Theoretische Informatik
Summary

September 21, 2020

Chapter 1

Alphabete, Wörter, Sprachen und die Darstellung von Problemen

1.1 2.2 Alphabete, Wörter und Sprachen

D2.1: Eine endliche nichtleere Menge $\sum$ heisst **Alphabet**. Die Elemente eines Alphabets werden **Buchstaben (Zeichen, Symbole)** genannt.

Häufig verwendete Alphabete:

- $\sum_{bool} = \{0, 1\}$
- $\sum_{lat} = \{a, b, c, ..., z\}$
- $\sum_{m} = \{0, 1, 2, ..., m-1\}$
- $\sum_{logic} = \{0, 1, x, (,), AND, OR, NOT\}$

D2.2: Ein **Wort** über $\sum$ ist eine endliche (eventuell leere) Folge von Buchstaben aus $\sum$. Das leere Wort λ (manchmal ϵ) ist die leere Buchstabenfolge. Die **Länge** $|w|$ eines Wortes w ist die Anzahl der Vorkommen von Buchstaben in w.

- $\sum^*$: Menge aller Wörter über $\sum$
- $\sum^+ = \sum^* - \{\lambda\}$

Das leere Wort λ ist ein Wort über jedem Alphabet.

Verabredung: Wir werden Wörter ohne Komma schreiben

D2.3: Die **Verkettung (Konkatenation)** für ein Alphabet $\sum$ ist eine Abbildung

$$Kon(x, y) = x \cdot y = xy$$

für alle $x, y \in \sum^*$ Die Verkettung ist eine assoziative Operation und $(\sum^*, Kon)$ ist eine Halbgruppe(Monoid) mit neutralen element λ
Für alle $x, y \in \sum^*$ gilt:

$$|xy| = |x \cdot y| = |x| + |y|$$

D2.4: Für ein Wort $a = a_1a_2a_3...a_n$ mit $a_i \in \sum$ für $i \in \{1, 2, ..., n\}$, bezeichnet $a^R = a_na_{n-1}...a_1$ die **Umkehrung** von a. **D2.5:** Sei $\sum$ ein Alphabet. Für alle $x \in \sum^*$ und alle $i \in \mathbb{N}$ definieren wir die i-te **Iteration** x^i von x als

$$x^0 = \lambda, x^1 = xundx^i = xx^{i-1}$$

Beispiel: aabbaaaaaa $= a^2b^2a^6$

D2.6: Seien v,w $\in \sum^*$ für ein Alphabet $\sum$

- v heisst ein **Teilwort** von w $\iff \exists x, y \in \sum^* : w = xvy$
- v heisst ein **Präfix** von w $\iff \exists y \in \sum^* : w = vy$
- v heisst ein **Suffix** von w $\iff \exists x \in \sum^* : w = xv$

D2.7:

- $|x|_a$ ist die Anzahl der Vorkommen von a in x
- $|A|$ die KArdinalität der Menge A
- $\mathbf{P(A)} = \{S | S \subseteq A\}$ die Potenzmenge von A

D2.8: Sei $\sum = \{s_1, s_2, ..., s_m\}, m \geq 1$ ein Alphabet und sei $s_1 < s_2 < ... < s_m$ eine Ordnung auf $\sum$. Wir definieren die **kanonische Ordnung** auf $\sum^*$ für u,v $\in \sum^*$ wie folgt:

$$u < v \iff |u| < |v| \vee |u| = |v| \wedge u = x \cdot s_i \cdot u' \wedge v = x \cdot s_j \cdot v' \text{ für irgendwelche } x, u', v' \in \sum^* \text{ und } i < j$$

D2.9:

Definition 2.9. *Eine* ***Sprache*** L *über einem Alphabet* Σ *ist eine Teilmenge von* Σ^*. *Das Komplement* $\boldsymbol{L^{\complement}}$ *der Sprache* L *bezüglich* Σ *ist die Sprache* $\Sigma^* - L$.

$\boldsymbol{L_\emptyset} = \emptyset$ *ist die* ***leere Sprache***.

