Script **generated by TTT**

Title: Seidl: Theoretische_Informatik (19.04.2012)

Date: Thu Apr 19 16:14:01 CEST 2012

Duration: 90:40 min

Pages: 80

Satz 1.8
Die Menge der Sprachen über einem nicht-leeren Alphabet ist überabzählbar.

Satz 1.8
Die Menge der Sprachen über einem nicht-leeren Alphabet ist überabzählbar.

Beweis:
Indirekt. Nimm an, die Menge der Sprachen sei abzählbar.

L_0	
L_1	
L_2	
$\vdots$	

Satz 1.8
Die Menge der Sprachen über einem nicht-leeren Alphabet ist überabzählbar.

Beweis:
Indirekt. Nimm an, die Menge der Sprachen sei abzählbar.

	w_0	w_1	w_2	$\ldots$	$= \Sigma^*$
L_0					
L_1					
L_2					
$\vdots$					

Satz 1.8

Die Menge der Sprachen über einem nicht-leeren Alphabet ist überabzählbar.

Beweis:

Indirekt. Nimm an, die Menge der Sprachen sei abzählbar.

	w_0	w_1	w_2	...	$= \Sigma^*$
L_0	0	1	1	...	
L_1					
L_2					
$\vdots$					

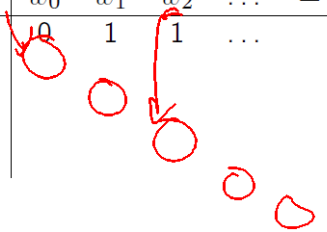
Satz 1.8

Die Menge der Sprachen über einem nicht-leeren Alphabet ist überabzählbar.

Beweis:

Indirekt. Nimm an, die Menge der Sprachen sei abzählbar.

	w_0	w_1	w_2	...	$= \Sigma^*$
L_0	0	1	1	...	
L_1					
L_2					
$\vdots$					



Satz 1.8

Die Menge der Sprachen über einem nicht-leeren Alphabet ist überabzählbar.

Beweis:

Indirekt. Nimm an, die Menge der Sprachen sei abzählbar.

	w_0	w_1	w_2	...	$= \Sigma^*$
L_0	0	1	1	...	
L_1	1	0	0	...	
L_2	1	1	1	...	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	

Eine Sprache L, die nicht in der Tabelle ist: ¬Diagonale!

Satz 1.8

Die Menge der Sprachen über einem nicht-leeren Alphabet ist überabzählbar.

Beweis:

Indirekt. Nimm an, die Menge der Sprachen sei abzählbar.

	w_0	w_1	w_2	...	$= \Sigma^*$
L_0	0	1	1	...	
L_1	1	0	0	...	
L_2	1	1	1	...	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	

Eine Sprache L, die nicht in der Tabelle ist: ¬Diagonale!

L	1	1	0	...

Definition 1.9

Eine Menge M ist entscheidbar gdw es einen Algorithmus gibt, der ihre charakteristische Funktion

$$\chi_M(x) = \begin{cases} 1 & \text{falls } x \in M \\ 0 & \text{falls } x \notin M \end{cases}$$

berechnet.

Beispiel 1.10 (Entscheidbare Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)

Beispiel 1.10 (Entscheidbare Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- $\{(ab)^n \mid n \in \mathbb{N}\}$

Beispiel 1.10 (Entscheidbare Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- $\{(ab)^n \mid n \in \mathbb{N}\}$
- $\{a^n b^n \mid n \in \mathbb{N}\}$

Beispiel 1.10 (Entscheidbare Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- $\{(ab)^n \mid n \in \mathbb{N}\}$
- $\{a^n b^n \mid n \in \mathbb{N}\}$
- Die Menge aller Primzahlen (dezimal kodiert)

Beispiel 1.10 (Entscheidbare Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- $\{(ab)^n \mid n \in \mathbb{N}\}$
- $\{a^n b^n \mid n \in \mathbb{N}\}$
- Die Menge aller Primzahlen (dezimal kodiert)
- $\emptyset$

Beispiel 1.10 (Entscheidbare Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- $\{(ab)^n \mid n \in \mathbb{N}\}$
- $\{a^n b^n \mid n \in \mathbb{N}\}$
- Die Menge aller Primzahlen (dezimal kodiert)
- $\emptyset$
- $\{\epsilon\}$

Satz 1.11

Es gibt nicht-entscheidbare Sprachen.

