Deterministische endliche Automaten

(engl.: deterministic finite automaton, kurz: DFA)

Definition 7.3 (DFA):

Ein deterministischer endlicher Automat

$$A = (\Sigma, Q, \delta, q_0, F)$$

besteht aus

- einer endlichen Menge Σ, dem so genannten Eingabealphabet
- einer endlichen Menge Q, der so genannten Zustandsmenge (die Elemente aus Q werden Zustände genannt)
- einer partiellen Funktion δ von $Q \times \Sigma$ nach Q, der so genannten (Zustands-)Übergangsfunktion (oder Überführungsfunktion)
- einem Zustand $q_0 \in Q$, dem sog. Startzustand
- einer Menge $F \subseteq Q$, der so genannten Menge der Endzustände bzw. akzeptierenden Zustände

(der Buchstabe "F" steht für "final states", also "Endzustände")

Graphische Darstellung endlicher Automaten

Endliche Automaten lassen sich anschaulich durch beschriftete Graphen darstellen (vgl Bsp 7.1 und 7.2):

- Für jeden Zustand $q \in Q$ gibt es einen durch (q) dargestellten Knoten.
- Der Startzustand q_0 wird durch einen in ihn hinein führenden Pfeil markiert, dh: $\rightarrow (q_0)$
- Jeder akzeptierende Zustand $q \in F$ wird durch eine doppelte Umrandung markiert, dh: $((q))$
- Ist $q \in Q$ ein Zustand und $a \in \Sigma$ ein Symbol aus dem Eingabealphabet, so dass $(q,a) \in \mathrm{Def}(\delta)$ liegt, so gibt es in der graphischen Darstellung von A einen mit dem Symbol a beschrifteten Pfeil von Knoten (q) zu Knoten $(\delta(q,a))$, d.h.: $(q) \xrightarrow{a} (\delta(q,a))$.

Beispiel 7.4

Die graphische Darstellung aus Bsp 7.2 repräsentiert den DFA $A = (\Sigma, Q, \delta, q_0, F)$ mit

- $\Sigma = \{$ Einwerfen einer 1€-Münze,
  Einwerfen einer 2€-Münze,
  Drücken der "Geld Rückgabe" Taste,
  Drücken der Taste "Kaffee kaufen" $\}$
- $Q = \{$ Grundzustand, Zustand 1, Zustand 2 $\}$
- $q_0 =$ Grundzustand
- $F = \{$ Grundzustand $\}$
- δ ist die partielle Funktion von $Q \times \Sigma$ nach Q mit

  $\delta($ Grundzustand, Einwerfen einer 1€-Münze$)$ = Zustand 1,
  $\delta($ Grundzustand, Einwerfen einer 2€-Münze$)$ = Zustand 2,
  $\delta($ Zustand 1, Einwerfen einer 1€-Münze$)$ = Zustand 2,
  $\delta($ Zustand 1, Drücken der "Geld Rückgabe" Taste$)$ = Grundzustand,
  $\delta($ Zustand 2, Drücken der "Geld Rückgabe" Taste$)$ = Grundzustand,
  $\delta($ Zustand 2, Drücken der Taste "Kaffee kaufen"$)$ = Grundzustand,

Die von einem DFA akzeptierte Sprache:

Ein DFA $A = (\Sigma, Q, \delta, q_0, F)$ erhält als Eingabe ein Wort $w \in \Sigma^*$, das eine Folge von "Aktionen" oder "Bedienoperationen" repräsentiert.

Ist das Eingabewort w von der Form $a_1 \cdots a_n$ mit $n \geq 0$ und $a_1 \in \Sigma, \ldots, a_n \in \Sigma$, so geschieht bei der "Verarbeitung" von w durch A folgendes:

A wird im "Startzustand" q_0 gestartet. Durch Lesen des ersten Buchstabens von w, also a_1, geht der Automat über in den Zustand $q_1 := \delta(q_0, a_1)$. In der graphischen Darstellung von A wird der Zustand $\rightarrow (q_0)$ durch die mit a_1 beschriftete Kante verlassen, und q_1 ist der Endknoten dieser Kante, dh $\rightarrow (q_0) \xrightarrow{a_1} (q_1)$.

