabet \{a, b\}:

- L1 is set of all words containing exactly three occurrences of a
- L2 is set of all words containing at least three occurrences of a
- L3 is set of words containing the substring aaa
The DFA define a language: the set of all strings that result in a sequence of state transitions from the start state to an accepting state.

Extended transition function

- Describes what happens when we start in any state and follow any sequence of inputs.
- If $\delta$ is our transition function, then the extended transition function is denoted by $\hat{\delta}$.
- The extended transition function is a function that takes a state $q$ and a string $w$ and returns a state $p$ (the state that the automaton reaches when starting in state $q$ and processing the sequence of inputs $w$).
Definition by induction on the length of the input string

**Basis:** $\hat{\delta}(q, \epsilon) = q$

If we are in a state $q$ and read no inputs, then we are still in state $q$

**Induction:** Suppose $w$ is a string of the form $xa$; that is $a$ is the last symbol of $w$, and $x$ is the string consisting of all but the last symbol

Then: $\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a)$

To compute $\hat{\delta}(q, w)$, first compute $\hat{\delta}(q, x)$, the state that the automaton is in after processing all but the last symbol of $w$

Suppose this state is $p$, i.e., $\hat{\delta}(q, x) = p$

Then $\hat{\delta}(q, w)$ is what we get by making a transition from state $p$ on input $a$ - the last symbol of $w$
Initially the state is $i$ and if the input word is $w = a_1 a_2 \ldots a_n$ then, as each letter is read, the state changes and we get $q_1, q_2, \ldots, q_n$ defined by

\[
egin{align*}
q_1 &= \phi(i, a_1) \\
q_2 &= \phi(q_1, a_2) \\
q_3 &= \phi(q_2, a_3) \\
& \vdots \\
q_n &= \phi(q_{n-1}, a_n)
\end{align*}
\]
Extend the definition of the transition function so that it tells us which state we reach after a word (not just a single letter) has been scanned:

In the above notation, extend the map $\phi : Q \times A \rightarrow Q$ to $\phi : Q \times A^* \rightarrow Q$ by defining:

$$\phi(q, \epsilon) = q \quad \text{for all } q \in Q$$
$$\phi(q, wa) = \phi(\phi(q, w), a) \quad \text{for all } q \in Q; w \in A^*; a \in A.$$

It is easy to show by induction that this extended map satisfies

$$\phi(q, vw) = \phi(\phi(q, v), w) \quad \text{for all } q \in Q; v, w \in A^*.$$
The language of a DFA $A = (Q, \Sigma, \delta, q_0, F)$, denoted $L(A)$ is defined by

$$L(A) = \{w \mid \hat{\delta}(q_0, w) \text{ is in } F\}$$

The language of $A$ is the set of strings $w$ that take the start state $q_0$ to one of the accepting states.

If $L$ is a $L(A)$ from some DFA, then $L$ is a regular language.
Language defined by a DFA

Suppose we have a DFA $A$. A word $w \in A^*$ is said to be accepted or recognised by $A$ if $\phi(i, w) \in T$, otherwise it is said to be rejected. The set of all words accepted by $A$ is called the language accepted by $A$ and will be denoted by $L(A)$. Thus

$$L(A) = \{w \in A^* : \phi(i, w) \in T\}.$$