Satz 1.11
Es gibt nicht-entscheidbare Sprachen.

Beweis:
Jeder Algorithmus ist ein Wort.

Satz 1.11
Es gibt nicht-entscheidbare Sprachen.

Beweis:
Jeder Algorithmus ist ein Wort.
Daher gibt es nur abzählbar viele Algorithmen.

Satz 1.11
Es gibt nicht-entscheidbare Sprachen.

Beweis:
Jeder Algorithmus ist ein Wort.
Daher gibt es nur abzählbar viele Algorithmen.
Aber überabzählbar viele Sprachen, also mehr als Algorithmen. □

Satz 1.11
Es gibt nicht-entscheidbare Sprachen.

Beweis:
Jeder Algorithmus ist ein Wort.
Daher gibt es nur abzählbar viele Algorithmen.
Aber überabzählbar viele Sprachen, also mehr als Algorithmen. □

Wir haben sogar gezeigt:

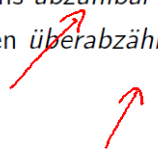
- Es gibt höchstens *abzählbar viele entscheidbare Sprachen.*
- Damit verbleiben *überabzählbar viele unentscheidbare Sprachen.*

Satz 1.11
Es gibt nicht-entscheidbare Sprachen.

Beweis:
Jeder Algorithmus ist ein Wort.
Daher gibt es nur abzählbar viele Algorithmen.
Aber überabzählbar viele Sprachen, also mehr als Algorithmen. □

Wir haben sogar gezeigt:

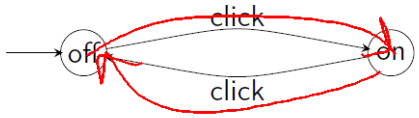
- Es gibt höchstens *abzählbar viele entscheidbare Sprachen*.
- Damit verbleiben *überabzählbar viele unentscheidbare Sprachen*.



2. Reguläre Sprachen

2.1 Deterministische endliche Automaten

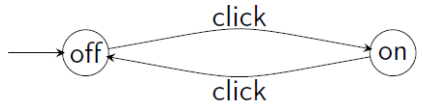
Ein Schalter:



2. Reguläre Sprachen

2.1 Deterministische endliche Automaten

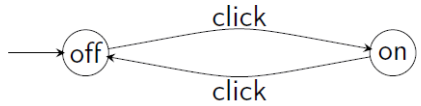
Ein Schalter:



UML state charts sind endliche Automaten.

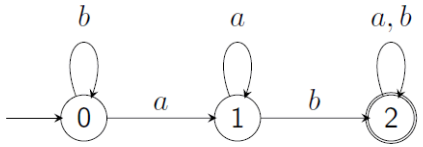
2. Reguläre Sprachen

2.1 Deterministische endliche Automaten

Ein Schalter: 

UML state charts sind endliche Automaten.

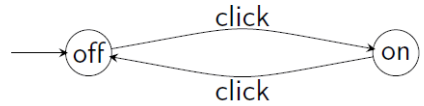
Ein endlicher Automat mit Endzustand:



Definition 2.1
Ein deterministischer endlicher Automat (*deterministic finite automaton*, DFA) $M = (Q, \Sigma, \delta, q_0, F)$ besteht aus

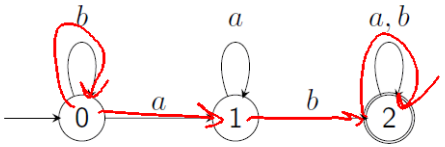
2. Reguläre Sprachen

2.1 Deterministische endliche Automaten

Ein Schalter: 

UML state charts sind endliche Automaten.

Ein endlicher Automat mit Endzustand:



Eingabewort $baba \rightsquigarrow$ Zustandsfolge 0,0,1,2,2.

