Finite state machines

- Finite automata:
  - simplest model of computation
  - describes so-called "regular languages"
  - works as follows:
    - is always in one of finitely many states
    - enters in some state
    - changes state in response to input
    - 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:
  \[
  \text{team 1} \, x-b \, \text{team 2} \, y-b \, \text{min:sec}
  \]
- list represented in HTML list:
  \[
  \langle \text{OL} \rangle \quad \text{ordered list}
  \langle \text{UL} \rangle \quad \text{unordered list}
  \langle \text{LI} \rangle \quad \text{list item}
  \]
  - accepts when it finds end of list

Example: \( \langle \text{OL} \rangle \)

\[
\langle \text{LI} \rangle \text{Rome - Lazio} \quad 2:0 \quad \langle /\text{LI} \rangle \\
\langle \text{LI} \rangle \text{Inter - Juve} \quad 10:2 \quad \langle /\text{LI} \rangle \\
\langle /\text{OL} \rangle
\]
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 a 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.

```
Solution: \( \Sigma = \{0, 1\} \)
\( L = \{ w \in \Sigma^* \mid w \) has substring 01 \} = \\
\{ \times 01 y \mid \times, y \in \Sigma^* \}
```

90 ... waiting for first 0
91 ... seen 0, waiting for 1
92 ... seen 01, waiting for rest of input
Note: DFA means read 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 a DFA: $L(A) = \{ w \in \Sigma^* | A \text{ accepts } w \}$

What we have seen are called Deterministic Finite Automata (DFAs)

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

- $Q$ ... 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
- as a transition table

\[
\begin{array}{c|cc}
\delta & 0 & 1 \\
\hline
q_0 & q_1 & q_0 \\
q_1 & q_1 & q_2 \\
q_2 & q_2 & q_2 \\
\end{array}
\]

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

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

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

denotes that starting at state $q$, portion $x$ of input string will take DFA to state $q$.

In other words: if $x = a_1 \ldots a_n$ 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:

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

Basis: \( \hat{\delta}(q, \varepsilon) = q \)

Induction: \( \hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a) \)

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

- $\varepsilon$ is a string
- if $x$ is a string and $a \in \Sigma$ then $ax$ is a string
- nothing else is a string

Example:

\[ \begin{array}{c}
\delta(q_0, \varepsilon) = q_0 \\
\delta(q_0, 1) = \delta(\delta(q_0, \varepsilon), 1) = \delta(q_0, 1) = q_0 \\
\delta(q_0, 10) = \delta(\delta(q_0, 1), 0) = \delta(q_0, 0) = q_1 \\
\delta(q_0, 101) = \delta(\delta(q_0, 10), 1) = \delta(q_1, 1) = q_2 \\
\end{array} \]
Language accepted by a DFA \( A = (Q, \Sigma, \delta, q_0, F) \)

Definition: \( L(A) = \{ w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F \} \)

Example:

![DFA Diagram]

What is \( L(A) \)?

Strings over \( \Sigma = \{0, 1\} \) that contain
an even number of 0's or
an even number of 1's

This DFA partitions the strings over \( \Sigma = \{0, 1\} \) in 4 equivalence
classes, depending on the parity of the numbers of 0's and 1's.

This is a general property:

- Each DFA partitions the strings into a finite number of
  equivalence classes, and conversely,
- Each partition of strings into a finite number of
  equivalence classes corresponds to a DFA

**Proof:**

\[
\hat{\delta}(q, a) = \delta(\delta(q, e), a) = \delta(q, a)
\]

Consequence: \( \delta \) and \( \hat{\delta} \) agree on strings of length 1.

Also, \( \hat{\delta} \) is defined only for strings of length 1.

Hence we can adopt the convention to call \( \hat{\delta} = \delta \).
Exercise 2.2.2: Prove that \( \forall q \in Q, \forall x, y \in \Sigma^* \):
\[
\hat{\delta}(q, x \cdot y) = \hat{\delta}(\hat{\delta}(q, x), y)
\]
Hint: use induction on |y|.

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

a) Each consecutive block of 5 symbols contains at least two 0's.

b) The 10th symbol from the right is a 1 (don't try to write down the whole DFA!)

c) The string either begins or ends (or both) with 01

d) The number of 0's is divisible by 5, and the number of 1's is divisible by 3.

Optional Exercises:

Exercise 2.2.8: Let \( A = (Q, \Sigma, \delta, q_0, F) \) be a DFA such that for some \( a \in \Sigma \) and all \( q \in Q \) we have \( \delta(q, a) = q \):

a) Show that for all \( n > 0 \), \( \hat{\delta}(q, a^n) = q \)

b) Show that either \( \{a\}^* \in L(A) \) or \( \{a\}^* \cap L(A) = \emptyset \)

Exercise 2.2.8: Let \( A = (Q, \Sigma, \delta, q_0, q_f) \) be a DFA such that for all \( x \in \Sigma \) we have \( \delta(q_0, x) = \delta(q_f, x) \):

a) Show that for all \( w \neq \varepsilon \), we have \( \hat{\delta}(q_0, w) = \hat{\delta}(q_f, w) \)

b) 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 F.A.: $\delta(q, a)$ is a unique state
- for each $w \in \Sigma^*$, the execution is completely determined