$\boldsymbol{L_\lambda} = \{\lambda\}$ *ist die einelementige Sprache, die nur aus dem leeren Wort besteht.*

Sind L_1 *und* L_2 *Sprachen über* Σ, *so ist*

$$\boldsymbol{L_1 \cdot L_2} = \boldsymbol{L_1 L_2} = \{vw \mid v \in L_1 \text{ und } w \in L_2\}$$

die ***Konkatenation*** *von* L_1 *und* L_2. *Ist* L *eine Sprache über* Σ, *so definieren wir*

$$\boldsymbol{L^0} := L_\lambda \text{ und } \boldsymbol{L^{i+1}} = L^i \cdot L \text{ für alle } i \in \mathbb{N},$$

$$\boldsymbol{L^*} = \bigcup_{i \in \mathbb{N}} L^i \text{ und } \boldsymbol{L^+} = \bigcup_{i \in \mathbb{N}-\{0\}} L^i = L \cdot L^*.$$

L^* *nennt man den* ***Kleene'schen Stern*** *von* L.

Die folgenden Mengen sind Sprachen über dem Alphabet $\Sigma = \{a, b\}$:

- $L_1 = \emptyset$,
- $L_2 = \{\lambda\}$,
- $L_3 = \{\lambda, ab, abab\}$,
- $L_4 = \Sigma^* = \{\lambda, a, b, aa, \ldots\}$,
- $L_5 = \Sigma^+ = \{a, b, aa, \ldots\}$,
- $L_6 = \{a\}^* = \{\lambda, a, aa, aaa, \ldots\} = \{a^i \mid i \in \mathbb{N}\}$,
- $L_7 = \{a^p \mid p \text{ ist eine Primzahl}\}$,
- $L_8 = \{a^i b^{2i} a^i \mid i \in \mathbb{N}\}$,
- $L_9 = \Sigma$,
- $L_{10} = \Sigma^3 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$.

L2.1 Seien L_1, L_2, L_3 Sprachen über einem Alphabet $\sum$. Dann gilt:

$$L_1 L_2 \cup L_1 L_3 = L_1 (L_2 \cup L_3)$$

L 2.2 Seien L_1, L_2, L_3 Sprachen über einem Alphabet $\sum$. Dann gilt:

$$L_1 (L_2 \cap L_3) \subseteq L_1 L_2 \cap L_1 L_3)$$

L2.3 Es existieren $U_1, U_2, U_3 \in (\sum_{bool})^*$ so dass

$$U_1 (U_2 \cap U_3) \subset U_1 U_2 \cap U_1 U_3$$

D2.10: Seiene $\sum_1$ und $\sum_2$ zwei beleibige Alphabete. Ein **Homomorphismus** von $\sum_1^*$ nach $\sum_2^*$ ist jede FUnktion $h : \sum_1^* \to \sum_2^*$ mit den folgendend Eigenschaften

- $h(\lambda) = \lambda$
- $h(uv) = h(u) \cdot h(v)$ für alle $u, v \in \sum_1^*$

1.2 2.3 Algorithmische Probleme

D2.11: Das**Entscheidungsproblem ($\sum$,L)** für ein gegebenes Alphabet $\sum$ und eine gegebene Sprache $L \subseteq \sum^*$ ist, für jedes $x \in \sum^*$ zu entscheiden, ob

$$x \in L \text{ oder } x \notin L$$

Wenn ein Algorithmus A die Entscheidungsproblem löst sagen wir auch, dass A die sprache L **erkennt**.
Wenn für eine Sprache L ein Algorithmus existiert, der L erkennt sagen wir dass L **rekursiv** ist

Üblicherweise stellen wir ein Entscheidungsproblem (Σ, L) wie folgt dar:

Eingabe: $x \in \Sigma^*$.
Ausgabe: $A(x) \in \Sigma_{\text{bool}} = \{0,1\}$, wobei

$$A(x) = \begin{cases} 1, & \text{falls } x \in L \text{ (Ja, } x \text{ hat die Eigenschaft)}, \\ 0, & \text{falls } x \notin L \text{ (Nein, } x \text{ hat die Eigenschaft nicht)}. \end{cases}$$

Beispielsweise ist $(\{a,b\}, \{a^n b^n \mid n \in \mathbb{N}\})$ ein Entscheidungsproblem, das man auch folgendermaßen darstellen kann:

Eingabe: $x \in \{a,b\}^*$.
Ausgabe: Ja, falls $x = a^n b^n$ für ein $n \in \mathbb{N}$.
Nein, sonst.

