Next: Finite automata

Ch. 1.1: Deterministic finite automata (DFA)

We will:

• Design automata for simple problems
• Study languages recognized by finite automata.
Recognizing finite languages

• Just need a lookup table and a search algorithm
• Problem – cannot express infinite sets, e.g. odd integers

Finite Automata

The simplest machine that can recognize an infinite language.

“Read once”, “no write” procedure.

Useful for describing algorithms also.
Used a lot in network protocol description.

Remember: DFA’s can accept finite languages as well.
A Simple Automaton (0)

states

transition rules

starting state

accepting state

A Simple Automaton (1)

on input “0110”, the machine goes:

$q_1 \rightarrow q_1 \rightarrow q_2 \rightarrow q_2 \rightarrow q_3$ = “reject”
A Simple Automaton (2)

on input “101”, the machine goes:
$q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_2 = “accept”$

A Simple Automaton (3)

010: reject
11: accept
010100100100100: accept
010000010010: reject
ε: reject
Examples of languages accepted by DFA

• $L = \{ w \mid w \text{ ends with } 1 \}$
• $L = \{ w \mid w \text{ contains sub-string } 00 \}$
• $L = \{ w \mid |w| \text{ is divisible by } 3 \}$
• $L = \{ w \mid |w| \text{ is odd or } w \text{ ends with } 1 \}$
• $L = \{ w \mid |w| \text{ is divisible by } 10^6 \}$

Note: $\Sigma = \{0,1\}$ in each case

DFA design

• Design DFA for language
  – $L = \{ w \in \{0,1\}^* \mid w \text{ contains substring } 01 \}$
• Three states to remember:
  – Have seen the substring 01
  – Not seen substring 01 and last symbol was 0
  – Not seen substring 01 and last symbol was not 0
• General principles?
DFA: Formal definition

- A deterministic finite automaton (DFA) \( M \) is defined by a 5-tuple \( M=(Q, \Sigma, \delta, q_0, F) \)
  
  - \( Q \): finite set of states
  - \( \Sigma \): finite alphabet
  - \( \delta \): transition function \( \delta:Q \times \Sigma \rightarrow Q \)
  - \( q_0 \in Q \): start state
  - \( F \subseteq Q \): set of accepting states

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

<table>
<thead>
<tr>
<th>States ( Q )</th>
<th>( q_1, q_2, q_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabet ( \Sigma )</td>
<td>{0,1}</td>
</tr>
<tr>
<td>Start state ( q_0 )</td>
<td>( q_1 )</td>
</tr>
<tr>
<td>Accept states ( F )</td>
<td>{q_2}</td>
</tr>
</tbody>
</table>

Transition function \( \delta \):

\[
\begin{array}{c|cc}
\delta & 0 & 1 \\
\hline
q_1 & q_1 & q_2 \\
q_2 & q_3 & q_2 \\
q_3 & q_2 & q_2 \\
\end{array}
\]
Recognizing Languages (defn)

A finite automaton \( M = (Q, \Sigma, \delta, q_0, F) \) accepts a string/word \( w = w_1 \ldots w_n \) if and only if there is a sequence \( r_0 \ldots r_n \) of states in \( Q \) such that:

1) \( r_0 = q_0 \)

2) \( \delta(r_i, w_{i+1}) = r_{i+1} \) for all \( i = 0, \ldots, n-1 \)

3) \( r_n \in F \)

Regular Languages

The language recognized by a finite automaton \( M \) is denoted by \( L(M) \).

A regular language is a language for which there exists a recognizing finite automaton.
Two DFA Questions

Given the description of a finite automaton $M = (Q, \Sigma, \delta, q_0, F)$, what is the language $L(M)$ that it recognizes?

In general, what kind of languages can be recognized by finite automata? (What are the regular languages?)

Union of Two Languages

**Theorem 1.12**: If $A_1$ and $A_2$ are regular languages, then so is $A_1 \cup A_2$. (The regular languages are ‘closed’ under the union operation.)