Non-deterministic F.A. (NFA): $\delta(q, a)$ is a set of states
- may be the empty set
- may contain several states

$\Rightarrow$ 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 quadruple $A_N = (Q, \Sigma, \delta_N, \epsilon, q_0, F)$

where: $Q, \Sigma, q_0, F$ as for a DFA
$
\delta_N$ is a total function

$\delta_N: Q \times \Sigma \rightarrow 2^Q$

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

i.e. $\delta_N(q, \alpha)$ is a subset of $Q$

Note: $\delta_N(q, \alpha)$ may be the empty set

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:

$\forall q \in Q, \forall \alpha \in \Sigma, \forall \epsilon \in \Sigma^*$

- $\hat{\delta}_N(q, \epsilon) = \{ q \}$
- $\hat{\delta}_N(q, \alpha \epsilon) = \bigcup \{ \delta_N(p, \alpha) | p \in \hat{\delta}_N(q, \epsilon) \}$

Definition: the language accepted by an NFA $A_N$ is
$\mathcal{L}(A_N) = \{ w \in \Sigma^* | \hat{\delta}_N(q_0, w) \cap F \neq \emptyset \}$
Example: $L_0 = \{ w \mid w$'s one but last symbol is $1 \}$

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

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

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

Given an input string $w$, we can represent the computation of $A_0$ on $w$ as a tree of possible executions (instead of a tree in a state-space)

e.g. for input 0111

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

for input 0101

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

(The string 0111 is accepted, because $\delta_n(q_0, 0111)$ contains at least one final state. D.e., there is at least one execution path that ends in a final state.)

(The string 0101 is not accepted. All execution paths either get stuck or end in a non-final state.)
Different kinds of non-determinism:

1) The NFA always makes the right choices to some 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 E2.1 Give NFA's for the languages in Exercise 2.2.5

6/10/2008

Relationship between DFA 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 \( \delta_N \) defined by the rule:
\[ \text{if } \delta_D(q, \sigma) = \emptyset \text{ then } \delta_N(q, \sigma) = \{ q \} \]
(Intuitively: we view the DFA as an NFA)

We can show by induction on \(|w|\) that if \( \delta_D(q_0, w) = \emptyset \) then \( \delta_N(q_0, w) = \{ q_0 \} \).

Exercise 2.3.5 (optional)

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 an execution.

\[ A_N: q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_1 \rightarrow q_1 \rightarrow q_2 \]

\[ A_D: \{ q_0 \} \rightarrow \{ q_0 \} \rightarrow \{ q_0, q_1 \} \rightarrow \{ q_0, q_1, q_2 \} \rightarrow \{ q_0, q_1, q_2 \} \rightarrow \{ q_0, q_1, q_2 \} \]

\( \Rightarrow \) A state in \( A_D \) corresponds to a subset of \( A_N \)'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

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

i.e. \( \delta_D(S, a) \) is the set of states of \( A_N \) reachable in \( A_N \) via \( a \) from some state in \( S \).

Example:

\[ A_N: \quad q_0 \xrightarrow{a} q_1 \xrightarrow{0,1} q_2 \]

\( \phi \) in 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 $\mathcal{L}(A_D) = \mathcal{L}(A_N)$.

**Optional part**

**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 $|w| = n$

**Show for $|w| = n + 1$**

Let $w = \varepsilon \cdot x$, with $|x| = m$, $|w| = n + 1$

By inductive hyp. we have $\hat{\delta}_D([q], x) = \hat{\delta}_N(q, x)$

$$\hat{\delta}_D([q], w) =$$

$$= \hat{\delta}_D([q], \varepsilon \cdot x)$$

$$= \hat{\delta}_D(\hat{\delta}_D([q], x), \varepsilon)$$

$$= \hat{\delta}_D(\hat{\delta}_N(q, x), \varepsilon)$$

$$= \cup_{r \in \hat{\delta}_N(q, x)} \delta_N(r, x)$$

$$= \hat{\delta}_N(q, \varepsilon \cdot x)$$

$$= \hat{\delta}_N(q, w)$$
We can finish now the proof that $L(A_D) = L(A_N)$.

$L(A_D) = \{ w \in \Sigma^* \mid \hat{\delta}_D(q_0, w) \in F_D \} = \text{def. of } F_D$

$= \{ w \in \Sigma^* \mid \hat{\delta}_D(q_0, w) \cap F_N \neq \emptyset \} = \text{lemma}

= \{ w \in \Sigma^* \mid \hat{\delta}_N(q_0, w) \cap F_N \neq \emptyset \} = \text{def. of } L(A_N)$

$= L(A_N)$

q.e.d.

End of optional part.

