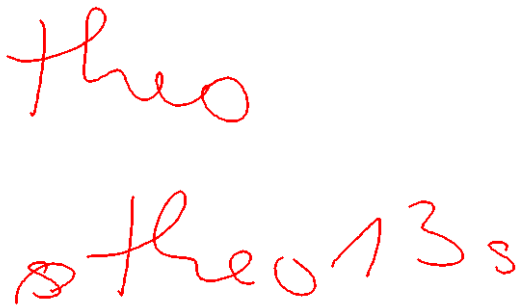
Script generated by TTT

Title: Seidl: Theoretische_Informatik (22.04.2013)

Date: Mon Apr 22 10:16:14 CEST 2013

Duration: 89:18 min

Pages: 78



Beweis:
Sei $N = (Q, \Sigma, \delta, q_0, F)$ ein NFA.
Definiere den DFA $M = (\mathcal{P}(Q), \Sigma, \bar{\delta}, \{q_0\}, F_M)$:

$$F_M := \{S \subseteq Q \mid S \cap F \neq \emptyset\}$$

Dann gilt:

$$\begin{array}{lcll} w \in L(N) & \Leftrightarrow & \hat{\bar{\delta}}(\{q_0\}, w) \cap F \neq \emptyset & \text{Def.} \\ & \Leftrightarrow & \hat{\bar{\delta}}(\{q_0\}, w) \in F_M & \text{Def.} \\ & \Leftrightarrow & w \in L(M) & \text{Def.} \end{array}$$ □

Dies nennt man die Potenzmengen- oder Teilmengenkonstruktion.

Beweis:
Sei $N = (Q, \Sigma, \delta, q_0, F)$ ein NFA.
Definiere den DFA $M = (\mathcal{P}(Q), \Sigma, \bar{\delta}, \{q_0\}, F_M)$:

$$F_M := \{S \subseteq Q \mid S \cap F \neq \emptyset\}$$

Dann gilt:

$$\begin{array}{lcll} w \in L(N) & \Leftrightarrow & \hat{\bar{\delta}}(\{q_0\}, w) \cap F \neq \emptyset & \text{Def.} \\ & \Leftrightarrow & \hat{\bar{\delta}}(\{q_0\}, w) \in F_M & \text{Def.} \\ & \Leftrightarrow & w \in L(M) & \text{Def.} \end{array}$$ □

Dies nennt man die Potenzmengen- oder Teilmengenkonstruktion.

Beweis:
Sei $N = (Q, \Sigma, \delta, q_0, F)$ ein NFA.
Definiere den DFA $M = (\mathcal{P}(Q), \Sigma, \bar{\delta}, \{q_0\}, F_M)$:

$$F_M := \{S \subseteq Q \mid S \cap F \neq \emptyset\}$$

Dann gilt:

$$\begin{aligned} w \in L(N) &\Leftrightarrow \hat{\bar{\delta}}(\{q_0\}, w) \cap F \neq \emptyset && \text{Def.} \\ &\Leftrightarrow \hat{\bar{\delta}}(\{q_0\}, w) \in F_M && \text{Def.} \\ &\Leftrightarrow w \in L(M) && \text{Def.} \end{aligned}$$

□

Warum NFAs?

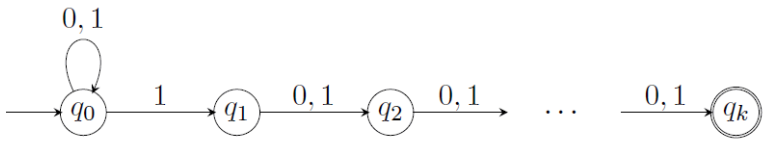
Mit NFAs lassen sich reguläre Sprachen
(u.U. exponentiell) kompakter darstellen.

Beweis:
Sei $N = (Q, \Sigma, \delta, q_0, F)$ ein NFA.

Beispiel 2.9

$$L_k := \{w \in \{0,1\}^* \mid \text{das } k\text{-letzte Bit von } w \text{ ist } 1\}$$

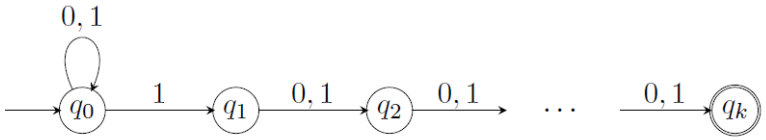
Ein NFA für diese Sprache ist gegeben durch:



Beispiel 2.9

$$L_k := \{w \in \{0,1\}^* \mid \text{das } k\text{-letzte Bit von } w \text{ ist 1}\}$$

Ein NFA für diese Sprache ist gegeben durch:

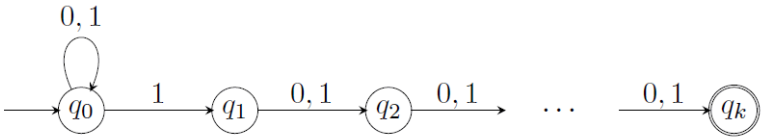


Die Potenzmengenkonstruktion liefert einen DFA für L_k mit 2^{k+1} Zuständen.

