Automatentheorie und Formale Sprachen

Teil 2 - Grundbegriffe und Wortfunktionen

Prof. Dr. Hans-Werner Sehring

NORDAKADEMIE
HOCHSCHULE DER WIRTSCHAFT

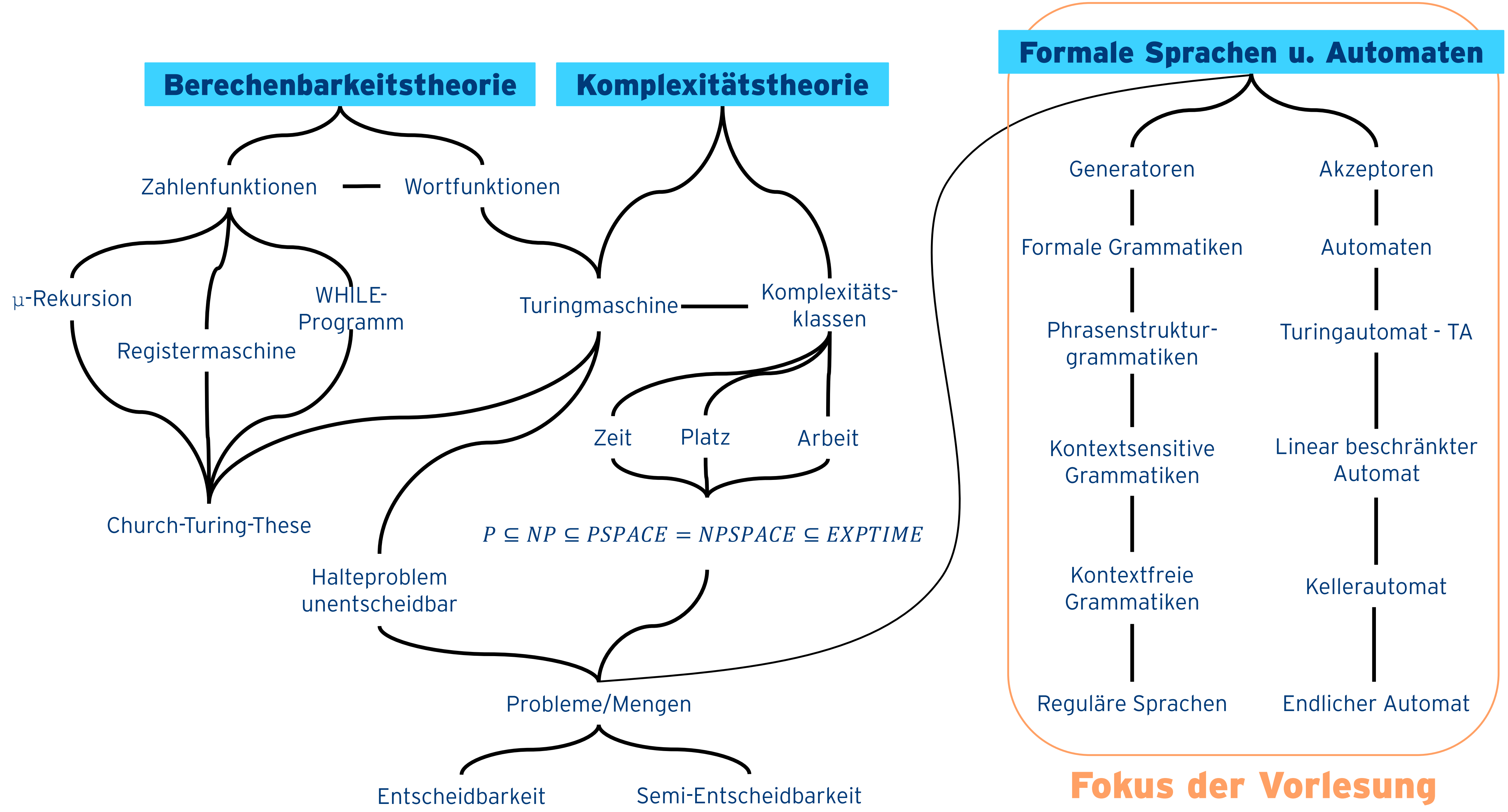
Agenda

01 Alphabete

02 Wörter

03 Formale Sprachen

Theoretische Informatik – Themenübersicht



Fragestellung

Was versteht man unter einer **formalen Sprache**?

Aufbau formaler Sprachen

Folgende Konvention gilt dabei aus Perspektive der formalen Sprachen:

Sprache: Menge von Zeichenreihen aus Eingabezeichen

- Menge der Zeichenreihen einer natürlichen Sprache
- Menge aller von einem endlichen Automaten akzeptierten Zeichenreihen

↑

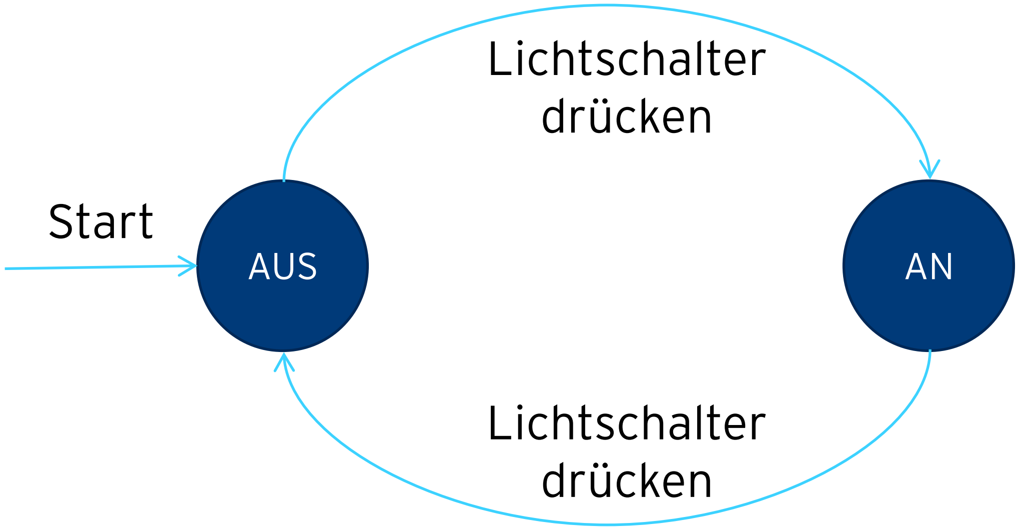
Zeichenreihe: Reihe aus Elementen eines Alphabets
bspw. Nutzung als Eingabe- oder Ausgabesequenz endlicher Automaten