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

$L(A_N^k) = \{ w \in \Sigma_1 \Sigma_2^* \mid \text{the } k\text{-th last symbol of } w \text{ is } 1 \}$

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 $k \geq 1$, construct an NFA $A_N^k$ such that

$L(A_N^k) = \{ w \in \Sigma_1^+ \mid w \text{ does not contain at least one of the symbols } \Sigma_k \}$

Try to construct an equivalent DFA $A_D^k$.

What are the numbers of states of $A_N^k$ and $A_D^k$?
Exercise 2.3.1: Convert the following NFA to a DFA

Exercise 2.3.4: Give NFA's that accept the following language:

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

d) The set of strings over \{0, \ldots, 5\} s.t. the final digit has not appeared before
Finite automates with $\varepsilon$-transitions

We add to NFA's $\varepsilon$-moves

$\varepsilon$-NFA is as an NFA, but allowing also $\varepsilon$-moves

Example:

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

Note: $\varepsilon$-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 E-NFA is a quintuple \( A_E = (Q, \Sigma, \delta, q_0, F) \)
where \( Q, \Sigma, q_0, F \) are as for an NFA
and \( \delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q \)

-\( q_i \) \( \in \) \( \varepsilon \)-closure \( (q_i) \)

\( \varepsilon \)-closure: \( (q_i) \) \( \in \) \( \text{closure} \) \( (q_i) \)

-\( q_i \in \varepsilon \)-closure \( (q_i) \)
-\( \forall \) \( q' \in \varepsilon \)-closure \( (q_i) \) \( \rightarrow \) \( q' \in \delta(q_i, \varepsilon) \)
-\( q_i \in \varepsilon \)-closure \( (q_i) \)

Note: always \( q_i \in \varepsilon \)-closure \( (q_i) \)

Example:

\[ \begin{array}{ccccc}
& & q_0 & & \\
& \varepsilon & \downarrow & \varepsilon & \\
\longleftarrow q_0 & \rightarrow q_1 & \rightarrow q_2 & \rightarrow q_3 & \\
& \downarrow & \varepsilon & \downarrow & \\
& q_0 & \rightarrow q_1 & \rightarrow q_2 & \\
& \varepsilon & \downarrow & \varepsilon & \\
\end{array} \]

\( \varepsilon \)-closure \( (q_0) \) \( = \) \{ \( q_0, q_1, q_3 \) \}

\( \varepsilon \)-closure \( (q_1) \) \( = \) \{ \( q_1, q_3 \) \}

We can extend \( \varepsilon \)-closure to sets of states: \( \varepsilon \)-closure \( (S) \) \( = \) \( \bigcup \varepsilon \)-closure \( (q_i) \) \( \forall \) \( q_i \in S \)

To define \( \hat{\delta} \), we need to take into account \( \varepsilon \)-closure:

- basis: \( \hat{\delta}(q_i, \varepsilon) = \varepsilon \)-closure \( (q_i) \)
- induction: \( \hat{\delta}(q_i, x \cdot \varepsilon) = \varepsilon \)-closure \( (\bigcup_{q_j \in \hat{\delta}(q_i, x)} \varepsilon \)-closure \( (\delta(q_j, x)) \) \)

\( = \bigcup_{q_j \in \hat{\delta}(q_i, x)} \varepsilon \)-closure \( (\delta(q_j, x)) \)
In more detail:
- let $\hat{\delta}(q, k) = \{q_1, \ldots, q_n\}$
- let $\hat{\delta}(q, e) = \delta(q_1, e) \cup \ldots \cup \delta(q_n, e) = \{q_1, \ldots, q_n\}$
- then $\hat{\delta}(q, \epsilon) = \text{Eclose}(\{q_1, \ldots, q_n\})$

In other words: $\hat{\delta}(q, w)$ is the set of all states reachable from $q$ along paths whose labels on arcs, 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 $\hat{\delta}(q, e) = \text{Eclose}(\bigcup_{q_i \in \delta(q, e)} \delta(q_i, e))$

Example (previous \(E\)-NFA)
>$\hat{\delta}(q_0, \epsilon) = \{q_0, q_1, q_3\}$
>$\hat{\delta}(q_0, \epsilon) = \{q_1\}$
>$\hat{\delta}(q_0, 1) = \text{Eclose}(\bigcup_{q_i \in \delta(q_0, 1)} \delta(q_i, 1)) = \text{Eclose}(\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_3, 1))$
>\[= \text{Eclose}(\emptyset \cup \{q_0\} \cup \{q_3\}) = \{q_3\}\]

Definition: language accepted by an \(E\)-NFA \(A_{\hat{\delta}}\)
>$\hat{L}(A_{\hat{\delta}}) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$

13/11/2008

Theorem: For each \(E\)-NFA \(A_{\hat{\delta}}\) there exists an NFA \(A_N\) such that \(\hat{L}(A_{\hat{\delta}}) = \hat{L}(A_N)\)

Idea: equivalent NFA has (almost) the same $Q$, $q_0$, and $F$. Only $\delta$ is changed by removing $E$-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

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

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

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

---

**Example**

![Diagram of NFA](image)

---

**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 $A_N$: 

- if $q_0 \in F$

We have to adjust for that:

- make $q_0$ a final state of $A_N$

**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)