Finite state machines

- Finite automata:
  - simplest model of computation
  - describes so-called "regular languages"
  - works as follows:
    1. is always in one of finitely many states
    2. starts in some state
    3. changes state in response to input
    4. accepts input by ending in an accepting (or final) state

Example: F.A. scanning HTML documents for a list of football game results

Observations:
- $\Sigma = \text{HTML tags} \cup \text{ASCII characters}$
- each result stored in the form:
  
  \[
  \text{team 1 } X - Y \text{ team 2 } X - Y \text{ in: } m:n
  \]
- list represented as HTML list:
  
  <ol>
  ... ordered list
  <li> ... unordered list
  <li> ... list item
  
  accepts when it finds end of list

Example:

  <ol>
  <li>Rome - Lazio - 2:0</li>
  <li>Inter - Juve - 10:2</li>
  </ol>
Notation in the state transition diagram:

- state 5
- start state 1 (or initial state)
- final state 8 (or accepting state)
- transition 3 → 4

meaning: when the F.A. is in state 3 and it sees '1' in the input, it moves to state 4 and advances on the input.

Example: describe using a set-former the language of all binary strings that contain the pattern 01.
- construct a F.A. that accepts the language.

Solution: \( \Sigma = \{ 0, 1 \} \)

\[ L = \{ w \in \Sigma^* \mid w \text{ has substring } 01 \} = \{ x01y \mid x, y \in \Sigma^* \} \]

90 ... waiting for first 0
91 ... seen 0, waiting for 1
92 ... seen 01, waiting for rest of input
Note: FA means input from left to right (cannot go back) making transitions

- accepts if it is in an accepting state when it reaches the end of the input

Language accepted by ε-FA A: $L(A) = \{ w \in \Sigma^* \mid A \text{ accepts } w \}$

What we have seen are called Deterministic Finite Automata (DFA).

Definition: a DFA is a quintuple $A = (Q, \Sigma, \delta, q_0, F)$

- $Q$ ... finite nonempty set of states e.g. $Q = \{ q_0, q_1, q_2 \}$
- $\Sigma$ ... input alphabet e.g. $\Sigma = \{ 0, 1 \}$
- $q_0$ ... initial (or start) state
  - $q_0 \in Q$
- $F$ ... set of final (or accepting) states
  - $F \subseteq Q$
  - e.g. $F = \{ q_2 \}$
- $\delta$ ... total function $\delta: Q \times \Sigma \rightarrow Q$
  - called state transition function
  - can be represented as a diagram or a transition table

<table>
<thead>
<tr>
<th>$q_0$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$q_1$</td>
<td>$q_0$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>$q_1$</td>
<td>$q_2$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$q_2$</td>
<td>$q_2$</td>
</tr>
</tbody>
</table>

  Note: $\delta(q_0, 0) = q_1$, $\delta(q_0, 1) = q_0$,
  $\delta(q_1, 0) = q_1$, $\delta(q_1, 1) = q_2$,
  $\delta(q_2, 0) = q_2$, $\delta(q_2, 1) = q_2$

Note: we have still not defined formally what the language accepted by a DFA is.
Extended transition function:

we want to extend \( \delta \) to multiple transitions

\[
\delta : Q \times \Sigma^* \rightarrow Q
\]

\[
\hat{\delta} : Q \times \Sigma^* \rightarrow Q
\]

meaning: \( \hat{\delta}(q, \xi) = q' \)

denotes that starting at state \( q \), portion \( \xi \) of input string will take DFA to state \( q' \).

In other words: if \( \xi = a_1 \ldots a_n \) and

\[
\delta(q, a_1) = q_1, \quad \delta(q_1, a_2) = q_2, \ldots, \delta(q_{n-1}, a_n) = q'
\]

then \( \hat{\delta}(q, a_1 \ldots a_n) = q' \).