Dies ist allerdings nur möglich, wenn $(q_0, a_1) \in \mathrm{Def}(\delta)$ liegt — dh wenn es in der graphischen Darstellung von A eine mit a_1 beschriftete Kante gibt, die

aus Zustand q_0 herausführt. Falls es keine solche Kante gibt, dh falls $(q_0, a_1) \notin Def(\delta)$, so "stürzt A ab", und die Verarbeitung des Worts w ist beendet. Ansonsten ist A nach Lesen des ersten Symbols von w im Zustand $q_1 := \delta(q_0, a_1)$. Durch Lesen des zweiten Symbols von w, also a_2, geht A nun in den Zustand $q_2 := \delta(q_1, a_2)$ über – bzw "stürzt ab", falls $(q_1, a_2) \notin Def(\delta)$. In der graphischen Darstellung wird q_1 durch die mit a_2 beschriftete Kante verlassen (falls eine solche Kante existiert); und $q_2 := \delta(q_1, a_2)$ ist der Endknoten dieser Kante, dh $q_1 \xrightarrow{a_2} q_2$.

Auf diese Weise wird nach und nach das gesamte Eingabewort $w = a_1 \cdots a_n$ abgearbeitet und – ausgehend vom Startzustand q_0 – werden nacheinander Zustände $q_1, \ldots, q_n$ erreicht.
In der graphischen Darstellung von A entspricht dies gerade dem Durchlaufen eines Weges der Länge n, der im Knoten $\rightarrow q_0$ startet und dessen Kanten mit den Buchstaben $a_1, \ldots, a_n$ beschriftet sind. Der

Knoten q_n, der am Ende dieses Weges erreicht wird (falls der Automat nicht zwischendurch abstürzt, d.h. falls es überhaupt einen mit $a_1, \dots, a_n$ beschrifteten in q_0 startenden Weg gibt),

ist <u>der von A bei Eingabe w erreichte Zustand</u>,

kurz: $q_n = \hat{\delta}(q_0, w)$.

(Im Fall, dass A bei Eingabe von w zwischendurch abstürzt, sagen wir: „$\hat{\delta}(q_0, w)$ ist undefiniert").

Präzise Definition von $\hat{\delta}$:

<u>Definition 7.6</u>:

Sei $A := (\Sigma, Q, \delta, q_0, F)$ ein DFA.

Die partielle Funktion $\hat{\delta}$ von $Q \times \Sigma^*$ nach Q ist rekursiv wie folgt definiert:

- f.a. $q \in Q$ ist $\hat{\delta}(q, \varepsilon) := q$
- f.a. $q \in Q$, $w \in \Sigma^*$ und $a \in \Sigma$ gilt:
  Falls $(q, w) \in \mathrm{Def}(\hat{\delta})$ und $(\hat{\delta}(q, w), a) \in \mathrm{Def}(\delta)$,
  so ist $\hat{\delta}(q, wa) := \delta(\hat{\delta}(q, w), a)$.

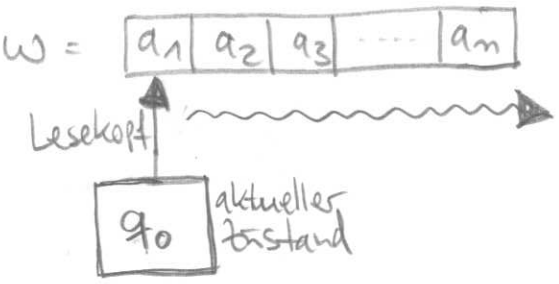
<u>Graphische Darstellung</u>:

Insgesamt gilt:

Falls $(q_0, w) \in \mathrm{Def}(\hat{\delta})$ so ist $\hat{\delta}(q_0, w)$ der Zustand, der durch Verarbeiten des Worts w erreicht wird; falls $(q_0, w) \notin \mathrm{Def}(\hat{\delta})$, so stürzt der Automat beim Verarbeiten des Worts w ab.