↑

Alphabet: Endliche Menge unzerlegbarer Einzelzeichen/Symbole
bspw. Nutzung als Eingabe- oder Ausgabe endlicher Automaten

Anwendungen formaler Sprachen

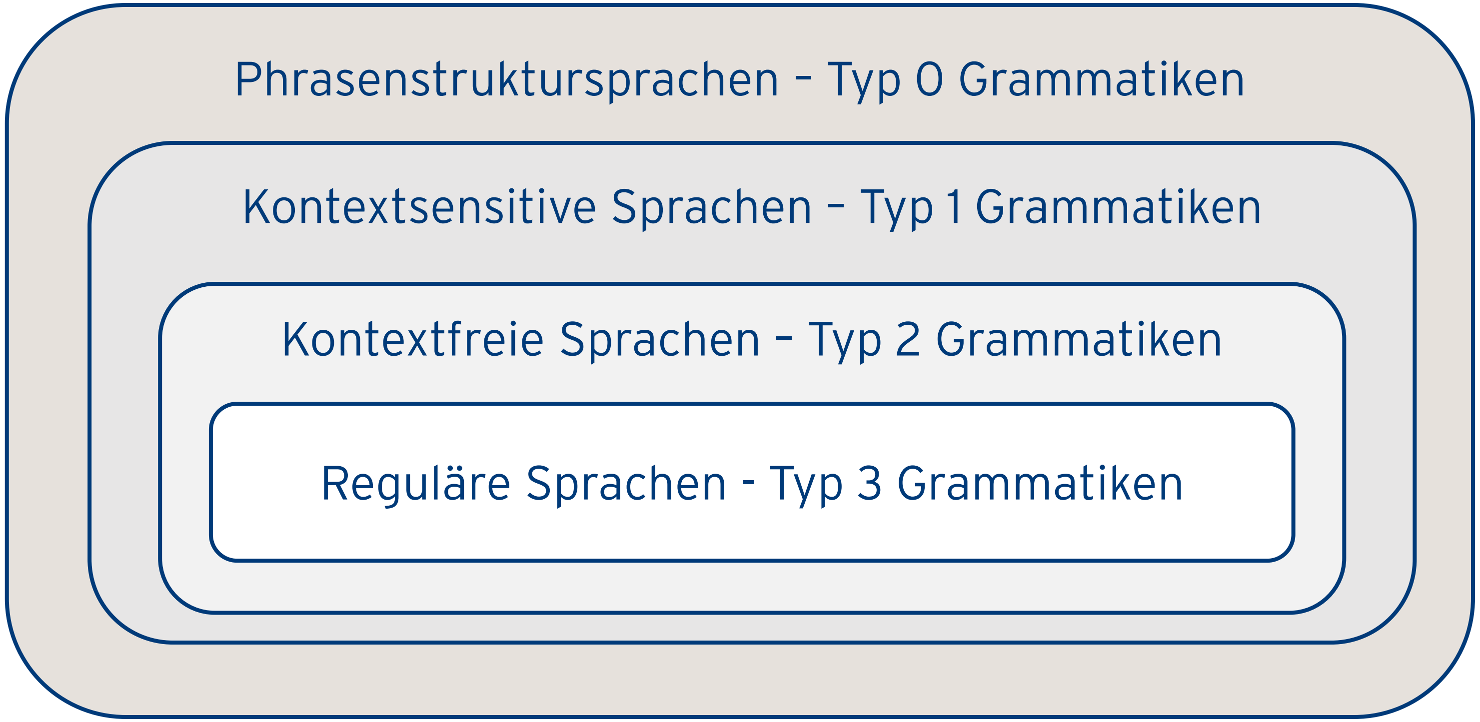
Wofür werden Zeichen, Zeichenketten und Sprachen benötigt?

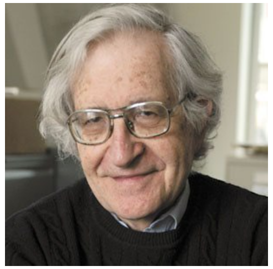


Chomsky-Hierarchie formaler Sprachen

Formale Sprachen werden gemäß der Chomsky-Hierarchie klassifiziert.

Details dazu folgen später.





Avram
Noam
Chomsky

Abschnitt 1

Alphabete

Alphabete

Definition:

Ein **Alphabet** $\boldsymbol{\Sigma}$ ist eine endliche, **abgeschlossene** Menge von vereinbarten, eindeutigen und unterscheidbaren Zeichen bzw. Symbolen.