D2.12: Seien $\sum$ und Γ zwei Alphabete. Wir sagen dass ein Algorithmus A eine **Funktion (Transformation)** $f : \sum^* \to \Gamma^*$ **berechnet (realisiert)** falls

$$A(x) = f(x) \text{ für alle } x \in \sum^*$$

D2.13: Seien $\sum$ und Γ zwei Alphabete und sei $R \subseteq \sum^* \times \Gamma^*$ eine Relation in $\sum^*$ und Γ^*. Ein Algorithmus **A berechnet R (oder löst das Relationsproblem R)** falls für jedes $x \in \sum^*$, für das ein $y \in \Gamma^*$ mit $(x,y) \in R$ existiert gilt:

$$(x, A(x)) \in R$$

Sei $R_{\text{fac}} \subseteq (\Sigma_{\text{bool}})^* \times (\Sigma_{\text{bool}})^*$, wobei $(x,y) \in R_{\text{fac}}$ genau dann, wenn entweder Nummer(y) ein Faktor[6] von Nummer(x) ist, oder $y = 1$, wenn Nummer(x) eine Primzahl ist, oder $y = 0$, wenn $x \in \{0,1\}$. Eine anschauliche Darstellung dieses Relationsproblems könnte wie folgt aussehen.

Eingabe: $x \in (\Sigma_{\text{bool}})^*$.
Ausgabe: $y \in (\Sigma_{\text{bool}})^*$, wobei

$$\text{Nummer}(y) = \begin{cases} 0, & \text{falls } x = 0 \text{ oder } x = 1, \\ 1, & \text{falls } x \text{ ist eine Primzahl}, \\ k, & \text{sonst, wobei } k \text{ ein Faktor von Nummer}(x) \text{ ist}. \end{cases}$$

D2.14: Optimierungsproblem:

Definition 2.14. *Ein* ***Optimierungsproblem*** *ist ein 6-Tupel* $\mathcal{U} = (\Sigma_I, \Sigma_O, L, \mathcal{M}, \text{cost}, \text{goal})$, *wobei:*

(i) Σ_I *ist ein Alphabet (genannt* ***Eingabealphabet****),*

(ii) Σ_O *ist ein Alphabet (genannt* ***Ausgabealphabet****),*

(iii) $L \subseteq \Sigma_I^*$ *ist die Sprache der* ***zulässigen Eingaben*** *(als Eingaben kommen nur Wörter in Frage, die eine sinnvolle Bedeutung haben). Ein* $x \in L$ *wird ein* ***Problemfall*** *(****Instanz****)* ***von*** $\mathcal{U}$ *genannt.*

(iv) $\mathcal{M}$ *ist eine Funktion von* L *nach* $\mathcal{P}(\Sigma_O^*)$, *und für jedes* $x \in L$ *ist* $\mathcal{M}(x)$ *die* ***Menge der zulässigen Lösungen für*** x,

(v) cost *ist eine Funktion,* $\text{cost} : \bigcup_{x \in L} (\mathcal{M}(x) \times \{x\}) \to \mathbb{R}^+$, *genannt* ***Kostenfunktion****,*

(vi) goal $\in \{\text{Minimum}, \text{Maximum}\}$ *ist das* ***Optimierungsziel****.*

Eine zulässige Lösung $\alpha \in \mathcal{M}(x)$ *heißt* ***optimal*** *für den Problemfall* x *des Optimierungsproblems* U, *falls*

$$\text{cost}(\alpha, x) = \mathbf{Opt}_{\mathcal{U}}(x) = \text{goal}\{\text{cost}(\beta, x) \mid \beta \in \mathcal{M}(x)\}.$$

Ein Algorithmus A ***löst*** $\mathcal{U}$, *falls für jedes* $x \in L$

(i) $A(x) \in \mathcal{M}(x)$, $\quad \{A(x)$ ist eine zulässige Lösung des Problemfalls x von $\mathcal{U}.\}$

(ii) $\text{cost}(A(x), x) = \text{goal}\{\text{cost}(\beta, x) \mid \beta \in \mathcal{M}(x)\}$.

Falls goal $=$ *Minimum, ist* $\mathcal{U}$ *ein* ***Minimierungsproblem****; falls* goal $=$ *Maximum, ist* $\mathcal{U}$ *ein* ***Maximierungsproblem****.*

Teilproblem: Ein Optimierungsproblem $U_1 = (\sum_I, \sum_O, L', M, cost, goal)$ ist ein Teilproblem des Optimierungsproblems $U_2 = (\sum_I, \sum_O, L, M, cost, goal)$ falls $L' \subseteq L$

Knotenüberdeckung: Eine Knotenüberdeckung eines Graphen ist jede Knotenmenge $U \subseteq V$, so dass jede Kante aus E zu mindestens einem Knoten aus U inzident ist.

D2.15: Sei $\sum$ ein Alphabet, und sei $x \in \sum^*$. Wir sagen, dass ein Algorithmus A das Wort x **generiert**, falls A für die Eingabe λ die Ausgabe x liefert.

D2.16: Sei $\sum$ ein Alphabetm und sei $L \subseteq \sum^*$. A ist ein **Aufzählungsalgorithmus für L**, falls A für jede Eingabe $n \in \mathbb{N} - \{0\}$ die Wortfolge $x_1, x_2, ..., x_n$ ausgibt, wobei $x_1, x_2, ..., x_n$ die kanonisch n ersten Wörter in L sind.

1.3 Kolmogorov-Komplexität

Komprimierung Die Erzeugung einer kürzeren Darstellung eines Wortes x.

D2.17: Für jedes Wort $x \in (\sum_{bool})^*$ ist die **Kolmogorov-Komplexität K(x)** des Wortes x das Minimum der binären Längen der Pascal-Programme die x generieren.

L2.4 Es existiert eine Konstante d, so dass für jede $x \in (\sum_{bool})^*$

$$K(x) \leq |x| + d$$

D2.18: Die Kolmogorov-Komplexität einer natürlichen Zahl n ist

$$K(w_n) = K(Bin(n))$$

L 2.5 Für jede Zahl $n \in \mathbb{N} - \{0\}$ existiert ein Wort $w_n \in (\sum_{bool})^n$ so dass

$$K(w_n) \geq |w_n| = n$$

d.h es existiert für jede Zahl n ein nichtkomprimierbares Wort der Länge n

Satz 2.1 Seien A und B programmiersprachen. Es existiert eine Konstante $c_{A,B}$, die nur von A und B abhängt so dass

$$|K_A(x) - K_B(x)| \leq c_{A,B}$$

für alle $x \in (\sum_{bool})^*$

D2.19: Ein Wort $x \in (\sum_{bool})^*$ heisst **zufällig**, falls $K(x) \geq |x|$. Eine Zahl n heisst **zufällig**, falls

$$K(n) = K(Bin(n)) \geq \lceil log_2(n+1) \rceil - 1$$

Satz 2.2 Sei L eine Sprache über $\sum_{bool}$. Sei für jedes $n \in \mathbb{N} - \{0\}, z_n$ das n-te Wort in L bezüglich der kanonischen Ordnung. Wenn ein Programm A_L existiert, das das Entscheidungsproblem $(\sum_{bool}, L)$ löst, dann gilt für alle $n \in \mathbb{N} - \{0\}$, dass

$$K(z_n) \leq \lceil log_2(n+1) \rceil + c$$

wobei c eine von n unabhängige Konstante ist.

Satz 2.3 (Primzahlsatz) Die Anzahl der Primzahlen wächst so schnell wie die Funktion $\frac{n}{ln(n)}$

$$\lim_{n \to \infty} \frac{Prim(n)}{\frac{n}{ln(n)}} = 1$$

wobei Prim(n) ist die Anzahl der Primzahlen kleiner gleich n.

L2.6: Sei $n_1, n_2, n_3...$ eine steigende unendliche Folge natürlicher Zahlen mit $K(n_i) \geq \frac{\lceil log_2 n_i \rceil}{2}$. Für jedes $i \in \mathbb{N} - \{0\}$ sei q_i die grösste Primzahl, die die Zahl n_i teilt. Dann ist die Menge:

$$Q = \{q_i | i \in \mathbb{N} - \{0\}\}$$

unendlich. L2.6 zeigt dass es unendlich viele Primzahlen gibt und auch dass die Menge der grössten Primzahlfaktoren einer beliebigen unendlichen FOlge natuürlicherZahlen mit nichtrivialer Kolmogorov Komplexität unendlich ist.

Satz 2.4 Für unendlich viele $k \in \mathbb{N}$ gilt

$$Prim(k) \geq \frac{k}{2^{17} log_2(k) \cdot (log_2(log_2(k)))^2}$$