

Theoretische Informatik

HS25

Übungsgruppe 10

Rita Nagy

BSc Mathematik

rinagy@student.ethz.ch

Links:

- Meine Notizen
<https://rinagy.github.io/>
- Altprüfungen,
Lösungen zum Buch...
<https://exams.vis.ethz.ch/category/TheoretischeInformatik>
- Buch - Theoretische Informatik
<https://link.springer.com/book/10.1007/978-3-658-06433-4>



Serien:

- Abgabe:
 - **Freitags bis 11.15 Uhr auf Moodle**
 - bis zu **Dreiergruppen** (verschiedene Übungsgruppen möglich)
 - es soll nur eine Person von der Gruppe abgeben
 - **Name und Leginummer** aller Beteiligten auf Blatt notieren
- In der Regel **30 Punkte** pro Serie möglich
- Lösungen begründen, ausführlich argumentieren

Prüfungen:

- **Midterm** (Anfang November)
50% der Punkte aus Serien für Zulassung
- **Endterm** (Mitte Dezember)
50% der Punkte aus Serien für Zulassung
- **Sessionsprüfung** (Prüfungsphase)
angemeldet bleiben!

2.2 Alphabete, Wörter und Sprachen

Def: Eine endliche nichtleere Menge Σ heißt Alphabet.

Def: Die Elemente eines Alphabets werden Buchstaben genannt.

Def: Sei Σ ein Alphabet.

Ein Wort über Σ ist eine Folge von Buchstaben aus Σ .

Def: Das leere Wort λ ist die leere Buchstabenfolge.

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

Def: Σ^* ist die Menge aller Wörter über dem Alphabet Σ .

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

Def: Σ^n ist die Menge aller Wörter der Länge n über Σ .

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:

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

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

Lösung: k^n

Def: Die Verkettung (Konkatenation) für ein Alphabet Σ 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: $\Sigma = \{0, 1\}$

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

$$\cdot (101)^3 = 101101101$$

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

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

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

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

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 Σ .

- v heisst Teilwort von w $\Leftrightarrow \exists x, y \in \Sigma^* : w = xv y$
- 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 Σ .

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

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

$$\vee |u| = |v| \wedge u = xs_i u' \wedge v = xs_j v'$$

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

In Worten:

- 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 Σ ist eine Teilmenge von Σ^* .
 $L = \{w \in \Sigma^* \mid \text{Bedingungen}\} \subseteq \Sigma^*$

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 Σ . Wir definieren:

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

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

$L^* := \bigcup_{i \in \mathbb{N}} L^i$ heißt der Kleene'sche Stern 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 $L_1 L_2$?

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 Σ_1^* nach Σ_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 Σ_1, Σ_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 Σ arbeiten
- (ii) $\forall x \in \Sigma^*: A(x) = B(x)$

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

Def: Ein Algo A löst das Entscheidungsproblem (Σ, 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.

2.4 Kolmogorov-Komplexität

Def: 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.

Def: Die Kolmogorov-Komplexität einer natürlichen Zahl n ist
 $K(n) = K(\text{Bin}(n))$

Aufgabe 1 aus Midterm HS04

Es sei $(w_j)_{j=0}^{\infty}$ die durch $w_j := b^{j^3}$ gegebene (unendliche) Folge von Wörtern. Zeigen Sie, dass es eine Konstante c gibt, so dass für alle j gilt:

$$K(w_j) \leq \frac{1}{3} \log |w_j| + c$$

Pseudo-Code

```
begin
    N := j;
    N := N · N · N;
    for i = 1 to N do
        write(b);
    end
```

Komplexitätsberechnung

$$\begin{aligned} K(w_j) &\leq \lceil \log_2(j+1) \rceil + c' \\ |w_j| &= |b^{j^3}| = j^{3^3} \Rightarrow j = |w_j|^{\frac{1}{3}} \\ \text{somit: } K(w_j) &\leq \lceil \log_2(1+|w_j|^{\frac{1}{3}}) \rceil + c' \\ &\leq \frac{1}{3} \lceil \log_2(|w_j|) \rceil + c \end{aligned}$$