Das Eingabewort w wird vom DFA A <u>akzeptiert</u>, falls er bei Eingabe von w nicht abstürzt und der durch Verarbeiten des Worts w erreichte Zustand zur Menge F der akzeptierenden Zustände gehört. In der graphischen Darstellung von A heißt das für ein Eingabewort $w = a_1 \cdots a_n$, dass es einen in $\rightarrow (q_0)$ startenden Weg der Länge n gibt, dessen Kanten mit den Symbolen $a_1, \ldots, a_n$ beschriftet sind, und der in einem akzeptierenden Zustand ◎ endet.

Verarbeitung eines Eingabeworts durch einen DFA A:



<u>Definition 7.7</u>: (Die von einem DFA A akzeptierte Sprache $L(A)$)

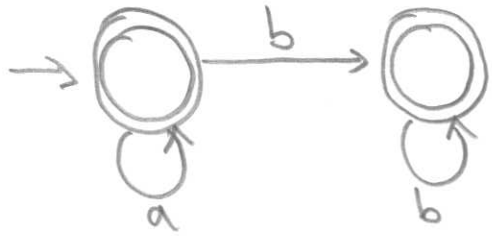
Die von einem DFA $A = (\Sigma, Q, \delta, q_0, F)$ akzeptierte Sprache $L(A)$ ist

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

D.h. ein Wort $w \in \Sigma^*$ gehört genau dann zur Sprache $L(A)$, wenn es vom DFA A akzeptiert wird.

<u>Beispiel 7.8</u>: Der Einfachheit halber betrachten wir das Eingabealphabet $\Sigma := \{a, b\}$.

<u>(a)</u> Sei A_1 ein DFA mit folgender graphischer Darstellung:



- A_1 akzeptiert z.B. folgende Worte: ε, a, b, aaa, aaab, aaaaabbbb, bbb, ...
- A_1 "stirbt ab" z.B. bei Eingabe von ba, aabbba

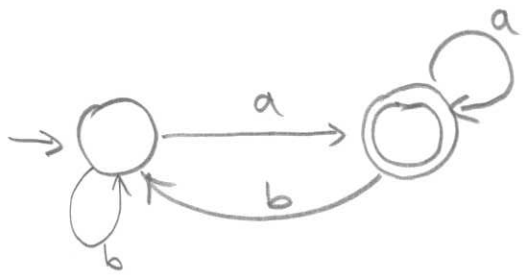
Insgesamt gilt: $L(A_1) = \{ a^n b^m : n \in \mathbb{N}, m \in \mathbb{N} \}$

(<u>Notation</u>: $a^n b^m$ bezeichnet das Wort $\underbrace{a\cdots a}_{n}\underbrace{b\cdots b}_{m}$ der Länge $n+m$, das aus n a's gefolgt von m b's besteht, z.B. ist a^3b^4 das Wort aaabbbb)

(<u>b</u>) Ein DFA A_2 mit

$L(A_2) = \{ w \in \{a,b\}^* :$ der letzte Buchstabe von w ist ein a $\}$:

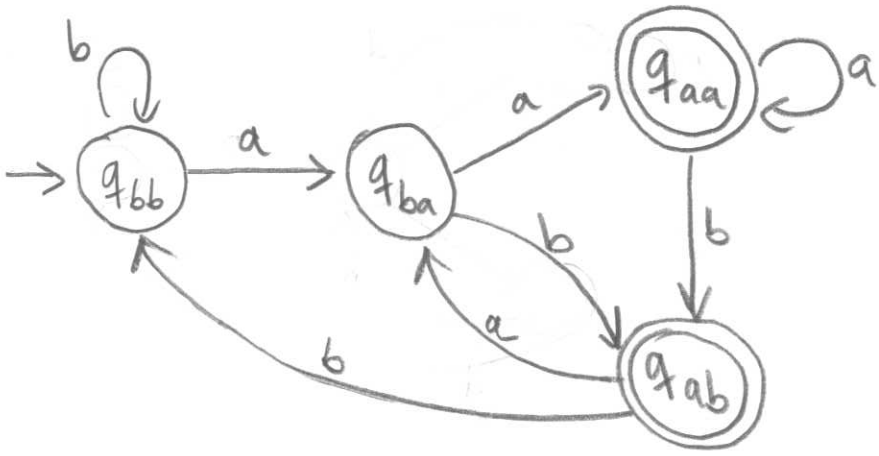
graphische Darstellung von A_2:



(<u>c</u>) Ein DFA A_3 mit

$L(A_3) = \{ w \in \{a,b\}^* :$ der <u>vorletzte</u> Buchstabe von w ist ein a $\}$:

graphische Darstellung von A_3:



Bemerkung 7.9

- Die in Definition 7.3 eingeführten DFAs $A = (\Sigma, Q, \delta, q_0, F)$ heißen <u>deterministisch</u>, weil es zu jedem Paar $(q,a) \in Q \times \Sigma$ höchstens einen "Nachfolgezustand" $\delta(q,a)$ gibt (da δ eine partielle Funktion von $Q \times \Sigma$ nach Q ist).

  Beim Verarbeiten eines Eingabeworts ist daher zu jedem Zeitpunkt klar, ob A "abstürzt" oder nicht – und falls nicht, welchen eindeutig festgelegten Nachfolgezustand A annimmt.

- Ein DFA A heißt <u>vollständig</u>, wenn die Übergangsfunktion δ eine totale Funktion $\delta: Q \times \Sigma \to Q$ ist.

  <u>Bsp</u>: Die DFAs A_2 und A_3 aus Bsp 7.8 sind vollständig; der DFA A_1 nicht, und auch die DFAs aus Bsp 7.1 und 7.2 sind nicht vollständig.

  Für die graphische Darstellung eines DFAs gilt: Der DFA ist genau dann vollständig, wenn für jeden Zustand q gilt: Für jedes Symbol $a \in \Sigma$ gibt es genau eine aus q herausführende Kante, die mit a beschriftet ist.

<u>Beachte</u>: In manchen Büchern weicht die Definition von DFAs von Definition 7.3 ab, indem gefordert wird, dass DFAs grundsätzlich vollständig sein müssen.

Nicht-deterministische endliche Automaten:

(engl.: non-deterministic finite automaton, kurz: NFA)

Für manche Modellierungsaufgaben ist die Forderung, dass es für jeden Zustand q und jedes Eingabesymbol a höchstens einen Nachfolgezustand $\delta(q,a)$ gibt, zu restriktiv, da man in manchen Zuständen für den Übergang mit einem Symbol a mehrere Möglichkeiten angeben will, ohne festzulegen, welche davon gewählt wird. Solche Entscheidungsfreiheiten in der Modellierung von Abläufen nennt man nicht-deterministisch. Nicht-deterministische Modelle sind häufig einfacher aufzustellen und leichter zu verstehen als deterministische.

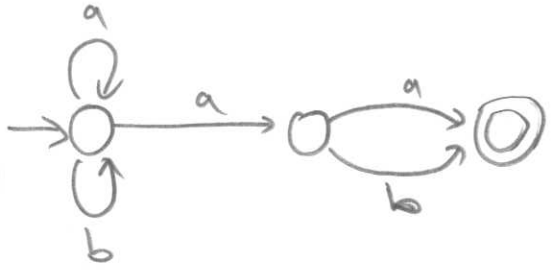
Beispiel 7.10:

In Beispiel 7.8 (c) haben wir einen (recht komplizierten) DFA A_3 kennengelernt mit

$L(A_3) = \{ w \in \{a,b\}^* :$ der vorletzte Buchstabe von w ist ein $a \}$.

Dieselbe Sprache auch vom folgenden, deutlich einfacheren nicht-deterministischen endlichen Automaten (kurz: NFA) A_4 akzeptiert:

graphische Darstellung von A_4:



Generell gilt: Ein Eingabewort w wird von dem NFA A_4 genau dann akzeptiert, wenn es in der graphischen Darstellung (mindestens) einen Weg gibt, der im Startzustand $\rightarrow\bigcirc$ beginnt, dessen Kanten mit w beschriftet sind und der im akzeptierenden Zustand ◎ endet.