Definition 2.1
Ein deterministischer endlicher Automat (*deterministic finite automaton*, DFA) $M = (Q, \Sigma, \delta, q_0, F)$ besteht aus

- einer endlichen Menge von Zuständen Q,

Definition 2.1
Ein deterministischer endlicher Automat (*deterministic finite automaton*, DFA) $M = (Q, \Sigma, \delta, q_0, F)$ besteht aus

- einer endlichen Menge von Zuständen Q,
- einem (endlichen) Eingabealphabet Σ,
- einer (totalen!) Übergangsfunktion $\delta : Q \times \Sigma \to Q$,

Definition 2.1
Ein deterministischer endlicher Automat (*deterministic finite automaton*, DFA) $M = (Q, \Sigma, \delta, q_0, F)$ besteht aus

- einer endlichen Menge von Zuständen Q,
- einem (endlichen) Eingabealphabet Σ,
- einer (totalen!) Übergangsfunktion $\delta : Q \times \Sigma \to Q$,
- einem Startzustand $q_0 \in Q$, und
- einer Menge von Endzuständen $F \subseteq Q$.

Definition 2.1
Ein deterministischer endlicher Automat (*deterministic finite automaton*, DFA) $M = (Q, \Sigma, \delta, q_0, F)$ besteht aus

- einer endlichen Menge von Zuständen Q,
- einem (endlichen) Eingabealphabet Σ,
- einer (totalen!) Übergangsfunktion $\delta : Q \times \Sigma \to Q$,
- einem Startzustand $q_0 \in Q$, und
- einer Menge von Endzuständen $F \subseteq Q$.

Endzustände heißen auch akzeptierenden Zustände.

Definition 2.1
Ein deterministischer endlicher Automat (*deterministic finite automaton*, DFA) $M = (Q, \Sigma, \delta, q_0, F)$ besteht aus

- einer endlichen Menge von Zuständen Q,
- einem (endlichen) Eingabealphabet Σ,
- einer (totalen!) Übergangsfunktion $\delta : Q \times \Sigma \to Q$,
- einem Startzustand $q_0 \in Q$, und
- einer Menge von Endzuständen $F \subseteq Q$.

Endzustände heißen auch akzeptierenden Zustände.
Die von M akzeptierte Sprache ist

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

Definition 2.1
Ein deterministischer endlicher Automat (*deterministic finite automaton*, DFA) $M = (Q, \Sigma, \delta, q_0, F)$ besteht aus

- einer endlichen Menge von Zuständen Q,
- einem (endlichen) Eingabealphabet Σ,
- einer (totalen!) Übergangsfunktion $\delta : \ Q \times \Sigma \to Q$,
- einem Startzustand $q_0 \in Q$, und
- einer Menge von Endzuständen $F \subseteq Q$.

Endzustände heißen auch akzeptierenden Zustände.
Die von M akzeptierte Sprache ist

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

wobei $\hat{\delta} : Q \times \Sigma^* \to Q$ rekursiv definiert ist durch

$$\begin{aligned} \hat{\delta}(q, \epsilon) &= q \\ \hat{\delta}(q, aw) &= \hat{\delta}(\delta(q, a), w) \quad \text{für } a \in \Sigma, w \in \Sigma^* \end{aligned}$$

Definition 2.2
Eine Sprache ist regulär gdw sie von einem DFA akzeptiert wird.

Offensichtlich gilt:

Fakt 2.3
Reguläre Sprachen sind entscheidbar.

Definition 2.2
Eine Sprache ist regulär gdw sie von einem DFA akzeptiert wird.

Definition 2.1
Ein deterministischer endlicher Automat (*deterministic finite automaton*, DFA) $M = (Q, \Sigma, \delta, q_0, F)$ besteht aus

- einer endlichen Menge von Zuständen Q,
- einem (endlichen) Eingabealphabet Σ,
- einer (totalen!) Übergangsfunktion $\delta : \ Q \times \Sigma \to Q$,
- einem Startzustand $q_0 \in Q$, und
- einer Menge von Endzuständen $F \subseteq Q$.

Endzustände heißen auch akzeptierenden Zustände.
Die von M akzeptierte Sprache ist

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

wobei $\hat{\delta} : Q \times \Sigma^* \to Q$ rekursiv definiert ist durch

$$\begin{aligned} \hat{\delta}(q, \epsilon) &= q \\ \hat{\delta}(q, aw) &= \hat{\delta}(\delta(q, a), w) \quad \text{für } a \in \Sigma, w \in \Sigma^* \end{aligned}$$

Definition 2.2
Eine Sprache ist regulär gdw sie von einem DFA akzeptiert wird.

Offensichtlich gilt:

Fakt 2.3
Reguläre Sprachen sind entscheidbar.

