

## 2.2 Alphabete, Wörter und Sprachen

Def: Eine endliche nichtleere Menge  $\Sigma$  heißt Alphabet.

Def: Die Elemente eines Alphabets werden Buchstaben genannt.

Def: Sei  $\Sigma$  ein Alphabet.

Ein Wort über  $\Sigma$  ist eine Folge von Buchstaben aus  $\Sigma$ .

Def: Das leere Wort  $\lambda$  ist die leere Buchstabenfolge.

Def: Die Länge  $|w|$  eines Wortes  $w$  ist die Anzahl der Buchstaben in  $w$ .

Def:  $\Sigma^*$  ist die Menge aller Wörter über dem Alphabet  $\Sigma$ .

Def:  $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$

Def:  $\Sigma^n$  ist die Menge aller Wörter der Länge  $n$  über  $\Sigma$ .

Bsp:  $\Sigma_{\text{bool}} = \{0, 1\}$  Alphabet

$$(\Sigma_{\text{bool}})^* = \{\lambda\} \cup \{x_1 x_2 \dots x_n \mid n \in \mathbb{N}, x_i \in \{0, 1\} \text{ für } i=1, \dots, n\}$$

$$\Sigma_{\text{bool}}^2 = \{00, 01, 10, 11\}$$

wahr/falsch Aussagen:

- $\lambda \in \Sigma_{\{a,b\}}^*$  ✓
- $\Sigma^*$  enthält immer unendlich viele Wörter. ✓
- $\lambda\lambda\lambda\lambda \in \Sigma^4$  ✗  $|\lambda|=0$
- Je nach Alphabet kann  $\Sigma^n$  unendlich viele Wörter enthalten. ✗  $|\Sigma|$  endlich

Aufg 1: Sei  $\Sigma$  ein Alphabet mit  $k$  Buchstaben.  
Bestimme die Anzahl Wörter in  $\Sigma^n$ .

Lösung:  $k^n$

Def: Die Verkettung (Konkatenation) für ein Alphabet  $\Sigma$  ist die Abbildung

$$\text{Kon}: \Sigma^* \times \Sigma^* \longrightarrow \Sigma^* \quad \text{s.d.}$$

$$\text{Kon}(x, y) = xy \quad \forall x, y \in \Sigma^*$$

Bsp:  $\Sigma = \{0, 1\}$

- $\text{Kon}(1011, 001) = 1011001$

- $(101)^3 = 101101101$

Bmk:

- Konkatenation ist assoziativ

$$\forall x, y, z \in \Sigma^*: \text{Kon}(\text{Kon}(x, y), z) = \text{Kon}(x, \text{Kon}(y, z))$$

- Konkatenation ist im Allgemeinen NICHT kommutativ

z.B.  $\text{kon}(11, 01) = 1101 \neq 0111 = \text{kon}(01, 11)$

- $\text{Kon}(x, \lambda) = \text{Kon}(\lambda, x) = x$

- $|xy| = |\text{Kon}(x, y)| = |x| + |y|$

Def: Seien  $a_i \in \Sigma$  für  $i = 1, \dots, n$ .

Sei  $a = a_1 a_2 \dots a_n \in \Sigma^*$  ein Wort.

Die Umkehrung (Reversal) von  $a$  ist  $a^R = a_n a_{n-1} \dots a_1$

Serie 1:  $(uv)^R = v^R u^R$  beweisen/widerlegen

Def: Seien  $v, w \in \Sigma^*$  für ein Alphabet  $\Sigma$ .

- $v$  heisst Teilwort von  $w$   $\Leftrightarrow \exists x, y \in \Sigma^* : w = xvy$
- $v$  heisst Präfix von  $w$   $\Leftrightarrow \exists y \in \Sigma^* : w = vy$
- $v$  heisst Suffix von  $w$   $\Leftrightarrow \exists x \in \Sigma^* : w = xv$
- $v \neq \lambda$  heisst echtes Teilwort/Präfix/Suffix von  $w$   
 $\Leftrightarrow v \neq w$  und  $v$  ist Teilwort/Präfix/Suffix von  $w$

Def: Sei  $x \in \Sigma^*$  und  $a \in \Sigma$ .

$|x|_a$  ist definiert als die Anzahl der Vorkommen von  $a$  in  $x$ .

Bsp:

- $|01101|_1 = 3$

- $|aba|_b = 1$

- $\forall x \in \Sigma^* : |x| = \sum_{a \in \Sigma} |x|_a$

Def: Sei  $A$  eine Menge.

- $|A|$  bezeichnet die Kardinalität von  $A$ .

- $P(A) = \{S \mid S \subseteq A\}$  bezeichnet die Potenzmenge von  $A$ .

Def: Sei  $\Sigma = \{s_1, s_2, \dots, s_m\}$  ein Alphabet ( $m \geq 1$ ).

Sei  $s_1 < s_2 < \dots < s_m$  eine Ordnung auf  $\Sigma$ .

Wir definieren die kanonische Ordnung auf  $\Sigma^*$  wie folgt für  $u, v \in \Sigma^*$ :

$u < v \Leftrightarrow |u| < |v|$

$v \mid u \mid = |v| \wedge u = xs_iu' \wedge v = xs_jv'$

für  $x, u', v' \in \Sigma^*$  und  $s_i < s_j$

In Wörtern:

- Das kürzere Wort ist kleiner.

- Bei gleich langen Wörtern betrachten wir den ersten Buchstaben, bei dem sich die Wörter unterscheiden.

Das Wort mit dem kleineren Buchstaben an dieser Stelle ist kleiner.

Def: Eine Sprache  $L$  über dem Alphabet  $\Sigma$  ist eine Teilmenge von  $\Sigma^*$ .  
 $L = \{w \in \Sigma^* \mid \text{Bedingungen}\} \subseteq \Sigma^*$

Def:

- $L^c = \Sigma^* \setminus L$  ist das Komplement von  $L$
- $L_\emptyset = \emptyset$  ist die leere Sprache
- $L_\lambda = \{\lambda\}$  ist die Sprache, die nur das leere Wort enthält.
- $L_1 \cdot L_2 = L_1 L_2 = \{vw \mid v \in L_1, w \in L_2\}$

Def: Sei  $L$  eine Sprache über  $\Sigma$ . Wir definieren:

$$L^0 := L_\lambda$$

$$L^{i+1} := L^i \cdot L \quad \forall i \in \mathbb{N}$$

$$L^* := \bigcup_{i \in \mathbb{N}} L^i \quad \text{heisst der } \underline{\text{Kleen'sche Stern}} \text{ von } L.$$

$$L^+ := \bigcup_{i \in \mathbb{N} \setminus \{0\}} L^i = L \cdot L^*$$

Bmk: Wann gilt  $L^* = L^+$ ?  
 gdw.  $\lambda \in L$

Aufg 2: Seien  $L_1 = \{\lambda, ab^2, ba^4\}$      $L_2 = \{a^3, b^2, ab\}$   
 Welche Wörter liegen in  $L_1 L_2$ ?

$$\text{Lösung: } L_1 L_2 = \{a^3, b^2, ab, ab^2a^3, ab^4, ab^2ab, ba^7, ba^4b^2, ba^5b\}$$

Def: Ein Homomorphismus von  $\Sigma_1^*$  nach  $\Sigma_2^*$  ist eine Funktion  
 $h: \Sigma_1^* \rightarrow \Sigma_2^*$  mit folgenden Eigenschaften:

- $h(\lambda) = \lambda$
- $\forall u, v \in \Sigma_1^* : h(uv) = h(u)h(v)$

- Hinweise:
- A1 a) c) es hilft, die Struktur der möglichen Wörter zu finden
  - b) zählt die Möglichkeiten für  $n=1, 2, \dots$   
Erkennt ihr eine Folge?
  - A2 Gegenbeispiele begründen
  - A3 Korrektheit der Konstruktion begründen

## 2.3 Algorithmische Probleme

Def: Ein Programm ist im Allgemeinen eine geordnete Abfolge von Anweisungen, die zum Lösen eines bestimmten Problems dienen soll.

Def: Ein Algorithmus A ist ein Programm, bzw. realisiert eine Abbildung  
 $A: \Sigma_1^* \rightarrow \Sigma_2^*$   
 für beliebige Alphabete  $\Sigma_1, \Sigma_2$ ,  
 s.d. das Programm für jede zulässige Eingabe  $x \in \Sigma_1^*$  hält (d.h. keine Endlosschleife) und eine Ausgabe  $A(x)$  liefert.

Def: Zwei Algorithmen A, B heißen äquivalent, falls

- (i) A und B mit dem gleichen Eingabealphabet  $\Sigma$  arbeiten
- (ii)  $\forall x \in \Sigma^*: A(x) = B(x)$

Def: Seien  $\Sigma$  ein Alphabet und  $L \subseteq \Sigma^*$  eine Sprache über  $\Sigma$ . Das Entscheidungsproblem ( $\Sigma, L$ ) ist,  $\forall x \in \Sigma^*$  zu entscheiden, ob  $x \in L$  oder  $x \notin L$  gilt.

Def: Ein Algo A löst das Entscheidungsproblem  $(\Sigma, L)$ , falls

$$\forall x \in \Sigma^*: A(x) \begin{cases} 1 & \text{falls } x \in L \\ 0 & \text{falls } x \notin L \end{cases}$$

Wir sagen auch, dass A die Sprache L erkennt.

Def: Eine Sprache L heißt rekursiv, falls es einen Algorithmus gibt, der L erkennt.