Theoretische Informatik

Prof. Dr. Peter Thiemann
Marius Weidner
Simon Dorer

Universität Freiburg
Institut für Informatik
Sommersemester 2026

Übungsblatt 10 – Lösungen

Abgabe: Dienstag, 07.07.2026, 16:00 Uhr

Hinweis Lehrevaluation:
Aktuell läuft die offizielle Lehrevaluation der Technischen Fakultät, in der Sie die Vorlesungen, die Sie im aktuellen Semester belegt haben bewerten können. Sie haben dazu bereits eine E-Mail mit einem Link erhalten.

Bitte nehmen Sie sich kurz Zeit für die Evaluation: Ihr Feedback hilft uns sehr dabei, die Vorlesung und die Übungen weiter zu verbessern und stärker an Ihre Wünsche und Bedürfnisse anzupassen. **Vielen Dank für Ihre Teilnahme!**

Aufgabe 10.1 (LOOP Makros; 6 Punkte)

In der Praxis verwendet man häufig Makros, um LOOP-Programme übersichtlicher zu notieren. Ein solches Makro ist kein neuer Grundbefehl, sondern steht immer für ein gewöhnliches LOOP-Programm mit festgelegtem Verhalten. In dieser Aufgabe sollen Sie zunächst eine einfache bedingte Ausführung als Makro realisieren und dieses Makro anschließend in einem LOOP-Programm verwenden.

(a) Sei $P \in \mathrm{LOOP}^k$ ein LOOP-Programm und sei x_i mit $1 \leq i \leq k$ ein Register. Konstruieren Sie ein LOOP-Programm, das die folgende Makro-Schreibweise realisiert:

$$\texttt{IF } x_i \neq 0 \texttt{ THEN } P \texttt{ END;}$$

Dabei soll P genau dann einmal ausgeführt werden, wenn der Wert von x_i zu Beginn ungleich 0 ist. Ist der Wert von x_i zu Beginn 0, so soll P nicht ausgeführt werden.

Abgesehen von den Änderungen durch eine mögliche Ausführung von P muss die Implementierung des Makros alle Registerwerte unverändert lassen. Sie dürfen dafür frische Hilfsregister verwenden, die in P nicht vorkommen und zu Beginn den Wert 0 haben.

(b) Verwenden Sie das Makro aus Teil (a), um ein LOOP-Programm zu konstruieren, das die ganzzahlige Division

$$\mathrm{div}\colon \mathbb{N} \times \mathbb{N}_{>0} \to \mathbb{N}, \qquad \mathrm{div}(n, m) = \left\lfloor \frac{n}{m} \right\rfloor$$

berechnet. Für Eingaben $m = 0$ darf sich ihr Programm beliebig verhalten.

Lösung:

(a) Wähle ein frisches Arbeitsregister x_ℓ mit $\ell > k$ und $\ell \neq i$. Wir nehmen an, dass x_ℓ zu Beginn 0 ist.

Wir verwenden folgende Idee: x_ℓ dient als Indikator dafür, ob x_i zu Beginn ungleich 0 war. Die erste Schleife setzt x_ℓ genau dann auf 1. Danach wird P genau x_ℓ mal ausgeführt und anschließend x_ℓ wieder auf 0 zurückgesetzt.

```
LOOP x_i DO
    LOOP x_ℓ DO x_ℓ := x_ℓ ∸ 1 END;
    x_ℓ := x_ℓ + 1
END;
LOOP x_ℓ DO P END;
x_ℓ := x_ℓ ∸ 1
```

(b) Wir verwenden folgende Idee: Wir ziehen wiederholt x_2 von x_1 ab und zählen in x_3, wie oft danach noch ein positiver Rest bleibt. Vorher erhöhen wir x_1 um 1, damit auch im Fall einer exakten Teilbarkeit der letzte volle Block noch gezählt wird. Am Ende enthält x_3 den Quotienten; die letzte Schleife überträgt diesen Wert zurück nach x_1.

```
x_1 := x_1 + 1;
LOOP x_1 DO
    LOOP x_2 DO x_1 := x_1 ∸ 1 END;
    IF x_1 ≠ 0 THEN x_3 := x_3 + 1 END
END;
LOOP x_3 DO x_1 := x_1 + 1 END
```

