Theoretische Informatik

Florin Manea
(basierend auf den Folien von Carsten Damm)

Stand: 22. April 2025

Jetzt:

1. Einführung
   - Bücher
   - Motivation
   - Symbole, Wörter und Sprachen
   - Operationen auf Sprachen
   - Sprachklassen
   - Reguläre Ausdrücke

Symbol und Alphabet

Symbole (Zeichen)

= kleinste bedeutungstragende Einheiten der Kommunikation.
Bedingung: Unterscheidbarkeit von anderen Zeichen.
Beispiele: 0, 1, a, ...
Sogar Grafiken oder Wörter/Wortgruppen können Zeichen sein, wenn sie stellvertretend für irgendeinen „Sinn“ stehen.
Beispiele: 🚹, ☕, STOP, Straßenbahn kreuzt, ...

Symbol und Alphabet

Symbole (Zeichen)

= kleinste bedeutungstragende Einheiten der Kommunikation.
Bedingung: Unterscheidbarkeit von anderen Zeichen.
Beispiele: 0, 1, a, ...
Sogar Grafiken oder Wörter/Wortgruppen können Zeichen sein, wenn sie stellvertretend für irgendeinen „Sinn“ stehen.
Beispiele: 🚹, ☕, STOP, Straßenbahn kreuzt, ...

Alphabet

Ein Alphabet ist eine nichtleere, endliche Menge von Symbolen.
Wir benutzen Σ als Bezeichnung(!) für ein vereinbartes Alphabet. Kommen weitere ins Spiel, so können wir sie mit $\Sigma_1, \Sigma_2, \Sigma', \ldots$ unterscheiden.

Wörter und Operationen auf Wörtern

Wörter

Ein Wort (String) über Σ ist eine endliche Folge w von Zeichen aus Σ. Die Länge von w wird mit $|w|$ bezeichnet. Beispiel: 101001 hat Länge 6.

Wörter und Operationen auf Wörtern

Wörter

Ein Wort (String) über Σ ist eine endliche Folge w von Zeichen aus Σ. Die Länge von w wird mit $|w|$ bezeichnet. Beispiel: 101001 hat Länge 6.
Das leere Wort ε ist die Zeichenfolge der Länge 0.

Wörter und Operationen auf Wörtern

Wörter

Ein Wort (String) über Σ ist eine endliche Folge w von Zeichen aus Σ. Die Länge von w wird mit $|w|$ bezeichnet. Beispiel: 101001 hat Länge 6.
Das leere Wort ε ist die Zeichenfolge der Länge 0.

Konkatenation von Wörtern

Die Konkatenation (Verkettung) von Wörtern $u, v \in \Sigma$ ist die Operation $(u, v) \mapsto uv$, wobei $uv = u \cdot v$ die Hintereinanderschreibung angibt.

Wörter und Operationen auf Wörtern

Wörter

Ein Wort (String) über Σ ist eine endliche Folge w von Zeichen aus Σ. Die Länge von w wird mit $|w|$ bezeichnet. Beispiel: 101001 hat Länge 6.
Das leere Wort ε ist die Zeichenfolge der Länge 0.

Konkatenation von Wörtern

Die Konkatenation (Verkettung) von Wörtern $u, v \in \Sigma$ ist die Operation $(u, v) \mapsto uv$, wobei $uv = u \cdot v$ die Hintereinanderschreibung angibt.
Offenbar gilt $|uv| = |u| + |v|$ und $u\varepsilon = \varepsilon u = u$ für alle Wörter u, v.

Wörter und Operationen auf Wörtern

Wörter

Ein Wort (String) über Σ ist eine endliche Folge w von Zeichen aus Σ. Die Länge von w wird mit $|w|$ bezeichnet. Beispiel: 101001 hat Länge 6.
Das leere Wort ε ist die Zeichenfolge der Länge 0.