Allgemeine Notation:

$$\boldsymbol{\Sigma = \{a_1, \dots, a_m\}} \text{ mit } \boldsymbol{a_i \neq a_j, i, j \in \mathbb{N} \wedge i \neq j}$$
$$\boldsymbol{0 \leq |\Sigma| < +\infty}$$

Beispiele:

- $\boldsymbol{\Sigma_1 = \{a_1, a_2, a_3\}}$ mit $\boldsymbol{a_1 = \alpha, a_2 = \beta, a_3 = \gamma}$
- $\boldsymbol{\Sigma_2 = \{a, b, c, d, e, f, \dots, x, y, z\}}$

Definition von Alphabeten (1)

Anmerkung:

- Alphabete lassen sich **extensional** mit Angabe ihrer Elemente darstellen, z.B. $\boldsymbol{\Sigma_3} = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}, \mathbf{6}, \mathbf{7}, \mathbf{8}, \mathbf{9}\}$. Alternativ gilt auch eine suggestive Darstellung à la $\boldsymbol{\Sigma_3} = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \dots, \mathbf{9}\}$, sofern Eindeutigkeit besteht.
- Eleganter ist die **intensionale** Angabe einer eingrenzenden Eigenschaft $\boldsymbol{\xi}(\boldsymbol{a_i})$, sofern eine Teilmengenbeziehung $\boldsymbol{\Sigma} \subseteq \boldsymbol{A}$ besteht, siehe nächste Folie.

Definition von Alphabeten (2)

Prinzip der Aussonderung:

Die Definition eines Alphabets mithilfe einer Eigenschaft $\xi(a_i)$ ist stets dann sinnvoll, falls die Elemente von Σ eine Teilmenge einer bestimmten Obermenge $A = \{a_1, \dots, a_n\}$ darstellen.

So gilt allgemein:

$$\Sigma = \{a \in A | \xi(a)\}$$

Beispiel:

Gegeben sei die Obermenge $\mathbb{N}^+$. So gilt für das Alphabet $\Sigma_4 = \{a \in \mathbb{N}^+ | \xi(a)\}$ mit $\xi(a) = \mathbf{a \bmod 2 = 0} \wedge a < \mathbf{10}$:
$\Sigma_4 = \{2, 4, 6, 8\}$

Für solche Alphabete gilt also:

$$\Sigma \subseteq A \text{ oder}$$

$$\Sigma \in P(A) \text{ (Potenzmenge von } A\text{, vgl. Folie 15)}$$

Kardinalität von Alphabeten

Falls

$$|\boldsymbol{\Sigma}| = \mathbf{0}\ ,$$

dann gilt

$$\boldsymbol{\Sigma} = \emptyset\ ,$$

das heißt, das Alphabet $\boldsymbol{\Sigma}$ ist leer.

Lexikographische Ordnung auf Zeichen

Für die Darstellung der Elemente aus $\boldsymbol{\Sigma}$ bietet sich die Nutzung einer **lexikographischen Ordnung** $\boldsymbol{R} \subseteq \boldsymbol{\Sigma} \times \boldsymbol{\Sigma}$ mit $\mathbf{R} = \prec$ an:

$$\boldsymbol{a_i \prec a_{i+1} \ldots \prec a_{n-1}, i, n} \in \boldsymbol{I} \subseteq \mathbb{N}$$

Die mithilfe der Indexmenge $\boldsymbol{I}$ indizierten Elemente $\boldsymbol{a_i}$ werden anhand eines vereinbarten Kriteriums sortiert (Anmerkung: Es handelt sich um einzelne Zeichen $\boldsymbol{a_i}$).

Beispiel Binäralphabet:

$$\boldsymbol{B = \{0, 1\}, a_1 = 0, a_2 = 1}$$

Die **Relation** $\boldsymbol{R}$ ist **transitiv**, was ebenfalls für die Inverse $\boldsymbol{R}^{-1}$ bzw. $\succ$ gilt.

$$\boldsymbol{a_i R a_{i+n}} \wedge \boldsymbol{a_{i+n} R a_{i+m}} \Rightarrow \boldsymbol{a_i R a_{i+m}} \text{ bzw. } \boldsymbol{a_i \prec a_{i+n}} \wedge \boldsymbol{a_{i+n} \prec a_{i+m}} \Rightarrow \boldsymbol{a_i \prec a_{i+m}}$$

Beispiel:

$$\boldsymbol{1R2} \ \wedge \ \boldsymbol{2R5} \Rightarrow \boldsymbol{1R5} \text{ bzw. } \boldsymbol{1 \prec 2} \ \wedge \ \boldsymbol{2 \prec 5} \Rightarrow \boldsymbol{1 \prec 5}$$

Abgeschlossenheit von Alphabeten

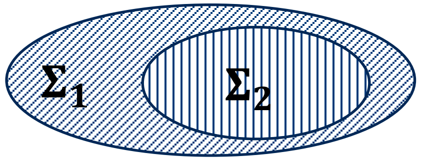
Aufgrund ihres Mengencharakters können Alphabete mit den üblichen Operationen behandelt werden.

Differenzmengenoperation:

$$\Sigma_1 \setminus \Sigma_2 = \Sigma_3$$

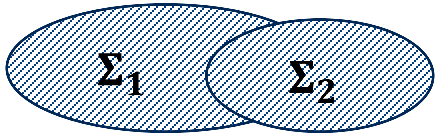
Vereinigungsoperation:

$$\Sigma_1 \cup \Sigma_2 = \Sigma_3$$



Schnittmengenbildung:

$$\Sigma_1 \cap \Sigma_2 = \Sigma_3$$



Symmetrische Differenz zweier Mengen:

$$\Sigma_1 \triangle \Sigma_2 = (\Sigma_1 \cup \Sigma_2) \setminus (\Sigma_1 \cap \Sigma_2)$$

Dabei gilt:

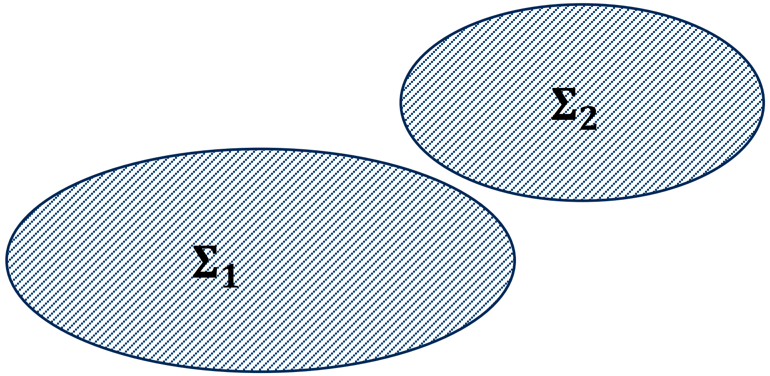
$$|\Sigma_1 \triangle \Sigma_2| = 0 \Rightarrow \Sigma_1 = \Sigma_2$$

Wichtig: Alphabete sind **abgeschlossen gegenüber Mengenoperationen**.

