ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
https://courses.ite.inf.ethz.ch/theo_inf_21

Lösungsvorschläge – Blatt 2

Zürich, 8. Oktober 2021

Lösung zu Aufgabe 4

(a) Wir definieren die Folge $(x_n)_{n=1}^{\infty}$ mit $x_n = 0^{2^n}$ für jedes $n \in \mathbb{N} - \{0\}$. Es ist offenbar, dass alle Wörter x_n paarweise unterschiedlich sind.

Wir geben jetzt für jedes $n \in \mathbb{N} - \{0\}$ ein Programm an, das x_n erzeugt:

```
begin
  s := n;
  j := 1;
  for i := 1 to s do
    j := j * 2;
  for i := 1 to j do
    write(0);
end.
```

Dieses Programm berechnet zuerst in einer Schleife $j = 2^n$. In einer weiteren Schleife gibt das Programm dann 2^n Mal das Zeichen 0 aus, was genau das Wort x_n ergibt.

Der einzige Teil des Quellcodes, der von x_n abhängt, ist die Darstellung von n in der zweiten Zeile. Der restliche Quellcode hat eine konstante Länge. Also ist die binäre Länge des Quellcodes höchstens $\lceil \log_2(n+1) \rceil + c$ für eine Konstante c.

Damit lässt sich die Kolmogorov-Komplexität von x_n von oben abschätzen durch

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

für eine Konstante c.

Die Länge von x_n ist $|x_n| = |0^{2^n}| = 2^n$. Also gilt $\log_2 \log_2 |x_n| = \log_2 n$ und somit erhalten wir eine obere Schranke von

$$K(x_n) \leq \log_2 \log_2 |x_n| + (c+1)$$

für die Kolmogorov-Komplexität von x_n.

(b) Wir zeigen indirekt, dass es keine solche Folge $(x_n)_{n=1}^{\infty}$ paarweise unterschiedlicher Wörter gibt. Nehmen wir an, $(x_n)_{n=1}^{\infty}$ sei eine solche Folge. Dann gibt es eine Konstante $c \in \mathbb{N}$, so dass für alle $n \in \mathbb{N} - \{0\}$

$$K(x_n) \leq \log_2 \sqrt{n} + c$$

gilt.

Für ein fixes $n \in \mathbb{N} - \{0\}$ und $t := \log_2 \sqrt{n} + c$ gibt es höchstens

$$\sum_{i=0}^{t} 2^i = 2^{t+1} - 1 = 2^{\log_2 \sqrt{n}+c+1} - 1 = \sqrt{n} \cdot 2^{c+1} - 1$$

Quellcodes der binären Länge höchstens t, also können höchstens $\sqrt{n} \cdot 2^{c+1} - 1$ paarweise unterschiedliche Wörter durch Programme mit Quellcodes der binären Länge höchstens t erzeugt werden. Für ein genügend grosses n, das nur von der Konstante c abhängt, gilt $\sqrt{n} \cdot 2^{c+1} - 1 < n$, was im Widerspruch dazu steht, dass die n Wörter $x_1, \ldots, x_n$ paarweise unterschiedlich sind und $K(x_i) \leq t$ für alle $i \in \{1, \ldots, n\}$ gilt, d.h., $x_1, \ldots, x_n$ durch Programme mit Quellcodes der binären Länge höchstens t erzeugt werden.

Lösung zu Aufgabe 5

Wir beweisen die Aussage indirekt. Nehmen wir an, es gebe unendlich viele zufällige Zahlen in der Menge $\{n^2 \mid n \in \mathbb{N}\}$. Nach Definition bedeutet dies, dass es unendlich viele $n \in \mathbb{N}$ gibt, sodass

$$K(n^2) \geq \lceil \log_2(n^2 + 1) \rceil - 1 \geq 2 \cdot \log_2 n - 1 \tag{1}$$

gilt.

Ferner lässt sich die Zahl n^2 für jedes $n \in \mathbb{N}$ mit einem Programm C_n berechnen, das die binäre Kodierung von n enthält, die Zahl n^2 berechnet und anschliessend ausgibt. Alle Bestandteile dieses Programms mit Ausnahme der Kodierung von n haben eine konstante Länge, also hat C_n die binäre Länge $\lceil \log_2(n+1) \rceil + c \leq \log_2 n + (c+1)$ für eine Konstante c. Es folgt, dass

$$K(n^2) \leq \log_2 n + (c + 1) \tag{2}$$

gilt.