Lösungsvorschläge – Blatt 1

26. September 2025

Lösung zu Aufgabe 1

- (a) Wenn ein Wort der Länge $n \in \mathbb{N} - \{0\}$ nicht das Teilwort 01 enthält, dann hat es die Form $1^l 0^m$ für irgendwelche $l, m \in \mathbb{N}$ mit $l + m = n$. Für ein gegebenes n gilt also $0 \leq l \leq n$ und m ist durch die Wahl von l eindeutig bestimmt. Damit gibt es genau $n + 1$ solche Wörter der Länge n .
- (b) Sei $\Sigma = \{0, 1\}$ und $n \in \mathbb{N} - \{0\}$. Wir bezeichnen mit L_n die Menge der Wörter aus Σ^n , die nicht das Teilwort 00 enthalten und setzen $N(n) = |L_n|$. Wir wollen $N(n)$ rekursiv bestimmen. Offenbar gilt $N(1) = 2$, weil die Wörter 0 und 1 beide 00 nicht als Teilwort enthalten. Ausserdem gilt $N(2) = 3$, weil alle Wörter der Länge 2 ausser 00 selbst das Teilwort 00 nicht enthalten.

Wenn wir nun ein Wort $w \in L_{n+1}$ mit $n \geq 2$ betrachten, dann können wir w schreiben als $w = xab$ mit $a, b \in \{0, 1\}$ und $x \in L_{n-1}$. Falls $b = 1$, dann ist für xa ein beliebiges Wort aus L_n möglich. Falls $b = 0$, dann muss $a = 1$ gelten, aber für x ist ein beliebiges Wort aus L_{n-1} möglich. Damit ergibt sich für $N(n)$ die Rekurrenzgleichung

$$N(n+1) = N(n) + N(n-1).$$

Dies ist offenbar die bekannte Rekurrenz der Fibonacci-Zahlen, die definiert sind durch $F(0) = 0$, $F(1) = 1$ und $F(n) = F(n-1) + F(n-2)$ für alle $n \geq 2$. Wegen der Anfangsbedingungen $N(1) = 2$ und $N(2) = 3$ gilt also

$$N(n) = F(n+2).$$

Die Folge der Fibonacci-Zahlen kann man explizit beschreiben durch die Formel von Moivre-Binet als

$$F(n) = \frac{\varphi^n - (1-\varphi)^n}{\sqrt{5}},$$

wobei $\varphi = \frac{1+\sqrt{5}}{2}$. Damit ergibt sich für $N(n)$ die explizite Darstellung

$$N(n) = \frac{\varphi^{n+2} - (1-\varphi)^{n+2}}{\sqrt{5}}.$$

- (c) Wir haben in Aufgabenteil (a) gesehen, dass jedes Wort der Länge n , dass das Teilwort 01 nicht enthält, die Form $1^l 0^m$ haben muss für $l + m = n$. Die einzigen Wörter dieser Form, die nicht das Teilwort 00 enthalten, sind 1^n und $1^{n-1}0$, also gibt es für jedes $n \in \mathbb{N} - \{0\}$ genau 2 solche Wörter.

Lösung zu Aufgabe 2

- (a) Sei Σ ein Alphabet und seien $u, v \in \Sigma^*$. Dann existieren $k, l \in \mathbb{N}$ mit $u = u_1 u_2 \dots u_k$ und $v = v_1 v_2 \dots v_l$, wobei $u_1, \dots, u_k, v_1, \dots, v_l \in \Sigma$. Damit ist

$$uv = u_1 u_2 \dots u_k v_1 v_2 \dots v_l$$

und somit

$$\begin{aligned}(uv)^R &= v_l \dots v_2 v_1 u_k \dots u_2 u_1 \\ &= v^R u^R,\end{aligned}$$

weil $v^R = v_l \dots v_2 v_1$ und $u^R = u_k \dots u_2 u_1$ gilt.