Konkatenation von Wörtern

Die Konkatenation (Verkettung) von Wörtern $u, v \in \Sigma$ ist die Operation $(u, v) \mapsto uv$, wobei $uv = u \cdot v$ die Hintereinanderschreibung angibt.
Offenbar gilt $|uv| = |u| + |v|$ und $u\varepsilon = \varepsilon u = u$ für alle Wörter u, v.

Potenzierung

Die n-te Potenz u^n eines Strings u ist die Konkatenation von n Exemplaren von u.
Potenzen beziehen sich nur auf den unmittelbar davor stehenden Bestandteil.
Beispiel: $0001^3 = 000111, (0001)^3 = 000100010001$

Wörter und Operationen auf Wörtern

Wörter

Ein Wort (String) über Σ ist eine endliche Folge w von Zeichen aus Σ. Die Länge von w wird mit $|w|$ bezeichnet. Beispiel: 101001 hat Länge 6.
Das leere Wort ε ist die Zeichenfolge der Länge 0.

Konkatenation von Wörtern

Die Konkatenation (Verkettung) von Wörtern $u, v \in \Sigma$ ist die Operation $(u, v) \mapsto uv$, wobei $uv = u \cdot v$ die Hintereinanderschreibung angibt.
Offenbar gilt $|uv| = |u| + |v|$ und $u\varepsilon = \varepsilon u = u$ für alle Wörter u, v.

Potenzierung

Die n-te Potenz u^n eines Strings u ist die Konkatenation von n Exemplaren von u.
Potenzen beziehen sich nur auf den unmittelbar davor stehenden Bestandteil.
Beispiel: $0001^3 = 000111, (0001)^3 = 000100010001$
Spezialfall: $u^0 = \varepsilon$ für alle Wörter u.

Python-Notation

Begriff	formelhaft	Python
String	$u = 010010001, v = 00$	$\mathtt{u} =' 010010001'$, $\mathtt{v} =' 00'$ oder $\mathtt{u} =$ "010010001", $\mathtt{v} =$ "00"
Länge	$\lvert u \rvert$	`len(u)`
Zeichen	$0, 1, a, \ldots$	$'0', '1', '\mathtt{a}', \ldots$ (Strings der Länge 1)
leeres Wort	ε	$''$
Alphabet	$\{0, 1\}$	`set({'0','1'})`
Konkatenation	$uv = u \cdot v = 01001000100$	$\mathtt{u} + \mathtt{v} ==' 01001000100'$
Potenzierung	u^3	`u * 3`

Teilwörter

Sei w ein Wort über Σ.

Ein Wort y heißt Teilwort (Teilstring, Faktor oder Infix) von w, falls es Wörter x, z gibt mit $w = xyz$.

Teilwörter

Sei w ein Wort über Σ.

Ein Wort y heißt Teilwort (Teilstring, Faktor oder Infix) von w, falls es Wörter x, z gibt mit $w = xyz$.

Präfix und Suffix

y heißt dann Präfix (bzw. Suffix) von w, falls darüberhinaus $x = \varepsilon$ (bzw. $z = \varepsilon$).

Teilwörter

Sei w ein Wort über Σ.

Ein Wort y heißt Teilwort (Teilstring, Faktor oder Infix) von w, falls es Wörter x, z gibt mit $w = xyz$.

Präfix und Suffix

y heißt dann Präfix (bzw. Suffix) von w, falls darüberhinaus $x = \varepsilon$ (bzw. $z = \varepsilon$).

SoSe2020 hat z.B. So als ein Präfix, e2020 als ein Suffix und oSe als Infix.

Sprachen

Kleenesche Hülle

Die Menge aller Wörter über Σ (inklusive ε) wird mit Σ^* bezeichnet (Kleenesche Hülle von Σ).

