
1. Alphabete, Wörter, Sprachen

In diesem Paragraphen führen wir die Grundobjekte sowohl für die Berechenbarkeitstheorie wie auch die Theorie Formaler Sprachen ein. Sprachen werden hier als Mengen von Wörtern über einem festen Alphabet aufgefasst. Zugleich werden Wörter als Ein- und Ausgabedaten von Algorithmen dienen.

1.1 DEFINITION. Ein *Alphabet* $(\Sigma, <)$ ist eine endliche, nichtleere Menge Σ zusammen mit einer totalen Ordnung $<$ auf Σ . Die Elemente von Σ heißen die *Buchstaben* (*Symbole*) des Alphabets.

Geben wir ein Alphabet $(\Sigma, <)$ durch Auflisten der Buchstaben an, so geschieht dies immer bezüglich der zugrundegelegten Ordnung: Für $\Sigma = \{a_0, \dots, a_n\}$ gilt $a_0 < a_1 < \dots < a_n$. Entsprechend lassen wir in der Bezeichnung eines Alphabets die Ordnung $<$ weg, falls diese implizit bekannt oder (an dieser Stelle) unwesentlich ist.

Ein Alphabet mit n Buchstaben heißt *n-äres Alphabet* und wird mit Σ_n bezeichnet. Insbesondere sind $\Sigma_1 = \{1\}$ und $\Sigma_2 = \{0, 1\}$ das *unäre* bzw. *binäre* Alphabet.

1.2 DEFINITION. Ein *Wort* über dem Alphabet $(\Sigma, <)$ ist eine endliche Folge über Σ , d.h. eine endliche Folge von Buchstaben aus Σ . Die Menge aller Wörter über Σ wird mit Σ^* bezeichnet.

Formal ist eine endliche Folge w über Σ eine Abbildung

$$w : \{0, \dots, n-1\} \rightarrow \Sigma \quad (1.1)$$

von einem Anfangsstück der natürlichen Zahlen in die Menge Σ . Hierbei lassen wir auch das leere Anfangsstück \emptyset , das wir auch mit $\{0, \dots, -1\}$ bezeichnen, zu. Die eindeutig bestimmte Folge $w : \emptyset \rightarrow \Sigma$ wird die *leere Folge* oder das *leere Wort* genannt und mit λ bezeichnet. Mit $\Sigma^+ = \Sigma^* - \{\lambda\}$ bezeichnen wir die Menge der nichtleeren Wörter über Σ .

Nichtleere Wörter w geben wir üblicherweise durch Auflisten der Buchstaben in der Reihenfolge ihrer Vorkommen in w an. So schreiben wir für das Wort w aus (1.1) $w = w(0) \dots w(n-1)$ (falls $n \geq 1$).

Die *Länge* $|w|$ eines Wortes w ist die Anzahl der Vorkommen von Buchstaben in w , d.h. $|w| = n$ für das Wort w aus (1.1). Insbesondere gilt $|\lambda| = 0$ und (für $n \geq 1$) $|w(0) \dots w(n-1)| = n$.

Die Menge der Wörter der Länge n über dem Alphabet Σ bezeichnen wir mit $\Sigma^{=n} = \{w \in \Sigma^* : |w| = n\}$. $\Sigma^{\leq n}$ und $\Sigma^{< n}$ sind entsprechend die Mengen der Wörter der Länge $\leq n$ bzw. $< n$. Man beachte, dass es für k -äres Alphabet Σ_k k^n Wörter $w = b_1 \dots b_n$ der Länge n gibt, da man für jeden Buchstaben b_m in w k Wahlmöglichkeiten hat. Formal beweist man

$$|\Sigma_k^{=n}| = k^n \quad (1.2)$$

durch Induktion nach n . (Hier bezeichnet $\|M\|$ die Kardinalität (Mächtigkeit) von M .) Insbesondere gelten:

$$\|\Sigma_k^{-1}\| = \|\Sigma_k\| = k \tag{1.3}$$

(Im Folgenden identifizieren wir Σ_k^{-1} mit Σ_k , indem wir das 1-buchstabile Wort a mit dem Buchstaben a identifizieren.)

$$\|\Sigma_1^{-n}\| = 1 \quad \text{und} \quad \|\Sigma_1^{\leq n}\| = n + 1 \tag{1.4}$$

$$\|\Sigma_2^{-n}\| = 2^n \quad \text{und} \quad \|\Sigma_2^{\leq n}\| = \sum_{m=0}^n \|\Sigma_2^{-m}\| = \sum_{m=0}^n 2^m = 2^{n+1} - 1 \tag{1.5}$$

Während also für das unäre Alphabet die Anzahl aller Wörter bis zur Länge n linear in n wächst, ist dieses Wachstum für das binäre Alphabet exponentiell. Letzteres Wachstumsverhalten gilt für alle Alphabete mit mindestens 2 Buchstaben (s. Übungen).

Die grundlegende Operation auf Wörtern ist die *Verkettung* oder *Konkatenation* \circ , anschaulich das Hintereinanderschreiben der Wörter. Für nichtleere Wörter $u = a_1 \dots a_m$ und $v = b_1 \dots b_n$ gilt $u \circ v = a_1 \dots a_m b_1 \dots b_n$ während die Konkatenation eines beliebigen Wortes w mit dem leeren Wort dieses unverändert lässt, d.h. $w \circ \lambda = \lambda \circ w = w$ (statt $v \circ w$ schreiben wir meist einfach vw). Verketteten wir ein Wort mit sich selbst, so sprechen wir von der Iteration des Wortes. Formal ist die *n-fache Iteration* von w ($n \geq 0$) induktiv durch

$$\begin{aligned} w^0 &= \lambda \\ w^{n+1} &= w^n w \end{aligned}$$

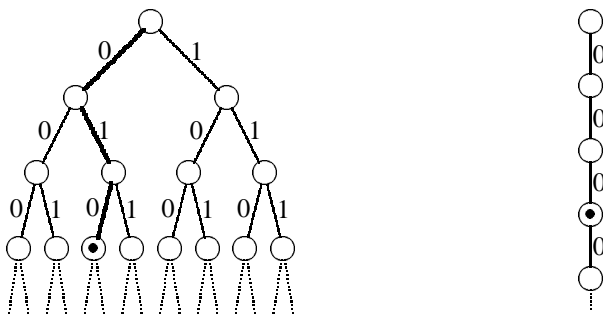
definiert. Wir sagen v ist ein (*echtes*) *Anfangsstück* von w , wenn sich w als Verkettung von v mit einem (nichtleeren) Wort darstellen lässt:

$$\begin{aligned} v \sqsubset w &\Leftrightarrow \exists x \in \Sigma^+ (vx = w) \\ v \sqsubseteq w &\Leftrightarrow \exists x \in \Sigma^* (vx = w) \Leftrightarrow v \sqsubset w \vee v = w. \end{aligned}$$

Ähnlich ist v ein *Teilwort* von w , falls $w = (x \circ v) \circ y$ für geeignetes $x, y \in \Sigma^*$.

Häufig ist es hilfreich die Wörter über dem k -ären Alphabet $\Sigma_k = \{a_1, \dots, a_k\}$ mit den Knoten des unendlichen, vollständigen und geordneten k -ären Baumes zu indentifizieren. Hierzu markiert man die k Kanten von einem Knoten zu seinen Söhnen von links nach rechts mit den Buchstaben a_1, \dots, a_k . Ein Knoten wird dann mit dem Wort identifiziert, mit dem der Weg von der Wurzel zu dem Knoten beschriftet ist.

In den folgenden Abbildungen haben wir (für den Fall $k = 2$ bzw $k = 1$) die Knoten, die das Wort 010 bzw 000 repräsentieren, hervorgehoben:



Allgemein entsprechen die Wörter der Länge n den Knoten der Tiefe n , insbesondere die Wurzel dem leeren Wort. Die Knoten des vollständigen Binärbaums der Tiefe n repräsentieren also $\Sigma_2^{\leq n}$, die Blätter desselben Σ_2^n . Sagen wir, dass ein Knoten v oberhalb eines Knotens w liegt, falls v auf dem Weg von der Wurzel λ zu dem Knoten w liegt, so liegt v genau dann oberhalb von w , wenn v Anfangsstück von w ist.

Die Menge Σ^* der Wörter über dem Alphabet Σ zusammen mit der Konkatenation $\circ : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ bildet eine *Halbgruppe* (Σ^*, \circ) mit *Eins*. D.h. \circ ist *assoziativ*

$$\forall x, y, z \in \Sigma^* ((x \circ y) \circ z = x \circ (y \circ z))$$

und für das leere Wort λ gilt

$$\forall x \in \Sigma^* (\lambda \circ x = x \circ \lambda = x).$$

Man nennt λ das *Einselement* oder *neutrale Element* von (Σ^*, \circ) . Wegen der Assoziativität von \circ können wir Klammern weglassen.

Eine mit der Konkatenation verträgliche Abbildung nennt man *Homomorphismus*. Formal: Ein *Homomorphismus* h ist eine Abbildung $h : \Sigma^* \rightarrow T^*$ sodass

$$\forall x, y \in \Sigma^* (h(x \circ y) = h(x) \circ h(y)).$$

Jeder Homomorphismus h ist bereits durch sein Verhalten auf den Buchstaben (d.h. Wörtern der Länge 1) bestimmt und jede Abbildung $h : \Sigma \rightarrow T^*$ lässt sich in eindeutiger Weise in einen Homomorphismus fortsetzen. (Man sagt, dass (Σ^*, \circ) die *freie Halbgruppe über Σ* ist).

1.3 LEMMA. *Zu jeder Abbildung $h_0 : \Sigma \rightarrow T^*$ gibt es genau einen Homomorphismus $h : \Sigma^* \rightarrow T^*$ mit $h(a) = h_0(a)$ für alle $a \in \Sigma$.*

BEWEISIDEE. Der gesuchte Homomorphismus ist durch $h(\lambda) = \lambda$ und

$$h(w(0) \dots w(n)) = h_0(w(0))h_0(w(1)) \dots h_0(w(n))$$

für Wörter w mit $|w| = n + 1 \geq 1$ definiert. □

Als nächstes führen wir eine Wohlordnung $<$ auf der Menge Σ^* der Wörter über dem Alphabet Σ ein. Hierzu werden Wörter zunächst nach ihrer Länge angeordnet und dann Wörter gleicher Länge mit Hilfe der Anordnung von Σ lexikographisch geordnet. Wir führen zunächst als Hilfsbegriff die partielle lexikographische Ordnung ein.

1.4 DEFINITION. Die *partielle lexikographische Ordnung* $<_{lex}$ auf der Menge Σ^* der Wörter über dem Alphabet $(\Sigma, <)$ ist definiert durch

$$v <_{lex} w \iff \exists x, y, z \in \Sigma^* \exists a, b \in \Sigma (a < b \ \& \ v = xay \ \& \ w = xbz).$$

Wie bei Ordnungsrelationen üblich betrachten wir neben jeder strikten Ordnung $<$ immer auch die zugehörige schwache Ordnung \preceq , bei der zusätzlich jedes Wort mit sich vergleichbar ist, d.h. hier

$$v \preceq_{lex} w \iff v <_{lex} w \text{ oder } v = w.$$

Identifizieren wir wie oben beschrieben Wörter mit den Knoten eines Baumes, so bedeutet $v <_{lex} w$, dass der Knoten v links des Knotens w liegt.

1.5 LEMMA. Die partielle lexikographische Ordnung \leq_{lex} ist eine partielle Ordnung (Halbordnung) auf Σ^* , d.h. es gilt:

- (i) $\forall w \in \Sigma^* (w \leq_{lex} w)$ (Reflexivität)
- (ii) $\forall u, v, w \in \Sigma^* (u \leq_{lex} v \ \& \ v \leq_{lex} w \Rightarrow u \leq_{lex} w)$ (Transitivität)
- (iii) $\forall v, w \in \Sigma^* (v \leq_{lex} w \ \& \ w \leq_{lex} v \Rightarrow v = w)$ (Antisymmetrie)

BEWEIS. (i) gilt nach Definition. Zum Nachweis von (ii) nehmen wir an, dass $u \leq_{lex} v$ und $v \leq_{lex} w$ gelte. Wir haben $u \leq_{lex} w$ nachzuweisen. Gilt $u = v$ oder $v = w$ so gilt dies nach Annahme. O.B.d.A. können wir also von $u <_{lex} v$ und $v <_{lex} w$ ausgehen. D.h. es gibt Zerlegungen der Wörter u, v, w

$$u = xay, v = xbz = x'a'y', w = x'b'z'$$

wobei $x, y, z, x', y', z' \in \Sigma^*$, $a, b, a', b' \in \Sigma$, $a < b$ und $a' < b'$ gilt. Hieraus erhält man Zerlegungen von u und w , die $u <_{lex} w$ bezeugen, durch Unterscheidung der folgenden 3 Fälle. Ist $|x| < |x'|$, so ist (wegen $xbz = x'a'y'$) xb Anfangsstück von x' und damit $u = xay$ und $w = xbz''$ für geeignetes z'' . Ist $|x| = |x'|$, so ist $x = x'$ und damit $u = xay$ und $w = x'b'z'$ wobei $a < b = a' < b'$, also $a < b'$. Ist schließlich $|x| > |x'|$, so gilt symmetrisch zum ersten Fall, dass $u = x'a'z'''$ und $w = x'b'z'$ für geeignetes $z''' \in \Sigma^*$.

Teil (iii) zeigen wir indirekt. Wir gehen von der Widerspruchsannahme aus, dass $v \leq_{lex} w$ und $w \leq_{lex} v$ aber $v \neq w$. Dann gilt $v <_{lex} w$ und $w <_{lex} v$, weshalb es Zerlegungen

$$v = xay = x'b'y' \quad \text{und} \quad w = xbz = x'a'z'$$

von v und w mit $x, y, z, x', y', z' \in \Sigma^*$, $a, b, a', b' \in \Sigma$, $a < b$ und $a' < b'$ gibt. Es folgt, dass $x = x'$ oder $x \sqsubset x'$ oder $x' \sqsubset x$. Ist $x = x'$, so gilt $a = b'$ und $b = a'$, also $a < a$ wegen $a < b = a' < b' = a$. Dies widerspricht der Wahl von $<$ als Ordnung auf Σ . Ist $x \sqsubset x'$, so gilt $xa \sqsubseteq x'$ und $xb \sqsubseteq x'$ also $a = b$ im Widerspruch zu $a < b$. Entsprechend impliziert $x' \sqsubset x$ dass $a' = b'$ im Widerspruch zu $a' < b'$. \square

Die Relation \leq_{lex} ist keine totale Ordnung, d.h. erfüllt die Forderung

$$\forall v, w \in \Sigma^* (v \leq_{lex} w \text{ oder } w \leq_{lex} v) \text{ (Konnexität)}$$

nicht. Wie man leicht einsieht wird diese Forderung genau dann verletzt, wenn v echtes Teilwort von w (oder umgekehrt) ist (d.h. die Knoten v und w auf demselben Pfad liegen):

$$(v \sqsubset w \text{ oder } w \sqsubset v) \Leftrightarrow (v \not\leq_{lex} w \ \& \ w \not\leq_{lex} v) \quad (1.6)$$

Insbesondere sind Wörter gleicher Länge also stets bezüglich \leq_{lex} vergleichbar:

1.6 LEMMA. Die Einschränkung von \leq_{lex} auf Σ^n für festes n ist eine totale Ordnung auf Σ^n . \square

Man kann $<_{lex}$ zu einer totalen Ordnung \prec auf Σ^* fortsetzen, indem man die Teilwortbeziehung als weiteres Ordnungskriterium hinzunimmt, also

$$\begin{aligned} v \prec w & \Leftrightarrow v <_{lex} w \text{ oder } v \sqsubset w \\ & \Leftrightarrow v <_{lex} w \text{ oder } [v \not\leq_{lex} w \ \& \ w \not\leq_{lex} v \ \& \ |v| < |w|] \end{aligned}$$

festlegt. Diese – auch *totale lexikographische Ordnung* genannte – Ordnung, der die lexikographische Ordnung als Hauptordnungskriterium zugrundeliegt und, wenn diese versagt, die Wortlänge als Nebenordnungskriterium dient, ist die in Lexika benutzte Ordnung. Sie hat jedoch den Nachteil, dass sie keine Nummerierung der Wörter liefert (d.h. – formal – (Σ^*, \prec) nicht isomorph zu $(\mathbb{N}, <)$ ist; s.u.). Z.B. gilt für den Fall des binären Alphabets $\Sigma = \Sigma_2$, dass $0^n \prec 1$ für alle $n \geq 0$ gilt, also $\{w : w \prec 1\}$ unendlich ist. Diesen Mangel beseitigt die längen-lexikographische Ordnung, bei der die Ordnungskriterien vertauscht, d.h. die Länge das primäre und die lexikographische Ordnung das sekundäre Ordnungsmerkmal sind.

1.7 DEFINITION. Die *längen-lexikographische Ordnung* $<_{ll}$ auf Σ^* ist definiert durch

$$v <_{ll} w \Leftrightarrow |v| < |w| \text{ oder } [|v| = |w| \ \& \ v <_{lex} w].$$

Beispielsweise gilt für das unäre Alphabet $\Sigma_1 = \{1\}$

$$\lambda(= 1^0) <_{ll} 1(= 1^1) <_{ll} 1^2 <_{ll} 1^3 <_{ll} 1^4 <_{ll} \dots$$

und für das binäre Alphabet $\Sigma_2 = \{0, 1\}$

$$\lambda <_{ll} 0 <_{ll} 1 <_{ll} 00 <_{ll} 01 <_{ll} 10 <_{ll} 11 <_{ll} 000 <_{ll} 001 <_{ll} \dots$$

1.8 SATZ. $(\mathbb{N}, <)$ ist isomorph zu $(\Sigma^*, <_{ll})$, d.h. es gibt eine Bijektion $f : \mathbb{N} \rightarrow \Sigma^*$, die ordnungserhaltend ist, d.h.

$$m < n \Leftrightarrow f(m) <_{ll} f(n)$$

erfüllt ($m, n \in \mathbb{N}$).

Zum Beweis von Satz 1.8 zeigen wir zunächst die beiden folgenden Lemmata.

1.9 LEMMA. Die *längen-lexikographische Ordnung* \leq_{ll} auf Σ^* ist eine totale Ordnung, d.h. \leq_{ll} ist reflexiv, transitiv, antisymmetrisch und konvex.

BEWEIS. Dies ergibt sich leicht aus der Definition von \leq_{ll} mit Lemma 1.6. \square

1.10 LEMMA. Jedes Wort $w \in \Sigma^*$ besitzt genau einen direkten Nachfolger $N(w)$ bzgl. \leq_{ll} und jedes nichtleere Wort $w \in \Sigma^+$ besitzt genau einen direkten Vorgänger $V(w)$ bzgl. \leq_{ll} , d.h.

$$w <_{ll} N(w) \quad \& \quad \forall v(w <_{ll} v \Rightarrow N(w) \leq_{ll} v) \quad (1.7)$$

$$V(w) <_{ll} w \quad \& \quad \forall v(v <_{ll} w \Rightarrow v \leq_{ll} V(w)) \quad (1.8)$$

BEWEIS. Wir definieren $N(w)$ und $V(w)$ und überlassen den Nachweis der Eigenschaften (1.7) und (1.8) als Übung.

Ist $\Sigma = \Sigma_1$, so definiert man $N(1^n) = 1^{n+1}$ und $V(1^{n+1}) = 1^n$ für $n \geq 0$. Ist $\Sigma = \Sigma_k = \{a_0, \dots, a_{k-1}\}$ k -är mit $k \geq 2$, so unterscheidet man folgende Fälle bei der Definition von $N(w)$ und $V(w)$, wobei $n = |w|$ die Länge von w sei.

Ist $w = (a_{k-1})^n$ (d.h. das bzgl. \leq_{lex} größte Wort in Σ^n), so setzt man $N(w) = (a_0)^{n+1}$ (d.h. wählt $N(w)$ als das bzgl. \leq_{lex} kleinste Wort der Länge $n+1$). Andernfalls wählt man $N(w)$ als den direkten Nachfolger von w bzgl. \leq_{lex} in Σ^n , d.h. $N(w)$ ist das bzgl. \leq_{lex} kleinste Wort in $\{v \in \Sigma^n : w <_{lex} v\}$. (Man beachte hierbei, dass jede

nichtleere endliche total geordnete Menge ein kleinstes Element besitzt. Explizit kann man $N(w)$ wie folgt bestimmen: Ist $w = a_{i_0} \dots a_{i_{n-1}} \neq (a_{k-1})^n$, so wählt man $m < n$ maximal mit $a_{i_m} \neq a_{k-1}$ und setzt $N(w) = a_{i_0} \dots a_{i_{m-1}} a_{i_m+1} a_0 \dots a_0$. Entsprechend definiert man für w mit $|w| = n > 0$: $V((a_0)^n) = (a_{k-1})^{n-1}$ und setzt $V(w) = w^-$ sonst, wobei w^- der direkte Vorgänger von w bzgl. \leq_{lex} in Σ^n ist. \square

BEWEIS SATZ 1.8. Die Funktion f wird unter Verwendung von Lemma 1.10 induktiv durch $f(0) = \lambda$ und $f(n+1) = N(f(n))$ definiert. Wegen Lemma 1.9 und (1.7) gilt

$$n < m \Leftrightarrow f(n) <_{ll} f(m)$$

(Induktion nach $k = m - n$) und damit auch, dass f injektiv ist. Die Surjektivität von f sieht man wie folgt. Als Widerspruchsannahme gehen wir davon aus, dass es ein Wort w gibt, das nicht im Bild von f liegt. Da $\{v : v \leq_{ll} w\} \subseteq \Sigma^{\leq |w|}$ endlich ist, gibt es ein \leq_{ll} -kleinstes solches w , etwa w_0 . Wegen $f(0) = \lambda$ ist $w_0 \neq \lambda$, d.h. $V(w_0)$ ist definiert und $V(w_0) <_{ll} w_0$. Wegen der Minimalität von w_0 gibt es also ein n mit $f(n) = V(w_0)$. Also $f(n+1) = N(V(w_0)) = w_0$. Widerspruch! \square

Satz 1.8 erlaubt uns die Wörter in Σ^* gemäß \leq_{ll} durchnummerieren. Wir bezeichnen mit $w_n^{(\Sigma, <)}$ das $(n+1)$ -te Wort in Σ^* bzgl. \leq_{ll} , d.h. $w_n^{(\Sigma, <)} = f(n)$ für f aus Satz 1.8 (und wir schreiben w_n statt $w_n^{(\Sigma, <)}$, wenn das zugrundegelegte Alphabet bekannt ist). Weiter können wir w_n mit n identifizieren, d.h. w_n als Darstellung von n auffassen. Wegen dieser Möglichkeit und zur Vereinfachung der Schreibweise werden wir in den folgenden Abschnitten meist \leq statt \leq_{ll} schreiben. Für das unäre Alphabet stimmt w_n mit der üblichen Unärarstellung $Un(n) = 1^n$ von n überein. Für das binäre Alphabet $\Sigma_2 = \{0, 1\}$ stimmt jedoch w_n nicht mit der üblichen Binärzahldarstellung $\text{Bin}(n)$ überein (da führende Nullen in einem Wort hier keinen Einfluss auf den Zahlwert des Wortes haben). Es gilt jedoch $\text{Bin}(n+1) = 1w_n$ (s. Übungen). Hierbei ist $\text{Bin}(0) = 0$ und $\text{Bin}(n)$ für eine Zahl $n \geq 1$ wie folgt definiert: Ist $n = \sum_{m=0}^k i_m \cdot 2^m$ mit $i_m \in \{0, 1\}$ und $i_k = 1$, so ist $\text{Bin}(n) = i_k i_{k-1} \dots i_0$.

Nach der ausführlichen Behandlung von Wörtern kommen wir nun zum Begriff der Sprache, die wir hier als beliebige Menge von Wörtern über einem festen Alphabet auffassen.

1.11 DEFINITION. Eine *Sprache* L über dem Alphabet Σ ist eine Teilmenge von Σ^* .

Zur Bezeichnung von Buchstaben, Wörtern und Sprachen werden wir uns an folgende Regeln halten. Kleine Buchstaben a, b, c vom Anfang des Alphabets reservieren wir für Buchstaben, kleine Buchstaben u, v, w, x, y, z vom Ende des Alphabets für Wörter. Sprachen bezeichnen wir mit großen (kursiven) Buchstaben A, B, C, \dots , Mengen von Sprachen, die wir auch kurz *Klassen* nennen, mit großen (normalen) Buchstaben A, B, C, \dots . Kleine Buchstaben k, l, m, n aus der Mitte des Alphabets werden in der Regel natürliche Zahlen beschreiben.

Die Menge aller Sprachen über Σ ist die *Potenzmenge* von Σ^* und wird mit

$$\text{POWER}(\Sigma^*) = \{L : L \subseteq \Sigma^*\}$$

bezeichnet.

Da Sprachen Mengen sind, können wir die üblichen mengentheoretischen Operationen verwenden ($L, L_1, L_2 \subseteq \Sigma^*$):

$$\begin{aligned} L_1 \cup L_2 &= \{x \in \Sigma^* : x \in L_1 \text{ oder } x \in L_2\} && \text{(Vereinigung)} \\ L_1 \cap L_2 &= \{x \in \Sigma^* : x \in L_1 \ \& \ x \in L_2\} && \text{(Durchschnitt)} \\ \bar{L} &= \{x \in \Sigma^* : x \notin L\} && \text{(Komplement)} \\ L_1 - L_2 &= \{x \in \Sigma^* : x \in L_1 \ \& \ x \notin L_2\} && \text{(Differenz)} \\ L_1 \triangle L_2 &= (L_1 - L_2) \cup (L_2 - L_1) && \text{(Symmetrische Differenz)} \end{aligned}$$

Daneben betrachten wir die durch Verkettung und Iteration induzierten Sprachoperationen:

$$\begin{aligned} L_1 L_2 &= \{xy : x \in L_1 \ \& \ y \in L_2\} \\ L^n &= \{x_1 \dots x_n : x_i \in L\} \quad (\text{d.h. } L^0 = \{\lambda\}, L^1 = L, L^2 = LL, \dots) \\ L^* &= \{x_1 \dots x_n : n \geq 0 \ \& \ x_i \in L\} = \bigcup_{n \geq 0} L^n \\ L^+ &= \{x_1 \dots x_n : n \geq 1 \ \& \ x_i \in L\} = \bigcup_{n \geq 1} L^n \end{aligned}$$

Hierbei schreiben wir aL statt $\{a\}L$.

Die *Kardinalität* oder *Mächtigkeit* einer Menge M wird mit $\|M\|$ bezeichnet. Für endliches M ist $\|M\|$ gerade die Anzahl der Elemente von M . Allgemein sagt man für Mengen M und M' , dass M und M' *gleichmächtig* sind – und schreibt $\|M\| = \|M'\|$ –, wenn es eine Bijektion von M auf M' gibt. Entsprechend bedeutet $\|M\| \leq \|M'\|$, dass es eine injektive Funktion $f : M \rightarrow M'$ gibt. In der Mengenlehre zeigt man, dass $\|M\| = \|M'\|$ g.d.w. $\|M\| \leq \|M'\|$ und $\|M'\| \leq \|M\|$ gilt. M ist *abzählbar unendlich*, falls $\|M\| = \|\mathbb{N}\|$. M ist (*höchstens*) *abzählbar*, wenn $\|M\| \leq \|\mathbb{N}\|$, d.h. wenn M endlich oder abzählbar unendlich ist. Offensichtlich ist jede Teilmenge einer abzählbaren Menge selbst wieder abzählbar.

1.12 LEMMA. Für jedes Alphabet Σ gilt:

- (i) Jede Sprache L über Σ ist abzählbar.
- (ii) Die Menge aller Sprachen über Σ ist nicht abzählbar.

BEWEIS. Da $L \subseteq \Sigma^*$ folgt (i) aus Satz 1.8, wo insbesondere gezeigt wurde, dass $\|\Sigma^*\| = \|\mathbb{N}\|$. Teil (ii) zeigt man mit einem auf Cantor zurückgehenden Diagonalisierungsargument. Offensichtlich ist $\text{POWER}(\Sigma^*)$ unendlich, weshalb es zu zeigen genügt, dass $\text{POWER}(\Sigma^*)$ nicht abzählbar unendlich ist. Wir gehen von $\|\text{POWER}(\Sigma^*)\| = \|\mathbb{N}\|$ als Widerspruchsannahme aus. Dann gibt es eine Bijektion $f : \mathbb{N} \rightarrow \text{POWER}(\Sigma^*)$, d.h. $\text{POWER}(\Sigma^*) = \{L_n : n \in \mathbb{N}\}$, wobei $L_n = f(n)$. Man definiert nun eine Sprache $L \subseteq \Sigma^*$ durch

$$w_n \in L : \Leftrightarrow w_n \notin L_n$$

für alle $n \geq 0$. Diese „Diagonale“ L unterscheidet sich von der n -ten Sprache L_n in dem n -ten Wort w_n , ist also nicht in $\{L_n : n \in \mathbb{N}\}$, d.h.

$$L \in \text{POWER}(\Sigma^*) - \{L_n : n \in \mathbb{N}\}.$$

Widerspruch. □

Man kann weite Teile der Vorlesung als eine Einführung in die Theorie der Darstellung von Sprachen auffassen. Hierbei ist eine *Darstellung* einer Sprache ein *endliche*,

eindeutige Beschreibung derselben. Endliche Sprachen kann man trivialerweise durch Auflisten aller ihrer Elemente darstellen. Z.B.

$$L_1 = \{0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24\}.$$

Solches direktes Auflisten kann jedoch sehr länglich und damit umständlich sein, so dass andere Darstellungen mitunter vorzuziehen sind. So ist z.B. die alternative Darstellung

$$L_1 = \{x \in \mathbb{N} : x \leq 24 \ \& \ x \text{ gerade}\}$$

kompakter und mehr instruktiv. Hierbei handelt es sich um eine explizite Definition von L_1 mit Hilfe einfacher mathematischer Begriffe.

Bei unendlichen Sprachen muss man stets auf Alternativen zum Auflisten zurückgreifen. Dabei können Darstellungen ein und derselben Sprache sehr unterschiedlich sein (wie das Beispiel oben schon zeigt). Neben expliziten Darstellungen sind hierbei auch implizite Darstellungen (d.h. induktive Charakterisierungen) interessant. Ein Beispiel:

Die Sprache

$$L_2 = \{w \in \Sigma_2^* : \#_0(w) = \#_1(w)\}$$

kann auch auf folgende *rekursive* (= *induktive*) Art charakterisiert werden: L_2 ist die kleinste Sprache $L \subseteq \Sigma^*$ mit folgenden 3 Eigenschaften:

- (i) $\lambda \in L$
- (ii) Gehört w zu L , so auch $0w1$ und $1w0$.
- (iii) Gehören v und w zu L , so auch vw .

Man beschreibt hier also L_2 , indem man zunächst eine einfache Teilsprache von L_2 (hier $\{\lambda\} \subseteq L_2$ gemäß (i)) und dann Abschlusseigenschaften von L_2 (hier (ii) und (iii)) angibt (Beweis: s. Übungen).

Diese Idee wird uns sowohl bei Grammatiken zur Beschreibung von Sprachen wiederbegegnen (hier werden die Grundannahmen ((i)) durch *Axiome* und die Abschlusseigenschaften ((ii), (iii)) durch *Regeln* modelliert) als auch bei induktiven Charakterisierungen von Algorithmen. Algorithmen selbst sind weitere Darstellungsmöglichkeiten. So wird durch den folgenden Algorithmus, der die Sprache L_2 erkennt, diese endlich und eindeutig beschrieben:

Bei Eingabe $w \in \Sigma_2^*$:
 Lege einen Zähler n an und initialisiere diesen ($n := 0$).
 Dann gehe w von links nach rechts Buchstabe für Buchstabe durch
 und inkrementiere dabei den Zähler ($n := n + 1$) beim Lesen einer Null
 und dekrementiere ($n := n - 1$) beim Lesen einer Eins.
 Ist $n = 0$ nach vollständigem Einlesen von w , so akzeptiere,
 sonst verwirfe.

Man spricht hier auch von einem Entscheidungsverfahren für L_2 . Eine andere algorithmische Charakterisierung der Sprache L_2 kann man durch ein Aufzählungsverfahren für L_2 geben (s. Übungen). Ein Aufzählungsverfahren für eine Sprache L gibt die zu L gehörenden Wörter (in beliebiger Reihenfolge) aus. Wir werden auf diese verschiedenen Algorithmientypen im nächsten Abschnitt näher eingehen.

Untersuchen werden wir meist nicht einzelne Darstellungen, sondern Darstellungsweisen. Eine *Darstellungsweise* \mathcal{D} ist eine Klasse von Darstellungen eines einheitlichen Typs zur Beschreibung von Sprachen über festem Alphabet. O.B.d.A. können wir davon ausgehen, dass Darstellungen durch endliche Texte gegeben, also Wörter über einem geeigneten Alphabet sind. Bei Darstellungen des gleichen Typs dürfte dann das Alphabet stets dasselbe sein, weshalb eine Darstellungsweise \mathcal{D} selbst wiederum eine Sprache über einem geeigneten Alphabet ist. Mit Lemma 1.12 folgt, dass die Klasse

$$L(\mathcal{D}) = \{L(D) : D \in \mathcal{D} \text{ \& } D \text{ stellt } L(D) \text{ dar}\}$$

der in \mathcal{D} darstellbaren Sprachen abzählbar ist, also eine Darstellungsweise nie die Beschreibung aller Sprachen über einem gegebenen Alphabet erlaubt.

In der Praxis ergeben sich damit zwei konkurrierende Anforderungen an Darstellungsweisen:

- *Mächtigkeit der Darstellungsweise*: Eine möglichst große Klasse von Sprachen soll darstellbar sein.
- *Güte der Darstellungsweise*: Den Darstellungen soll möglichst viel Information über die dargestellte Sprache effektiv und möglichst effizient entnehmbar sein.

Wie wir sehen werden, wird die Mächtigkeit einer Darstellungsweise stets auf Kosten deren Güte (und umgekehrt) gehen. Typische Merkmale für die Güte einer Darstellungsweise \mathcal{D} (für Sprachen über dem Alphabet Σ) sind die Komplexität der folgenden Probleme:

- *Wortproblem für \mathcal{D}* : Stelle für gegebenes $D \in \mathcal{D}$ und $w \in \Sigma^*$ fest, ob w in der von D dargestellten Sprache $L(D)$ liegt.
- *Äquivalenzproblem für \mathcal{D}* : Stelle für gegebene Darstellungen $D, D' \in \mathcal{D}$ fest, ob diese dieselbe Sprache beschreiben, d.h. $L(D) = L(D')$ gilt.

Ähnlich betrachtet man das *Korrektheitsproblem* („Ist $L(D) = L^*$ für vorgegebenes L “), *Leerheitsproblem* („Ist $L(D) = \emptyset^*$ “), *Endlichkeitsproblem* („Ist $L(D)$ endlich?“), *Durchschnittsproblem* („Ist $L(D) \cap L(D') = \emptyset^*$ “) etc. Man sagt, dass solch ein Problem *lösbar* ist, falls es einen Algorithmus zur Beantwortung der zugehörigen Frage gibt.