Präzise Definition von NFAs:

<u>Definition 7.11</u> (NFA)

Ein <u>nicht-deterministischer endlicher Automat</u>

$A = (\Sigma, Q, \delta, q_0, F)$ besteht aus

- einer endlichen Menge Σ, dem so genannten <u>Eingabealphabet</u>
- einer endlichen Menge Q, der sog. <u>Zustandsmenge</u>
- einer Funktion $\delta: Q \times \Sigma \to \mathcal{P}(Q)$, der sog. <u>Übergangsfunktion</u>, die jedem Zustand $q \in Q$ und jedem Symbol $a \in \Sigma$ eine <u>Menge $\delta(q,a)$ von möglichen Nachfolgezuständen</u> zuordnet
  (Beachte: möglicherweise ist $\delta(q,a) = \emptyset$ — dann "stürzt" der Automat ab, wenn er im Zustand q ist und das Symbol a liest).
- einem Zustand $q_0 \in Q$, dem sog. <u>Startzustand</u>
- einer Menge $F \subseteq Q$, der sog. Menge der <u>Endzustände</u> bzw. <u>akzeptierenden Zustände</u>.

Graphische Darstellung von NFAs:

Wie bei DFAs. Ist $q \in Q$ ein Zustand und ist $a \in \Sigma$ ein Eingabesymbol, so gibt es <u>für jeden Zustand $q' \in \delta(q,a)$</u> in der graphischen Darstellung des NFAs einen mit dem Symbol a beschrifteten Pfeil von Knoten (q) zu Knoten (q'), dh $(q) \xrightarrow{a} (q')$.

Die von einem NFA A akzeptierte Sprache L(A):

Definition 7.12

Sei $A = (\Sigma, Q, \delta, q_0, F)$ ein NFA

(a) Sei $n \in \mathbb{N}$ und sei $w = a_1 \cdots a_n$ ein Eingabewort der Länge n.

Das Wort w wird genau dann vom NFA A akzeptiert, wenn es in der graphischen Darstellung von A einen im Startzustand $\to (q_0)$ beginnenden Weg der Länge n gibt, dessen Kanten mit den Symbolen $a_1, \ldots, a_n$ beschriftet sind und der

in einem akzeptierenden Zustand endet.

(b) Die von A akzeptierte Sprache $L(A)$ ist

$$L(A) := \{ w \in \Sigma^* : A \text{ akzeptiert } w \}$$

Ein Anwendungsbeispiel: Stichwort-Suche in Texten

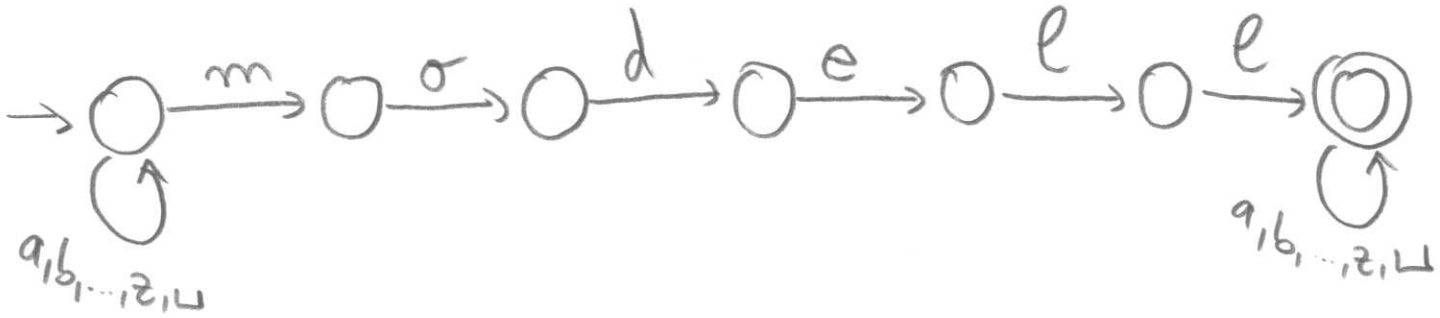
Gegeben: Ein Stichwort, z.B. "modell"

Eingabe: Ein Text, der aus den Buchstaben $a, \dots, z, \sqcup$ besteht

Frage: Kommt das Stichwort "modell" irgendwo im Eingabetext vor?