Beispiel 2.9

$$L_k := \{w \in \{0,1\}^* \mid \text{das } k\text{-letzte Bit von } w \text{ ist 1}\}$$

Ein NFA für diese Sprache ist gegeben durch:



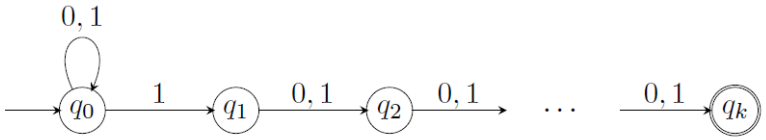
Die Potenzmengenkonstruktion liefert einen DFA für L_k mit 2^{k+1} Zuständen.

Geht es kompakter?

Beispiel 2.9

$$L_k := \{w \in \{0,1\}^* \mid \text{das } k\text{-letzte Bit von } w \text{ ist 1}\}$$

Ein NFA für diese Sprache ist gegeben durch:



Die Potenzmengenkonstruktion liefert einen DFA für L_k mit 2^{k+1} Zuständen. Geht es kompakter?

Lemma 2.10

Jeder DFA M mit $L(M) = L_k$ hat mindestens 2^k Zuständen.

Beweis:

Indirekt. Sei M ein DFA mit $< 2^k$ Zuständen, so dass $L(M) = L_k$.

Beweis:
Indirekt. Sei M ein DFA mit $< 2^k$ Zuständen, so dass $L(M) = L_k$.

- Es gibt $w_1, w_2 \in \{0,1\}^k$ mit $w_1 \neq w_2$ aber $\hat{\delta}(q_0, w_1) = \hat{\delta}(q_0, w_2)$

□

Beweis:
Indirekt. Sei M ein DFA mit $< 2^k$ Zuständen, so dass $L(M) = L_k$.

- Es gibt $w_1, w_2 \in \{0,1\}^k$ mit $w_1 \neq w_2$ aber $\hat{\delta}(q_0, w_1) = \hat{\delta}(q_0, w_2)$
- Sei $w_1 = wa_i \dots a_k$ und $w_2 = wb_i \dots b_k$ mit $a_i \neq b_i$

□

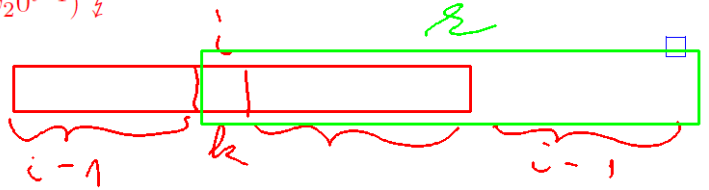
Beweis:
Indirekt. Sei M ein DFA mit $< 2^k$ Zuständen, so dass $L(M) = L_k$.

- Es gibt $w_1, w_2 \in \{0,1\}^k$ mit $w_1 \neq w_2$ aber $\hat{\delta}(q_0, w_1) = \hat{\delta}(q_0, w_2)$
- Sei $w_1 = wa_i \dots a_k$ und $w_2 = wb_i \dots b_k$ mit $a_i \neq b_i$
- OE sei $a_i = 1$, $b_i = 0$
- $w_1 0^{i-1} = wa_i \dots a_k 0^{i-1} \in L_k$ und $w_2 0^{i-1} = wb_i \dots b_k 0^{i-1} \notin L_k$

□

Beweis:
Indirekt. Sei M ein DFA mit $< 2^k$ Zuständen, so dass $L(M) = L_k$.

- Es gibt $w_1, w_2 \in \{0,1\}^k$ mit $w_1 \neq w_2$ aber $\hat{\delta}(q_0, w_1) = \hat{\delta}(q_0, w_2)$
- Sei $w_1 = wa_i \dots a_k$ und $w_2 = wb_i \dots b_k$ mit $a_i \neq b_i$
- OE sei $a_i = 1$, $b_i = 0$
- $w_1 0^{i-1} = wa_i \dots a_k 0^{i-1} \in L_k$ und $w_2 0^{i-1} = wb_i \dots b_k 0^{i-1} \notin L_k$
- $\hat{\delta}(q_0, w_1 0^{i-1}) = \hat{\delta}(\hat{\delta}(q_0, w_1), 0^{i-1}) = \hat{\delta}(\hat{\delta}(q_0, w_2), 0^{i-1}) = \hat{\delta}(q_0, w_2 0^{i-1})$ ↯



□

2.4 NFAs mit ϵ-Übergängen

Definition 2.11
Ein NFA mit ϵ-Übergängen (auch ϵ-NFA) ist ein NFA mit einem speziellen Symbol $\epsilon \notin \Sigma$ und mit

