Übungsstunde 28.09.2022

Samstag, 24. September 2022 15:45

Motivation Warum sind Alphabete, Wörter & Sprachen für die Informatik interessant?

In der algorithmischen Datenverarbeitung können wir Programme als Texte über dem Alphabet der Rechnertastatur verstehen. Jedes Programm realisiert also eine Transformation von Eingabetexten in Ausgabetexte. Wir starten damit den Formalismus für den Umgang mit Texten als Informationsträger einzuführen

Theorie

Definition 2.1. *Eine endliche nichtleere Menge Σ heißt* ***Alphabet****. Die Elemente eines Alphabets werden* ***Buchstaben*** *(****Zeichen****,* ***Symbole****) genannt.*

Wichtig: Es dürfen beliebige, aber nur endlich viele Symbole für das Alphabet verwendet werden

Definition 2.2. *Sei Σ ein Alphabet. Ein* ***Wort*** *über Σ ist eine endliche (eventuell leere) Folge von Buchstaben aus Σ. Das* ***leere Wort*** $\boldsymbol{\lambda}$ *ist die leere Buchstabenfolge. (Manchmal benutzt man ε statt λ.)*

Die ***Länge*** $\boldsymbol{|w|}$ *eines Wortes w ist die Länge des Wortes als Folge, d. h. die Anzahl der Vorkommen von Buchstaben in w.*

$\boldsymbol{\Sigma^*}$ *ist die Menge aller Wörter über Σ,* $\boldsymbol{\Sigma^+} = \Sigma^* - \{\lambda\}$.

Definition 2.3. *Die* ***Verkettung*** *(****Konkatenation****) für ein Alphabet Σ ist eine Abbildung* $\mathrm{Kon}\colon \Sigma^* \times \Sigma^* \to \Sigma^*$, *so dass*

$$\mathrm{Kon}(x, y) = x \cdot y = xy$$

für alle $x, y \in \Sigma^*$.

Wichtig: Konkatenation ist assoziativ und $|xy| = |x| + |y|$

Definition 2.6. *Seien $v, w \in \Sigma^*$ für ein Alphabet Σ.*

- *v heißt ein* ***Teilwort*** *von $w \iff \exists x, y \in \Sigma^* : w = xvy$.*
- *v heißt ein* ***Präfix*** *von $w \iff \exists y \in \Sigma^* : w = vy$.*
- *v heißt ein* ***Suffix*** *von $w \iff \exists x \in \Sigma^* : w = xv$.*
- *$v \neq \lambda$ heißt ein* ***echtes*** *Teilwort (Präfix, Suffix) von w genau dann, wenn $v \neq w$ und v ein Teilwort (Präfix, Suffix) von w ist.*

Definition 2.9. *Eine* ***Sprache*** L *über einem Alphabet* Σ *ist eine Teilmenge von* Σ^*. *Das Komplement* $\boldsymbol{L^C}$ *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*

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

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.

Aufgabe: Betrachte das Alphabet $\Sigma = \{a, b, c\}$ und $n \in \mathbb{N}$. Bestimme die Anzahl der Wörter aus Σ^n (= Wörter der Länge n), die a als Teilwort enthalten.

Beweis: Oft ist es einfacher zunächst das Gegenteil zu zeigen. Wir zählen also wie viele Wörter aus Σ^n den Buchstaben a nicht enthalten. Für jede der n Positionen kann also entweder der Buchstabe b oder c gewählt werden. Aber dies sind genau alle möglichen Wörter der Länge n aus dem Alphabet $\{b, c\}$.

Da $|\{b,c\}^n| = 2^n$ folgern wir, dass es genau 2^n Wörter in Σ^n gibt, welche den Buchstaben a nicht enthalten. Aus $|\Sigma^n| = 3^n$ schliessen wir, dass es $3^n - 2^n$ Wörter in Σ^n gibt, welche das Teilwort a enthalten. □

Aufgabe: Beweise oder widerlege

a) Falls für eine Sprache $L^2 = L$ gilt, dann gilt auch $L^3 = L$

b) Es gibt eine Sprache L, die die Bedingung $L^2 = L$ erfüllt.

c) $L_2 \cdot (L_2 - L_1) = L_2^2 - L_2 \cdot L_1$, falls $L_1, L_2 \subseteq \Sigma^*$ (Σ beliebig).

Beweis:

a) Direkter Beweis. Angenommen $L^2 = L$. Dann gilt: $L^3 \overset{\text{Def}}{=} L^2 \cdot L \overset{\text{Annahme}}{=} L \cdot L \overset{\text{Def}}{=} L^2 \overset{\text{Annahme}}{=} L$ □

b) Wir geben Beispiele: $L = \emptyset$, $L = \{\lambda\}$, $L = \Sigma^*$

c) Wir geben ein Gegenbeispiel: Sei $L_1 = \{\lambda\}$, $L_2 = \{a\}^*$. Dann ist $L_2 - L_1 = \{a\}^+$, also $L_2 \cdot (L_2 - L_1) = \{a\}^* \cdot \{a\}^+ = \{a\}^+$. Andererseits ist $L_2^2 = \{a\}^* \{a\}^* = \{a\}^*$ und $L_2 \cdot L_1 = \{a\}^* \{\lambda\} = \{a\}^*$. Also $L_2^2 - L_2 L_1 = \{a\}^* - \{a\}^* = \emptyset$.