Sprachen

Kleenesche Hülle

Die Menge aller Wörter über Σ (inklusive ε) wird mit Σ^* bezeichnet (Kleenesche Hülle von Σ). Die Menge der *nichtleeren Wörter* über Σ wird mit Σ^+ bezeichnet (positive Hülle).
Es gilt: $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$

Sprachen

Kleenesche Hülle

Die Menge aller Wörter über Σ (inklusive ε) wird mit Σ^* bezeichnet (Kleenesche Hülle von Σ). Die Menge der *nichtleeren Wörter* über Σ wird mit Σ^+ bezeichnet (positive Hülle).
Es gilt: $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$

Formale Sprache

Jede Teilmenge $L \subseteq \Sigma^*$ wird formale Sprache über Σ genannt. Die Menge kann *leer*, endlich oder unendlich sein.

Sprachen

Kleenesche Hülle

Die Menge aller Wörter über Σ (inklusive ε) wird mit Σ^* bezeichnet (Kleenesche Hülle von Σ). Die Menge der *nichtleeren Wörter* über Σ wird mit Σ^+ bezeichnet (positive Hülle).
Es gilt: $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$

Formale Sprache

Jede Teilmenge $L \subseteq \Sigma^*$ wird formale Sprache über Σ genannt. Die Menge kann *leer*, endlich oder unendlich sein.

Beispiele

Alphabet Σ	Sprache	Beispielwörter	Größe
beliebig	Zero $= \emptyset$	–	0
beliebig	Unit $= \{\varepsilon\}$	ε	1
$\{a, b, c\}$	MyLang $= \{\varepsilon, aa, aba\}$	ε, aa	3
$\{a, \ldots, z\}$	Σ^*	b	∞
$\{a, b, c\}$	Pref$(abc) = \{w : w$ Präfix von $abc\}$	ε, a, ab, abc	4
$\{0, 1\}$	$L_U = \{0^i 1 : i \geq 0\}$	$1, 0001, \underbrace{00\ldots0}_{100}1$	∞
$\{a, b\}$	$L_{\#a<\#b} = \{a^i b^j : i, j \geq 0 \wedge i < j\}$	$b, abb, aabbb, \ldots, a^5 b^{55}$	∞

Sprachen mit Python definieren

- endliche Sprachen können wir durch Auflisten aller Wörter angeben:

```
MyLang = {'', 'aa', 'aba'}
Zero = set()          # leere Menge = leere Sprache
Unit = set({''})      # Sprache, die nur das leere Wort enthält
```

Sprachen mit Python definieren

- endliche Sprachen können wir durch Auflisten aller Wörter angeben:

```
MyLang = {'', 'aa', 'aba'}
Zero = set()          # leere Menge = leere Sprache
Unit = set({''})      # Sprache, die nur das leere Wort enth lt
```

- unendliche Sprachen können wir auf diese Weise höchstens teilweise angeben, z.B. bis zu einer bestimmten Länge

```
L01Rep9 = {'01'*i for i in range(10)} # range(10) = {0,1,2,3,4,5,6,7,8,9}
La_lt_b_9 = {'a'*i + 'b'*j for i in range(10) for j in range(i)}
```

Lexikographische Ordnung auf Σ^*

Sei $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_\ell\}$ mit einer Ordnung auf den Zeichen: $\sigma_1 < \sigma_2 < ... < \sigma_\ell$.

Lexikographische Ordnung auf Σ^*

Sei $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_\ell\}$ mit einer Ordnung auf den Zeichen: $\sigma_1 < \sigma_2 < ... < \sigma_\ell$.
Nun definieren wir „Wort u ist lexikographisch kleiner als Wort v ($u \prec v$)“:

- $u = u_1 u_2 \cdots u_m \prec v = v_1 v_2 \cdots v_n$, falls u Präfix von v ist oder aber $\exists i : u_i < v_i \wedge \forall j < i : u_j = v_j$ (d.h. die Zeichen an der ersten Unterscheidungsstelle stehen in Alphabetordnung)