- (b) Die Aussage gilt nicht, wir geben ein Gegenbeispiel an: Sei $\Sigma = \{a\}$ und seien $L_1 = \{a\}$ und $L_2 = \{aa\}$. Dann gilt $L_1 \cap L_2 = \emptyset$, also $(L_1 \cap L_2)^* = \{\lambda\}$. Aber es sind $L_1^* = \{a\}^*$ die Sprache aller Wörter, die beliebig viele Buchstaben a enthalten, und $L_2^* = \{aa\}^*$ die Sprache aller Wörter mit einer geraden Anzahl von Buchstaben a . Damit gilt $L_2^* \subseteq L_1^*$ und somit $L_1^* \cap L_2^* = L_2^* \neq \{\lambda\}$.
- (c) Auch diese Aussage ist falsch, wir geben wieder ein Gegenbeispiel an: Sei $L_1 = \{aa\}$ über dem Alphabet $\Sigma = \{a\}$. Dann ist $L_1^C = \{a\}^* \setminus \{aa\}$. Es gilt $L_1^* = \{a^{2i} \mid i \in \mathbb{N}\}$, also $(L_1^*)^C = \{a^{2i+1} \mid i \in \mathbb{N}\}$, und $(L_1^C)^* = \{a\}^*$.

Lösung zu Aufgabe 3

- (a) Wir wählen $\Sigma = \{a\}$ und

$$L_1 = \{a, aa, \dots, a^k\} = \{a^i \mid 1 \leq i \leq k\}.$$

Dann gilt für

$$L_2 = \{\lambda, a\},$$

dass

$$\begin{aligned}L_1 \cdot L_2 &= L_1 \cdot \{\lambda\} \cup L_1 \cdot \{a\} \\ &= L_1 \cup \{a^i a \mid 1 \leq i \leq k\} \\ &= L_1 \cup \{a^i \mid 2 \leq i \leq k+1\} \\ &= \{a^i \mid 1 \leq i \leq k+1\},\end{aligned}$$

also ist $|L_1 L_2| = k+1$.

- (b) Hier lässt sich ein Beispiel analog zu Aufgabenteil (a) konstruieren. Wir wählen wieder $\Sigma = \{a\}$ und

$$L_1 = \{a, aa, \dots, a^k\} = \{a^i \mid 1 \leq i \leq k\}.$$

Dann gilt für

$$L_2 = \{\lambda, a, aa, aaa, aaaa, aaaaa\},$$

dass

$$\begin{aligned} L_1 \cdot L_2 &= \bigcup_{j=0}^5 L_1 \cdot \{a^j\} \\ &= \bigcup_{j=0}^5 \{a^i a^j \mid 1 \leq i \leq k\} \\ &= \{a^i \mid 1 \leq i \leq k+5\}, \end{aligned}$$

also ist $|L_1 L_2| = k+5$.

- (c) Wir wählen $\Sigma = \{a, b\}$ und

$$L_1 = \{a^i \mid 1 \leq i \leq k\}$$

und

$$L_2 = \{b^i \mid 1 \leq i \leq l\}$$

Dann ist

$$L_1 L_2 = \{a^i b^j \mid 1 \leq i \leq k \text{ und } 1 \leq j \leq l\},$$

also $|L_1 L_2| = k \cdot l$.

- (d) Um dieselbe Grösse von $L_1 \cdot L_2$ über einem einbuchstabigen Alphabet zu erreichen, müssen wir sicherstellen, dass jedes Paar von Wörtern aus L_1 und L_2 ein unterschiedliches Gesamtwort erzeugt. Dies können wir erreichen, wenn wir die Längen der Wörter in L_1 so wählen, dass wir für jedes $v \in L_1$ durch das Anhängen eines beliebigen Wortes $w \in L_2$ nicht das nächstlängere Wort aus L_1 erreichen können. Die Idee ist also, die Längen der Wörter in L_2 möglichst klein zu halten und die Längen der Wörter in L_1 hinreichend unterschiedlich zu wählen. Sei $\Sigma = \{a\}$. Dann setzen wir

$$L_1 = \{a^{li} \mid 0 \leq i \leq k-1\}$$

und

$$L_2 = \{a^j \mid 0 \leq j \leq l-1\}.$$

Dann ist

$$L_1 L_2 = \{a^{li} a^j \mid 0 \leq i \leq k-1 \text{ und } 0 \leq j \leq l-1\}.$$

Alle Wörter in $L_1 L_2$ haben paarweise unterschiedliche Längen, weil alle Wortlängen in L_1 Vielfache von l sind und alle Wörter in L_2 kürzer als l sind. Also gilt $|L_1 L_2| = k \cdot l$.

