Determismistic & Nondeterministic Finite Automata

Martin Fränzle

Informatics and Mathematical Modelling
The Technical University of Denmark

Preliminaries for Formal Languages:

Alphabets, strings, languages
What you’ll learn

You will be introduced to the basic notions of

**Alphabet**: A set of “symbols”,

**String/word**: a sequence of symbols from the alphabet,

**Language**: a set of strings.

Alphabets

**Def.**: An alphabet is a *finite, nonempty set*.

**Ex.**: • The binary alphabet \{0, 1\},
  • the set of all printable ASCII characters,
  • the set \{red, green, blue, yellow, magenta, cyan\} of basic colors.

**N.B.**: This notion of alphabet is distinctly different from everyday use:
  • alphabets need not feature an ordering,
  • the members of an alphabet need not be letters in the usual sense, but can well be compounds of some kind (e.g., tuples).

We will often denote alphabets by the greek letter \(\Sigma\).
Strings (a.k.a. words)

**Def.:** A string (or word) over an Alphabet $\Sigma$ is a *finite* sequence of symbols from $\Sigma$.

The length of a string $w$, denoted $|w|$, is the number of positions for symbols in the string.

**Ex.:**
- $0100$ (or, more formally, $\langle 0, 1, 0, 0 \rangle$) is a string of length 4 over alphabet $\{0, 1\}$.
- $\langle \rangle$ is the string of length 0 over alphabet $\{a, \ldots, z\}$ (as well as over any other alphabet). It is called the empty string and is generically denoted $\epsilon$, irrespective of the particular alphabet.

**Def.:** Given strings $v = \langle v_1, v_2, \ldots, v_n \rangle$ and $w = \langle w_1, w_2, \ldots, w_m \rangle$, the concatenation of $v$ and $w$ is the string

$$vw \overset{\text{def}}{=} \langle v_1, v_2, \ldots, v_n, w_1, w_2, \ldots, w_m \rangle$$

of length $n + m$.

Powers of an alphabet

Given an alphabet $\Sigma$ and a natural number $n \in \mathbb{N}$,
- $\Sigma^n$ is the set of all strings over $\Sigma$ of length $n$,
- $\Sigma^+ \overset{\text{def}}{=} \Sigma^1 \cup \Sigma^2 \cup \ldots \overset{\text{def}}{=} \bigcup_{i=1}^{\infty} \Sigma^i$ is the set of all *nonempty* strings over $\Sigma$,
- $\Sigma^* \overset{\text{def}}{=} \{\epsilon\} \cup \Sigma^1 \cup \Sigma^2 \cup \ldots \overset{\text{def}}{=} \{\epsilon\} \cup \Sigma^+$ is the set of all strings over $\Sigma$. 
Languages

**Def.** A set of strings over an alphabet $\Sigma$ (i.e., a subset of $\Sigma^*$) is called a language over $\Sigma$.

**Ex.:**
- The set of strings consisting of $n$ 0's followed by $n$ 1's for arbitrary $n \in \mathbb{N}$ forms a language over $\{0, 1\}$:
  
  \[ \{\varepsilon, 01, 0011, 000111, 00001111, \ldots \} = \{0^n1^n | n \in \mathbb{N}\} \]

- The set of strings over $\{0, \ldots, 9\}$ representing a prime number forms a language over $\{0, \ldots, 9\}$:
  
  \[ \{2, 3, 5, 7, 11, 13, 17, \ldots \} \]

- $\Sigma^*$ is a language over $\Sigma$.
- $\emptyset$ is a language (over arbitrary alphabet).
- $\{\varepsilon\}$ is a language (over arbitrary alphabet).  

*Note:* Languages can be finite or infinite!

Problems

**Def.:** Given a language $L$ over some alphabet $\Sigma$, the problem $L$ is:

*Given a string $w$ over $\Sigma$, decide whether or not $w \in L$.*

**Ex.:**
- Problem of deciding whether some $w \in \{0, 1\}^*$ is a binary representation of a prime number.
- Problem of deciding whether some $w \in \{\alpha, \ldots, \varepsilon\}^*$ is the imperfect of an English verb.
- Problem of deciding whether a string $w \in (\{A, \ldots, H\} \times \{1, \ldots, 8\} \times \{A, \ldots, H\} \times \{1, \ldots, 8\})^*$ describes
  - a game of chess where 'white' has won,
  - a game of chess that has reached a position where 'white' can enforce a win.

Questions concerning a given problem are in particular
- whether it can be solved by a computer,
- if so, what would be the best algorithm to do so,
- how much computational effort this requires.
Finite Automata

The deterministic variant: DFAs

DFAs — informally

- An automaton is a computational device that changes its state depending on the inputs it receives.
- A finite automaton has finitely many states for remembering its computation history.
- In a deterministic finite automaton, the current state is completely determined by the sequence of inputs seen so far.
- Automata define languages through the following mechanism:
  1. Some states are marked as being “accepting”;
  2. whenever the automaton is in an accepting state, the string of inputs seen so far is a member of the language accepted by the automaton.

\[
\{a, ab, abb, abbb, abbbb, \ldots\} \cup \\
\{b, baa, baaaa, baaaaa, \ldots\}
\]
DFAs — formally

A deterministic finite automaton (DFA) \( A = (Q, \Sigma, \delta, q_0, F) \) consist of

1. a finite set of states \( Q \),
2. a finite set of input symbols \( \Sigma \), called the “alphabet of \( A \)”,
3. a transition function \( \delta : Q \times \Sigma \rightarrow Q \),
4. a start state \( q_0 \in Q \),
5. a set of final or accepting states \( F \subseteq Q \).