We can define \( \hat{\delta} \) formally by induction:

\[
\forall q \in Q, \forall a \in \Sigma, \forall \xi \in \Sigma^*
\]

\[\text{Basis: } \hat{\delta}(q, \epsilon) = q\]

\[\text{Induction: } \hat{\delta}(q, \xi a) = \delta(\hat{\delta}(q, \xi), a)\]

Note: we exploit the fact that strings are defined inductively:

- \( \epsilon \) is a string
- if \( \xi \) is a string and \( a \in \Sigma \) then \( \xi a \) is a string.
- nothing else is a string.

Example:

\[
\begin{array}{c}
\overset{q_0}{\xrightarrow{a}} \overset{q_1}{\xrightarrow{a}} \overset{q_2}{\xrightarrow{a}} \overset{q_1}{\xrightarrow{a}} \\
\end{array}
\]

\[
\hat{\delta}(q_0, \epsilon) = q_0 \]

\[
\hat{\delta}(q_0, a) = \delta(\hat{\delta}(q_0, \epsilon), a) = \delta(q_0, a) = q_0
\]

\[
\hat{\delta}(q_0, 10) = \delta(\hat{\delta}(q_0, 1), 0) = \delta(q_0, 0) = q_1
\]

\[
\hat{\delta}(q_0, 101) = \delta(\hat{\delta}(q_0, 10), 1) = \delta(q_1, 1) = q_1
\]
Consequence: $S_0$ and $S_1$ agree on strings of length 4.

Example: $S(0, e) = S(3, 0)$,

$S(1, e) = S(3, 0)$,

$S(2, e) = S(3, 0)$.

This is a crucial property because each DFA partitions $\Sigma$ into a finite number of strings. Next, we define $A = (Q, \Sigma, \delta, q_0, F)$ as the DFA accepting $L(A)$.

What is $L(A)$? Could $L(A)$ contain an even number of $1$s?
Exercise 2.2.2: Show that $\forall q \in Q$, $\forall \alpha, \beta \in \Sigma^*$

$$\delta(q, \alpha \beta) = \delta(\delta(q, \alpha), \beta)$$

Hint: use induction on $|\gamma|$

Exercise 2.2.5: Give DFAs that accept the set of all strings over $\Sigma = \{0, 1\}$ such that:

- Each consecutive block of 5 symbols contains at least two 0's
- The 10th symbol from the right is a 1
- There are no 0's on either end (or both) with 01
- The number of 0's is divisible by 5, and the number of 1's is divisible by 3

Exercise 2.2.8: Let $A$ be a DFA such that for some $a \in \Sigma$ and all $q \in Q$ we have $\delta(q, a) = q$

- Show that for all $n > 0$, $\delta(q, a^n) = q$
- Show that either $\{a\}^* \in L(A)$ or $\{a\}^* \cap \Sigma(A) = \emptyset$

Exercise 2.2.9: Let $A = (Q, \Sigma, \delta, q_0, \{q_f\})$ be a DFA such that for all $a \in \Sigma$ we have $\delta(q_0, a) = \delta(q_f, a)$

- Show that for all $w \neq \varepsilon$, we have $\delta(q_0, w) = \delta(q_f, w)$
- Show that for all $x \in L(A)$ with $x \neq \varepsilon$, we have $x^k \in L(A)$ for all $k > 0$. 
Non-determinism

Deterministic FA: \( \delta(q, a) \) is a unique state

\[ \Rightarrow \text{for each } w \in \Sigma^*, \text{ the execution is completely determined} \]

Non-deterministic FA (NFA): \( \delta(q, a) \) is a set of states

\[ \Rightarrow \text{may be the empty set} \]
\[ \Rightarrow \text{contain several states} \]

\[ \Rightarrow \text{multiple choices allow NFA to "guess" the right move.} \]

Accepts a string \( w \) if there is a sequence of guesses that leads to a final state.