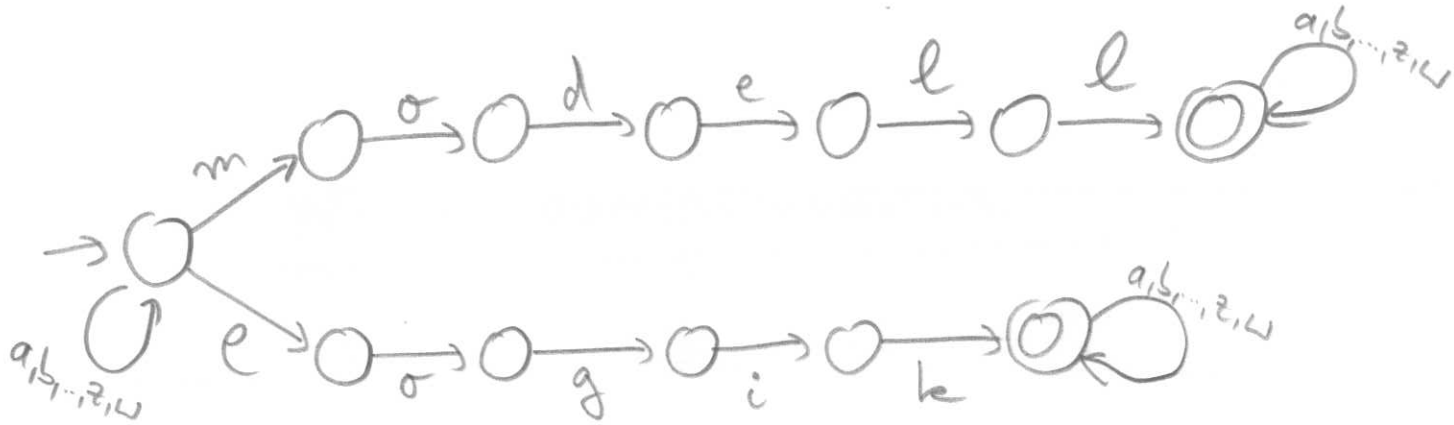
— Der Eingabetext soll genau dann akzeptiert werden, wenn er das Stichwort "modell" enthält.

Graphische Darstellung eines NFAs, der dies bewerkstelligt:



Variante: Kommt mindestens eins der Stichworte "modell" bzw. "logik" im Eingabetext vor?

Graphische Darstellung eines NFAs, der dies bewerkstelligt:



NFAs vs. DFAs

Frage: Können NFAs wirklich "mehr" als DFAs?

Antwort: Nein:

Satz 7.13:
Für jeden NFA $A = (\Sigma, Q, \delta, q_0, F)$ gibt es einen DFA $A' = (\Sigma, Q', \delta', q_0', F')$ mit $L(A') = L(A)$.

D.h: NFAs und DFA können genau dieselben Sprachen akzeptieren.

Beweis: In der Vorlesung GL-2.

Bemerkung 7.15

Typische Fragen bzgl. DFAs bzw NFAs:

(a) Welche Sprachen können prinzipiell durch DFAs (bzw, äquivalent dazu, NFAs) akzeptiert werden; welche nicht?

Bsp: Die Sprache $L_1 := \{ a^n b^m : n \in \mathbb{N}, m \in \mathbb{N} \}$ wird von dem DFA A_1 aus Beispiel 7.8(a) akzeptiert, dh $L_1 = L(A_1)$.

Satz: Es gibt keinen DFA, der genau die Sprache $L_2 := \{ a^n b^n : n \in \mathbb{N} \}$ akzeptiert

Beweis: In der Vorlesung GL-2.

(b) Gegeben sei ein DFA oder NFA A.
Wie kann man herausfinden, ob $L(A) \neq \emptyset$ ist, d.h. ob es (mind.) ein Eingabewort gibt, das von A akzeptiert wird (↗ vgl. das Flüssiberquerungsproblem aus Beispiel 1.1).
— Antwort: Teste, ob es in der graphischen Darstellung von A einen Weg gibt, der vom Startzustand

zu einem akzeptierenden Zustand führt.

(c) Wie schwierig ist es, zu einem gegebenen NFA einen DFA zu finden, der dieselbe Sprache akzeptiert?

Antwort(en): In der Vorlesung GL-2.

Reguläre Ausdrücke

Reguläre Ausdrücke beschreiben Mengen von Worten, die nach bestimmten Regeln bzw "Mustern" aufgebaut sind.