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q) .$$

2.4 NFAs mit ϵ-Übergängen

Definition 2.11
Ein NFA mit ϵ-Übergängen (auch ϵ-NFA) ist ein NFA mit einem speziellen Symbol $\epsilon \notin \Sigma$ und mit

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q) .$$

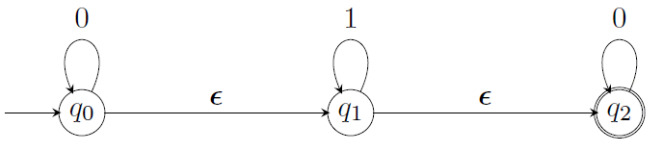
Ein ϵ-Übergang darf ausgeführt werden, ohne dass ein Eingabezeichen gelesen wird.

2.4 NFAs mit ϵ-Übergängen

Definition 2.11
Ein NFA mit ϵ-Übergängen (auch ϵ-NFA) ist ein NFA mit einem speziellen Symbol $\epsilon \notin \Sigma$ und mit

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q) .$$

Ein ϵ-Übergang darf ausgeführt werden, ohne dass ein Eingabezeichen gelesen wird.

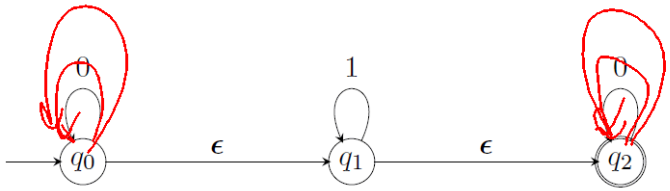


2.4 NFAs mit ϵ-Übergängen

Definition 2.11
Ein NFA mit ϵ-Übergängen (auch ϵ-NFA) ist ein NFA mit einem speziellen Symbol $\epsilon \notin \Sigma$ und mit

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q) .$$

Ein ϵ-Übergang darf ausgeführt werden, ohne dass ein Eingabezeichen gelesen wird.



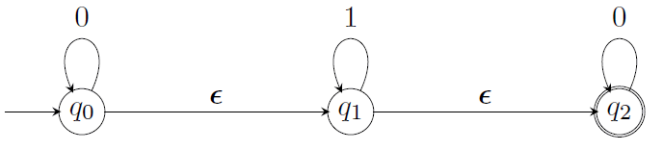
Akzeptiert: ϵ, 00, 11, ...

2.4 NFAs mit ϵ-Übergängen

Definition 2.11
Ein NFA mit ϵ-Übergängen (auch ϵ-NFA) ist ein NFA mit einem speziellen Symbol $\epsilon \notin \Sigma$ und mit

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q) .$$

Ein ϵ-Übergang darf ausgeführt werden, ohne dass ein Eingabezeichen gelesen wird.



Akzeptiert: ϵ, 00, 11, ... Nicht akzeptiert: 101, ...

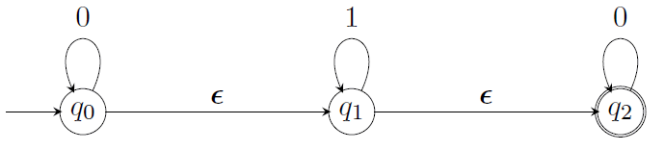
Formal betrachten wir einen ϵ-NFA $N = (Q, \Sigma, \delta, q_0, F)$ als kompakte Repräsentation eines ϵ-freien NFA $N' = (Q, \Sigma, \delta', q_0, F')$

2.4 NFAs mit ϵ-Übergängen

Definition 2.11
Ein NFA mit ϵ-Übergängen (auch ϵ-NFA) ist ein NFA mit einem speziellen Symbol $\epsilon \notin \Sigma$ und mit

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q) .$$

Ein ϵ-Übergang darf ausgeführt werden, ohne dass ein Eingabezeichen gelesen wird.



Akzeptiert: ϵ, 00, 11, ... Nicht akzeptiert: 101, ...

Bemerkung: $\epsilon \neq \epsilon$: ϵ ist ein einzelnes Symbol. ϵ das leere Wort.

Formal betrachten wir einen ϵ-NFA $N = (Q, \Sigma, \delta, q_0, F)$ als kompakte Repräsentation eines ϵ-freien NFA $N' = (Q, \Sigma, \delta', q_0, F')$

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

$$\delta'(q, a) := \bigcup_{i \geq 0, j \geq 0} \hat{\delta}(\{q\}, \epsilon^i a \epsilon^j) .$$

Formal betrachten wir einen ϵ-NFA $N = (Q, \Sigma, \delta, q_0, F)$ als kompakte Repräsentation eines ϵ-freien NFA $N' = (Q, \Sigma, \delta', q_0, F')$