Definition: an NFA is a quintuple \( A_N = (Q, \Sigma, \delta_N, q_0, F) \)

where \( Q, \Sigma, q_0, F \) are as for a DFA

\[ \delta_N \text{ is a total function} \]

\[ \delta_N : Q \times \Sigma \rightarrow 2^Q \]

(i.e. the powerset of \( Q \) (i.e. the set of all subsets of \( Q \))

\[ \text{i.e. } \delta(q, a) \text{ is a subset of } Q \]

Note: \( \delta(q, a) \) may be the empty set.

\[ \text{i.e. the NFA makes no transition on that input} \]

Definition: the extended transition function of an NFA \( A_N \)

is the function \( \hat{\delta}_N : Q \times \Sigma^* \rightarrow 2^Q \) defined as follows:

\[ \forall q \in Q, \forall a \in \Sigma, \forall w \in \Sigma^* \]

\( \hat{\delta}_N(q, \epsilon) = \{ q \} \)

\( \hat{\delta}_N(q, va) = \bigcup_{p \in \delta_N(q, v)} \hat{\delta}_N(p, a) \)

\[ \text{i.e. if } \hat{\delta}_N(q, va) = \{ q_1, \ldots, q_k \} \]

and \( \hat{\delta}_N(q, a) = \{ q_i \} \]

\[ \forall a \in \Sigma \]

then \( \hat{\delta}_N(q, \epsilon) = \phi \)

Definition: the language accepted by an NFA \( A_N \) is

\[ L(A_N) = \{ w \in \Sigma^* \mid \hat{\delta}_N(q_0, w) \cap F \neq \phi \} \]
Example: \( L_{a_2} = \{ w \mid w \text{ o n e b u t l e a st s y m b l } is 1 \} \)

Idea: NFA "guesses" the end of input using nondeterminism and looks for 10 or 11

\[ q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \]

(note: transitions from \( q_2 \) are all to \( \emptyset \))

Given an input string \( w \), we can represent the computation of \( A_{a_2} \) on \( w \) as a tree of possible executions (instead of a tree in a state-space)

Example: For input 0111

\[ q_0 \xrightarrow{0} q_0 \]
\[ q_0 \xrightarrow{1} q_1 \]
\[ q_0 \xrightarrow{1} q_1 \]
\[ q_0 \xrightarrow{0} q_1 \]
\[ q_1 \xrightarrow{1} q_2 \]
\[ q_2 \xrightarrow{1} q_2 \]

(Stuck)

The string 0111 is accepted, because \( \delta^*(q_0, 0111) \) contains at least one final state.

I.e., there is at least one execution path that ends in a final state.

For input 0101

\[ q_0 \xrightarrow{0} q_0 \]
\[ q_0 \xrightarrow{1} q_1 \]
\[ q_1 \xrightarrow{1} q_2 \]
\[ q_2 \xrightarrow{1} q_2 \]
\[ (\text{Stuck}) \]

The string 0101 is not accepted.

All execution paths either get stuck or end in a non-final state.
Different views of non-determinism:

1) The NFA always makes the right choices to ensure acceptance (if possible at all).

2) The NFA spawns off multiple copies at each non-deterministic choice point.

3) The NFA explores multiple paths in parallel.

Note: The various paths/computations evolve completely independently from each other.
(different e.g. from parallel computations which may synchronize at a certain point)

Exercise E.2.4: Give NFA's for the languages in Exercise 2.2.5.

Yesterday: 13/10/2004

Relationship between DFA's and NFAs:

Let $\mathcal{L}(\text{DFA})$ be the class of languages accepted by some DFA.

$\mathcal{L}(\text{NFA})$

What is the relationship between $\mathcal{L}(\text{DFA})$ and $\mathcal{L}(\text{NFA})$?

We show now that $\mathcal{L}(\text{DFA}) = \mathcal{L}(\text{NFA})$, i.e. DFA's and NFAs have the same expressive power.

We show the two directions separately.
Theorem: $L(DFA) \subseteq L(NFA)$

i.e., for every DFA $A_D$, there is an NFA $A_N$ such that $L(A_N) = L(A_D)$

Proof: Easy. Let $A_D = (Q, \Sigma, \delta_D, q_0, F)$ be a DFA.