The automaton

1. starts in \( q_0 \),
2. proceeds from state \( q \) to state \( \delta(q, a) \) if the current input is \( a \),
3. accepts a string (= sequence of symbols) iff that sequence of inputs drives it from \( q_0 \) to some \( q \in F \).

Transition diagrams

are a graphical notation for FAs.

A transition diagram for a DFA \( A = (Q, \Sigma, \delta, q_0, F) \) is a graph s.t.

1. For each state in \( Q \) there is a node,
2. if \( \delta(q, a) = q' \) then (and only then) there is an arc from \( q \) to \( q' \) labelled \( a \), (we allow ourselves to collate multiple arcs between the same nodes into a single one labelled with a list)
3. there is an arrow without source node into the start state,
4. nodes representing accepting states are marked with double circles.
### Transition tables

Transition tables are tabular representations of transition functions, where information about initiality and finality of states is added:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>→q₀</td>
<td>q₂</td>
<td>q₀</td>
</tr>
<tr>
<td>*q₁</td>
<td>q₁</td>
<td>q₁</td>
</tr>
<tr>
<td>q₂</td>
<td>q₂</td>
<td>q₁</td>
</tr>
</tbody>
</table>

As the state space and the alphabet can be deduced from the table, transition tables are a complete representation of DFAs!

### Extending the trans. fct. to strings

Given $\delta : Q \times \Sigma \to Q$, we define a function $\hat{\delta} : Q \times (\Sigma^*) \to Q$ via recursion:

**Base case:** $\hat{\delta}(q, \varepsilon) \overset{\text{def}}{=} q$ for each $q \in Q$;

**Recursion:**

$\hat{\delta}(q, xa) \overset{\text{def}}{=} \hat{\delta}(\hat{\delta}(q, x), a)$ for each $q \in Q$, $x \in \Sigma^*$, $a \in \Sigma$.

**N.B.:** We could as well have defined

$$\hat{\delta}(q, ax) \overset{\text{def}}{=} \hat{\delta}(\delta(q, a), x)$$

without altering the function defined.
The language of a DFA

**Def:** The language $L(\mathcal{A})$ of a DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ is

$$L(\mathcal{A}) \overset{\text{def}}{=} \{ w \in \Sigma^* | \hat{\delta}(q_0, w) \in F \}.$$  

The language $L(\mathcal{A})$ thus is the set of strings that drive $\mathcal{A}$ from the initial state to some accepting state.

**Def:** If $L = L(\mathcal{A})$ for some DFA $\mathcal{A}$ then we call $L$ a regular language. The regular languages are exactly those languages $L$ which are definable by DFAs in the sense that $L = L(\mathcal{A})$ for some DFA $\mathcal{A}$.

---

**Finite Automata**

The nondeterministic variant: NFAs
NFAs — informally

- Has the freedom to do various different moves when in a state and seeing some input,
- has the help of an “oracle” for performing the right guess,
- this is modeled mathematically as
  1. the ability to be in various states at once, and
  2. accepting a string whenever at least one of those states is accepting.

```
               a, b
              /   \
             /     \
            /       \
           /         \\
  (a) -> a, b <- (b)
```

NFAs — formally

An *nondeterministic finite automaton (NFA)* \( A = (Q, \Sigma, \delta, q_0, F) \) consist of

1. a **finite set of states** \( Q \),
2. a **finite set of input symbols** \( \Sigma \), called the “alphabet of \( A \)”,
3. a **transition function** \( \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q) \), (which may be identified with a transition *relation* \( \delta' \subseteq Q \times \Sigma \times Q \)),
4. a **start state** \( q_0 \in Q \),
5. a **set of final or accepting states** \( F \subseteq Q \).

The automaton

1. starts in state set \( \{q_0\} \),
2. proceeds from state set \( P \) to state set \( \delta(P, a) \) if the current input is \( a \),
3. accepts a string (= sequence of symbols) iff that sequence of inputs drives it from \( q_0 \) to a state set containing some \( q \in F \).
Transition diagrams

are a graphical notation for FAs. A transition diagram for a NFA \( A = (Q, \Sigma, \delta, q_0, F) \) is a graph s.t.

1. For each state in \( Q \) there is a node,
2. if \( q' \in \delta(q, a) \) then (and only then) there is an arc from \( q \) to \( q' \) labelled \( a \), (we allow ourselves to collate multiple arcs between the same nodes into a single one labelled with a list)
3. there is an arrow without source node into the start state,
4. nodes representing accepting states are marked with double circles.

Transition tables

Do now assign sets of states:

\[
\begin{array}{c|c|c}
    & a & b \\
    \rightarrow q_0 & \{q_0\} & \{q_0, q_1\} \\
    q_1 & \emptyset & \{q_2\} \\
    *q_2 & \{q_2\} & \emptyset
\end{array}
\]
Extending the trans. fct. to strings

Given $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$, we define a function $\hat{\delta} : Q \times (\Sigma^*) \rightarrow \mathcal{P}(Q)$ via recursion:

**Base case:** $\hat{\delta}(q, \varepsilon) \overset{\text{def}}{=} \{q\}$ for each $q \in Q$;

**Recursion:** $\hat{\delta}(q, xa) \overset{\text{def}}{=} \bigcup_{p \in \hat{\delta}(q, x)} \delta(p, a)$ for each $q \in Q$, $x \in \Sigma^*$, $a \in \Sigma$.

The language of an NFA

**Def:** The language $L(\mathcal{A})$ of an NFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ is

$$L(\mathcal{A}) \overset{\text{def}}{=} \{ w \in \Sigma^* | \hat{\delta}(q_0, w) \cap F \neq \emptyset \}.$$ 

The language $L(\mathcal{A})$ thus is the set of strings that drive $\mathcal{A}$ from the initial state to a set of states containing some accepting state.