Python

```
def lexlt(s, t):
    if t == '':
        return False
    if s == '' or s[0] < t[0]:
        return True
    return (s[0] == t[0]) & lexlt(s[1:],t[1:])
```

Lexikographische Ordnung auf Σ^*

Sei $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_\ell\}$ mit einer Ordnung auf den Zeichen: $\sigma_1 < \sigma_2 < ... < \sigma_\ell$.
Nun definieren wir „Wort u ist lexikographisch kleiner als Wort v ($u \prec v$)“:

- $u = u_1 u_2 \cdots u_m \prec v = v_1 v_2 \cdots v_n$, falls u Präfix von v ist oder aber $\exists i : u_i < v_i \wedge \forall j < i : u_j = v_j$ (d.h. die Zeichen an der ersten Unterscheidungsstelle stehen in Alphabetordnung)

Python

```
def lexlt(s, t):
    if t == '':
        return False
    if s == '' or s[0] < t[0]:
        return True
    return (s[0] == t[0]) & lexlt(s[1:],t[1:])
```

Problem

Lexikographische Anordnung ist ungeeignet für systematische Auflistungen:

$$\{0,1\}^* = \{\varepsilon, 0, 00, 000, 0000, 00000, ...\} \text{ ???}$$

Numerische Ordnung auf Σ^*

längen-lexikographisch kleiner als $(<)$

- falls $|u| < |v|$, so $u < v$
- falls $|u| = |v|$, so $u < v$ gdw. $u \prec v$

Beispiel: $\{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, ...\}$

Numerische Ordnung auf Σ^*

längen-lexikographisch kleiner als ($<$)

- falls $|u| < |v|$, so $u < v$
- falls $|u| = |v|$, so $u < v$ gdw. $u \prec v$

Beispiel: $\{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, ...\}$

Längen-lexikographisch = numerisch

Eine Abbildung $f : \mathbb{N}_+ \to M$ mit $M = \{f(1), f(2), ...\}$ heißt Aufzählung von M.
Beispiel: $\{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, ...\}$ entspricht der Aufzählung

$$n \mapsto \text{Binärdarstellung von } n \text{ ohne führende 1.}$$

Anders ausgedrückt: für das n-te Wort w gilt $1w = \mathrm{bin}(n)$.

Numerische Ordnung auf Σ^*

längen-lexikographisch kleiner als ($<$)

- falls $|u| < |v|$, so $u < v$
- falls $|u| = |v|$, so $u < v$ gdw. $u \prec v$

Beispiel: $\{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, ...\}$

Längen-lexikographisch = numerisch

Eine Abbildung $f : \mathbb{N}_+ \to M$ mit $M = \{f(1), f(2), ...\}$ heißt Aufzählung von M.
Beispiel: $\{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, ...\}$ entspricht der Aufzählung

$$n \mapsto \text{Binärdarstellung von } n \text{ ohne führende 1.}$$

Anders ausgedrückt: für das n-te Wort w gilt $1w = \text{bin}(n)$.

Python

```
# Python-Funktion bin, Bsp.: bin(5) == '0b101'
def nthbitstring(n):
    return bin(n)[3:] # Praefix '0b1' weglassen
```

Jetzt:

1. Einführung
   - Bücher
   - Motivation
   - Symbole, Wörter und Sprachen
   - Operationen auf Sprachen
   - Sprachklassen
   - Reguläre Ausdrücke

Produkt zweier Sprachen

= Konkatenationen der Wörter der ersten mit denen der zweiten Sprache:
$L_1 \circ L_2 = L_1 L_2 := \{xy : x \in L_1 \wedge y \in L_2\}$

Produkt zweier Sprachen

= Konkatenationen der Wörter der ersten mit denen der zweiten Sprache:
$L_1 \circ L_2 = L_1 L_2 := \{xy : x \in L_1 \wedge y \in L_2\}$

