Prof. Dr. M. Chimani
Dipl.-Inf. S. Beyer
Dipl.-Math. I. Hedtke

Universität Osnabrück
Theoretische Informatik
Sommersemester 2014

Übungsblatt 9 zur Informatik C̈: Einführung in die Theoretische Informatik

Ausgabe: 27. Juni

Besprechung: 7.–9. Juli

Aufgabe 9.1. Paradoxon

Das Wort „Substantiv“ ist ein Substantiv, das Wort „deutsch“ ist deutsch, aber das Wort „Verb“ ist kein Verb und das Wort „englisch“ ist nicht englisch. Wir nennen Wörter die selbst unter den durch das Wort bezeichneten Begriff fallen als *selbstidentifizierend* und alle anderen als *nichtselbstidentifizierend*.

Zeigen Sie, dass es Wörter gibt, die weder selbstidentifizierend noch nichtselbstidentifizierend sind.

Aufgabe 9.2. Semi-Entscheidbarkeit

Betrachten Sie das folgende Entscheidungsproblem:

Gegeben: Turingmaschine M und ein Zustand Z_i von M
Frage: Kann Z_i in M erreicht werden?

Beweisen Sie, dass dieses Problem *semi-entscheidbar* ist.

Beachten Sie, dass hier nicht nach der Erreichbarkeit im Automaten (Graphen) gefragt wird, sondern danach, ob der Zustand Z_i bei einer Ausführung von M bei *irgendeiner* Eingabe besucht wird.

Aufgabe 9.3. Collatz-Vermutung beweisen

Aus Aufgabe 8.2 kennen Sie schon die Collatz-Folge $n, C(n), C(C(n)), \ldots$, wobei

$$C(n) := \begin{cases} \frac{n}{2} & \text{wenn } n \text{ gerade ist,} \\ 3n+1 & \text{wenn } n \text{ ungerade ist.} \end{cases}$$

Es gibt die Vermutung, dass die Folge für alle $n > 0$ in einen Zyklus $4, 2, 1$ gelangt.

Angenommen, es gäbe einen Algorithmus $\mathcal{A}(\mathcal{P})$ mit

$$\mathcal{A}(\mathcal{P}) := \begin{cases} 0, & \text{falls } \mathcal{P} \text{ kein (gültiges) Programm ist,} \\ 1, & \text{falls } \mathcal{P} \text{ ein Programm ist, das immer hält,} \\ 2, & \text{falls } \mathcal{P} \text{ ein Programm ist, das bei einigen, aber nicht allen Eingaben hält,} \\ 3, & \text{falls } \mathcal{P} \text{ ein Programm ist, das nie hält.} \end{cases}$$

Wie würden Sie $\mathcal{A}(\mathcal{P})$ benutzen können, um die Collatz-Vermutung zu beweisen?

Aufgabe 9.4. Entscheidbarkeit

Beweisen oder widerlegen Sie:

Es ist entscheidbar, ob eine Turingmaschine auf keiner möglichen Eingabe hält.

Aufgabe 9.5. Post'sches Korrespondenzproblem

Betrachten Sie die folgenden PCP-Instanzen. Geben Sie jeweils eine zulässige Lösung an, oder begründen Sie, warum keine existiert.

(a)

db	dort	h	ier	in	rum	un	we
ier	ort	un	d	ier	mb	b	weinh

(b)

a	m	p	p	pa
m	a	aam	paa	pam

Aufgabe 9.6. Kolmogorov-Komplexität doch berechenbar?

Wir haben in der Vorlesung gesehen, dass die Kolmogorov-Komplexität nicht berechenbar ist. Wir betrachten nun die Programmiersprache PURL („pott-ugly run-length language"), mit folgenden Anweisungen. Sei $\mathbb{D}(c) \in \{0,\ldots,9\}^+$ die Dezimaldarstellung der Zahl $c \in \mathbb{N}$.

ε (Leerwort)	—	tut nichts,
AB	—	führt Anweisungen A und B nacheinander aus,
φ`:`ϱ	—	gibt den String $\varrho \in \Sigma^c$ aus, wobei $\varphi = \mathbb{D}(c)$,
φ`(`A`)`	—	Anweisung A wird c mal ausgeführt, wobei $\varphi = \mathbb{D}(c)$.

Zum besseren Verständnis ein paar Beispiele:

PURL-Programm	Ausgabewort
`1:m2(3:iss)4:ippi`	`mississippi`
`3:abc 3(5(1:a)2:bb1:p)`	`abc aaaaabbp aaaaabbp aaaaabbp`
`2:3: 19:abc3(5(1:a)2:bb1:p)`	`3:abc 3(5(1:a)2:bb1:p)`

Der folgende Algorithmus $\mathcal{K}(w)$ berechnet die Länge des kürzesten PURL-Programms, das als Ausgabe w erzeugt.

```
if |w| = 0 then return 0
ℓ := ⌊log10 |w|⌋ + 1
for i = 1, 2, ..., (|w| + ℓ):
    for alle Strings P mit |P| = i:
        if P ist ein gültiges PURL-Programm:
            interpretiere P als PURL-Programm und führe es aus
            if w = Ausgabe von P then return i
return |w| + ℓ + 1
```

(a) Warum ist der Rückgabewert von $\mathcal{K}(w)$ immer korrekt? $\mathcal{K}(w)$ ist nie größer als $|w| + \ell + 1$. Warum stimmt das?

(b) Terminiert $\mathcal{K}(w)$ immer? Begründen Sie Ihre Antwort.

(c) Begründen Sie, warum $\mathcal{K}(w)$ dennoch keinen Widerspruch zur Nicht-Berechenbarkeit der Kolmogorov-Komplexität darstellt.