Finite Automata
(FAs)

- Automata: The General Idea
- Diagrams and Language Recognition
- Formal Notation
- Nondeterministic Finite Automata
- Removing Nondeterminism
Why FAs?

- **Recognition**: Is string $x$ in language $L$?
- **Simplest** way to specify some languages
- **Equivalences** among representation systems
- First in a family of *models of computation*
States: 
the essence of FAs

- *State*: a model of the entire internal memory
- FA's *memory*: knowing what state it’s in.
- State to state *transitions*, based on input
- State-transition *diagrams* (next)
### State Transition Diagrams for some languages

<table>
<thead>
<tr>
<th>Language Set</th>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>{a, bc}</td>
<td><img src="https://via.placeholder.com/150" alt="Diagram" /></td>
</tr>
<tr>
<td>{aa,ac,ba,bc, aaa,aea,baa,ber}</td>
<td><img src="https://via.placeholder.com/150" alt="Diagram" /></td>
</tr>
<tr>
<td>{\emptyset, a,b,aa,ab,bb,aaa,aab,abb,bbb,\ldots}</td>
<td><img src="https://via.placeholder.com/150" alt="Diagram" /></td>
</tr>
<tr>
<td>{ab, ac}</td>
<td><img src="https://via.placeholder.com/150" alt="Diagram" /></td>
</tr>
<tr>
<td>{ab, ac}</td>
<td><img src="https://via.placeholder.com/150" alt="Diagram" /></td>
</tr>
<tr>
<td>{ab, ac}</td>
<td><img src="https://via.placeholder.com/150" alt="Diagram" /></td>
</tr>
</tbody>
</table>


Reading a State Transition Diagram

<table>
<thead>
<tr>
<th>What to do with the...</th>
<th>state:</th>
<th>string:</th>
</tr>
</thead>
<tbody>
<tr>
<td>start at the beginning:</td>
<td>start state;</td>
<td>start of string</td>
</tr>
<tr>
<td>go right on to the end:</td>
<td>follow an arrow;</td>
<td>use up a symbol</td>
</tr>
<tr>
<td>then stop:</td>
<td>accepting state;</td>
<td>end of string</td>
</tr>
</tbody>
</table>
Conventions for State Transition Diagrams

- Arrow with 2 labels: shorthand for 2 transitions.
- If the start state is an accepting state, it is accepted.
- Stopping at an accepting state is optional.
- Rejection: if input ends at a non-accepting state.
- Rejection: if (state, input) pair has no arrow.
From Description to Diagram

Find a state-transition diagram for strings over $\Sigma = \{a, b\}$ that have an odd number of b’s.

Using states to keep track of whether the number of b’s so far is odd or even.
Find a state-transition diagram for those strings over $\Sigma = \{a,b\}$ that have both an odd number of $a$’s and an odd number of $b$’s.

Using states to reflect even/odd combinations.
Formal Notation:  
**The Quintuple**

A finite automaton, $M$, is a quintuple, $(Q, S, q_0, \delta, A)$, where

- $Q$ is a finite set of states,
- $S$ is a finite set of symbols,
- $q_0 \in Q$ and $q_0$ is the start state,
- $\delta: Q \times S \rightarrow Q$,
- $A \subseteq Q$ and $A$ is the set of accepting states.
Example of a Quintuple

Let $M = (Q, \square, q_0, \square, A)$ be the FA with

$Q = \{q_0\}$,

$\square = \{a\}$,

$A = \{q_0\}$, and

$\square(q_0, a) = q_0$.

• Draw the state-transition diagram.

• What is its language?
Let $M = (\{q_0, q_1, q_2\}, \{a, d\}, q_0, \square, \{q_1\}), \ldots$

where $\square$ is specified by this table:

<table>
<thead>
<tr>
<th></th>
<th>$a$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$q_1$</td>
<td>$q_2$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>$q_1$</td>
<td>$q_1$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$q_2$</td>
<td>$q_2$</td>
</tr>
</tbody>
</table>

- Which is the trap state?
- Draw the Diagram.
The Diagram, with Trap State

• *Trap state* is shown dotted …

• … as are the transitions to and from it.

• What (else) tells you it’s the trap state?

• What language is expressed?
The Language

- The language of the diagram is \( \{a\} \{a,d\}^* \)

- What has this to do with programming languages?
is a Function

Let $M = (Q, \delta, q_0, \alpha, A)$ be an FA.

- $d^*$ is a function.
- It maps a state and a string ... to a resulting state:

\[ d^* : Q \times \Sigma^* \rightarrow Q \]
Recursive Definition of $d^*$

For any $q \in Q$,

$$d^*(q, \square) = q$$

For any $q \in Q$, any $s \in S$ and any $x \in S^*$,

$$d^*(q, x\square) = (d^*(q, x), \square).$$
The Language of an Automaton

Given a finite automaton $M$, where

\[ M = (Q, \delta, q_0, \bar{q}, A), \]

$M$'s language, $L(M)$, the strings it accepts:

\[ L(M) = \{ x \delta^* \delta^*(q_0, x) \bar{q} A \} \]

$M$ rejects the strings of (the complement) $\delta^* \setminus L(M)$. 
Recognizing the Language of an Automaton

Let $M$ be the FA, $(Q, \delta, q_0, A, \square)$, $x$ the input string, and “$\leftarrow$” the assignment operator.

Let dequeue remove and return the first symbol of a string.

$q \leftarrow q_0$
while $x \neq \square$ do
    $\delta \leftarrow \text{dequeue}(x)$
    $q \leftarrow \delta(q, \square)$
return $q \in A$
NonDetermism

• Definition: 2+ transitions/arrows
  with the same input/label, and
  from the same state

• Motivating Examples
  All strings ending in abb, or
  Results of \( \square \)-removal (later)
Removing NonDeterminism:
The Subset Construction

- Think of partial knowledge as a set of states.
- New machine: states are ‘states of mind.’
- So new state is a set of old states.
- New start state contains just the old one.
- New state accepts if it includes old accepters.
\[\mathcal{L}\]-Elimination

- For each \(\mathcal{L}^*a_i\) path from \(q_j\) to \(q_k\), add an \(a_i\) transition from \(q_j\) to \(q_k\).
- For each \(\mathcal{L}^*\) path from \(q_j\) to an accepting state \(q_k\), make \(q_j\) accepting.
- Remove all \(\mathcal{L}\)-transitions.
Tidying Up after $\emptyset$-Elimination

• No Unreachables: Remove unreachable states and transitions from them.

• Careful Merging: A set of states can be merged if they agree about acceptance and destination

  ✓ all or none of them are accepting and

  ✓ all would have the same transition destination, – if any – for each label, after completion of the merger.
Example Using 3 Simplifications

Each diagram accepts the language of \{a\}^* 

✓ The first one is from \{a\}^* in Figure 10.2.
✓ \smallfrown\text{-elimination yields the second.}
✓ Eliminate an unreachable state to get the second.
✓ Then carefully merge to get the last one.
Equivalences we have shown

• the languages from 3 language ops
• ... are within the capacity of DFA diagrams
• ... and of NFA-Łs
• ... and NFAs (without Łs)
• ...and DFAs