Python-Funktion

```
def cat(L1, L2):
    return set({x+y for x in L1 for y in L2})
```

Potenzierung von Sprachen

Die leere Sprache $\emptyset$

Für alle Sprachen L gilt: $L \circ \emptyset = \emptyset \circ L = \emptyset$.
Vergleich: $\forall x \in \mathbb{N} : 0 \cdot x = x \cdot 0 = 0$

Die Einheitssprache $\{\varepsilon\}$

Für alle Sprachen L gilt: $L \circ \{\varepsilon\} = \{\varepsilon\} \circ L = L$.
Vergleich: $\forall x \in \mathbb{N} : 1 \cdot x = x \cdot 1 = x$

Potenzierung von Sprachen

Die leere Sprache $\emptyset$

Für alle Sprachen L gilt: $L \circ \emptyset = \emptyset \circ L = \emptyset$.
Vergleich: $\forall x \in \mathbb{N} : 0 \cdot x = x \cdot 0 = 0$

Die Einheitssprache $\{\varepsilon\}$

Für alle Sprachen L gilt: $L \circ \{\varepsilon\} = \{\varepsilon\} \circ L = L$.
Vergleich: $\forall x \in \mathbb{N} : 1 \cdot x = x \cdot 1 = x$

Das motiviert diese Festlegungen

- $L^0 := \{\varepsilon\}$
- $L^n := LL^{n-1}$ für $n > 0$

Insbesondere

$$\emptyset^i = \begin{cases} \{\varepsilon\} & i = 0 \\ \emptyset & \text{sonst.} \end{cases}$$

Potenzierung von Sprachen

Die leere Sprache $\emptyset$

Für alle Sprachen L gilt: $L \circ \emptyset = \emptyset \circ L = \emptyset$.
Vergleich: $\forall x \in \mathbb{N} : 0 \cdot x = x \cdot 0 = 0$

Die Einheitssprache $\{\varepsilon\}$

Für alle Sprachen L gilt: $L \circ \{\varepsilon\} = \{\varepsilon\} \circ L = L$.
Vergleich: $\forall x \in \mathbb{N} : 1 \cdot x = x \cdot 1 = x$

Das motiviert diese Festlegungen

- $L^0 := \{\varepsilon\}$
- $L^n := LL^{n-1}$ für $n > 0$

Insbesondere

$$\emptyset^i = \begin{cases} \{\varepsilon\} & i = 0 \\ \emptyset & \text{sonst.} \end{cases}$$

```
def power(L,n):
    return Unit if n==0 else cat(L,power(L,n-1))
```

Hülle einer formalen Sprache

Ist $L \subseteq \Sigma^*$, so ist die Kleenesche Hülle von L definiert durch

$$L^* = \bigcup_{i \geq 0} L^i.$$

Die positive Hülle von L ist die Menge

$$L^+ = \bigcup_{i > 0} L^i.$$

Hülle einer formalen Sprache

Ist $L \subseteq \Sigma^*$, so ist die Kleenesche Hülle von L definiert durch

$$L^* = \bigcup_{i \geq 0} L^i.$$

Die positive Hülle von L ist die Menge

$$L^+ = \bigcup_{i > 0} L^i.$$

Beispiel

Sei $L = \{0, 11\}$. Dann enthält L^* diese Wörter

$$\varepsilon, 0, 00, 11, 000, 011, 110, 0000, 0011, 0110, 1100, 1111, \ldots$$

und $L^+ = L^* \setminus \{\varepsilon\}$.

Einfache Mengenoperationen

Seien L, L_1, L_2 formale Sprachen über Σ.

Vereinigung und Schnitt

$$L_1 \cup L_2 = \{w : w \in L_1 \vee w \in L_2\}$$
$$L_1 \cap L_2 = \{w : w \in L_1 \wedge w \in L_2\}$$

Einfache Mengenoperationen

Seien L, L_1, L_2 formale Sprachen über Σ.

Vereinigung und Schnitt

$$L_1 \cup L_2 = \{w : w \in L_1 \vee w \in L_2\}$$
$$L_1 \cap L_2 = \{w : w \in L_1 \wedge w \in L_2\}$$

Differenz, symmetrische Differenz und Komplement

$$L_1 \setminus L_2 = \{w : w \in L_1 \wedge w \notin L_2\}$$
$$L_1 \Delta L_2 = (L_1 \setminus L_2) \cup (L_2 \setminus L_1)$$
$$\overline{L} = \Sigma^* \setminus L$$

Beispiele

- Felder des Schachbretts

$$\{A, B, C, D, E, F, G, H\} \circ \{1, 2, 3, 4, 5, 6, 7, 8\}$$

[1] vierstellig

Beispiele

- Felder des Schachbretts

$$\{A, B, C, D, E, F, G, H\} \circ \{1, 2, 3, 4, 5, 6, 7, 8\}$$

- Uhrzeiten

$$(\{2\} \circ \{0, 1, 2, 3\} \cup \{\varepsilon, 1\} \circ \Sigma) \circ \{:\} \circ \{0, 1, ..., 5\} \circ \Sigma, \text{ mit } \Sigma = \{0, 1, ..., 9\}$$

[1] vierstellig

Beispiele

- Felder des Schachbretts

$$\{A, B, C, D, E, F, G, H\} \circ \{1, 2, 3, 4, 5, 6, 7, 8\}$$

- Uhrzeiten

$$(\{2\} \circ \{0, 1, 2, 3\} \cup \{\varepsilon, 1\} \circ \Sigma) \circ \{:\} \circ \{0, 1, ..., 5\} \circ \Sigma, \text{ mit } \Sigma = \{0, 1, ..., 9\}$$

- Schaltjahre[1] (Jahr durch 400 teilbar oder durch 4 aber nicht durch 100)

$$S = VH \cup (V \cap \overline{\Sigma^* \circ \{00\}}),$$

wobei $V = G_1 \cup \Sigma^2 \circ (G \circ G_1 \cup U \circ G_2)$, $VH = V \circ \{00\}$, mit $G_1 = \{0, 4, 8\}, G_2 = \{2, 6\}, G = G_1 \cup G_2, U = \overline{G}$

[1] vierstellig

Σ^* als Monoid

Ein Monoid $(M, \circ, e)$ besteht aus einer Menge M mit assoziativer Verknüpfungsoperation $\circ : M \times M \to M$ und neutralem Element $e \in M$, d.h.

- $\forall a, b, c \in M : a \circ (b \circ c) = (a \circ b) \circ c =: a \circ b \circ c$
- $\forall a \in M : a \circ e = e \circ a = a$.

Σ^* als Monoid

Ein Monoid $(M, \circ, e)$ besteht aus einer Menge M mit assoziativer Verknüpfungsoperation $\circ : M \times M \to M$ und neutralem Element $e \in M$, d.h.

- $\forall a, b, c \in M : a \circ (b \circ c) = (a \circ b) \circ c =: a \circ b \circ c$
- $\forall a \in M : a \circ e = e \circ a = a$.

Beispiele

- $(\Sigma^*, \cdot, \varepsilon)$
- $(\mathbb{N}, +, 0)$ — natürliche Zahlen mit Addition
- $(\mathbb{N}_+, \cdot, 1)$ — positive natürliche Zahlen mit Multiplikation

Σ^* als Monoid

Ein Monoid $(M, \circ, e)$ besteht aus einer Menge M mit assoziativer Verknüpfungsoperation $\circ : M \times M \to M$ und neutralem Element $e \in M$, d.h.

- $\forall a, b, c \in M : a \circ (b \circ c) = (a \circ b) \circ c =: a \circ b \circ c$
- $\forall a \in M : a \circ e = e \circ a = a$.

Beispiele

- $(\Sigma^*, \cdot, \varepsilon)$
- $(\mathbb{N}, +, 0)$ — natürliche Zahlen mit Addition
- $(\mathbb{N}_+, \cdot, 1)$ — positive natürliche Zahlen mit Multiplikation

Freies Monoid

Ein Monoid heißt frei erzeugt von $A \subseteq M$, wenn jedes Element *eindeutig* darstellbar ist als Verknüpfung $a_1 \circ a_2 \circ \cdots \circ a_n$ endlich vieler Elemente aus A.

- $(\Sigma^*, \cdot, \varepsilon)$ ist frei erzeugt von Σ
- $(\mathbb{N}, +, 0)$ ist frei erzeugt von $\{1\}$
- $(\mathbb{N}_+, \cdot, 1)$ ist *nicht* frei erzeugt, z.B. $2 \cdot 3 = 3 \cdot 2$

Homomorphismen

Eine Abbildung $h : \Sigma^* \to \Gamma^*$ heißt Homomorphismus, falls

$$\forall u, v \in \Sigma^* : h(uv) = h(u)h(v).$$

Homomorphismen

Eine Abbildung $h : \Sigma^* \to \Gamma^*$ heißt Homomorphismus, falls

$$\forall u, v \in \Sigma^* : h(uv) = h(u)h(v).$$

Willkürlich gewählte Abbildungen haben diese Eigenschaft nicht.
Beispiel: $\Sigma = \Gamma = \{0, 1\}, h(0) := 0, h(w) = 1$ falls $w \neq 0$.

Homomorphismen

Eine Abbildung $h : \Sigma^* \to \Gamma^*$ heißt Homomorphismus, falls

$$\forall u, v \in \Sigma^* : h(uv) = h(u)h(v).$$

Willkürlich gewählte Abbildungen haben diese Eigenschaft nicht.
Beispiel: $\Sigma = \Gamma = \{0, 1\}, h(0) := 0, h(w) = 1$ falls $w \neq 0$.

Universalität

Sei $h : \Sigma \to \Gamma^*$ beliebig. Da Σ^* frei ist, gibt es genau eine Fortsetzung von h zu einem Homomorphismus $\hat{h} : \Sigma^* \to \Gamma^*$:

$$\hat{h}(w) = \begin{cases} \varepsilon & \text{falls } w = \varepsilon \\ h(w_1)h(w_2)\cdots h(w_n) & \text{falls } w = w_1 w_2 \cdots w_n \wedge \forall i\, w_i \in \Sigma. \end{cases}$$

Homomorphismen

Eine Abbildung $h : \Sigma^* \to \Gamma^*$ heißt Homomorphismus, falls

$$\forall u, v \in \Sigma^* : h(uv) = h(u)h(v).$$

Willkürlich gewählte Abbildungen haben diese Eigenschaft nicht.
Beispiel: $\Sigma = \Gamma = \{0, 1\}, h(0) := 0, h(w) = 1$ falls $w \neq 0$.

Universalität

Sei $h : \Sigma \to \Gamma^*$ beliebig. Da Σ^* frei ist, gibt es genau eine Fortsetzung von h zu einem Homomorphismus $\hat{h} : \Sigma^* \to \Gamma^*$:

$$\hat{h}(w) = \begin{cases} \varepsilon & \text{falls } w = \varepsilon \\ h(w_1)h(w_2)\cdots h(w_n) & \text{falls } w = w_1 w_2 \cdots w_n \wedge \forall i\, w_i \in \Sigma. \end{cases}$$

Konvention

Die Fortsetzung $\hat{h}$ wird meist ebenfalls mit h bezeichnet.