We define an NFA $A_N = (Q, \Sigma, \delta_N, q_0, F)$, with

\[ \delta_N (q, a) = \{ p \} \]

if $\delta_D (q, a) = p$

(Informally: we view the DFA as an NFA)

We can show by induction on $|w|$ that if $\delta_D (q_0, w) = p$
then $\delta_N (q_0, w) = \{ p \}$. \[ \text{Exercize 2.3.5} \]

Since $A_D$ and $A_N$ coincide in the initial and final states, we get that $L(A_D) = L(A_N)$. q.e.d.

Theorem: $L(NFA) \subseteq L(DFA)$

i.e., for every NFA $A_N$ there is a DFA $A_D$ such that $L(A_D) = L(A_N)$

Idea for the construction of $A_D$:

$A_D$ simulates the entire execution tree of $A_N$ on one exec.

E.g.: $A_D$: $q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2$

$A_N$: $\{ q_0 \} \rightarrow \{ q_0 \} \rightarrow \{ q_0, q_1 \} \rightarrow \{ q_0, q_2 \} \rightarrow \{ q_0, q_3 \}$

=> A state in $A_N$ corresponds to a subset of $A_D$’s states.
Subset construction:

given \( A_N = (Q_N, \Sigma, \delta_N, q_0, F_N) \)

define \( A_D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D) \) with

- \( Q_D = 2^{Q_N} \)
- \( F_D = \{ S \subseteq Q_N \mid S \cap F_N \neq \emptyset \} \)
- \( \delta_D(S, \alpha) = \bigcup_{p \in S} \delta_N(p, \alpha) \)

i.e. \( \delta_D(S, \alpha) \) is the set of states of \( A_N \)

reachable in \( A_N \) via \( \alpha \) from some state in \( S \).

Example:

\[
A_N: \quad \xrightarrow{0} q_0 \xrightarrow{0,1} q_0 \xrightarrow{0,1} q_2
\]

\[
A_D:
\begin{cases}
\{q_0\} \quad \xrightarrow{1} \{q_0, q_1\} \\
\{q_0, q_1\} \quad \xrightarrow{1} \{q_0, q_1, q_2\} \\
\{q_0, q_2\} \quad \xrightarrow{0,1} \{q_0, q_2\}
\end{cases}
\]

\( \emptyset \) is a dead state: we cannot leave it

The computation is stuck.

Note: Some states cannot be reached from the start state

\( \Rightarrow \) can be eliminated.
We still have to show that for the DFA $A_D$ constructed from $A_N$ via the subset construction, we have $L(A_D) = L(A_N)$.

**Lemma:** \( \forall q \in Q_N, \forall w \in \Sigma^* \)
\[
\hat{\delta}_D(q, w) = \hat{\delta}_N(q, w)
\]

**Proof:** by induction on \(|w|\)

- **Base:** \(|w| = 0\), i.e., \(w = \varepsilon\)
\[
\hat{\delta}_D(\{q\}, \varepsilon) = \{q\} = \hat{\delta}_N(q, \varepsilon)
\]

- **Induction:** assume claim holds for \(|w| = m\)

  Show for \(|w| = m+1\)

Let \(w = \alpha \cdot \varepsilon\), with \(|\alpha| = m\), \(|w| = m+1\).

By induction hypothesis, we have \(\hat{\delta}_D(\{q\}, \alpha) = \hat{\delta}_N(q, \alpha)\)
\[
\hat{\delta}_D(\{q\}, w) = \hat{\delta}_D(\{q\}, \alpha \cdot \varepsilon) = \hat{\delta}_D(\hat{\delta}_D(\{q\}, \alpha), \varepsilon) = \hat{\delta}_D(\hat{\delta}_N(q, \alpha), \varepsilon) = \bigcup_{\bar{p} \in \hat{\delta}_N(q, \alpha)} \hat{\delta}_N(\bar{p}, \varepsilon) = \hat{\delta}_N(q, \alpha \cdot \varepsilon) = \hat{\delta}_N(q, w)
\]
We can finish now the proof that \( L(A_D) = L(A_N) \) \(^{2.13}\)
\[ L(A_D) = \{ w \in \Sigma^* \mid \delta_D(\{ q_0 \}, w) \in F_D \} = \text{def of } F_D \]
\[ = \{ w \in \Sigma^* \mid \delta_N(\{ q_0 \}, w) \cap F_N \neq \emptyset \} \]
\[ = \{ w \in \Sigma^* \mid \delta_N(q_0, w) \cap F_N \neq \emptyset \} \]
\[ = L(A_N) \]
q.e.d