Aufgabe 10.2 (μ-Rekursion; 5 Punkte)
Sei $\mathrm{col} : \mathbb{N} \to \mathbb{N}$ die einstellige Collatz-Funktion mit

$$\mathrm{col}(n) = \begin{cases} \frac{n}{2}, & \text{falls } n \text{ gerade ist,} \\ 3n+1, & \text{falls } n \text{ ungerade ist.} \end{cases}$$

Wir betrachten die potentiell partielle Funktion[1]

$$\mathrm{colsteps} : \mathbb{N} \hookrightarrow \mathbb{N},$$

die einer Eingabe n die kleinste Anzahl von Collatz-Schritten zuordnet, nach denen zum ersten Mal der Wert 1 erreicht wird. Falls ausgehend von n nie der Wert 1 erreicht wird, ist $\mathrm{colsteps}(n)$ undefiniert. Insbesondere gilt $\mathrm{colsteps}(1) = 0$.

[1] In der Praxis wird vermutet, dass colsteps total ist, d.h. für alle $n \in \mathbb{N}$ terminiert. Es existiert allerdings noch kein Beweis, der diese Vermutung bestätigt oder widerlegt. Wenn wir es also schaffen würden colsteps primitiv rekursiv zu definieren, dann hätten wir die Collatz-Vermutung gezeigt.

In dieser Aufgabe sollen Sie diese unbeschränkte Suche mithilfe von μ-Rekursion modellieren. Dabei dürfen Sie ohne formalen Beweis verwenden, dass col und die Funktion

$$\text{isOne}(n) = \begin{cases} 0, & \text{falls } n = 1, \\ 1, & \text{sonst} \end{cases}$$

primitiv rekursiv sind.

(a) Geben Sie einen Term $\mathtt{C} \in \text{PREC}^2$ an, dessen Semantik $C := [\![\mathtt{C}]\!]_2$ die k-fache Iteration der Collatz-Funktion darstellt:

$$C(0, n) = n \qquad \text{und} \qquad C(k+1, n) = \text{col}(C(k, n)).$$

(b) Verwenden Sie anschließend μ-Minimierung, um daraus einen Term $\mathtt{colsteps} \in \mu\text{REC}^1$ anzugeben, dessen Semantik die Funktion colsteps ist.

Lösung:

(a) Sei $\mathtt{col} \in \text{PREC}^1$ ein Term mit

$$[\![\mathtt{col}]\!]_1 = \text{col}\,.$$

Da col nach Voraussetzung primitiv rekursiv ist, existiert ein solcher Term.

Wir definieren einen Term $\mathtt{C} \in \text{PREC}^2$, der die k-fache Iteration von col beschreibt. Dazu setzen wir

$$\mathtt{C} := \text{PR}\big(\pi^1_1,\ \mathtt{col} \circ [\pi^3_1]\big) \in \text{PREC}^2\,.$$

Für $C := [\![\mathtt{C}]\!]_2$ gilt dann nach der Semantik der primitiven Rekursion:

$$C(0, n) = n$$

und

$$C(k+1, n) = \text{col}(C(k, n)).$$

Also ist $C(k, n)$ genau der Wert, den man nach k Collatz-Schritten ausgehend von n erhält.