Partnerarbeit - Bitte schau dir die Lösung nicht vorher an

Aufgabe: Sei $\Sigma := \{0,1\}$, $n \in \mathbb{N}$. Bestimme die Anzahl der Wörter der Länge n über Σ welche

a) nicht das Teilwort 01 enthalten

b) keines der Teilwörter 01 und 00 enthalten

Beweis:

a) Wenn ein Wort der Länge n nicht das Teilwort 01 enthält, dann hat es die Form 1^L0^m für irgendwelche $L,m \in \mathbb{N}$ s.d. $L+m=n$. Für gegebenes n gilt also: $0 \leq L \leq n$ und m ist durch die Wahl von L eindeutig bestimmt. Da es $n+1$ Möglichkeiten zur Wahl von L gibt, gibt es genau $n+1$ solche Wörter der Länge n. □

b) Falls $n \leq 1$, so erfüllen alle Wörter die Bedingung. Sei nun $n \geq 2$. Aus a) wissen wir bereits, dass die gültigen Wörter in der Menge $\{1^L0^m \mid L,m \in \mathbb{N}, L+m=n\}$ sein müssen.
Da 00 nicht enthalten sein darf, gilt $m \in \{0,1\}$. Folglich sind 1^n, $1^{n-1}0$ die einzigen Möglichkeiten. □

Kolmogorov - Komplexität

Motivation: Nachdem wir das Regelwerk für Texte als Informationsgrösse gebildet haben, können wir uns nun nach dem Aufwand einer solchen Darstellung fragen. Wie in der Informatik üblich beschränken wir uns auf das bool'sche Alphabet $\Sigma_{bool} = \{0,1\}$ zur Darstellung von Texten.
Intuitiv wissen wir, dass $(01)^5$ schneller geschrieben werden kann als 0101010101.
Intelligente Leute würden diesen Unterschied so beschreiben: $(01)^5$ ist eine komprimierte Darstellung von
Diese Komprimierung können wir als kleineren Informationsgehalt auffassen. Die KK verallgemeinert diese Idee von "weniger Speicherplatz" auf die binäre Länge von Pascal-Programmen welche die Information als Output erzeugt.

Theorie:

Definition 2.17. *Für jedes Wort $x \in (\Sigma_{\text{bool}})^*$ ist die* ***Kolmogorov-Komplexität $K(x)$ des Wortes x*** *das Minimum der binären Längen der Pascal-Programme, die x generieren.*

↳ i.e. wähle aus allen Maschienencodes (0110...) von Pascal-Programmen welche x generieren das kürzeste (generiert: Output von Programm ohne Input ist x)

↳ Wenn wir ein Programm schreiben, welches x generiert, so ist dies sicherlich eine obere Schranke für K(x), aber eventuell nicht die beste obere Schranke!

Aufgabe: Gibt es eine Konstante $c \in \mathbb{N}$, s.d. $\forall n \in \mathbb{N} - \{0\}$: $K(x_n) \leq \log_2(\sqrt{n}) + c$ für eine unendliche Folge von paarweise verschiedenen Wörtern $(x_n)_{n \in \mathbb{N}-\{0\}}$?

Beweis: Aufgaben von diesem Typ lassen sich immer gleich lösen. Entweder man führt einen Widerspruchsbeweis mit einem Kombinatorikargument (ähnlich zu dem in Lemma 2.5) durch oder man entwirft eine solche Folge, schreibt ein Pascal-Programm und schätzt die KK nach oben ab (das sehen wir nächste Woche). Mit göttlicher Intuition probieren wir nun Variante 1.

Beweis per Widerspruch. Angenommen es existiert eine solche Folge. Sei $n \in \mathbb{N} - \{0\}$ fix und setze

$t := \log_2(\sqrt{n}) + c = \log_2(n^{1/2}) + c = \frac{1}{2}\log_2(n) + c$

Es gibt $\sum_{i=0}^{t} 2^t = 2^{t+1} - 1$ verschiedene Wörter mit binärer Länge höchstens t.

$2^{t-1} - 1 = 2^{\frac{1}{2}\log_2(n) + c + 1} - 1 < \infty$. Folglich können höchstens $2^{t+1} - 1$ paarweise verschiedene Wörter durch Programme mit binärer Länge $\leq t$ erzeugt werden.

Der entscheidende Punkt ist die Endlichkeit!

Mathematisch gesehen können wir nun ein n finden/wählen, s.d. $2^{t+1} - 1 < n$, was der Annahme paarweise verschieden widerspricht. Formell: Für ein genügend großes n, welches nur von der Konstante c abhängt, gilt: $2^{t-1} - 1 < n$. Also $\exists i, j \in \{1, \dots, n\}$, $i < j$, s.d. $x_i = x_j$ ↯ was unserer Annahme ($x_i \neq x_j$ für $i \neq j$) widerspricht. □