Note: the DFA \( A_D \) obtained from an NFA \( A_N \) has in general a number of states that is exponential in the number of states of \( A_N \).

Can we do better? NO!

There are languages accepted by an NFA of \( n \) states, and for which the minimum size DFA has \( O(2^n) \) states.

Exercise E2.2: For \( k \geq 1 \), define an NFA \( A_N^k \), such that
\[ L(A_N^k) = \{ w \in \{ 0, 1 \}^* \mid \text{the } k\text{-th last symbol of } w \text{ is } 1 \} \]
Try to construct a DFA \( A_D^k \) s.t. \( L(A_D^k) = L(A_N^k) \) by applying the subset construction.
What are the numbers of states of \( A_N^k \) and \( A_D^k \)?

Exercise E2.3: For \( \Sigma = \{ e_1, \ldots, e_k \} \) construct an NFA \( A_N^e \), such that
\[ L(A_N^e) = \{ w \in \Sigma^* \mid w \text{ does not contain at least one of the symbols } e_1, \ldots, e_k \} \]
Try to construct an equivalent DFA \( A_D^e \).
What are the numbers of states of \( A_N^e \) and \( A_D^e \)?
Exercise 2.3.1: Convert the following NFA to a DFA

Exercise 2.3.4: Give NFA’s that accept the following languages:

a) The set of strings over \{0, ..., 5\} s.t. the final digit has appeared before

b) The set of strings over \{0, ..., 5\} s.t. the final digit has not appeared before
We add to NFA's *E-moves*

\[ 1 \xrightarrow{E} 0 \]

meaning: the automaton can do a transition without consuming an input symbol

**E-NFA** is as an NFA, but allowing also *E-moves*

**Example:**

\[
\begin{array}{c}
A_{0,1} \\
A_{1,0}
\end{array}
\]

strings that end in 01

strings that end in 10

We want an automaton accepting all strings that end either in 01 or in 10

**Note:** *E-moves are another form of non-determinism:*

the automaton can non-deterministically choose to change state

**Why are they useful?**

- useful descriptive tool (for specifications), to take into account "external" events
- useful for composing NFA's
- conversion to DFA's is still possible
**Definition:** An \( \varepsilon \)-NFA is a quintuple \( A_\varepsilon = (Q, \Sigma, \delta, q_0, F) \),

where \( Q, \Sigma, q_0, F \) are as for an NFA

and \( \delta: Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q \)

i.e., we may have: \( \delta(q, \varepsilon) = \{q_2, q_3, q_4\} \)

---

**\( \varepsilon \)-closure:**

For \( q \in Q \), \( \text{Eclose}(q) \) is the set of all states reachable from \( q \) using a sequence of \( \varepsilon \)-moves (including the empty sequence).

Can be defined inductively:

- \( q \in \text{Eclose}(q) \)
- If \( p \in \text{Eclose}(q) \) and \( p' \in \delta(p, \varepsilon) \), then \( p' \in \text{Eclose}(q) \)
- Nothing else is in \( \text{Eclose}(q) \)

**Note:** always \( q \in \text{Eclose}(q) \)

---

**Example:**

\[ \varepsilon \]

\[ q_0 \xrightarrow{\varepsilon} q_1 \xrightarrow{1} q_2 \xrightarrow{\varepsilon} q_3 \]

\( \text{Eclose}(q_0) = \{q_0, q_1, q_3\} \)

\( \text{Eclose}(q_1) = \{q_1, q_3\} \)

We can extend \( \text{Eclose} \) to sets of states: \( \text{Eclose}(S) = \bigcup_{q \in S} \text{Eclose}(q) \)

