

# Serie 2

André Graubner, Lukas Walker, Leonard von Kleist

4. Oktober 2018

## 1 Aufgabe 4

Sei  $L$  eine unendliche Sprache über einem Alphabet  $\Sigma$ .

Richtung 1 ( $\Rightarrow$ ): Sei  $L$  rekursiv. Das heisst per Definition, dass ein Algorithmus  $A_{erk}$  existiert, der  $L$  erkennt. Wir konstruieren einen Aufzählungsalgorithmus (mit Eingabe  $n$ ) für  $L$ : Iteriere in kanonischer Reihenfolge durch  $\Sigma^*$  und überprüfe mit  $A_{erk}$  für jedes Element, ob es  $\in L$  ist. Falls ja, gebe es aus. Fahre so lange fort, bis wir  $n$  Elemente ausgegeben haben.

Richtung 2 ( $\Leftarrow$ ): Sei  $A_{enum}$  (mit Eingabe  $n$ ) ein Aufzählungsalgorithmus für  $L$ . Wir konstruieren einen Algorithmus (mit Eingabe  $x$ ), der  $L$  erkennt: Bezeichne  $k$  die Stelle, an der das Wort  $x$  kanonisch in  $\Sigma^*$  steht. Rufe  $A_{enum}$  mit Eingabe  $k$  auf. Wenn eines der von  $A_{enum}$  ausgegebenen Wörter  $x$  ist, so ist  $x \in L$ , also geben wir 1 (bzw. "ja", etc) aus, sonst 0.

## 2 Aufgabe 5

Im Intervall  $[2^n, 2^{n+1} - 1]$  gibt es  $2^n$  verschiedene Zahlen. Wir betrachten jeweils lediglich die kürzesten Maschinencodes, die diese Zahlen generieren. Weiterhin ist klar, dass diese Codes für zwei verschiedene Zahlen unterschiedlich sein müssen. Es reicht also zu zeigen, dass von  $2^n$  paarweise unterschiedlichen binären Strings für jedes  $i < n$  mindestens  $2^n - 2^{n-i}$  die Länge  $n - i$  haben.

Die Anzahl der (nichtleeren) Bitstrings von Länge kleiner als  $n - i$  ist

$$\sum_{k=1}^{n-i-1} 2^k = 2^{n-i} - 2$$

Wenn wir diese Anzahl nun jedoch von der Menge an Zahlen in unserem Intervall abziehen, sehen wir:

$$2^n - (2^{n-i} - 2) = 2^n - 2^{n-i} + 2 \geq 2^n - 2^{n-i}$$

Also gilt für mindestens  $2^n - 2^{n-i}$  der Zahlen  $K(x) \geq n - i$ .

## 3 Aufgabe 6a

Mit einem Programm der Form "Print "0"  $8 * n^4$  times" mit der Eingabe  $n$  können wir jedes  $w_n$  generieren. Hierbei hat alles in dem Programm ausser  $n$  eine konstante Länge.

Demnach gilt:

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

Um die Komplexität nun im Verhältnis zu  $w_n$  anzugeben, müssen wir also nun die Zahl  $n$  als Funktion von der Länge von  $w_n$  angeben. Hierbei gilt (unter der Annahme, dass jedes Potenzzeichen eine weitere Iteration des Strings darstellt, also der Term  $0^{2^3} = 00^3 = 0^6$  statt  $0^8$  heisst):

$$|w_n| = 1 * 2 * 2 * 2 * n^4 = 8 * n^4 \Rightarrow n = \sqrt[4]{\frac{|w_n|}{8}}$$

Also ist eine obere Schranke für  $K(w_n)$ :

$$K(w_n) \leq \left\lceil \log_2 \left( \sqrt[4]{\frac{|w_n|}{8}} + 1 \right) \right\rceil + c$$

## 4 Aufgabe 6b

Sei  $y_i = (2^{2^i})^4$  für ein  $i \in \mathbb{N}_{>0}$ . Dann gibt es ein Programm, welches  $y_i$  aus  $i$  konstruiert. Es gibt also eine (feste) Konstante  $c$ , so dass für alle  $y_i$  gilt::

$$K(y_i) \leq \lceil \log_2(i+1) \rceil + c$$

Hierbei ist zu beachten, dass  $i$  in folgendem Verhältnis zu  $y_i$  steht:

$$i = \log_2(\log_2(\log_2(\sqrt[4]{y_i})))$$

Es gilt also für alle  $y_i$ :

$$K(y_i) \leq \lceil \log_2(i+1) \rceil + c \leq i + c = \lceil \log_2(\log_2(\log_2(\sqrt[4]{y_i}))) \rceil + c$$