Beispiel 7.16:

Die Menge aller Wörter über dem Alphabet $\{a,b\}$, deren vorletzter Buchstabe ein a ist, wird durch den folgenden regulären Ausdruck beschrieben:

$$(a|b)^* a (a|b)$$

Definition 7.17 (Reguläre Ausdrücke – Syntax)

Sei Σ ein endliches Alphabet.

Die Menge aller regulären Ausdrücke über Σ ist rekursiv wie folgt definiert:

Basisregeln:

- $\emptyset$ ist ein regulärer Ausdruck über Σ ("leere Menge")
- ε ist ein regulärer Ausdruck über Σ ("leeres Wort")
- für jedes $a \in \Sigma$ gilt:
  a ist ein regulärer Ausdruck über Σ

Rekursive Regeln:

- Ist R ein regulärer Ausdruck über Σ, so ist auch R^* ("Kleene-Stern") ein regulärer Ausdruck über Σ
- Sind R und S reguläre Ausdrücke über Σ, so ist auch
  - $(R \cdot S)$ ("Konkatenation") ein regulärer Ausdruck über Σ
  - $(R \mid S)$ ("Vereinigung") ein regulärer Ausdruck über Σ

Definition 7.18 (Reguläre Ausdrücke – Semantik)

Sei Σ ein endliches Alphabet.
Jeder reguläre Ausdruck R über Σ <u>beschreibt</u> (oder: definiert) eine Sprache $L(R) \subseteq \Sigma^*$, die induktiv wie folgt definiert ist:

- $L(\emptyset) := \emptyset$
- $L(\varepsilon) := \{\varepsilon\}$
- für jedes $a \in \Sigma$ gilt: $L(a) := \{a\}$
- Ist R ein regulärer Ausdruck über Σ, so ist

  $L(R^*) := \{\varepsilon\} \cup \{w_1 \cdots w_k : k \in \mathbb{N}_{>0},\ w_1 \in L(R), \ldots, w_k \in L(R)\}$
- Sind R und S reguläre Ausdrücke über Σ, so ist
  - $L((R \cdot S)) := \{wu : w \in L(R),\ u \in L(S)\}$
  - $L((R \mid S)) := L(R) \cup L(S)$.

Notation 7.19

Zur vereinfachten Schreibweise und Lesbarkeit von regulären Ausdrücken vereinbaren wir folgende Konventionen:

- Den "Punkt" bei der Konkatenation $(R \cdot S)$ darf man weglassen
- Bei Ketten gleichartiger Operatoren darf man Klammern weglassen: z.B. schreiben wir kurz $(R_1 | R_2 | R_3 | R_4)$ statt $(((R_1 | R_2) | R_3) | R_4)$ und $(R_1 R_2 R_3 R_4)$ statt $(((R_1 R_2) R_3) R_4)$
- "Präzedenzregeln":
  1) $*$ bindet stärker als $\cdot$
  2) $\cdot$ bindet stärker als $|$
- äußere Klammern, die einen regulären Ausdruck umschließen, dürfen weggelassen werden.
- Zur besseren Lesbarkeit dürfen zusätzliche Klammern benutzt werden.

Beispiel 7.20

(a) $a|bc^*$ ist eine verkürzte Schreibweise für den regulären Ausdruck $(a|(b \cdot c^*))$.

Die von diesem regulären Ausdruck beschriebene Sprache ist $L(a|bc^*) = \{a\}$

$\cup \{ w \in \{a,b,c\}^* :$ der erste Buchstabe von w ist ein b und alle weiteren Buchstaben von w sind c's $\}$

(b) $L((a|b)^*) = \{a,b\}^*$

(c) Die Menge aller Worte über dem Alphabet $\{a,b,c\}$, in denen abb als Teilwort vorkommt, wird durch den folgenden regulären Ausdruck beschrieben: $(a|b|c)^* \, abb \, (a|b|c)^*$

(d) Die Menge aller Worte über $\{a,b,c\}$, deren letzter oder vorletzter Buchstabe ein a ist, wird durch den folgenden regulären Ausdruck beschrieben:

$$(a|b|c)^* \, a \, (\varepsilon|a|b|c)$$