To define \( \hat{\delta} \), we have to take into account \( \text{Eclose} \):

- **basis:** \( \hat{\delta}(q, \varepsilon) = \text{Eclose}(q) \)
- **induction:** \( \hat{\delta}(q, x \cdot \varepsilon) = \text{Eclose} \left( \bigcup_{p \in \delta(q, x)} \hat{\delta}(q, x) \right) = \bigcup_{p \in \delta(q, x)} \text{Eclose}(\hat{\delta}(q, x)) \)

---
In more detail:

- Let $\hat{\delta}(q, k) = \{ \hat{q}_1, ..., \hat{q}_k \}$
- Let $U \delta(q_i, e) = \delta(q_1, e) \cup \cdots \cup \delta(q_n, e) = \hat{\delta}(q, k)$
- Then $\hat{\delta}(q, k, e) = \text{Eclose}(\{q_1, ..., q_n\})$

In other words: $\hat{\delta}(q, w)$ is the set of all states reachable from $q$ along paths whose labels on edges, apart from $\epsilon$, yield $w$

**Note:**
- $q \in \hat{\delta}(q, \epsilon)$
- $\delta(q, e) \neq \hat{\delta}(q, e)$ (different from DFA/NFA)

In fact $\hat{\delta}(q, e) = \text{Eclose}(U_{q_i \in \hat{\delta}(q, e)} \delta(q_i, e))$

**Example** (previous E-NFA)

\[
\hat{\delta}(q_0, \epsilon) = \{q_0, q_1, q_3\} \quad \delta(q_0, \epsilon) = \{q_1\}
\]

\[
\hat{\delta}(q_0, 1) = \text{Eclose}(U_{q_i \in \hat{\delta}(q_0, \epsilon)} \delta(q_i, 1)) = \text{Eclose}(\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_3, 1)) = \text{Eclose}(\emptyset \cup \{q_2\} \cup \{q_2, q_8\}) = \{q_2, q_8\}
\]

**Definition:** language accepted by an E-NFA $A_\epsilon$

$L(A_\epsilon) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$

**Theorem:** for each E-NFA $A_\epsilon$ there exists an NFA $A_\epsilon$ such that $L(A_\epsilon) = L(A_\epsilon)$

**Idea:** equivalent NFA has (almost) the same $Q$, $q_0$, and $F$, only $\delta_\epsilon$ is changed by removing $\epsilon$-moves and adding new moves instead.
Formally: Let $A_\varepsilon = (Q, \Sigma, \delta_\varepsilon, q_0, F)$ be an $\varepsilon$-NFA.

We construct the NFA $A_N = (Q, \Sigma, \delta_N, q_0, F)$ with

$\forall q \in Q, \ \forall a \in \Sigma$

$\delta_N(q, a) = \delta_\varepsilon(q, a) = \varepsilon\text{close}(\bigcup_{q_i \in \varepsilon\text{close}(q)} \delta(q_i, a))$

Note: $\delta_N(q, \varepsilon)$ is not defined (and it should not be).

**Example:**

![Diagram of NFA with states $q_0, q_1, q_2, q_3$ and transitions]

**Question:** Do we have that $L(A_N) = L(A_\varepsilon)$?

Yes, except possibly for $\varepsilon$.

In $A_\varepsilon$, we have that $\varepsilon \in L(A_\varepsilon)$ if $\varepsilon\text{close}(q_0) \cap F \neq \emptyset$

In $A_N$, we have that $\varepsilon \in L(A_N)$ if $q_0 \in F$

We have to adjust for that:

- make $q_0$ a final state of $A_N$ if in $A_\varepsilon$ $\varepsilon\text{close}(q_0) \cap F \neq \emptyset$

**Exercise E2.4** Prove that $L(A_N) = L(A_\varepsilon)$

**Note:** Combining the elimination of $\varepsilon$-transition with the subset construction, we can convert an $\varepsilon$-NFA to a DFA.

(Textbook provides a direct construction)