(b) Sei $\mathtt{isOne} \in \text{PREC}^1$ ein Term mit

$$[\![\mathtt{isOne}]\!]_1 = \text{isOne}\,.$$

Nach Voraussetzung existiert ein solcher Term. Wir definieren

$$\mathtt{h} := \mathtt{isOne} \circ [\mathtt{C}] \in \text{PREC}^2,$$

also semantisch

$$[\![\mathtt{h}]\!]_2(k, n) = \text{isOne}(C(k, n)).$$

Damit gilt

$$[\![\mathtt{h}]\!]_2(k,n) = 0 \quad \Longleftrightarrow \quad C(k,n) = 1.$$

Nun setzen wir mithilfe von μ-Minimierung

$$\mathtt{colsteps} := \mu\mathtt{h} \in \mu\mathrm{REC}^1.$$

Nach der Semantik der Minimierung gilt

$$[\![\mathtt{colsteps}]\!]_1 = \mu([\![\mathtt{h}]\!]_2).$$

Also gilt für $n \in \mathbb{N}$:

$$[\![\mathtt{colsteps}]\!]_1(n) = \min C_{[\![\mathtt{h}]\!]_2}(n),$$

falls diese Menge nicht leer ist, und ist sonst undefiniert. Dabei ist

$$C_{[\![\mathtt{h}]\!]_2}(n) = \{k \in \mathbb{N} \mid [\![\mathtt{h}]\!]_2(k,n) = 0 \text{ und } \forall j < k : [\![\mathtt{h}]\!]_2(j,n) \downarrow \neq 0\}.$$

Da $\mathtt{h} \in \mathrm{PREC}^2$ ist, ist $[\![\mathtt{h}]\!]_2$ total. Also ist die Zusatzbedingung $\downarrow$ hier automatisch erfüllt.

Damit ist $[\![\mathtt{colsteps}]\!]_1(n)$ genau das kleinste k, sodass

$$C(k,n) = 1$$

gilt. Existiert kein solches k, so ist $C_{[\![\mathtt{h}]\!]_2}(n) = \emptyset$, und $[\![\mathtt{colsteps}]\!]_1(n)$ ist undefiniert.

Also gilt

$$[\![\mathtt{colsteps}]\!]_1 = \mathrm{colsteps}\,.$$

Aufgabe 10.3 (Turing-Schritte sind primitiv rekursiv; 9 Punkte)
In der Vorlesung haben Sie gesehen, dass das Prädikat

$$P(f,g) = \forall \bar{a} \in \mathbb{N}^k : f(\bar{a}) = g(\bar{a})$$

für primitiv rekursive Funktionen f und g unentscheidbar ist.

Im Beweis haben wir u.a. skizziert, wie wir mithilfe von primitiv rekursiven Funktionen eine deterministische Turingmaschine simulieren können. In dieser Aufgabe sollen Sie beschreiben, wie wir Konfigurationen durch natürliche Zahlen codieren und darauf aufbauend die Funktionen start_M und step_M primitiv rekursiv definieren können.

Sei M also eine feste, aber beliebige deterministische Turingmaschine, die eine partielle Funktion $f : \mathbb{N}^k \hookrightarrow \mathbb{N}$ berechnet. Wir bezeichnen Konfigurationen mit (v,q,w), wobei w die Bandsymbole ab der Kopfposition nach rechts codiert und v die Bandsymbole links vom Kopf in *umgekehrter* Reihenfolge codiert, also beginnend mit dem Symbol direkt links vom Kopf. Außerdem ist q eine Codierung des aktuellen

Zustands. Die Übergangsfunktion von M ist durch eine endliche Tabelle gegeben. Jeder Eintrag hat die Form

$$\delta(q_i, a_j) = (q_\ell, b, d)$$

mit Zuständen q_i, q_ℓ, Bandsymbolen a_j, b und einer Richtung $d \in \{L, R, N\}$. Ist für ein Paar (q, a) kein Eintrag definiert, so hält M in der entsprechenden Konfiguration.