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

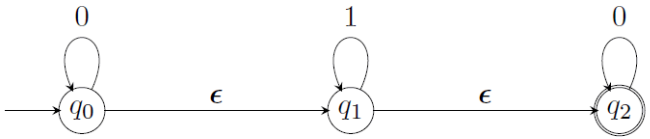
$$\delta'(q, a) := \bigcup_{i \geq 0, j \geq 0} \hat{\delta}(\{q\}, \epsilon^i a \epsilon^j) \ .$$

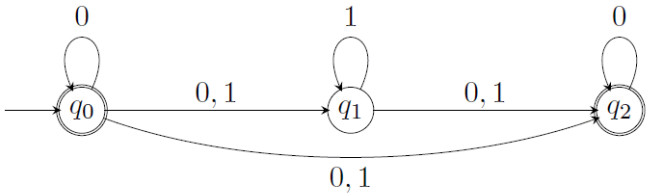
- Falls N das leere Wort ϵ akzeptiert, also falls

$$\exists i \geq 0. \ \hat{\delta}(\{q_0\}, \epsilon^i) \cap F \neq \emptyset$$

dann setze $F' := F \cup \{q_0\}$, sonst setze $F' := F$.

Beispiel 2.12





Formal betrachten wir einen ϵ-NFA $N = (Q, \Sigma, \delta, q_0, F)$ als kompakte Repräsentation eines ϵ-freien NFA $N' = (Q, \Sigma, \delta', q_0, F')$

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

$$\delta'(q, a) := \bigcup_{i \geq 0, j \geq 0} \hat{\delta}(\{q\}, \epsilon^i a \epsilon^j) \ .$$

- Falls N das leere Wort ϵ akzeptiert, also falls

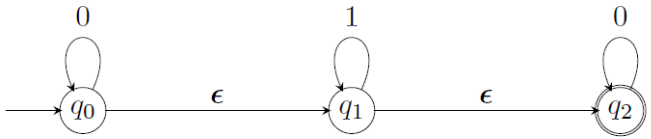
$$\exists i \geq 0. \ \hat{\delta}(\{q_0\}, \epsilon^i) \cap F \neq \emptyset$$

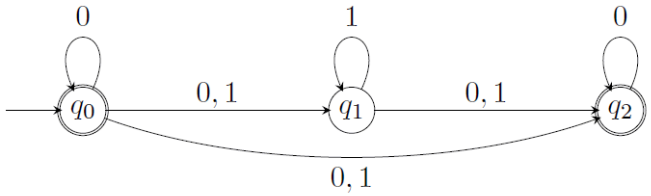
dann setze $F' := F \cup \{q_0\}$, sonst setze $F' := F$.

Damit gilt per definitionem:

Jeder ϵ-NFA ist äquivalent zu einem NFA.

Beispiel 2.12





Formal betrachten wir einen ϵ-NFA $N = (Q, \Sigma, \delta, q_0, F)$ als kompakte Repräsentation eines ϵ-freien NFA $N' = (Q, \Sigma, \delta', q_0, F')$

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

$$\delta'(q, a) := \bigcup_{i \geq 0, j \geq 0} \hat{\delta}(\{q\}, \epsilon^i a \epsilon^j) \ .$$

- Falls N das leere Wort ϵ akzeptiert, also falls

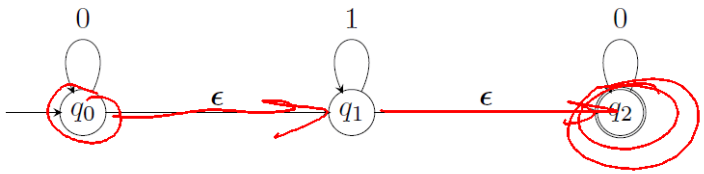
$$\exists i \geq 0. \ \hat{\delta}(\{q_0\}, \epsilon^i) \cap F \neq \emptyset$$

dann setze $F' := F \cup \{q_0\}$, sonst setze $F' := F$.

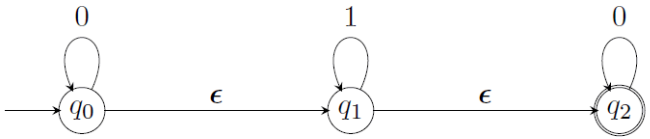
Damit gilt per definitionem:

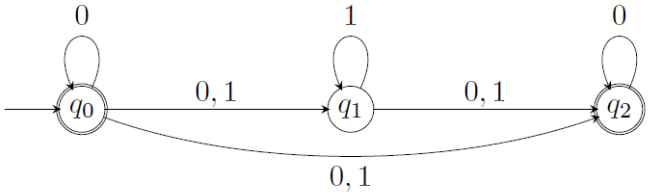
Jeder ϵ-NFA ist äquivalent zu einem NFA.

Beispiel 2.12



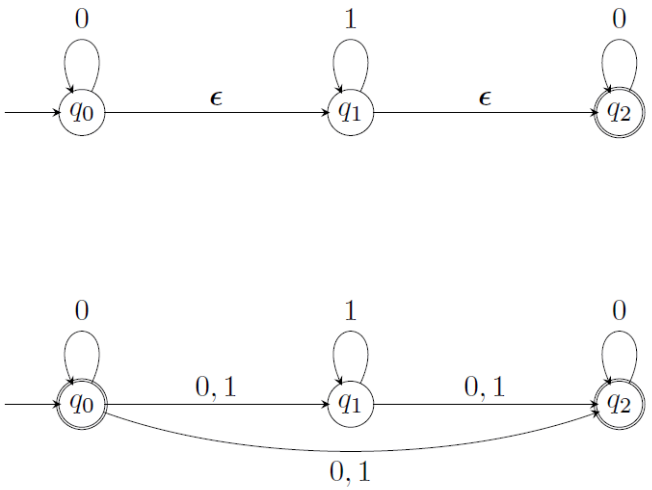
Beispiel 2.12





Warum ϵ-NFAs?

Beispiel 2.12



Warum ϵ-NFAs?

Sie sind praktisch.

Ab jetzt: „ϵ-Übergang“ und „ϵ-NFA“

Warum ϵ-NFAs?

Sie sind praktisch.

Formal betrachten wir einen ϵ-NFA $N = (Q, \Sigma, \delta, q_0, F)$ als kompakte Repräsentation eines ϵ-freien NFA $N' = (Q, \Sigma, \delta', q_0, F')$

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

$$\delta'(q, a) := \bigcup_{i \geq 0, j \geq 0} \hat{\delta}(\{q\}, \epsilon^i a \epsilon^j) \ .$$

- Falls N das leere Wort ϵ akzeptiert, also falls

$$\exists i \geq 0. \ \hat{\delta}(\{q_0\}, \epsilon^i) \cap F \neq \emptyset$$

dann setze $F' := F \cup \{q_0\}$, sonst setze $F' := F$.

Damit gilt per definitionem:

Jeder ϵ-NFA ist äquivalent zu einem NFA.

Beispiel 2.12

Formal betrachten wir einen ϵ-NFA $N = (Q, \Sigma, \delta, q_0, F)$ als kompakte Repräsentation eines ϵ-freien NFA $N' = (Q, \Sigma, \delta', q_0, F')$

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

$$\delta'(q, a) := \bigcup_{i \geq 0, j \geq 0} \hat{\delta}(\{q\}, \epsilon^i a \epsilon^j) \ .$$

- Falls N das leere Wort ϵ akzeptiert, also falls

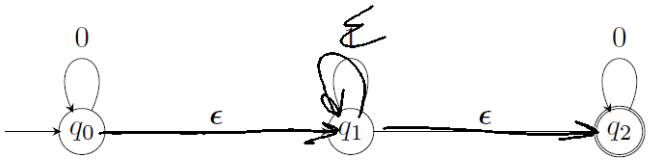
$$\exists i \geq 0. \ \hat{\delta}(\{q_0\}, \epsilon^i) \cap F \neq \emptyset$$

dann setze $F' := F \cup \{q_0\}$, sonst setze $F' := F$.

Damit gilt per definitionem:

Jeder ϵ-NFA ist äquivalent zu einem NFA.

Beispiel 2.12



Formal betrachten wir einen ϵ-NFA $N = (Q, \Sigma, \delta, q_0, F)$ als kompakte Repräsentation eines ϵ-freien NFA $N' = (Q, \Sigma, \delta', q_0, F')$

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

$$\delta'(q, a) := \bigcup_{i \geq 0, j \geq 0} \hat{\delta}(\{q\}, \epsilon^i a \epsilon^j) \ .$$

- Falls N das leere Wort ϵ akzeptiert, also falls

$$\exists i \geq 0. \ \hat{\delta}(\{q_0\}, \epsilon^i) \cap F \neq \emptyset$$

dann setze $F' := F \cup \{q_0\}$, sonst setze $F' := F$.

Damit gilt per definitionem:

Jeder ϵ-NFA ist äquivalent zu einem NFA.

Warum ϵ-NFAs?

Sie sind praktisch.

Ab jetzt: „ϵ-Übergang“ und „ϵ-NFA“

Vorläufiges Fazit:
Die folgenden Automatentypen sind gleich mächtig:

- DFA
- NFA
- ϵ-NFA

Vorläufiges Fazit:
Die folgenden Automatentypen sind gleich mächtig:

- DFA
- NFA
- ϵ-NFA

2.5 Reguläre Ausdrücke

Reguläre Ausdrücke sind eine kompakte Notation für formale Sprachen.

2.5 Reguläre Ausdrücke

Reguläre Ausdrücke sind eine kompakte Notation für formale Sprachen.

Definition 2.13
Reguläre Ausdrücke (*regular expressions*, REs) sind induktiv definiert:

- $\emptyset$ ist ein regulärer Ausdruck.

2.5 Reguläre Ausdrücke

Reguläre Ausdrücke sind eine kompakte Notation für formale Sprachen.

Definition 2.13
Reguläre Ausdrücke (*regular expressions*, REs) sind induktiv definiert:

- $\emptyset$ ist ein regulärer Ausdruck.
- ϵ ist ein regulärer Ausdruck.

2.5 Reguläre Ausdrücke

Reguläre Ausdrücke sind eine kompakte Notation für formale Sprachen.

Definition 2.13
Reguläre Ausdrücke (*regular expressions*, REs) sind induktiv definiert:

- $\emptyset$ ist ein regulärer Ausdruck.
- ϵ ist ein regulärer Ausdruck.
- Für jedes $a \in \Sigma$ ist a ist ein regulärer Ausdruck.
- Wenn α und β reguläre Ausdrücke sind, dann auch
  - $\alpha\beta$
  - $\alpha \mid \beta$ (oft $\alpha + \beta$ geschrieben)
  - α^*.

Nichts sonst ist ein regulärer Ausdruck.

2.5 Reguläre Ausdrücke

Reguläre Ausdrücke sind eine kompakte Notation für formale Sprachen.

Definition 2.13
Reguläre Ausdrücke (*regular expressions*, REs) sind induktiv definiert:

- $\emptyset$ ist ein regulärer Ausdruck.
- ϵ ist ein regulärer Ausdruck.
- Für jedes $a \in \Sigma$ ist a ist ein regulärer Ausdruck.
- Wenn α und β reguläre Ausdrücke sind, dann auch
  - $\alpha\beta$
  - $\alpha \mid \beta$ (oft $\alpha + \beta$ geschrieben)
  - α^*.

Nichts sonst ist ein regulärer Ausdruck.

$1 * 3 + 3 =$

$(1 * 3) + 3$

Notation:

- Reguläre Ausdrücke können bzw. müssen geklammert werden.
- Bindungsstärke: * stärker als Konkatenation stärker als |

Notation:

- Reguläre Ausdrücke können bzw. müssen geklammert werden.
- Bindungsstärke: * stärker als Konkatenation stärker als |
- $ab^* = a(b^*) \neq (ab)^*$

Notation:

- Reguläre Ausdrücke können bzw. müssen geklammert werden.
- Bindungsstärke: * stärker als Konkatenation stärker als |
- $ab^* = a(b^*) \neq (ab)^*$
- $ab \mid c = (ab) \mid c \neq a(b \mid c)$

Definition 2.14

Zu einem regulären Ausdruck γ ist die zugehörige Sprache $L(\gamma)$ rekursiv definiert:

- $L(\emptyset) = \emptyset$

Definition 2.14

Zu einem regulären Ausdruck γ ist die zugehörige Sprache $L(\gamma)$ rekursiv definiert:

- $L(\emptyset) = \emptyset$
- $L(\epsilon) = \{\epsilon\}$

Definition 2.14

Zu einem regulären Ausdruck γ ist die zugehörige Sprache $L(\gamma)$ rekursiv definiert:

- $L(\emptyset) = \emptyset$
- $L(\epsilon) = \{\epsilon\}$
- $L(a) = \{a\}$

Definition 2.14

Zu einem regulären Ausdruck γ ist die zugehörige Sprache $L(\gamma)$ rekursiv definiert:

- $L(\emptyset) = \emptyset$
- $L(\epsilon) = \{\epsilon\}$
- $L(a) = \{a\}$
- $L(\alpha\beta) = L(\alpha)L(\beta)$

Definition 2.14

Zu einem regulären Ausdruck γ ist die zugehörige Sprache $L(\gamma)$ rekursiv definiert:

- $L(\emptyset) = \emptyset$
- $L(\epsilon) = \{\epsilon\}$
- $L(a) = \{a\}$
- $L(\alpha\beta) = L(\alpha)L(\beta)$
- $L(\alpha \mid \beta) = L(\alpha) \cup L(\beta)$

Definition 2.14

Zu einem regulären Ausdruck γ ist die zugehörige Sprache $L(\gamma)$ rekursiv definiert:

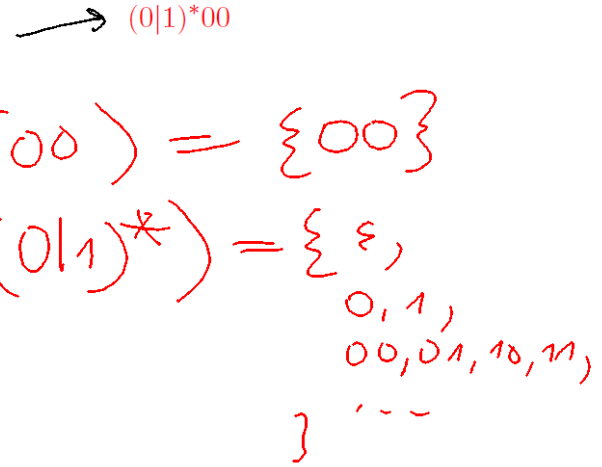
- $L(\emptyset) = \emptyset$
- $L(\epsilon) = \{\epsilon\}$
- $L(a) = \{a\}$
- $L(\alpha\beta) = L(\alpha)L(\beta)$
- $L(\alpha \mid \beta) = L(\alpha) \cup L(\beta)$
- $L(\alpha^*) = L(\alpha)^*$

Beispiel 2.15

Sei das zugrunde liegende Alphabet $\Sigma = \{0, 1\}$.