Definition 2.1
Ein deterministischer endlicher Automat (*deterministic finite automaton*, DFA) $M = (Q, \Sigma, \delta, q_0, F)$ besteht aus

- einer endlichen Menge von Zuständen Q,
- einem (endlichen) Eingabealphabet Σ,
- einer (totalen!) Übergangsfunktion $\delta : Q \times \Sigma \to Q$,
- einem Startzustand $q_0 \in Q$, und
- einer Menge von Endzuständen $F \subseteq Q$.

Endzustände heißen auch akzeptierenden Zustände.
Die von M akzeptierte Sprache ist

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

wobei $\hat{\delta} : Q \times \Sigma^* \to Q$ rekursiv definiert ist durch

$$\begin{aligned} \hat{\delta}(q, \epsilon) &= q \\ \hat{\delta}(q, aw) &= \hat{\delta}(\delta(q, a), w) \quad \text{für } a \in \Sigma, w \in \Sigma^* \end{aligned}$$

Bemerkungen

- Endliche Automaten können durch gerichtete und markierte Zustandsgraphen veranschaulicht werden:

Definition 2.2
Eine Sprache ist regulär gdw sie von einem DFA akzeptiert wird.

Offensichtlich gilt:

Fakt 2.3
Reguläre Sprachen sind entscheidbar.

Bemerkungen

- Endliche Automaten können durch gerichtete und markierte Zustandsgraphen veranschaulicht werden:
  - Knoten $\widehat{=}$ Zuständen

Bemerkungen

- Endliche Automaten können durch gerichtete und markierte Zustandsgraphen veranschaulicht werden:
  - Knoten $\widehat{=}$ Zuständen
  - Kanten $\widehat{=}$ Übergängen: $p \xrightarrow{a} q \;\widehat{=}\; \delta(p,a) = q$

  Der Anfangszustand wird durch einen Pfeil, Endzustände werden durch doppelte Kreise gekennzeichnet.

- Die Operation $\hat{\cdot}$ ist ein Funktional. In OCaml:

```
let rec dach f q = function
    [] -> q |
    a::w -> dach f (f q a) w
```

Bemerkungen

- Endliche Automaten können durch gerichtete und markierte Zustandsgraphen veranschaulicht werden:
  - Knoten $\widehat{=}$ Zuständen
  - Kanten $\widehat{=}$ Übergängen: $p \xrightarrow{a} q \;\widehat{=}\; \delta(p,a) = q$

  Der Anfangszustand wird durch einen Pfeil, Endzustände werden durch doppelte Kreise gekennzeichnet.

- Die Operation $\hat{\cdot}$ ist ein Funktional. In OCaml:

```
let rec dach f q = function
    [] -> q |
    a::w -> dach f (f q a) w
```

  D.h. `dach` = `fold_left` !

Bemerkungen

- Endliche Automaten können durch gerichtete und markierte Zustandsgraphen veranschaulicht werden:
  - Knoten $\widehat{=}$ Zuständen
  - Kanten $\widehat{=}$ Übergängen: $p \xrightarrow{a} q \;\widehat{=}\; \delta(p,a) = q$

  Der Anfangszustand wird durch einen Pfeil, Endzustände werden durch doppelte Kreise gekennzeichnet.

- Die Operation $\hat{\cdot}$ ist ein Funktional. In OCaml:

```
let rec dach f q = function
    [] -> q |
    a::w -> dach f (f q a) w
```

  D.h. `dach` = `fold_left` !

- Die Erweiterung von f auf $\hat{f}$ funktioniert für beliebige $f : A \times B \to A$

Bemerkungen

- Endliche Automaten können durch gerichtete und markierte Zustandsgraphen veranschaulicht werden:
  - Knoten $\widehat{=}$ Zuständen
  - Kanten $\widehat{=}$ Übergängen: $p \xrightarrow{a} q \ \widehat{=}\ \delta(p,a) = q$

  Der Anfangszustand wird durch einen Pfeil, Endzustände werden durch doppelte Kreise gekennzeichnet.

- Die Operation $\hat{\cdot}$ ist ein Funktional. In OCaml:

```
let rec dach f q = function
    [] -> q |
    a::w -> dach f (f q a) w
```

  D.h. `dach` = `fold_left` !