Sie dürfen ohne Beweis verwenden, dass Zustände, Bandsymbole und Wörter über dem Bandalphabet durch natürliche Zahlen codiert sind. Außerdem dürfen Sie die folgenden Hilfsfunktionen als gegeben und primitiv rekursiv annehmen:

$$\text{first}\colon \mathbb{N} \to \mathbb{N}, \quad \text{tail}\colon \mathbb{N} \to \mathbb{N}, \quad \text{cons}\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}, \quad \text{input}_M\colon \mathbb{N}^k \to \mathbb{N}.$$

Dabei liefert $\text{first}(u)$ das erste Symbol von u, $\text{tail}(u)$ entfernt das erste Symbol von u, und $\text{cons}(a, u)$ stellt das Bandsymbol a vorne an das codierte Wort u. Die Funktion

$$\text{input}_M\colon \mathbb{N}^k \to \mathbb{N}$$

übersetzt eine Eingabe $\bar{n} = (n_1, \ldots, n_k)$ in das codierte Startwort auf dem Band von M.

Geben Sie für die folgenden Schritte jeweils eine Definition auf semantischer Ebene an und begründen Sie kurz, warum die dadurch beschriebene Funktion primitiv rekursiv ist. Sie müssen die entsprechenden Terme nicht vollständig ausarbeiten.

(a) Geben Sie eine Codierung von Konfigurationen (v, q, w) durch natürliche Zahlen an, also eine Funktion

$$\text{encode} : \mathbb{N}^3 \to \mathbb{N}$$

sowie geeignete Decodierungsfunktionen, mit denen aus dem Code einer Konfiguration wieder die Komponenten v, q und w erhalten werden. Begründen Sie, warum diese Funktionen primitiv rekursiv sind.

Hinweis: Erinnern Sie sich an die Cantorsche Paarfunktion.

(b) Definieren Sie auf dieser Codierung eine Funktion

$$\text{start}_M : \mathbb{N}^k \to \mathbb{N},$$

die zu einer Eingabe $\bar{n} = (n_1, \ldots, n_k)$ die codierte Startkonfiguration von M liefert. Begründen Sie, warum start_M primitiv rekursiv ist.

(c) Definieren Sie auf dieser Codierung eine Funktion

$$\text{step}_M : \mathbb{N} \to \mathbb{N},$$

die zu einer codierten Konfiguration c die codierte Nachfolgekonfiguration nach einem Schritt von M liefert. Für Haltekonfigurationen soll $\text{step}_M(c) = c$ gelten. Begründen Sie, warum step_M primitiv rekursiv ist.

Hinweis: Erinnern Sie sich an das letzte Übungsblatt, in dem Sie gezeigt haben, dass ite *und $\chi_=$ primitiv rekursiv sind!*

Lösung:

(a) Wir verwenden die Cantorsche Paarfunktion aus der Vorlesung

$$\mathrm{cp}\colon \mathbb{N}^2 \to \mathbb{N}$$

sowie ihre beiden Umkehrfunktionen

$$\mathrm{px}, \mathrm{py}\colon \mathbb{N} \to \mathbb{N}.$$

Eine Konfiguration (v, q, w) codieren wir durch

$$\mathrm{encode}(v, q, w) := \mathrm{cp}(v, \mathrm{cp}(q, w)).$$

Die drei Decodierungsfunktionen seien

$$\mathrm{left}(c) := \mathrm{px}(c), \qquad \mathrm{state}(c) := \mathrm{px}(\mathrm{py}(c)), \qquad \mathrm{right}(c) := \mathrm{py}(\mathrm{py}(c)).$$

Damit gilt für $c = \mathrm{encode}(v, q, w)$:

$$\mathrm{left}(c) = v, \qquad \mathrm{state}(c) = q, \qquad \mathrm{right}(c) = w.$$

Die Funktionen encode, left, state und right sind primitiv rekursiv, da sie nur durch Komposition der primitiv rekursiven Funktionen cp, px und py entstehen.

(b) Wir definieren

$$\mathrm{start}_M(n_1, \ldots, n_k) := \mathrm{encode}\big(\ulcorner \varepsilon \urcorner, \ulcorner q_0 \urcorner, \mathrm{input}_M(n_1, \ldots, n_k)\big).$$

Diese Funktion liefert also den Code der Startkonfiguration

$$(\varepsilon, q_0, \text{Eingabewort}).$$

Sie ist primitiv rekursiv, weil sie eine Komposition aus Konstanten, der nach Aufgabenstellung primitiv rekursiven Funktion input_M und encode ist.