Zeichenmengen

Weiterhin gilt:

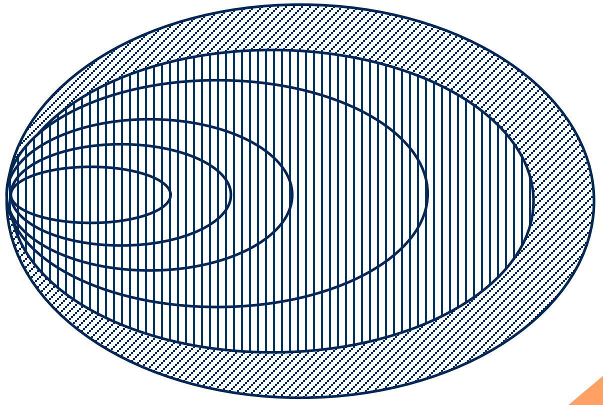
$$|\boldsymbol{\Sigma_1} \cup \boldsymbol{\Sigma_2}| = |\boldsymbol{\Sigma_1}| + |\boldsymbol{\Sigma_2}| \text{ gdw. } \boldsymbol{\Sigma_1} \cap \boldsymbol{\Sigma_2} = \emptyset$$



Die Menge aller möglichen Teilmengen $\boldsymbol{\sigma}$ von $\boldsymbol{\Sigma}$ ergibt sich über das **Potenzmengenaxiom**:

$$\boldsymbol{P}(\boldsymbol{\Sigma}) = \{\boldsymbol{\sigma} | \boldsymbol{\sigma} \subseteq \boldsymbol{\Sigma}\}$$

Dabei gilt:



$$|\boldsymbol{P}(\boldsymbol{\Sigma})| = \boldsymbol{2}^{|\boldsymbol{\Sigma}|}$$

$$\boldsymbol{P}(\boldsymbol{\Sigma} = \emptyset) = \{\emptyset\}$$

(Die Potenzmenge eines Alphabets ist nicht leer.)

Abschnitt 2

Wörter

Wörter

Wörter $\boldsymbol{w}$ (Zeichenketten, Strings) ergeben sich durch die Aneinanderreihung der Elemente aus $\boldsymbol{\Sigma}$.

Beispiel:

$$\boldsymbol{w} = \boldsymbol{abc} \text{ mit } \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \boldsymbol{\Sigma}$$

Formal wird die **Konkatenation** $\circ$ einzelner Zeichen aus $\boldsymbol{\Sigma}$ über die Abbildung

$$\circ: \boldsymbol{\Sigma} \times \boldsymbol{\Sigma} \rightarrow \boldsymbol{\Sigma}^*$$

beschrieben:

$$\boldsymbol{w} = \boldsymbol{a} \circ \boldsymbol{b} \circ \boldsymbol{c}$$

Dabei gilt:

$$\forall\, \boldsymbol{a} \in \boldsymbol{\Sigma}:\ \boldsymbol{\varepsilon} \circ \boldsymbol{a} = \boldsymbol{a} \circ \boldsymbol{\varepsilon} = \boldsymbol{a}$$

D.h., das leere Wort $\boldsymbol{\varepsilon}$ (international auch: $\boldsymbol{\lambda}$) ist das **Neutralelement der Verkettungsoperation**.
Das leere Wort $\boldsymbol{\varepsilon}$ dient dabei als theoretisches Konstrukt, das für weiterführende Überlegungen von Belang ist.

Anmerkung:
Jedes Wort $\boldsymbol{w}$ lässt sich als Verkettung beliebiger Zeichen $\boldsymbol{a} \in \boldsymbol{\Sigma}$ mit dem leeren Wort $\boldsymbol{\varepsilon}$ in beliebiger Häufigkeit darstellen – z.B. $\boldsymbol{w} = \boldsymbol{a_1 \varepsilon a_2 \varepsilon\varepsilon\varepsilon a_1 \varepsilon} = \boldsymbol{a_1 a_2 a_1}$.

Wörter – Konkatenation (1)

Die wiederholte Konkatenation eines Zeichens $\boldsymbol{a} \in \boldsymbol{\Sigma}$ mit sich selbst führt zur **Potenz eines Zeichens**: $\boldsymbol{a}^{\mathrm{n}}$

Beispiel: $\boldsymbol{a} \circ \boldsymbol{a} = \boldsymbol{a}^2$

Analog zur Arithmetik gilt:

$$\boldsymbol{a}^0 = \boldsymbol{\varepsilon} \equiv \boldsymbol{x}^0 = \mathbf{1}$$

Die rekursive Definition der Konkatenation lässt sich wie folgt ausdrücken:

$$\boldsymbol{a}^1 = \boldsymbol{a}^0 \circ \boldsymbol{a}^1 = \boldsymbol{a}$$

$$\boldsymbol{a}^2 = \boldsymbol{a}^0 \circ \boldsymbol{a}^1 \circ \boldsymbol{a}^1 = \boldsymbol{a} \circ \boldsymbol{a}^1 = \boldsymbol{a} \circ (\boldsymbol{a}^0 \circ \boldsymbol{a}^1) = \boldsymbol{aa}$$