**Proof idea**: $A_1$ and $A_2$ are regular, hence there are two DFA $M_1$ and $M_2$, with $A_1 = L(M_1)$ and $A_2 = L(M_2)$. Out of these two DFA, we will make a third automaton $M_3$ such that $L(M_3) = A_1 \cup A_2$. 
Proof Union-Theorem (1)

\( M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \) and \( M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \)

Define \( M_3 = (Q_3, \Sigma, \delta_3, q_3, F_3) \) by:

- \( Q_3 = Q_1 \times Q_2 = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\} \)

- \( \delta_3((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)) \)

- \( q_3 = (q_1, q_2) \)

- \( F_3 = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\} \)

Proof Union-Theorem (2)

The automaton \( M_3 = (Q_3, \Sigma, \delta_3, q_3, F_3) \) runs \( M_1 \) and \( M_2 \) in ‘parallel’ on a string \( w \).

In the end, the final state \((r_1, r_2)\) ‘knows’ if \( w \in L_1 \) (via \( r_1 \in F_1 \)) and if \( w \in L_2 \) (via \( r_2 \in F_2 \)).

The accepting states \( F_3 \) of \( M_3 \) are such that \( w \in L(M_3) \) if and only if \( w \in L_1 \) or \( w \in L_2 \), for:

\( F_3 = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\} \).
Concatenation of $L_1$ and $L_2$

Definition: $L_1 \bullet L_2 = \{ xy | x \in L_1 \text{ and } y \in L_2 \}$

Example: $\{a,b\} \bullet \{0,11\} = \{a0,a11,b0,b11\}$

Theorem 1.13: If $L_1$ and $L_2$ are regular languages, then so is $L_1 \bullet L_2$.
(The regular languages are ‘closed’ under concatenation.)

---

Proving Concatenation Thm.

Consider the concatenation:
$\{1,01,11,001,011,\ldots\} \bullet \{0,000,00000,\ldots\}$
(That is: the bit strings that end with a “1”, followed by an odd number of 0’ s.)

Problem is: given a string $w$, how does the automaton know where the $L_1$ part stops and the $L_2$ substring starts?

We need an $M$ with ‘lucky guesses’.
Nondeterminism

Nondeterministic machines are capable of being lucky, no matter how small the probability.

A nondeterministic finite automaton has transition rules/possibilities like

A Nondeterministic Automaton

This automaton accepts “0110”, because there is a possible path that leads to an accepting state, namely:

$q_1 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_4 \rightarrow q_4$
A Nondeterministic Automaton

The string 1 gets rejected: on “1” the automaton can only reach: \{q_1, q_2, q_3\}.

Nondeterminism ~ Parallelism

For any (sub)string \(w\), the nondeterministic automaton can be in a set of possible states.

If the final set contains an accepting state, then the automaton accepts the string.

“The automaton processes the input in a parallel fashion. Its computational path is no longer a line, but a tree.” (Fig. 1.16)
Nondeterministic FA (def.)

- A nondeterministic finite automaton (NFA) M is defined by a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, with

  - $Q$: finite set of states
  - $\Sigma$: finite alphabet
  - $\delta$: transition function $\delta: Q \times \Sigma \rightarrow P(Q)$
  - $q_0 \in Q$: start state
  - $F \subseteq Q$: set of accepting states

Nondeterministic $\delta: Q \times \Sigma_\varepsilon \rightarrow P(Q)$

The function $\delta: Q \times \Sigma_\varepsilon \rightarrow P(Q)$ is the crucial difference. It means: “When reading symbol “a” while in state $q$, one can go to one of the states in $\delta(q, a) \subseteq Q$.”

The $\varepsilon$ in $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ takes care of the empty string transitions.
Recognizing Languages (def)

A nondeterministic FA $M = (Q, \Sigma, \delta, q, F)$ accepts a string $w = w_1 \ldots w_n$ if and only if we can rewrite $w$ as $y_1 \ldots y_m$ with $y_i \in \Sigma^*$ and there is a sequence $r_0 \ldots r_m$ of states in $Q$ such that:

1) $r_0 = q_0$

2) $r_{i+1} \in \delta(r_i, y_{i+1})$ for all $i=0,\ldots,m-1$

3) $r_m \in F$

Exercises

[Sipser 1.5]: Give NFAs with the specified number of states that recognize the following languages over the alphabet $\Sigma=\{0,1\}$:

1. $\{ w \mid w \text{ ends with } 00 \}$, three states
2. $\{0\}$; two states
3. $\{ w \mid w \text{ contains even number of } 0\text{s, or exactly two } 1\text{s} \}$, six states
4. $\{0^n \mid n \in \mathbb{N} \}$, one state
Proof the following result:
“If L₁ and L₂ are regular languages, then L₁ ∩ L₂ is a regular language too.”

Describe the language that is recognized by this nondeterministic automaton: