Extended transition function of a DFA

The next two pages describe the extended transition function of a DFA in a more detailed way than Handout 3.
We define the **extended transition function** \( \hat{\delta} \). It takes a state \( q \) and an input string \( w \) to the resulting state. The definition proceeds by **induction** over the length of \( w \).

- **Induction basis** (\( w \) has length 0): in this case, \( w \) is the **empty string**, i.e. the string of length 0, for which we write \( \epsilon \). We define

\[
\hat{\delta}(q, \epsilon) = q.
\]
Induction step (from length $l$ to length $l + 1$): in this case, $w$, which has length $l + 1$, is of the form $va$, where $v$ is a string of length $l$ and $a$ is a symbol. We define

$$\hat{\delta}(q, va) = \delta(\hat{\delta}(q, v), a).$$

This works because, by induction hypothesis, $\hat{\delta}(q, v)$ is already defined.
Non-deterministic finite automata (NFA’s)
Non-deterministic FA (NFA)

- An NFA is like a DFA, except that it can be in several states at once.
- This can be seen as the ability to guess something about the input.
- Useful for searching texts.
NFA: example

An NFA accepting all strings that end in 01:

It is non-deterministic because input 0 in state $q_0$ can lead to both $q_0$ and $q_1$. 
The example on the next few pages is a corrected version of the *wrong* example in Handout 3!
Using the NFA

Suppose the input string is 100101. The NFA starts in state $q_0$, as indicated by the token.
Using the NFA

The remaining input string is 100101. The NFA reads the first symbol, 1. It remains in state $q_0$. 
Using the NFA

The remaining input string is 00101. The NFA reads the next symbol, 0. The resulting possible states are $q_0$ or $q_1$. 
Using the NFA

The remaining input string is $0101$. The NFA reads the next symbol, $0$. The resulting possible states are still $q_0$ or $q_1$. 

![NFA Diagram]

- $q_0$ is the start state.
- $q_1$ and $q_2$ are possible states.
- Transitions are labeled with symbols: $0$ and $1$.
Using the NFA

The remaining input string is 101. The NFA reads the next symbol, 1. The resulting possible states are $q_0$ and $q_2$. (Because $q_2$ is a final states, this means that the word so far, 1001, would be accepted.)
The remaining input string is 01. The NFA reads the next symbol, 0. There is no transition for 0 from $q_2$, so the token on $q_2$ dies. The resulting possible states are $q_0$ or $q_1$. 
Using the NFA

The remaining input string is 1. The NFA reads the next symbol, 1. The possible states are $q_0$ and $q_2$. Because $q_2$ is final, the NFA accepts the word, 100101.
Formal definition of NFA

Definition. A non-deterministic finite automaton (NFA) consists of

- a finite set of **states**, often denoted \( Q \),
- a finite set \( \Sigma \) of **input symbols**, 
- a **transition function** \( \delta : Q \times \Sigma \rightarrow P(Q) \),
- a **start state** \( q_0 \in Q \), and
- a set \( F \subseteq Q \) of **final** or **accepting states**.
Difference between NFA and DFA

Suppose that \( q \) is a state and \( a \) is an input symbol.

- In a DFA, we have \( \delta(q, a) \in Q \), that is, \( \delta(q, a) \) is a state.

- In a NFA, we have \( \delta(q, a) \in P(Q) \), that is, \( \delta(q, a) \) is a set of states; it can be seen as the possible states that can result from input \( a \) in state \( q \).
Formal approach to accepted strings

- We are aiming to describe the language $L(A)$ accepted by a NFA $A$.
- This description is similar to the DFA case, but a bit more sophisticated.
- As in the DFA case, we first define the extended transition function:
  \[ \hat{\delta} : Q \times \Sigma \rightarrow P(Q). \]
- That function $\hat{\delta}$ will be used to define $L(A)$. 
Before reading any symbols, the set of possible states is $\hat{\delta}(q_0, \epsilon) = \{q_0\}$. 
Example of $\hat{\delta}$ (input string: 100101)

We have $\hat{\delta}(q_0, 1) = \{q_0\}$. 
Example of $\hat{\delta}$ (input string: 100101)

We have $\hat{\delta}(q_0, 10) = \{q_0, q_1\}$. 
We have $\hat{\delta}(q_0, 100) = \{q_0, q_1\}$. 

Example of $\hat{\delta}$ (input string: 100101)
Example of $\hat{\delta}$ (input string: 100101)

We have $\hat{\delta}(q_0, 1001) = \{q_0, q_2\}$. 
Example of $\hat{\delta}$ (input string: 100101)

We have $\hat{\delta}(q_0, 10010) = \{q_0, q_1\}$. 
Example of $\hat{\delta}$ (input string: 100101)

We have $\hat{\delta}(q_0, 100101) = \{q_0, q_2\}$. Because $\{q_0, q_2\} \cap F = \{q_0, q_2\} \cap \{q_2\} = \{q_2\} \neq \emptyset$, the NFA accepts.
Formal definition of $\hat{\delta}$

**Definition.** The extended transition function $\hat{\delta} : Q \times \Sigma \rightarrow P(Q)$ of an NFA is defined inductively as follows:

- **Induction basis (length 0):**
  \[
  \hat{\delta}(q, \epsilon) = \{q\}
  \]

- **Induction step (from length $l$ to length $l+1$):**
  \[
  \hat{\delta}(q, va) = \bigcup_{q' \in \hat{\delta}(q, v)} \delta(q', a).
  \]
The language of an NFA

Intuitively, the language of a DFA $A$ is the set of strings $w$ that lead from the start state to an accepting possible state.

Formally, the language $L(A)$ accepted by the FA $A$ is defined as follows:

$$L(A) = \{w | \hat{\delta}(q_0, w) \cap F \neq \emptyset\}.$$
Exercise

Give NFA to accept the following languages.

1. The set of strings over an alphabet \( \{0, 1, \ldots, 9\} \) such that the final digit has appeared before.

2. The set of strings over an alphabet \( \{0, 1, \ldots, 9\} \) such that the final digit has \textbf{not} appeared before.

3. The set of strings of 0’s and 1’s such that there are two 0’s separated by a number of positions that is a multiple of 4.
DFA’s and NFA’s

- Evidently, DFA’s are precisely those NFA’s for which the set of states $\delta(q, a)$ has exactly one element for all $q$ and $a$.

- So, trivially, every language accepted by a DFA is also accepted by some NFA.

- Is every language accepted by an NFA also accepted by some DFA?

- Surprisingly, the answer is “yes”!
Simulation of an NFA by a DFA

Let $N = (Q_N, \Sigma, \delta_N, q_0^N, F_N)$ be a NFA. The equivalent DFA $D$ is obtained from the so-called powerset construction (also called “subset construction”). We define

$$D = (Q_D, \Sigma, \delta_D, q_0^D, F_D),$$

where...
Simulation of an NFA by a DFA

- The alphabet of $D$ is that of $N$.
- The states of $D$ are sets of states of $N$:
  \[ Q_D = P(Q_N) \]
- The initial state $q_0^D$ of $D$ is $\{q_0^N\}$. 
Simulation of an NFA by a DFA

- The final states of $D$ are those sets that contain the final state of $N$:

$$F_D = \{ S \in P(Q_N) \mid S \cap F_N \neq \emptyset \}$$

- The transition function of $D$ arises from the transition function of $N$ as follows:

$$\delta_D(S, a) = \bigcup_{q' \in S} \delta_N(q', a)$$
### Example of powerset construction: table

<table>
<thead>
<tr>
<th>NFA</th>
<th>DFA</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="NFA Diagram" /></td>
<td><img src="#" alt="DFA Table" /></td>
</tr>
</tbody>
</table>

#### NFA Diagram
- **Start**: $q_0$
- **States**: $q_0, q_1, q_2$
- **Transitions**:
  - From $q_0$ on input 0, transition to $q_1$.
  - From $q_1$ on input 0, transition to $q_0$.
  - From $q_1$ on input 1, transition to $q_2$.
  - From $q_2$ on input 0, transition to $q_0$.
  - From $q_2$ on input 1, transition to $q_1$.

#### DFA Table
<table>
<thead>
<tr>
<th>Transition</th>
<th>DFA State</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>${q_0}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>${q_1}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>${q_2}$</td>
</tr>
<tr>
<td>${q_2}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>${q_0, q_1}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0, q_2}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_2}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>${q_1, q_2}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>${q_2}$</td>
</tr>
<tr>
<td>${q_0, q_1, q_2}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0, q_2}$</td>
<td>${q_0}$</td>
</tr>
</tbody>
</table>
Example of powerset construction: graph

Transition graph of the resulting DFA.
Example of powerset construction: graph

Optionally, we can remove the unreachable states of the DFA.
Proposition about the simulation

**Proposition.** For every NFA $N$, there is a DFA $D$ such that $L(D) = L(N)$. 
First, we show that for every string \( w \) we have \[
\delta_D(\{q_0\}, w) = \delta_N(q_0, w) \quad (1)
\]

We proceed by induction on the length \( l \) of \( w \).

- **Base case** \( (l = 0) \): in this case, \( w \) is the empty string, \( \varepsilon \). We have

\[
\delta_D(\{q_0\}, \varepsilon) = \{q_0\} \quad \text{(by defn. of } \delta_D) \\
= \delta_N(q_0, \varepsilon) \quad \text{(by defn. of } \delta_N).
\]
Proof of the proposition (part 2/3)

- Induction step (from $l$ to $l + 1$): in this case, $w$, which is of length $l + 1$, is of the form $va$, where $v$ is a string of length $l$ and $a$ is a symbol. We have

$$\widehat{\delta_D}(\{q_0\}, va) = \delta_D(\widehat{\delta_D}(\{q_0\}, v), a) \quad \text{(by defn. of } \widehat{\delta_D}\text{)}$$

$$= \delta_D(\widehat{\delta_N}(q_0, v), a) \quad \text{(by indn. hypoth.)}$$

$$= \bigcup_{q' \in \widehat{\delta_N}(q_0, v)} \delta_N(q', a) \quad \text{(by defn. of } \delta_D\text{)}$$

$$= \widehat{\delta_N}(q_0, va) \quad \text{(by defn. of } \widehat{\delta_N}\text{).}$$
Proof of the proposition (part 3/3)

Finally, we use Equation ??, which we just proved, to prove that the languages of $D$ and $N$ are equal:

\[
\begin{align*}
  w \in L(D) &\iff \widehat{\delta}_D(\{q_0\}, w) \in F_D & \text{(by defn. of $L(D)$)} \\
  &\iff \widehat{\delta}_N(q_0, w) \in F_D & \text{(by Equation ??)} \\
  &\iff \widehat{\delta}_N(q_0, w) \cap F_N \neq \emptyset & \text{(by defn. of $F_D$)} \\
  &\iff w \in L(N) & \text{(by defn. of $L(N)$)}.
\end{align*}
\]
The proposition implies:

**Corollary.** A language $L$ is accepted by some DFA if and only if $L$ is accepted by some NFA.

**Proof.** $\Rightarrow$: this is the powerset construction we have just seen.

$\Leftarrow$: this is true because every DFA is a special case of an NFA, as observed earlier.
Warning

- Let $N$ be an NFA, and let $D$ be the DFA that arises from the powerset construction.
- As we have seen, we have $Q_D = P(Q_N)$.
- So, if $Q_N$ has size $k$, then the size of $Q_D$ is $2^k$.
- This exponential growth of the number of states makes the powerset construction unusable in practice.
- It can be shown that removing unreachable states does not prevent this exponential growth.
Exercise

Convert the following NFA to a DFA:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>→p</td>
<td>{p, q}</td>
</tr>
<tr>
<td>q</td>
<td>{r}</td>
</tr>
<tr>
<td>r</td>
<td>{s}</td>
</tr>
<tr>
<td>*s</td>
<td>{s}</td>
</tr>
</tbody>
</table>
Exercise

Convert the following NFA to a DFA:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>→ p</td>
<td>{q, s}</td>
<td>{q}</td>
</tr>
<tr>
<td>*q</td>
<td>{r}</td>
<td>{q, r}</td>
</tr>
<tr>
<td>r</td>
<td>{s}</td>
<td>{p}</td>
</tr>
<tr>
<td>*s</td>
<td>{}</td>
<td>{p}</td>
</tr>
</tbody>
</table>
Exercise

Convert the following NFA to a DFA:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>→ p</td>
<td>{p, q}</td>
<td>{p}</td>
</tr>
<tr>
<td>q</td>
<td>{r, s}</td>
<td>{t}</td>
</tr>
<tr>
<td>r</td>
<td>{p, r}</td>
<td>{t}</td>
</tr>
<tr>
<td>*s</td>
<td>{}</td>
<td>{}</td>
</tr>
<tr>
<td>*t</td>
<td>{}</td>
<td>{}</td>
</tr>
</tbody>
</table>

Describe informally the language accepted by this NFA accept? (Don’t worry if you need tutor’s help for this.)