- Alle Wörter, die mit 00 enden:

$$(0|1)^*00$$



Beispiel 2.15

Sei das zugrunde liegende Alphabet $\Sigma = \{0, 1\}$.

- Alle Wörter, die mit 00 enden:

$$(0|1)^*00$$

- Alle Wörter gerader Länge, in denen 0 und 1 alternieren:

$$(01)^* \mid (10)^*$$

Beispiel 2.15

Sei das zugrunde liegende Alphabet $\Sigma = \{0, 1\}$.

- Alle Wörter, die mit 00 enden:

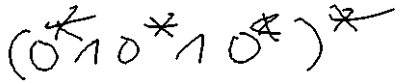
$$(0|1)^*00$$

- Alle Wörter gerader Länge, in denen 0 und 1 alternieren:

$$(01)^* \mid (10)^*$$

- Alle Wörter, die eine gerade Anzahl von 1'en enthalten:

$$(0^*10^*1)^*0^*$$



Beispiel 2.15

Sei das zugrunde liegende Alphabet $\Sigma = \{0, 1\}$.

- Alle Wörter, die mit 00 enden:

$$(0|1)^*00$$

- Alle Wörter gerader Länge, in denen 0 und 1 alternieren:

$$(01)^* \mid (10)^*$$

- Alle Wörter, die eine gerade Anzahl von 1'en enthalten:

$$(0^*10^*1)^*0^*$$

- Alle Wörter, die die Binärdarstellung einer durch 3 teilbaren Zahl darstellen, also

$$0, 11, 110, 1001, 1100, 1111, 10010, \ldots$$

Beispiel 2.16

Gleitkommazahlen: $\Sigma = \{+, -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, .\}$

Beispiel 2.16

Gleitkommazahlen: $\Sigma = \{+, -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, .\}$

$$(+ \mid - \mid \epsilon)(\cancel{DD^*} \mid DD^*.D^* \mid D^*.DD^*)$$

wobei $D = (0|1|2|3|4|5|6|7|8|9)$

Erweiterte reguläre Ausdrücke in UNIX:

$. \quad = \quad a_1|\ldots|a_n$ wobei $\Sigma = \{a_1, \ldots a_n\}$

Beispiel 2.16

Gleitkommazahlen: $\Sigma = \{+, -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, .\}$

$$(+ \mid - \mid \epsilon)(DD^* \mid DD^*.D^* \mid D^*.DD^*)$$

wobei $D = (0|1|2|3|4|5|6|7|8|9)$

Erweiterte reguläre Ausdrücke in UNIX:

$. \quad = \quad a_1|\ldots|a_n$ wobei $\Sigma = \{a_1, \ldots a_n\}$

Erweiterte reguläre Ausdrücke in UNIX:

$. \quad = \quad a_1|\ldots|a_n$ wobei $\Sigma = \{a_1, \ldots a_n\}$

$[a_1 \ldots a_n] \quad = \quad a_1|\ldots|a_n$

[0,,9]

Erweiterte reguläre Ausdrücke in UNIX:

$. \quad = \quad a_1|\ldots|a_n$ wobei $\Sigma = \{a_1, \ldots a_n\}$

$[a_1 \ldots a_n] \quad = \quad a_1|\ldots|a_n$

$[\hat{}a_1 \ldots a_n] \quad = \quad b_1|\ldots|b_m$ wobei $\{b_1, \ldots, b_m\} = \Sigma \setminus \{a_1, \ldots a_n\}$

Erweiterte reguläre Ausdrücke in UNIX:

$$
\begin{aligned}
. &= a_1|\ldots|a_n \text{ wobei } \Sigma = \{a_1,\ldots a_n\} \\
[a_1\ldots a_n] &= a_1|\ldots|a_n \\
[\hat{}\,a_1\ldots a_n] &= b_1|\ldots|b_m \text{ wobei } \{b_1,\ldots,b_m\} = \Sigma \setminus \{a_1,\ldots a_n\} \\
\alpha? &= \epsilon|\alpha
\end{aligned}
$$

