II.3.1. String Recognition (13.1)

II.3.2. Nondeterministic Finite State Automata (13.2)

II.3.3. Examples of Automata (13.3)

II.3.4. Automata with Empty Move Transitions (13.4)

II.3.5. Deterministic Finite State Automata (13.6)

Example Recognising Strings, Step 1

We want to define a program which recognises strings “start” and “stop”. We start by defining a program which recognises the letter “s”. This can be given by a system given by the following diagram, which will be our first automaton (automata will be introduced soon).

This automaton has the following ingredients.

- States \(q_0, q_1\).
- State \(q_0\) is the starting state, indicated by the arrow into it coming from nowhere.
- State \(q_1\), the state indicating that we have recognised letter “s”.
- A transition, which when recognising letter “s”, goes from state \(q_0\) to \(q_1\).
II.3.1. String Recognition (13.1)

Example Recognising Strings, Step 2

In order to recognise the letter “t”, we extend this automaton as follows:

- $q_0$ is the start state.
- $q_1$ indicates we have read string “s”.
- $q_2$ indicates we have read string “st”.

II.3.1. String Recognition (13.1)

Example Recognising Strings, Step 3

For the 3rd letter, we have two choices: “a” as part of the word “start”, and “o” as part of the word “stop”.

- $q_0$ is the start state.
- $q_1$ indicates we have read string “s”.
- $q_2$ indicates we have read string “st”.
- $q_3$ indicates we have read string “sta”.
- $q_6$ indicates we have read string “sto”.
Example Recognising Strings, Step 3

We can complete our automaton and obtain the following

This diagram contains a new ingredient: States $q_5$ and $q_7$ are accepting states. If we have processed a word and reached such a state then the word is accepted as a string of the language of the automaton.
(In our example $L = \{\text{start, stop}\}$.)

Automata with Loops

In order to recognise infinite languages, we need automata with loops. The following automaton recognises $L = \cdots$
II.3.1. String Recognition (13.1)

Automata with Loops

The following automaton recognises $L = \ldots$:

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} \]

Nondeterministic Automata

The language $\{ \text{start, stop} \}$ can be as well recognised by the following nondeterministic automaton:

\[ q_0 \xrightarrow{s} q_1 \xrightarrow{t} q_2 \xrightarrow{a} q_3 \xrightarrow{r} q_4 \xrightarrow{t} q_5 \]
\[ q_6 \xrightarrow{s} q_7 \xrightarrow{t} q_8 \xrightarrow{o} q_9 \xrightarrow{p} \]

- The automaton chooses in state $q_0$, non-deterministically, when in state $q_0$ and recognising a letter $s$, whether to go to $q_1$ or $q_6$.
- The accepted language is the set of strings such that for each of them we obtain an accepting state for at least one non-deterministic choice.

- If we try to accept the word “stop” by moving $q_0 \xrightarrow{s} q_1 \xrightarrow{t} q_2$ we get stuck at $q_2$.
- That we fail to accept a word for one specific non-deterministic choice doesn’t imply that this word is not in the language.
- A word is not accepted only if for all non-deterministic choices the corresponding run of the automaton doesn’t accept the string.
Why Nondeterministic Automata?

- When translating regular grammars into automata, we will obtain non-deterministic automata.
- We will show later that from a non-deterministic automaton we can obtain an equivalent deterministic automaton.
- Non-deterministic machine models play an important role in the theory of algorithms and complexity.
  - In some cases (as for automata), deterministic and non-deterministic are equivalent.
  - Sometimes they are not.
  - In other cases it is an open problem whether they are equivalent.
  - One example is the famous open “P = NP?” problem.
    Whether the problems recognised by deterministic (P) and non-deterministic (NP) Turing machines in polynomial time coincide or not.

Definition NFA

A **non-deterministic finite state automaton**, in short **NFA** \((Q, q_0, F, T, \rightarrow)\) is given by
- A finite set \(Q\) of **states**.
- A single **initial state** \(q_0\).
- A set \(F \subseteq Q\) of **accepting states**.
- A finite set of **terminal symbols** \(T\).
- \(\rightarrow\) is a relation between states \(q \in Q, a \in T\) and states \(q' \in Q\). We write \(q \xrightarrow{a} q'\) if \(q, a, q'\) are in this relation.

More formally \(\rightarrow\) is a subset of \(Q \times T \times Q\) and \(q \xrightarrow{a} q'\) is an abbreviation for \((q, a, q') \in \rightarrow\).