D.h.,

$$\boldsymbol{a}^{n+1} = \boldsymbol{a}^n \circ \boldsymbol{a}^1$$

Für die Verkettung unterschiedlicher Zeichen gelten dieselben Regeln und Notationen

$$\boldsymbol{a}^{n+1}\boldsymbol{b}^{n+1} = \boldsymbol{a}^n \circ \boldsymbol{a}^1 \circ \boldsymbol{b}^n \circ \boldsymbol{b}^1$$

Potenzen unterschiedlicher Zeichen können konkateniert werden, z.B. $\boldsymbol{a}^n\boldsymbol{b}^{n+m}\boldsymbol{c}^{n+t}$

Wörter – Konkatenation (2)

Beispiel:

$$\boldsymbol{\Sigma} = \{\boldsymbol{a_1}, \boldsymbol{a_2}, \boldsymbol{a_3}\}$$

mit

$$\boldsymbol{a_1} = \S, \boldsymbol{a_2} = !, \boldsymbol{a_3} = \text{ß}$$

So ergibt sich mit

$$\boldsymbol{a_1}^{\boldsymbol{5}} = \S\S\S\S\S$$
$$\boldsymbol{a_2}^{\boldsymbol{3}} = !!!$$
$$\boldsymbol{a_3}^{\boldsymbol{7}} = \text{ßßßßßßß}$$

für das Wort $\boldsymbol{w}$ folgende Darstellung:

$$w = \boldsymbol{a_2}^{\boldsymbol{3}} \circ \boldsymbol{a_1}^{\boldsymbol{5}} \circ \boldsymbol{a_3}^{\boldsymbol{7}} = !!!\,\S\S\S\S\S\text{ßßßßßßß}$$

Kleenesche Hülle

Die Potenzbildung eines Zeichens lässt sich auch auf Zeichenmengen übertragen:

$$\boldsymbol{\Sigma^0 = \{\varepsilon\}}$$

$$\boldsymbol{\Sigma^1 = \Sigma \circ \Sigma^0 = \{\varepsilon a_1 \dots \varepsilon a_n\} = \{a_1 \dots a_n\}}, \text{ wenn } \boldsymbol{\Sigma = \{a_1, \dots, a_n\}}$$

$$\boldsymbol{\Sigma^2 = \Sigma^1 \circ \Sigma^1 = \{a_1 a_1 \dots a_n a_n\}}$$

$$\boldsymbol{\Sigma^3 = \Sigma \circ \Sigma^2 = \{a_1 a_1 a_1 \dots a_n a_n a_n\}}$$

$$\boldsymbol{\Sigma^n = \Sigma^{n-1} \circ \Sigma = \{\dots\}}$$

Die Vereinigung aller Potenzen über $\boldsymbol{\Sigma}$ bildet den **Kleene-Abschluss** (**Kleenesche Hülle**, **Sternhülle**):

$$\boldsymbol{\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^{\,n}}$$

Für die Menge aller Wörter (**Plushülle**) ohne das leere Wort gilt:

$$\boldsymbol{\Sigma^* \setminus \{\varepsilon\} = \Sigma^+}$$

Wichtig: $\{\boldsymbol{\varepsilon}\} \neq \emptyset$

Wortoperationen – Konkatenation

Die Konkatenation lässt sich auch auf Elemente $\boldsymbol{w} \in \boldsymbol{\Sigma}^*$ anwenden.

$$\boldsymbol{w_3 = w_1 \circ w_2}$$

Beispiel: mit $\boldsymbol{w_1} = \S^5$ und $\boldsymbol{w_2} = !^2$ ist $\boldsymbol{w_3 = w_1 \circ w_2} = \S^5 \circ !^2$

Auch gilt hier

$$\boldsymbol{{w_3}^0 = \varepsilon}$$

bzw.

$$\forall \boldsymbol{w} \in \boldsymbol{\Sigma}^*: \boldsymbol{w^0 = \varepsilon}$$

Somit bildet die algebraische Struktur $(\boldsymbol{\Sigma}^*, \circ)$ ein **Monoid** bzw. eine Halbgruppe **mit einem Neutralelement**, da

- Innere zweistellige Verknüpfung: $\circ: \boldsymbol{\Sigma}^* \times \boldsymbol{\Sigma}^* \to \boldsymbol{\Sigma}^*$
- Assoziativität der Operation: $\forall \boldsymbol{a, b, c} \in \boldsymbol{\Sigma}^*: \boldsymbol{(a \circ b) \circ c = a \circ (b \circ c)}$
- Neutralelement: $\boldsymbol{a \circ \varepsilon = a}$

Wörter – Ordnung (1)

Analog zur Darstellung von $\boldsymbol{\Sigma}$ unterliegen auch die Elemente von $\boldsymbol{\Sigma}^*$ einer **lexikographischen Ordnung** $\prec$:

$$\prec \subseteq \boldsymbol{\Sigma}^* \times \boldsymbol{\Sigma}^*$$

So gilt für die Elemente $\boldsymbol{v}, \boldsymbol{w} \in \boldsymbol{\Sigma}^*$

$$\boldsymbol{v} = \boldsymbol{v}_{\boldsymbol{i}=\mathbf{1}} \dots \boldsymbol{v}_{\boldsymbol{i}=\boldsymbol{m}},\ \boldsymbol{v}_{\boldsymbol{i}} \in \boldsymbol{\Sigma}, \mathbf{1} \leq \boldsymbol{i} \leq \boldsymbol{m}, \qquad \boldsymbol{w} = \boldsymbol{w}_{\boldsymbol{j}=\mathbf{1}} \dots \boldsymbol{w}_{\boldsymbol{j}=\boldsymbol{n}}, \boldsymbol{w}_{\boldsymbol{j}} \in \boldsymbol{\Sigma}, \mathbf{1} \leq \boldsymbol{j} \leq \boldsymbol{n}$$

$$\boldsymbol{v} \prec \boldsymbol{w} \text{ bzw. } (\boldsymbol{v}, \boldsymbol{w}) \in \prec$$

genau dann, wenn

$$\boldsymbol{m} < \boldsymbol{n}$$

D.h., die Sortierung erfolgt gemäß der Länge der Wörter.