Erweiterte reguläre Ausdrücke in UNIX:

$$
\begin{aligned}
. &= a_1|\ldots|a_n \text{ wobei } \Sigma = \{a_1,\ldots a_n\} \\
[a_1\ldots a_n] &= a_1|\ldots|a_n \\
[\hat{}\,a_1\ldots a_n] &= b_1|\ldots|b_m \text{ wobei } \{b_1,\ldots,b_m\} = \Sigma \setminus \{a_1,\ldots a_n\} \\
\alpha? &= \epsilon|\alpha
\end{aligned}
$$

$$\alpha+ = \alpha\alpha^*$$

Erweiterte reguläre Ausdrücke in UNIX:

$$
\begin{aligned}
. &= a_1|\ldots|a_n \text{ wobei } \Sigma = \{a_1,\ldots a_n\} \\
[a_1\ldots a_n] &= a_1|\ldots|a_n \\
[\hat{}\,a_1\ldots a_n] &= b_1|\ldots|b_m \text{ wobei } \{b_1,\ldots,b_m\} = \Sigma \setminus \{a_1,\ldots a_n\} \\
\alpha? &= \epsilon|\alpha \\
\alpha+ &= \alpha\alpha^*
\end{aligned}
$$

$$\alpha\{n\} = \alpha\ldots\alpha \ (n \text{ copies})$$

Erweiterte reguläre Ausdrücke in UNIX:

$$
\begin{aligned}
. &= a_1|\ldots|a_n \text{ wobei } \Sigma = \{a_1,\ldots a_n\} \\
[a_1\ldots a_n] &= a_1|\ldots|a_n \\
[\hat{}\,a_1\ldots a_n] &= b_1|\ldots|b_m \text{ wobei } \{b_1,\ldots,b_m\} = \Sigma \setminus \{a_1,\ldots a_n\} \\
\alpha? &= \epsilon|\alpha \\
\alpha+ &= \alpha\alpha^* \\
\alpha\{n\} &= \alpha\ldots\alpha \ (n \text{ copies})
\end{aligned}
$$

Strukturelle Induktion

Da die regulären Ausdrücke induktiv definiert sind, gilt für sie das Prinzip der strukturellen Induktion:

Strukturelle Induktion

Da die regulären Ausdrücke induktiv definiert sind, gilt für sie das Prinzip der strukturellen Induktion:

Um zu beweisen, dass Eigenschaft P für alle regulären Ausdrücke γ gilt, also $P(\gamma)$, beweise

Strukturelle Induktion

Da die regulären Ausdrücke induktiv definiert sind, gilt für sie das Prinzip der strukturellen Induktion:

Um zu beweisen, dass Eigenschaft P für alle regulären Ausdrücke γ gilt, also $P(\gamma)$, beweise

- $P(\emptyset)$
- $P(\epsilon)$
- $P(a)$ für alle $a \in \Sigma$
- $P(\alpha) \wedge P(\beta) \Rightarrow P(\alpha\beta)$
- $P(\alpha) \wedge P(\beta) \Rightarrow P(\alpha \mid \beta)$
- $P(\alpha) \Rightarrow P(\alpha^*)$

Strukturelle Induktion

Da die regulären Ausdrücke induktiv definiert sind, gilt für sie das Prinzip der strukturellen Induktion:

Um zu beweisen, dass Eigenschaft P für alle regulären Ausdrücke γ gilt, also $P(\gamma)$, beweise

- $P(\emptyset)$
- $P(\epsilon)$
- $P(a)$ für alle $a \in \Sigma$
- $P(\alpha) \wedge P(\beta) \Rightarrow P(\alpha\beta)$
- $P(\alpha) \wedge P(\beta) \Rightarrow P(\alpha \mid \beta)$
- $P(\alpha) \Rightarrow P(\alpha^*)$

Komfortabler Spezialfall der Induktion über die Länge von γ.

Satz 2.17 (Kleene 1956)

Eine Sprache $L \subseteq \Sigma^*$ ist genau dann durch einen regulären Ausdruck darstellbar, wenn sie regulär ist.

Strukturelle Induktion

Da die regulären Ausdrücke induktiv definiert sind, gilt für sie das Prinzip der strukturellen Induktion:

Um zu beweisen, dass Eigenschaft P für alle regulären Ausdrücke γ gilt, also $P(\gamma)$, beweise

- $P(\emptyset)$
- $P(\epsilon)$
- $P(a)$ für alle $a \in \Sigma$
- $P(\alpha) \wedge P(\beta) \Rightarrow P(\alpha\beta)$
- $P(\alpha) \wedge P(\beta) \Rightarrow P(\alpha \mid \beta)$
- $P(\alpha) \Rightarrow P(\alpha^*)$

Komfortabler Spezialfall der Induktion über die Länge von γ.

Satz 2.17 (Kleene 1956)

Eine Sprache $L \subseteq \Sigma^*$ ist genau dann durch einen regulären Ausdruck darstellbar, wenn sie regulär ist.

Beweis:

"$\Longrightarrow$":

Sei $L = L(\gamma)$.

Wir konstruieren einen ϵ-NFA N mit $L = L(N)$ mit Hilfe struktureller Induktion über γ.