

Theoretische Informatik

Prof. Dr. Peter Thiemann
Marius Weidner
Simon Dorer

Universität Freiburg
Institut für Informatik
Sommersemester 2026

Übungsblatt 10

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 $P \in \text{LOOP}^k$ 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:

IF $x_i \neq 0$ THEN P 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

$$\text{div}: \mathbb{N} \times \mathbb{N}_{>0} \rightarrow \mathbb{N}, \quad \text{div}(n, m) = \left\lfloor \frac{n}{m} \right\rfloor$$

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

Aufgabe 10.2 (μ -Rekursion; 5 Punkte)

Sei $\text{col} : \mathbb{N} \rightarrow \mathbb{N}$ die einstellige Collatz-Funktion mit

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

Wir betrachten die potentiell partielle Funktion¹

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

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 $\text{colsteps}(n)$ undefiniert. Insbesondere gilt $\text{colsteps}(1) = 0$.

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 $\mathbf{C} \in \text{PREC}^2$ an, dessen Semantik $C := \llbracket \mathbf{C} \rrbracket_2$ die k -fache Iteration der Collatz-Funktion darstellt:

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

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

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) ,

¹In 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.

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:

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

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

$$\text{input}_M: \mathbb{N}^k \rightarrow \mathbb{N}$$

übersetzt eine Eingabe $\bar{n} = (n_1, \dots, n_k)$ 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

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

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

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

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

- (c) Definieren Sie auf dieser Codierung eine Funktion

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

die zu einer codierten Konfiguration c die codierte Nachfolgekongfiguration 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!

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.