Presentation of NFA by Picture

An NFA can be presented by a picture, like the following diagram:

- The arrow from nowhere into state \(q_0\) denotes that \(q_0\) is the initial state.
- Circles denote states.
- Arrows from a state \(q\) to \(q'\) labelled by \(a\) mean that \(q \xrightarrow{a} q'\).
- Double circle like for \(q_5\) and \(q_9\) denote the accepting states.
- The alphabet needs to be stated, unless all elements of the alphabet occur as labels of arrows, or it is clear from the context.

Alphabet = \(\{a, \ldots, z\}\)
**Presentation of NFA by Table**

**automaton** \( A \)

**states** \( q_0, \ldots, q_n \)

**terminals** \( a_0, \ldots, a_m \)

**start** \( q_i \)

**final** \( q_{i_0}, q_{i_1}, \ldots, q_{i_m} \)

**transitions** \( q_{k_1} \xrightarrow{a_{i_k}} q_{m_1} \)

---

**Example**

The automaton for alphabet \( \{a, \ldots, z\} \) given by the diagram

![Diagram of automaton]

is represented by the table on the next slide.

---

**The Extended Transition Relation**

We define an extended transition relation which determines, whether the NFA from a state \( q \) when reading word \( w \in T^* \) can reach state \( q' \):

**Definition**

Let \( A = (Q, q_0, F, T, \xrightarrow{} ) \) be an NFA. Then we define \( \xrightarrow{}^* \) between states \( q \in Q \), words \( w \in T^* \) and \( q' \in Q \).

We write \( q \xrightarrow{w}^* q' \), if \( q, w, q' \) are in this relation.

- If \( w = a_0 \cdots a_n \), then

  \[
  q \xrightarrow{w}^* q' \text{ iff } q \xrightarrow{a_0} q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n = q' \text{ for some } q_i \in Q
  \]

Especially \( q \xrightarrow{\epsilon} q' \text{ iff } q' = q \).
The Extended Transition Relation

Again, more formally $\rightarrow^*$ is a subset of $Q \times T^* \times Q$ and $q \overset{w}{\rightarrow}^* q'$ means that $(q, w, q') \in \rightarrow^*$

Example

For the automaton

we give some definitions of $\rightarrow^*$:

The Language accepted by an NFA

Definition

Let $A = (Q, q_0, F, T, \rightarrow)$ be an NFA. The language accepted by $A$ is defined as

$$L(A) := \{ w \in T^* \mid q_0 \overset{w}{\rightarrow}^* q \text{ for some } q \in F \}$$
II.3.2. Nondeterministic Finite State Automata (13.2)

Operational Understanding of NFAs

To an NFA \( A = (Q, q_0, F, T, \rightarrow) \) corresponds a non-deterministic program for checking whether an input string is in \( L(A) \).

The program will have

- a variable \( q \in Q \),
- and a pointer to a position in the input string,
- a Boolean variable \( \text{stopped} \), which will be true if the automaton has stopped without having reached the end of the input string.

A run of the NFA on an input string \( s \) is as given by the pseudo-code on the next slide:

\[
\begin{align*}
{\text{Initialisation}} & \quad q := q_0; \\
& \quad p := \text{beginning of word } s; \\
& \quad \text{stopped} := \text{false}; \\
\text{while } (p \neq \text{end of word } s) \land \neg \text{stopped} \text{ do } & \quad \{ a := \text{next symbol in } s \text{ at position } p; \\
& \quad \text{if } (\neg \exists q'. (q \xrightarrow{a} q')) \\
& \quad \text{then } \{ \text{stopped} := \text{true}; \} \\
& \quad \text{else } \{ \text{choose } q' \text{ s.t. } q \xrightarrow{a} q'; \\
& \quad \quad q := q'; \\
& \quad \quad p := \text{position of next symbol in } s \text{ from position } p; \} \}
\end{align*}
\]

If at the end of the execution \( \text{stopped} = \text{true} \) (therefore we have not reached the end of the string), the string is not accepted in this run.

Otherwise the string is accepted if \( q \in F \).

A string is accepted if there exist a sequence of non-deterministic choices, such that the string is accepted in the corresponding run.

One can now see that a string \( s \) is accepted by some run of the program for automaton \( A \) iff \( s \in L(A) \).
II.3.1. String Recognition (13.1)

II.3.2. Nondeterministic Finite State Automata (13.2)

II.3.3. Examples of Automata (13.3)

II.3.4. Automata with Empty Move Transitions (13.4)

II.3.5. Deterministic Finite State Automata (13.6)

Example 1: An Automaton accepting \{1, 2, 3\}

Display style:

<table>
<thead>
<tr>
<th>automaton</th>
<th>( A^{1,2,3} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>states</td>
<td>( q_0, q_1, q_2, q_3 )</td>
</tr>
<tr>
<td>terminals</td>
<td>1, 2, 3</td>
</tr>
<tr>
<td>start</td>
<td>( q_0 )</td>
</tr>
<tr>
<td>final</td>
<td>( q_1, q_2, q_3 )</td>
</tr>
</tbody>
</table>
| transitions     | \( q_0 \xrightarrow{1} q_1 \)  
|                 | \( q_0 \xrightarrow{2} q_2 \)  
|                 | \( q_0 \xrightarrow{3} q_3 \)  |

Simplified Version
II.3.3. Examples of Automata (13.3)

Simplified Version

Shorter pictorial presentation of the same automaton:

Example 2: Automaton accepting ???

Display style:

assign automaton $A^{1,2,3'}$

assign states $q_0, q_1$

assign terminals $1, 2, 3$

assign start $q_0$

assign final $q_1$

assign transitions $q_0 \xrightarrow{0} q_1$

assign transitions $q_0 \xrightarrow{1} q_1$

assign transitions $q_0 \xrightarrow{2} q_1$
Example 3: Automaton accepting $0, 1, \ldots, 9$

Display Style

<table>
<thead>
<tr>
<th>automaton</th>
<th>$A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>states</td>
<td>$q_0, q_1, q_2$</td>
</tr>
<tr>
<td>terminals</td>
<td>$0, \ldots, 9$</td>
</tr>
<tr>
<td>start</td>
<td>$q_0$</td>
</tr>
<tr>
<td>final</td>
<td>$q_1, q_2$</td>
</tr>
</tbody>
</table>
| transitions   | $q_0 \xrightarrow{0} q_2$
|               | $q_0 \xrightarrow{a} q_1$, ($a \in \{1, \ldots, 9\}$) |
|               | $q_1 \xrightarrow{a} q_1$, ($a \in \{0, \ldots, 9\}$) |

The material for this section has been moved to Additional Material
II.3.5. Deterministic Finite State Automata (13.6)

**Definition DFA**

Let $A = (Q, q_0, F, T, \rightarrow)$ be an NFA. $A$ is a **deterministic finite state automaton**, in short **DFA**, if for all $q \in Q$, there exist at most one $q'$ s.t. $q \xrightarrow{a} q'$

So deterministic finite state automata are those automata corresponding to real programs: we have never to make a choice.

**Example (DFA)**

The following NFA is a DFA:

The following NFA is not a DFA:

Note that there is no transition from $q_0$ labelled by letter $\neq t$.
For a DFA from each state and element of the alphabet there need to be at most one transition – it is possible to have no transition.
Theorem II.3.5.1.

Let \( A \) be an NFA. Then there exists a DFA \( A' \) s.t. \( L(A) = L(A') \). \( A' \) can be computed from \( A \).

Definition of the DFA

- We define a DFA as follows:
  - The states are all possible sets of states.
    - Note that since the set of states is finite, the set of sets of states is finite, too.
  - The initial state is the set of initial states, here \( \{ q_0 \} \).

Example

Consider the NFA above which was not an DFA:

![Diagram](image-url)
II.3.5. Deterministic Finite State Automata (13.6)

Definition of the DFA

- $\emptyset$ is not an accepting state, and $\emptyset \xrightarrow{\cdot} \emptyset$.
  - So $\emptyset$ is a sink of the DFA. It's non-accepting and all transitions from it go back to it (why?). So we can't escape from it (why?).
  - We can omit it and all transitions into it.
- We can as well omit all states which are unreachable from the initial state.
- So we get a (usually much smaller) automaton, by starting from the initial state $\{q_0\}$, and successively determine all transitions of the DFA which don't end up in $\emptyset$.

Example above Repeated

Corresponding DFA

Using the technique above we obtain the following automaton:

Note that this was the original DFA above (up to renaming of states).

Example 2

Consider the following NFA accepting $L = \ldots$:
Corresponding DFA

Proof Idea

- We denote the transitions of the NFA by $\rightarrow$ and and those of the DFA by $\rightarrow'$.
- One can easily see that in the DFA we have that
  $$\{q_0\} w^* \rightarrow' \{q_1, \ldots, q_n\} =: B$$
  iff
  $$B = \{q \mid q_0 \xrightarrow{w^*} q\}$$
  so $B$ is the set of states which we can reach from $q_0$ by a $w$-transition.
- Therefore in the DFA we reach an accepting state (one containing an element of $F$) by a $w$-transition, iff
  the NFA can reach an accepting state by a $w$-transition.
- Therefore the language accepted by this DFA is the same as the language accepted by the original NFA.
- A formal proof can be found in the Additional Material.