Beispiel für $\boldsymbol{L} \subseteq \boldsymbol{\Sigma}^*$ mit $\boldsymbol{\Sigma} = \{\boldsymbol{a}\}$:

$$\prec_{\boldsymbol{\Sigma}^*} = \{(\boldsymbol{a}, \boldsymbol{aa}), (\boldsymbol{a}, \boldsymbol{aaa}), \dots, (\boldsymbol{aa}, \boldsymbol{aaa}), \dots, (\boldsymbol{aaa}, \boldsymbol{aaaa}), \dots\}$$
$$\boldsymbol{L} = \{\boldsymbol{a}, \boldsymbol{aa}, \boldsymbol{aaa}, \boldsymbol{aaaa}\}$$

Die Relation $\prec_{\boldsymbol{\Sigma}^*}$ ist ebenfalls transitiv.

Wörter – Ordnung (2)

Ferner gilt für die Elemente $\boldsymbol{v}, \boldsymbol{w} \in \boldsymbol{\Sigma}^*$

$$\boldsymbol{v = v_1 \dots v_m,\ v_i \in \Sigma, 1 \leq i \leq m, \qquad w = w_1 \dots w_n, w_j \in \Sigma, 1 \leq j \leq n}$$

$\boldsymbol{v} \prec \boldsymbol{w}$ bzw. $(\boldsymbol{v}, \boldsymbol{w}) \in \prec$ genau dann,

falls $\boldsymbol{m = n = 1}$ und $\boldsymbol{v_1 \prec w_1}$ oder

$\boldsymbol{m = n > 1}$ und es ein $\boldsymbol{k}$ gibt, welches folgende Kriterien erfüllt:

$$\boldsymbol{1 \leq k < m}$$

und $\boldsymbol{v_1 \dots v_k = w_1 \dots w_k}$ und $\boldsymbol{v_{k+1} \prec w_{k+1}}$

Das heißt, Wörter $\boldsymbol{w} \in \boldsymbol{L}$ gleicher Länge werden **stellenwertig** hinsichtlich der enthaltenen Alphabetzeichen $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{\Sigma}$ sortiert.

Beispiel:

$$\boldsymbol{\Sigma = \{\mathrm{d}, \mathrm{e}\}}$$

$$\boldsymbol{L = \{d, e, dd, de, ed, ee, ddd, \dots\}}$$

Als Alternative gibt es die bekannte **lexikalische Ordnung** (wie im Lexikon):
Wörter werden unabhängig von ihrer Länge nach alphabetischer Reihenfolge sortiert.

Wortoperationen – Länge

Die Länge eines Wortes $\boldsymbol{w}$ lässt sich mit der rekursiv definierten Funktion $\boldsymbol{len}$ bestimmen:

$$\boldsymbol{len}: \boldsymbol{\Sigma}^* \rightarrow \mathbb{N}_\mathbf{0}$$

$$\boldsymbol{len}(\boldsymbol{aw}) = \boldsymbol{len}(\boldsymbol{w}) + \mathbf{1}, \boldsymbol{a} \in \boldsymbol{\Sigma}, \boldsymbol{w} \in \boldsymbol{\Sigma}^*$$

Dabei gilt:

$$\boldsymbol{len}(\boldsymbol{\varepsilon}) = \mathbf{0}$$

Beispiel:

$$\boldsymbol{w} = \boldsymbol{abc}$$

$$\boldsymbol{len}(\boldsymbol{abc}) = \boldsymbol{len}(\boldsymbol{bc}) + \mathbf{1} = \boldsymbol{len}(\boldsymbol{c}) + \mathbf{1} + \mathbf{1} = \boldsymbol{len}(\boldsymbol{\varepsilon}) + \mathbf{1} + \mathbf{1} + \mathbf{1} = \mathbf{0} + \mathbf{1} + \mathbf{1} + \mathbf{1}$$

$$\boldsymbol{len}(\boldsymbol{abc}) = \mathbf{3}$$

Zeichenketten/Wörter/Strings werden sequentiell abgearbeitet.

Alternativ:
Die Länge eines Wortes $\boldsymbol{w}$ kann auch über die Nutzung von Betragsstrichen $|\boldsymbol{w}|$ angezeigt werden.

Wörter – Prä-, In- und Suffix (1)

Aufbau eines Wortes/Strings $\boldsymbol{w}$:

$$(\boldsymbol{w} \in \boldsymbol{\Sigma}^*) \wedge (\boldsymbol{w} = \boldsymbol{xyz}; \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \boldsymbol{\Sigma}^*)$$

Dabei ist $\boldsymbol{x}$ das **Präfix** des Wortes, $\boldsymbol{y}$ das **Infix** des Wortes und $\boldsymbol{z}$ das **Suffix** des Wortes $\boldsymbol{w}$.
Infixe, Suffixe und Präfixe sind als Worte $\boldsymbol{w}$ Elemente des Kleene-Stern-Produkts über $\boldsymbol{\Sigma}$.

$\boldsymbol{x}$ ist **echtes Präfix** von $\boldsymbol{w}$, wenn gilt:

$$\boldsymbol{x} \text{ ist Präfix von } \boldsymbol{w} \text{ und } \boldsymbol{x} \neq \boldsymbol{w}.$$

Analog dazu gelten auch die Definitionen für **echte** Infixe und Suffixe.