(c) Sei c der Code einer Konfiguration. Wir decodieren zunächst

$$v := \mathrm{left}(c), \qquad q := \mathrm{state}(c), \qquad w := \mathrm{right}(c).$$

Das aktuell gelesene Bandsymbol ist

$$a := \mathrm{first}(w).$$

Da M fest ist, besitzt M nur endlich viele Übergänge. Für jeden möglichen Übergang

$$\delta(q_i, a_j) = (q_\ell, b, d)$$

können wir mit den primitiv rekursiven Funktionen $\chi_=$ und ite testen, ob $q = q_i$ und $a = a_j$ gilt, und dann die entsprechende Nachfolgekonfiguration auswählen.

Wir beschreiben die drei möglichen Fälle. Zunächst sei

$$r := \mathrm{tail}(w)$$

der rechte Rest des Bandes nach dem gelesenen Symbol.

1. Falls der Übergang nach rechts geht, also $\delta(q, a) = (q', b, R)$, dann wird a durch b ersetzt und der Kopf bewegt sich nach rechts. Die neue Konfiguration ist
$$\big(\mathrm{cons}(b, v),\ q',\ r\big).$$

2. Falls der Übergang nach links geht, also $\delta(q, a) = (q', b, L)$, dann wird $\mathrm{first}(v)$ zum neuen gelesenen Symbol und das alte, mit b überschriebene Kopfsymbol wird rechts davon gespeichert. Die neue Konfiguration ist
$$\big(\mathrm{tail}(v),\ q',\ \mathrm{cons}\big(\mathrm{first}(v), \mathrm{cons}(b, r)\big)\big).$$

3. Falls der Übergang stehen bleibt, also $\delta(q, a) = (q',\ b,\ N)$, dann ist die neue Konfiguration
$$\big(v,\ q',\ \mathrm{cons}(b, r)\big).$$

Dabei verwenden wir die Konvention

$$\mathrm{first}(\ulcorner\varepsilon\urcorner) = \ulcorner\mathrm{blank}\urcorner \qquad \text{und} \qquad \mathrm{tail}(\ulcorner\varepsilon\urcorner) = \ulcorner\varepsilon\urcorner.$$

Dadurch sind auch Bewegungen über den bisher explizit gespeicherten Bandabschnitt hinaus abgedeckt. Falls die gegebenen Wortfunktionen diese Konvention nicht erfüllen, kann man dasselbe Verhalten durch eine zusätzliche Fallunterscheidung mit $\chi_=$ und ite erzwingen.

Existiert kein Übergang für (q, a), dann ist c bereits eine Haltekonfiguration. In diesem Fall setzen wir
$$\mathrm{step}_M(c) := c.$$

Formal erhält man step_M, indem man über die endlich vielen Übergänge von M eine verschachtelte ite-Fallunterscheidung baut. Jede Bedingung verwendet nur Gleichheitstests auf natürlichen Zahlen, und jeder Fall verwendet nur die oben genannten primitiv rekursiven Hilfsfunktionen sowie encode. Daher ist step_M primitiv rekursiv.

Aufgabe 10.4 (Erfahrungen; Datei: `NOTES.md`)

Notieren Sie Ihre Erfahrungen mit diesem Übungsblatt (benötigter Zeitaufwand, Probleme, Bezug zur Vorlesung, Interessantes, etc.).

Editieren Sie hierzu die Datei `NOTES.md` im Abgabeordner dieses Übungsblattes auf unserer Webplattform. Halten Sie sich an das dort vorgegebene Format, da wir den Zeitbedarf automatisch statistisch auswerten. Die Zeitangabe `4.5 h` steht dabei für 4 Stunden und 30 Minuten.