- Die Erweiterung von f auf $\hat{f}$ funktioniert für beliebige $f : A \times B \to A$

- Für $a \in \Sigma$ gilt $\hat{\delta}(q,a)$

Das Beweisprinzip Induktion über die Länge von Wörtern hat folgenden häufigen Spezialfall:

- Basis: Zeige die Behauptung für $w = \epsilon$

Bemerkungen

- Endliche Automaten können durch gerichtete und markierte Zustandsgraphen veranschaulicht werden:
  - Knoten $\widehat{=}$ Zuständen
  - Kanten $\widehat{=}$ Übergängen: $p \xrightarrow{a} q \ \widehat{=}\ \delta(p,a) = q$

  Der Anfangszustand wird durch einen Pfeil, Endzustände werden durch doppelte Kreise gekennzeichnet.

- Die Operation $\hat{\cdot}$ ist ein Funktional. In OCaml:

```
let rec dach f q = function
    [] -> q |
    a::w -> dach f (f q a) w
```

  D.h. `dach` = `fold_left` !

- Die Erweiterung von f auf $\hat{f}$ funktioniert für beliebige $f : A \times B \to A$

- Für $a \in \Sigma$ gilt $\hat{\delta}(q,a) = \hat{\delta}(q,a\epsilon) = \hat{\delta}(\delta(q,a),\epsilon) = \delta(q,a)$

Das Beweisprinzip Induktion über die Länge von Wörtern hat folgenden häufigen Spezialfall:

- Basis: Zeige die Behauptung für $w = \epsilon$

- Induktionsannahme: Die Behauptung stimmt für w.

Das Beweisprinzip Induktion über die Länge von Wörtern hat folgenden häufigen Spezialfall:

- Basis: Zeige die Behauptung für $w = \epsilon$
- Induktionsannahme: Die Behauptung stimmt für w.
  Schritt: Zeige die Behauptung für aw, $a \in \Sigma$.

Das Beweisprinzip Induktion über die Länge von Wörtern hat folgenden häufigen Spezialfall:

- Basis: Zeige die Behauptung für $w = \epsilon$
- Induktionsannahme: Die Behauptung stimmt für w.
  Schritt: Zeige die Behauptung für aw, $a \in \Sigma$.

Lemma 2.4
Für $x \in \Sigma$ und $w \in \Sigma^*$ gilt $\hat{\delta}(q, wx) = \delta(\hat{\delta}(q, w), x)$

$\hat{\delta}(q, aw) =$

$\hat{\delta}(\delta(q, a), w)$

Das Beweisprinzip Induktion über die Länge von Wörtern hat folgenden häufigen Spezialfall:

- Basis: Zeige die Behauptung für $w = \epsilon$
- Induktionsannahme: Die Behauptung stimmt für w.
  Schritt: Zeige die Behauptung für aw, $a \in \Sigma$.
  Alternative: Zeige die Behauptung für wa, $a \in \Sigma$.

Lemma 2.4
Für $x \in \Sigma$ *und* $w \in \Sigma^*$ *gilt* $\hat{\delta}(q, wx) = \delta(\hat{\delta}(q, w), x)$

Lemma 2.4
Für $x \in \Sigma$ *und* $w \in \Sigma^*$ *gilt* $\hat{\delta}(q, wx) = \delta(\hat{\delta}(q, w), x)$

Beweis:
Mit Induktion über die Länge von w:

- Basis: $w = \epsilon$:
  $\hat{\delta}(q, \epsilon x) =$

Lemma 2.4
Für $x \in \Sigma$ *und* $w \in \Sigma^*$ *gilt* $\hat{\delta}(q, wx) = \delta(\hat{\delta}(q, w), x)$

Beweis:
Mit Induktion über die Länge von w:

- Basis: $w = \epsilon$:
  $\hat{\delta}(q, \epsilon x) = \hat{\delta}(q, x\epsilon) = \hat{\delta}(\delta(q, x), \epsilon) =$

Lemma 2.4
Für $x \in \Sigma$ *und* $w \in \Sigma^*$ *gilt* $\hat{\delta}(q, wx) = \delta(\hat{\delta}(q, w), x)$

Beweis:
Mit Induktion über die Länge von w:

- Basis: $w = \epsilon$:
  $\hat{\delta}(q, \epsilon x) = \hat{\delta}(q, x\epsilon) = \hat{\delta}(\delta(q, x), \epsilon) = \delta(q, x) =$

Lemma 2.4

Für $x \in \Sigma$ und $w \in \Sigma^*$ gilt $\hat{\delta}(q, wx) = \delta(\hat{\delta}(q, w), x)$

Beweis:

Mit Induktion über die Länge von w:

- Basis: $w = \epsilon$:
  $\hat{\delta}(q, \epsilon x) = \hat{\delta}(q, x\epsilon) = \hat{\delta}(\delta(q, x), \epsilon) = \delta(q, x) = \delta(\hat{\delta}(q, \epsilon), x)$

Lemma 2.4

Für $x \in \Sigma$ und $w \in \Sigma^*$ gilt $\hat{\delta}(q, wx) = \delta(\hat{\delta}(q, w), x)$

Beweis:

Mit Induktion über die Länge von w:

- Basis: $w = \epsilon$:
  $\hat{\delta}(q, \epsilon x) = \hat{\delta}(q, x\epsilon) = \hat{\delta}(\delta(q, x), \epsilon) = \delta(q, x) = \delta(\hat{\delta}(q, \epsilon), x)$

- Schritt:

$$\hat{\delta}(q, awx)$$

Lemma 2.4

Für $x \in \Sigma$ und $w \in \Sigma^*$ gilt $\hat{\delta}(q, wx) = \delta(\hat{\delta}(q, w), x)$

Beweis:

Mit Induktion über die Länge von w:

- Basis: $w = \epsilon$:
  $\hat{\delta}(q, \epsilon x) = \hat{\delta}(q, x\epsilon) = \hat{\delta}(\delta(q, x), \epsilon) = \delta(q, x) = \delta(\hat{\delta}(q, \epsilon), x)$
- Schritt:

$$\begin{aligned} &\hat{\delta}(q, awx) \\ &= \hat{\delta}(\delta(q, a), wx) && \text{Def. von } \hat{\delta} \end{aligned}$$

Lemma 2.4

Für $x \in \Sigma$ und $w \in \Sigma^*$ gilt $\hat{\delta}(q, wx) = \delta(\hat{\delta}(q, w), x)$

Beweis:

Mit Induktion über die Länge von w:

- Basis: $w = \epsilon$:
  $\hat{\delta}(q, \epsilon x) = \hat{\delta}(q, x\epsilon) = \hat{\delta}(\delta(q, x), \epsilon) = \delta(q, x) = \delta(\hat{\delta}(q, \epsilon), x)$
- Schritt:

$$\begin{aligned} &\hat{\delta}(q, awx) \\ &= \hat{\delta}(\delta(q, a), wx) && \text{Def. von } \hat{\delta} \\ &= \delta(\hat{\delta}(\delta(q, a), w), x) && \text{Ind.Hyp.} \end{aligned}$$

Lemma 2.4

Für $x \in \Sigma$ und $w \in \Sigma^*$ gilt $\hat{\delta}(q, wx) = \delta(\hat{\delta}(q, w), x)$

Beweis:

Mit Induktion über die Länge von w:

- Basis: $w = \epsilon$:
  $\hat{\delta}(q, \epsilon x) = \hat{\delta}(q, x\epsilon) = \hat{\delta}(\delta(q, x), \epsilon) = \delta(q, x) = \delta(\hat{\delta}(q, \epsilon), x)$
- Schritt:

$$\begin{aligned} & \hat{\delta}(q, awx) \\ &= \hat{\delta}(\delta(q, a), wx) && \text{Def. von } \hat{\delta} \\ &= \delta(\hat{\delta}(\delta(q, a), w), x) && \text{Ind.Hyp.} \\ &= \delta(\hat{\delta}(q, aw), x) && \text{Def. von } \hat{\delta} \end{aligned}$$

□

Lemma 2.4

Für $x \in \Sigma$ und $w \in \Sigma^*$ gilt $\hat{\delta}(q, wx) = \delta(\hat{\delta}(q, w), x)$

Beweis:

Mit Induktion über die Länge von w:

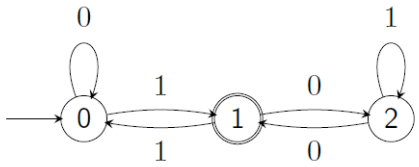
- Basis: $w = \epsilon$:
  $\hat{\delta}(q, \epsilon x) = \hat{\delta}(q, x\epsilon) = \hat{\delta}(\delta(q, x), \epsilon) = \delta(q, x) = \delta(\hat{\delta}(q, \epsilon), x)$
- Schritt:

$$\begin{aligned} & \hat{\delta}(q, awx) \\ &= \hat{\delta}(\delta(q, a), wx) && \text{Def. von } \hat{\delta} \\ &= \delta(\hat{\delta}(\delta(q, a), w), x) && \text{Ind.Hyp.} \\ &= \delta(\hat{\delta}(q, aw), x) && \text{Def. von } \hat{\delta} \end{aligned}$$

□

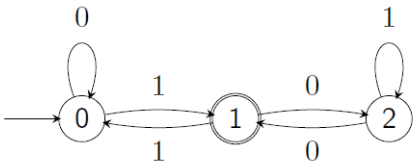
Beispiel 2.5

Ein DFA M:



Beispiel 2.5

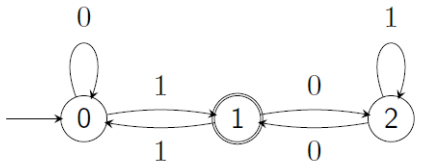
Ein DFA M:



Für $w \in \{0, 1\}^*$ sei $\#w$ die von w binär repräsentierte Zahl, zB $\#100 = 4$.

Beispiel 2.5

Ein DFA M:



Für $w \in \{0,1\}^*$ sei $\#w$ die von w binär repräsentierte Zahl, zB $\#100 = 4$.

$$L(M) = \{w \mid \#w \bmod 3 = 1\}$$

$w = \varepsilon \quad \hat{\delta}(0,\varepsilon) = 0$ ✓

Beweis:

1. $\delta(q,b) =$ $(q \cdot 2 + b) \,\%\, 3$
2. $\hat{\delta}(0,w) =$ $(\#w) \,\%\, 3$

$\left| \begin{array}{l} \hat{\delta}(0,wb) = \\ \delta(\hat{\delta}(0,w),b) \end{array} \right.$ □

2.2 Nichtdeterministische endliche Automaten

Verallgemeinerung: $\delta : Q \times \Sigma \to \mathcal{P}(Q)$

Finde geschlossenen numerischen Ausdruck für

$$\hat{\delta}(q,w)$$

$= \delta(\hat{\delta}(q,w),b)$
$= \delta(\#w \,\%\, 3, b)$
$= ((\#w \,\%\, 3) * 2 + b) \,\%\, 3$
$= (\#(wb)) \,\%\, 3$

2.2 Nichtdeterministische endliche Automaten

Verallgemeinerung: $\delta : Q \times \Sigma \to \mathcal{P}(Q)$

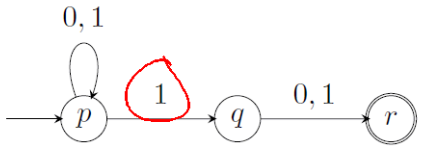
[$\mathcal{P}(Q)$ = Potenzmenge = Menge aller Teilmengen = 2^Q]

2.2 Nichtdeterministische endliche Automaten

Verallgemeinerung: $\delta : Q \times \Sigma \to \mathcal{P}(Q)$
[$\mathcal{P}(Q)$ = Potenzmenge = Menge aller Teilmengen = 2^Q]

Beispiel 2.6

Erkennung der Binärzahlen, deren vorlezte Ziffer 1 ist:

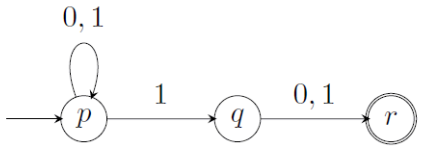


2.2 Nichtdeterministische endliche Automaten

Verallgemeinerung: $\delta : Q \times \Sigma \to \mathcal{P}(Q)$
[$\mathcal{P}(Q)$ = Potenzmenge = Menge aller Teilmengen = 2^Q]

Beispiel 2.6

Erkennung der Binärzahlen, deren vorlezte Ziffer 1 ist:



Wort wird akzepiert gdw es einen Weg zum Endzustand gibt.