Wörter – Prä-, In- und Suffix (2)

Sonderfälle:

- Falls $\boldsymbol{x = w \rightarrow (x = Präfix, x = Infix\, und\, x = Suffix)}$
- Falls $\boldsymbol{w = xy \rightarrow (x = Präfix, (x = Infix \vee y = Infix), y = Suffix)}$

Präfixfreiheit: Eine Sprache $\boldsymbol{L \subseteq \Sigma^*}$ heißt **präfixfrei**, falls

$$\boldsymbol{\forall w \in L : w = xyz \ \wedge x \notin L}$$

Beispiele:

- $\boldsymbol{L_1 = \{a^n b^n | n \geq 1\}}$
  Sprache $\boldsymbol{L_1}$ ist präfixfrei, da für alle echten Präfixe $\boldsymbol{a^n b^i}$ für Elemente $\boldsymbol{w}$ aus $\boldsymbol{L_1}$ mit $\boldsymbol{n > i}$ gilt: $\boldsymbol{a^n b^i \notin L_1}$.
- $\boldsymbol{L_2 = \{w \in \{a, b\}^* || w|_a = |w|_b\}}$
  Sprache $\boldsymbol{L_2}$ hingegen enthält echte Präfixe ($\boldsymbol{w = ababab}$, $\boldsymbol{Präfix = ab}$).

Allgemein gilt, falls $\boldsymbol{\varepsilon \notin L}$:

$$\boldsymbol{L \cap \{L \circ \Sigma^+\} = \emptyset}$$

Wörter – Teilwörter (1)

Indizierung von Einzelzeichen:

Will man einzelne Zeichen eines Wortes $\boldsymbol{w}$ anzeigen oder Teilstrings $\boldsymbol{s}$ des Wortes, bietet sich die Verwendung eines Laufindex $\boldsymbol{i}$ an.

$$\boldsymbol{w = w_{i=1}w_{i=2}w_{i=3}, \dots, w_{i=n}}$$

Beispiel:

$$\boldsymbol{w = abbabba}$$
$$\boldsymbol{w_3 = b}$$
$$\boldsymbol{w_{2:5} = bbab}$$

Möchte man anzeigen, dass ein wiederkehrendes Element gemeint ist (z. B. jedes zweite oder dritte Element des Wortes $\boldsymbol{w}$), lässt sich folgende Notation verwenden:

$$\boldsymbol{w_{i \bmod j=0} = a \in \Sigma}$$

Beispiel:

$$\boldsymbol{\Sigma = \{a, b, c, d\}, L \subseteq \Sigma^+, L = \{w \in \Sigma^+ | w_{i \bmod 2=0} = d\} = \{a, b, c, d, ad, bd, cd, dd, \dots, adad, \dots\}}$$

Wörter – Teilwörter (2)

Finden eines Substring:

$$substr: \Sigma^* \times \Sigma^* \rightarrow \{ja, nein\}$$

mit

$$substr(w, v) = \begin{cases} ja, falls\ v\ Infix\ von\ w \\ nein, sonst \end{cases}$$

Beispiel:

$$w = \ aaabbbccc$$
$$substr(w, bbb) = ja$$
$$substr(w, aba) = nein$$

Beispiel in Java-Syntax:

```java
String w = "Haustuer";
String s = "tu";
boolean sIsInfixOfW = w.contains(s);
```

Wörter – Teilwörter (3)

Ausgeben eines Substring:

$$\boldsymbol{substr}: \boldsymbol{\Sigma}^* \times \mathbb{N} \times \mathbb{N} \rightarrow \boldsymbol{\Sigma}^*$$

mit

$$\boldsymbol{substr}(\boldsymbol{w}, \boldsymbol{i}, \boldsymbol{j}) = \begin{cases} \boldsymbol{w_i} \ldots \boldsymbol{w_j}, \text{falls } \boldsymbol{i} \leq \boldsymbol{j} \\ \boldsymbol{\varepsilon}, \text{sonst} \end{cases}$$

$$\text{für } \boldsymbol{w} = \boldsymbol{w_1} \ldots \boldsymbol{w_n}, \boldsymbol{1} \leq \boldsymbol{i} \leq \boldsymbol{n}, \boldsymbol{1} \leq \boldsymbol{j} \leq \boldsymbol{n}.$$

Beispiel:

$$\boldsymbol{w} = \boldsymbol{aabaaabbb}, \boldsymbol{substr}(\boldsymbol{w}, \boldsymbol{3}, \boldsymbol{7}) = \boldsymbol{baaab}$$

Beispiel in Java-Syntax:

```
String w = "Haustuer";
String s = w.substring(0, 4); // Achtung: Index des ersten Zeichens ist 0.
System.out.print(s);          // Erster Index inklusive, zweiter exklusive
```

Wörter –Teilwörtern (4)

Zählen von Elementen in w: $anz: \Sigma^* \times \Sigma \to \mathbb{N}_0$

Mit $$anz(aw, b) = \begin{cases} anz(w, b) + 1, a = b \\ anz(w, b), a \neq b \end{cases}$$

Ferner gilt: $\forall b \in \Sigma\colon\ anz(\varepsilon, b) = 0$

Alternative Notation: $anz(w, a) = |w|_a$

Beispiel: $w = aabccaac$

$anz(w, c) = anz(abccaac, c) = anz(bccaac, c) = anz(ccaac, c) = anz(caac, c) + 1 = anz(aac, c) + 1 + 1$
$= anz(ac, c) + 1 + 1 = anz(c, c) + 1 + 1 = anz(\varepsilon, c) + 1 + 1 + 1 = 0 + 1 + 1 + 1 = 3$

Beispiel in Java-Syntax:

```
String w = "Haustueren"; int numOfChars = 0;
for (int i = 0; i < w.length(); i++)
    if (w.charAt (i) == 'e') numOfChars++;
System.out.print (numOfChars);
```

Wörter – Modifikation

Austauschen von Symbolen im String:

$$\boldsymbol{tausche}: \boldsymbol{\Sigma}^* \times \boldsymbol{\Sigma} \times \boldsymbol{\Sigma} \to \boldsymbol{\Sigma}^*$$