Aus den beiden Schranken (1) und (2) für die Kolmogorov-Komplexität ergibt sich, dass

$$2 \cdot \log_2 n - 1 \leq \log_2 n + (c + 1)$$

für unendlich viele $n \in \mathbb{N}$ gelten muss. Hieraus folgt wiederum, dass für diese n auch

$$\log_2 n \leq c + 2$$

gelten muss, was unmöglich ist, da c eine Konstante ist und $\log_2 n$ mit n beliebig wächst. Somit ist unsere Annahme falsch und die zu beweisende Aussage folgt.

Lösung zu Aufgabe 6

Offenbar gibt es ein Programm A_L, das für ein gegebenes Wort $x \in \Sigma^*_{\text{bool}}$ entscheidet, ob $x \in L$ gilt. Nach Satz 2.2 aus dem Buch gilt damit für das kanonisch n-te Wort x_n aus L

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

für eine von n unabhängige Konstante c.

Wir schätzen nun n in Abhängigkeit von $|x_n|$ ab, indem wir die Anzahl Wörter der Form $1^i0^j1^k$ abschätzen, deren Länge $i+j+k$ höchstens $|x_n|$ beträgt. Die ersten n Wörter $x_1, \ldots, x_n$ aus L in der kanonischen Reihenfolge haben die Form $1^i0^j1^k$ mit $i+j+k \leq |x_n|$, aus $i+j+k \leq |x_n|$ folgt, dass $i, j, k \leq |x_n|$ gilt, und deshalb gilt $n \leq |x_n|^3$. Daraus folgt schliesslich, dass

$$K(x_n) \leq \log_2 n + (c+1) \leq \log_2 |x_n|^3 + (c+1) = 3 \cdot \log_2 |x_n| + (c+1)$$

gilt.

Bemerkung: Man könnte eine alternative Lösung vorschlagen, in der ein Programm einfach die binären Kodierungen der drei Werte i, j und k enthält und daraus die Ausgabe berechnet. Leider ist dies nicht so einfach möglich, wie es zunächst scheint. Weil i, j und k beliebige natürliche Zahlen sein können, kann ihre binäre Kodierung in manchen Fällen mit der Darstellung anderer Teile im Quellcode (z.B. Funktionsnamen) übereinstimmen. Wir müssten also im Quellcode genau angeben, wo genau die binären Kodierungen der Werte i, j und k im Quellcode stehen, damit sie richtig interpretiert werden. Für eine einzelne Zahl i können wir dies tun, indem wir die Längen der konstanten Teile im Quellcode davor und dahinter mit konstantem Platz angeben. Ab zwei Zahlen beliebiger Länge ist aber auch mit der Angabe von mehreren solchen konstanten Längen (davor, dazwischen und dahinter) eine eindeutige Interpretation des Quellcodes noch nicht gewährleistet. Es könnte passieren, dass es mehrere gültige Möglichkeiten gibt, die Werte i, j und k und den restlichen Quellcode zu interpretieren.

Deshalb muss man die Zahlen i, j und k selbstbeschränkend kodieren (vgl. die Kodierung im Beweis des Primzahlsatzes), was aber allgemein zu einer grösseren binären Länge des Quellcodes führt. Eine einfache selbstbeschränkende Kodierung (nach jedem Bit ausser dem letzten wird eine Null angehängt, nach dem letzten Bit wird eine Eins angehängt, die zu einer eindeutigen Interpretation der Kodierung führt) liefert einen Quellcode der binären Länge $2 \cdot (\log_2 i + \log_2 j + \log_2 k) + c$ für eine Konstante c, was eine binäre Länge bis zu $6 \cdot \log_2 |x_n| + c$ ergibt.

Um die binäre Länge $3 \cdot \log_2 |x_n| + c$ zu erreichen, müssten die binären Darstellungen von i, j, k dieselbe Länge haben. Dann kann man wie bei einer einzigen Zahl die Längen der konstanten Teile im Quellcode vor und hinter der konkatenierten Darstellung der drei Zahlen mit konstantem Platz angeben. Anschliessend kann man die Zahlen i, j, k eindeutig ablesen, indem man die binäre Kodierung in drei gleich lange Teile zerlegt. Falls die Zahlen i, j, k nicht dieselbe binäre Länge haben, muss man ihre binären Darstellungen mit zusätzlichen Nullen am Anfang ergänzen.