2.2 Nichtdeterministische endliche Automaten

Verallgemeinerung: $\delta : Q \times \Sigma \to \mathcal{P}(Q)$
[$\mathcal{P}(Q)$ = Potenzmenge = Menge aller Teilmengen = 2^Q]

Definition 2.7

Ein nichtdeterministischer endlicher Automat (*nondeterministic finite automaton*, NFA) ist ein 5-Tupel $N = (Q, \Sigma, \delta, q_0, F)$, so dass

Definition 2.7

Ein nichtdeterministischer endlicher Automat (*nondeterministic finite automaton*, NFA) ist ein 5-Tupel $N = (Q, \Sigma, \delta, q_0, F)$, so dass

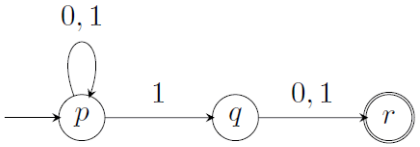
- Q, Σ, q_0 und F sind wie bei einem DFA
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$

Definition 2.7

Ein nichtdeterministischer endlicher Automat (*nondeterministic finite automaton*, NFA) ist ein 5-Tupel $N = (Q, \Sigma, \delta, q_0, F)$, so dass

- Q, Σ, q_0 und F sind wie bei einem DFA
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$

Beispiel



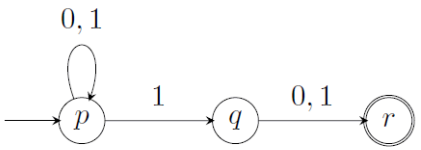
δ	0	1
p	$\{p\}$	$\{p, q\}$
q	$\{r\}$	$\{r\}$
r	$\emptyset$	$\emptyset$

Definition 2.7

Ein nichtdeterministischer endlicher Automat (*nondeterministic finite automaton*, NFA) ist ein 5-Tupel $N = (Q, \Sigma, \delta, q_0, F)$, so dass

- Q, Σ, q_0 und F sind wie bei einem DFA
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$

Beispiel



δ	0	1
p	$\{p\}$	$\{p, q\}$
q	$\{r\}$	$\{r\}$
r	$\emptyset$	$\emptyset$

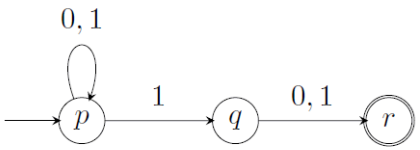
Alternative: Relation $\delta \subseteq Q \times \Sigma \times Q$

Definition 2.7

Ein nichtdeterministischer endlicher Automat (*nondeterministic finite automaton*, NFA) ist ein 5-Tupel $N = (Q, \Sigma, \delta, q_0, F)$, so dass

- Q, Σ, q_0 und F sind wie bei einem DFA
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$

Beispiel



δ	0	1
p	$\{p\}$	$\{p, q\}$
q	$\{r\}$	$\{r\}$
r	$\emptyset$	$\emptyset$

Alternative: Relation $\delta \subseteq Q \times \Sigma \times Q$

Erweiterung von $\delta : Q \times \Sigma \to \mathcal{P}(Q)$
auf $\bar{\delta} : \mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q)$:

Erweiterung von $\delta : Q \times \Sigma \to \mathcal{P}(Q)$
auf $\bar{\delta} : \mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q)$:

$$\bar{\delta}(S, a) := \bigcup_{q \in S} \delta(q, a)$$

Erweiterung von $\delta : Q \times \Sigma \to \mathcal{P}(Q)$
auf $\bar{\delta} : \mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q)$:

$$\bar{\delta}(S, a) := \bigcup_{q \in S} \delta(q, a)$$

$\Rightarrow \hat{\bar{\delta}} : \mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q)$

Erweiterung von $\delta : Q \times \Sigma \to \mathcal{P}(Q)$
auf $\bar{\delta} : \mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q)$:

$$\bar{\delta}(S, a) := \bigcup_{q \in S} \delta(q, a)$$

$\Rightarrow \hat{\bar{\delta}} : \mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q)$

Intuition:

$\hat{\bar{\delta}}(S, w)$ ist Menge aller Zustände,
die sich von einem Zustand in S aus mit w erreichen lassen.

Die von $N = (Q, \Sigma, \delta, q_0, F)$ akzeptierte Sprache ist

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