Definiert wird die Funktion mittels:

$$\boldsymbol{tausche}(\boldsymbol{aw}, \boldsymbol{b}, \boldsymbol{c}) = \begin{cases} \boldsymbol{b} \circ \boldsymbol{tausche}(\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{c}), \boldsymbol{a} = \boldsymbol{c} \\ \boldsymbol{a} \circ \boldsymbol{tausche}(\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{c}), \boldsymbol{a} \neq \boldsymbol{c} \end{cases}$$

Ferner gilt:

$$\boldsymbol{tausche}(\boldsymbol{\varepsilon}, \boldsymbol{b}, \boldsymbol{c}) = \boldsymbol{\varepsilon}$$

Beispiel in Java-Syntax:

```java
String w = "Haustueren";
w = w.replaceAll("a", "e"); // Achtung: Parameter in umgekehrter Reihenfolge
System.out.print(w);
```

Abschnitt 3

Formale Sprachen

Formale Sprachen

Eine Teilmenge von $\boldsymbol{\Sigma}^*$ wird als **formale Sprache** $\boldsymbol{L}$ über $\boldsymbol{\Sigma}$ bezeichnet, $\boldsymbol{L} \subseteq \boldsymbol{\Sigma}^*$

Beispiel:

$$\boldsymbol{\Sigma} = \{\mathbf{a}, \mathbf{b}\}$$
$$\boldsymbol{\Sigma}^* = \{\boldsymbol{\varepsilon}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{aa}, \boldsymbol{ab}, \boldsymbol{ba}, \boldsymbol{bb}, \boldsymbol{aaa} \ldots.\}$$
$$\boldsymbol{L} = \{\boldsymbol{\varepsilon}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{aa}, \boldsymbol{ab}, \boldsymbol{ba}, \boldsymbol{bb}, \boldsymbol{aaa}\}$$

Sprachen $\boldsymbol{L}$ können mit den üblichen Methoden der Mengenlehre behandelt werden. So gelten z.B. auch

$$\boldsymbol{L_3} = \boldsymbol{L_1} \cup \boldsymbol{L_2}$$

oder

$$\boldsymbol{L_3} = \boldsymbol{L_1} \cap \boldsymbol{L_2}$$

oder

$$\boldsymbol{L_3} = \boldsymbol{L_1} \backslash \boldsymbol{L_2}$$

oder

$$\boldsymbol{L_3} = \boldsymbol{L_1} \triangle \boldsymbol{L_2}$$

Formale Sprachen – Konkatenation

Die Konkatenation der Sprachen L_i erfolgt über die Konkatenation der Elemente:

$$L_n \circ L_m = \{vw | v \in L_n, w \in L_m\}$$

Beispiel:

$$L_n = \{aa, ab\}$$
$$L_m = \{\varepsilon, cc, da\}$$
$$L_n \circ L_m = \{aa, ab, aacc, aada, abcc, abda\}$$

Allgemein gilt:

$$L_n \subseteq L_n \circ L_m \text{ gdw. } \varepsilon \in L_m$$

Mehrfach wiederholte Konkatenationen über Sprachen L_i lassen sich als Potenzen ausdrücken:

$$L^0 = \{\varepsilon\}$$
$$L^1 = L^1 \circ L^0 = L^1 = L$$

An dieser Stelle ist $\{\varepsilon\}$ das Neutralelement:

$$L \circ \{\varepsilon\} = L$$

Formale Sprachen – Konkatenation

Allgemein gilt:

$$L^{n+1} = L^n \circ L^1 = L^n \circ L$$

Das Kleene-Stern-Produkt ist die Vereinigung aller Potenzen einer Sprache L über Σ:

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

Unter Aussonderung der leeren Sprache $L = \{\varepsilon\}$ gilt

$$\bigcup_{i=0}^{\infty} L^i \setminus \{\varepsilon\} = L^+$$

$$\forall L^i \colon L^i \subseteq L^* \subseteq \Sigma^*$$

Die Potenzmenge $P(\Sigma^*)$ ist somit gegenüber der Operation

$$\circ \colon P(\Sigma^*) \times P(\Sigma^*) \rightarrow P(\Sigma^*)$$

abgeschlossen ($L \in P(\Sigma^*)$).

Formale Sprachen – Spiegelung

Die **Spiegelung** von Elementen aus $\boldsymbol{L}$ lässt sich wie folgt formulieren:

$$\boldsymbol{mirr}: \boldsymbol{\Sigma}^* \to \boldsymbol{\Sigma}^*$$

mit der Definition

$$\boldsymbol{mirr}(\boldsymbol{aw}) = \boldsymbol{mirr}(\boldsymbol{w}) \circ \boldsymbol{a}, \boldsymbol{w} \in \boldsymbol{\Sigma}^*, \boldsymbol{a} \in \boldsymbol{\Sigma}$$

$$\boldsymbol{mirr}(\boldsymbol{\varepsilon}) = \boldsymbol{\varepsilon}$$

Verallgemeinert auf Sprachen ergibt sich:

$$\boldsymbol{mirr}: \boldsymbol{P}(\boldsymbol{\Sigma}^*) \to \boldsymbol{P}(\boldsymbol{\Sigma}^*)$$

Damit gilt:

$$\boldsymbol{mirr}(\boldsymbol{L}) = \{\mathbf{mirr}(\boldsymbol{w}) | \boldsymbol{w} \in \boldsymbol{L}\}$$

NORDAKADEMIE
HOCHSCHULE DER WIRTSCHAFT

NORDAKADEMIE gAG Hochschule der Wirtschaft
Köllner Chaussee 11 · 25337 Elmshorn · Tel.: +49 (0) 4121 4090-0 · E-Mail: info@nordakademie.de · Web: www.nordakademie.de