Beispiele

Sei $\Sigma = \{0, 1, 2\}$ und $\Gamma = \{a, b\}$. Wir definieren Homomorphismen h, g durch Fortsetzung:

Beispiele

Sei $\Sigma = \{0, 1, 2\}$ und $\Gamma = \{a, b\}$. Wir definieren Homomorphismen h, g durch Fortsetzung:

- Sei $h : \Sigma^* \to \Gamma^*$ festgelegt durch

$$0 \mapsto a, 1 \mapsto b, 2 \mapsto b.$$

Dann ist z.B. $h(0012) = aabb$ und das Bild von $L_1 = \{0^n 12^{n-1} : n \geq 1\}$ unter h ist

$$h(L_1) = \{a^n b^n : n \geq 1\}.$$

Beispiele

Sei $\Sigma = \{0, 1, 2\}$ und $\Gamma = \{a, b\}$. Wir definieren Homomorphismen h, g durch Fortsetzung:

- Sei $h : \Sigma^* \to \Gamma^*$ festgelegt durch

$$0 \mapsto a, 1 \mapsto b, 2 \mapsto b.$$

Dann ist z.B. $h(0012) = aabb$ und das Bild von $L_1 = \{0^n 12^{n-1} : n \geq 1\}$ unter h ist

$$h(L_1) = \{a^n b^n : n \geq 1\}.$$

- Sei $g : \Sigma^* \to \Sigma^*$ festgelegt durch

$$0 \mapsto 0, 1 \mapsto 10, 2 \mapsto 0.$$

Dann ist $g(021) = 0010$ und $g^{-1}(0010) = \{021, 001, 221, 201\} = \{0, 2\}^2\{1\}$ und $g^{-1}(0^n 10^n) = \{0, 2\}^n\{1\}\{0, 2\}^{n-1}$.

Beispiele

Sei $\Sigma = \{0, 1, 2\}$ und $\Gamma = \{a, b\}$. Wir definieren Homomorphismen h, g durch Fortsetzung:

- Sei $h : \Sigma^* \to \Gamma^*$ festgelegt durch

$$0 \mapsto a, 1 \mapsto b, 2 \mapsto b.$$

Dann ist z.B. $h(0012) = aabb$ und das Bild von $L_1 = \{0^n 12^{n-1} : n \geq 1\}$ unter h ist

$$h(L_1) = \{a^n b^n : n \geq 1\}.$$

- Sei $g : \Sigma^* \to \Sigma^*$ festgelegt durch

$$0 \mapsto 0, 1 \mapsto 10, 2 \mapsto 0.$$

Dann ist $g(021) = 0010$ und $g^{-1}(0010) = \{021, 001, 221, 201\} = \{0, 2\}^2\{1\}$ und $g^{-1}(0^n 10^n) = \{0, 2\}^n\{1\}\{0, 2\}^{n-1}$.

Für das Urbild $g^{-1}(L_2)$ von $L_2 = \{0^n 10^n : n \geq 1\}$ unter g gilt

$$g^{-1}(L_2) \cap \{0\}^*\{1\}\{2\}^* = \{0^n 12^{n-1} : n \geq 1\}.$$

Reflexion

Reflexion eines Worts

Für $w \in \Sigma^*$ ist w^R rekursiv definiert durch

$$w^R = \begin{cases} \varepsilon & \text{für } w = \varepsilon \\ v^R a & \text{falls } w = av \text{ für } a \in \Sigma, v \in \Sigma^* \end{cases}$$

Reflexion

Reflexion eines Worts

Für $w \in \Sigma^*$ ist w^R rekursiv definiert durch

$$w^R = \begin{cases} \varepsilon & \text{für } w = \varepsilon \\ v^R a & \text{falls } w = av \text{ für } a \in \Sigma, v \in \Sigma^* \end{cases}$$

Die Reflexion einer Sprache

$L^R := \{w^R : w \in L\}$

```
def revs(s):          # String s umkehren
    return s[::-1]
def revl(L):          # jeden String aus L umkehren
    return set({revs(x) for x in L})
```

Frage

Ist Reflexion $w \mapsto w^R$ ein Homomorphismus?