Korrekturhinweise – Blatt 1

19. September 2025

Wie schon auf dem Aufgabenblatt erwähnt, legen wir Wert darauf, dass die Lösungen vernünftig begründet werden. Wir erwarten, dass die abgegebenen Lösungen vollständig ausformuliert sind und nicht nur aus einer Ansammlung von unkommentierten Formeln bestehen. Als Anhaltspunkt für die Art der Formulierung, die wir gerne hätten, könnt Ihr jeweils die Musterlösung nehmen, die aber teilweise auch zusätzliche Erklärungen enthält, die wir von den Studierenden nicht erwarten.

Hier wie bei allen zukünftigen Korrekturhinweisen sind die Angaben nur als Richtlinie zu verstehen. Es ist zum Teil schwierig, die Punkte für eine Aufgabe genau auf die einzelnen Schritte der Argumentation zu verteilen, hier ist auch Euer Augenmaß gefordert.

Korrekturhinweise zu Aufgabe 1

- (a) Hier würde ich 1 Punkt vergeben für die Form solcher Wörter und 1 Punkt für das korrekte Zählen.
- (b) Rekursiver Ansatz 1 Punkt, Anfangsbedingungen 1 Punkt, Herleitung der Rekurrenz 3 Punkte, Zurückführen auf Fibonacci 1 Punkt. Dabei reicht uns die Angabe, dass es sich um die (verschobene) Fibonaccifolge handelt, die Herleitung der expliziten Formel ist nicht nötig.
- (c) Dies ist der einfachste Aufgabenteil, hier würde ich 1 Punkt dafür vergeben, dass der Fall $n = 1$ berücksichtigt wird, und 1 Punkt für den Fall $n \geq 2$.

Korrekturhinweise zu Aufgabe 2

Teilaufgabe (a) ist so einfach, dass die meisten Teilnehmer hier alle Punkte bekommen werden. Achtet bitte insbesondere auf eine saubere Argumentation, die genaue Punkteverteilung überlasse ich Euch.

Bei Teilaufgabe (b) würde ich 1 Punkt für das Gegenbeispiel vergeben und zwei Punkte für die Argumentation, warum die angegebenen Sprachen ein Gegenbeispiel sind. Bei (c) ist das Finden des Gegenbeispiels etwas schwieriger, deshalb gibt es hier 2 Punkte für das Gegenbeispiel und 2 Punkte für die Argumentation.

(bitte wenden)

Korrekturhinweise zu Aufgabe 3

Zu Teil (a) bis (c): Jeweils 1 Punkt für das Beispiel und 1 Punkt für die Argumentation. Wenn bei (c) jemand darauf verweist, dass das direkt aus (d) folgt, ist das auch in Ordnung, unter der Voraussetzung, dass (d) sinnvoll bearbeitet wurde. Zu Teil (d): Dieses Beispiel ist etwas komplizierter als die aus (a) bis (c). Ich würde hier wieder eine ähnliche Verteilung ansetzen, also 2 Punkte für das Beispiel (dabei muss man aufpassen, dass wirklich keine Überlappung der Bereiche auftritt). Weitere 2 Punkte dann für eine saubere Begründung.

Veit

Péter

✓ Valeria

Bilyana

Enrico

Jean

✓ Annika

Janne

Boris

✓ Fadri

✓ Relja

Nicolas

Kirill

Leo

Alexander

Kira

Martin

Dario

✓ Sascha

Neli

Annika

Guiheng

Hehui

Mateo