Universität Siegen
Lehrstuhl Theoretische Informatik
Markus Lohrey



# Übungsblatt 14

**Aufgabe 1.** Bestimmen Sie $\mu f$ für die folgenden Funktionen.

(a) $f(n, x) = n + x$

(b) $f(n, x) = n - x$

(c) $f(n, x) = x - n$

(d) $f(n, x, y) = x - n \cdot y$

**Lösung zu Aufgabe 1.** Offensichtlich sind alle Funktionen in dieser Aufgabe total, d.h. es genügt, die erste Nullstelle zu finden.

(a) $(\mu f)(x) = \begin{cases} 0 & \text{falls } x = 0, \\ \text{undefiniert} & \text{sonst.} \end{cases}$

Für $x = 0$ erhalten wir mit Hilfe von $n = 0$ eine Nullstelle. Falls $x > 0$ ist, so hat $f(n, x)$ keine Nullstellen, da wir nur natürliche Zahlen betrachten.

(b) $(\mu f)(x) = 0$, denn $n = 0$ ist die kleinste Nullstelle von $f(n, x) = n - x$ für alle $x \in \mathbb{N}$: $f(0, x) = 0 - x = 0$.

(c) $(\mu f)(x) = x$, denn für alle $n, x \in \mathbb{N}$ gilt, dass $f(n, x) = 0$ genau dann, wenn $n \geq x$. D.h. für alle $x \in \mathbb{N}$ gilt, dass

$$\min\{n \in \mathbb{N} \mid f(n, x) = 0\} = \min\{n \in \mathbb{N} \mid n \geq x\} = x.$$

(d) Sei zunächst $y \neq 0$. Dann ist $(\mu f)(x, y) = \lceil x/y \rceil$, denn

$$\begin{aligned} \min\{n \in \mathbb{N} \mid f(n, x, y) = 0\} &= \min\{n \in \mathbb{N} \mid x - n \cdot y = 0\} \\ &= \min\{n \in \mathbb{N} \mid n \cdot y \geq x\} \\ &= \min\{n \in \mathbb{N} \mid n \geq x/y\} \\ &= \lceil x/y \rceil. \end{aligned}$$

Mit Betrachtung des Falls $y = 0$ erhalten wir insgesamt

$$(\mu f)(x, y) = \begin{cases} \lceil x/y \rceil & \text{falls } y \neq 0, \\ 0 & \text{falls } x = y = 0, \\ \text{undefiniert} & \text{sonst.} \end{cases}$$

**Aufgabe 2.** Beweisen Sie, dass folgende Funktionen $\mu$-rekursiv sind:

(a) $f(x, y) = \lceil \log_y(x) \rceil, y \geq 2$ (hierbei sei $\log_y(0) = 0$)

(b) $g(x, y) = \begin{cases} y & \text{wenn } x = 0, \\ \text{undefiniert} & \text{sonst.} \end{cases}$

**Lösung zu Aufgabe 2.**

(a) Idee: Probiere alle Potenzen von $y$ durch und schaue, welche als erste größer oder gleich $x$ wird. Es gilt

$$\begin{aligned} \lceil \log_y(x) \rceil &= \min\{n \in \mathbb{N} \mid n \geq \log_y(x)\} \\ &= \min\{n \in \mathbb{N} \mid y^n \geq x\} \\ &= \min\{n \in \mathbb{N} \mid x - y^n \leq 0\}. \end{aligned}$$

Die Umformung $n \geq \log_y(x) \Rightarrow y^n \geq x$ stimmt nur für $y \geq 2$, aber dies ist laut Aufgabenstellung gegeben. Sei nun $h \colon \mathbb{N}^3 \to \mathbb{N}$ definiert als $h(n, x, y) = x - y^n$. Dann gilt $f = \mu h$.

(b) Sei $k(n, x, y) = (y - n) + x$. Behauptung: Es gilt $g = \mu k$. Es gilt

$$g(0, y) = \mu k(0, y) = \min\{n \in \mathbb{N} \mid (y - n) + 0 = 0\} = y.$$

Außerdem gilt für alle $x > 0$, dass

$$g(x, y) = \mu k(x, y) = \min\{n \in \mathbb{N} \mid (y - n) + x = 0\} = \min \varnothing = \text{undef.},$$

weil die Subtraktion (in diesem Fall $y - n$) in unserem Kontext immer bei 0 abgeschnitten ist. Von daher ist $(y - n) + x > 0$ für $x > 0$.

**Aufgabe 3.** Die Softwarefirma Halting & Co. KG bietet folgende Produkte zur Programmverifikation an.

(a) Produkt A überprüft, ob ein gegebenes Programm auf einer gegebenen Eingabe höchstens 1,000 Rechenschritte durchführt.

(b) Produkt B überprüft, ob ein gegebenes Programm auf einer gegebenen Eingabe höchstens 1 GB Speicher benötigt.

(c) Produkt C überprüft, ob ein gegebenes Programm niemals die Ausgabe 123 produziert.

Welche Produkte kann es tatsächlich geben?

**Lösung zu Aufgabe 3.**

(a) Wahr. Das Programm wird unter der gegebenen Eingabe simuliert und die Rechenschritte mitgezählt. Falls das simulierte Programm bis zum 1000. Rechenschritt terminiert, meldet der Simulator ≪terminiert≫. Andernfalls wird die Simulation abgebrochen und der Simulator meldet ≪terminiert nicht bei bis zu 1000 Rechenschritten≫ .

(b) Wahr. Das Programm wird unter der gegebenen Eingabe simuliert und alle bereits erreichten Konfigurationen gespeichert. Wichtig ist, dass es mit 1 GB Speicher nur endlich viele Konfigurationen gibt, die in diesem begrenzten Speicher arbeiten. Sollte eine Konfiguration auftauchen, die mehr als 1 GB Speicher benötigt, bricht der Simulator mit ≪braucht mehr als 1GB Speicher≫ ab. Sollte das simulierte Programm terminieren ohne jemals eine zu große Konfiguration zu erreichen, meldet der Simulator ≪das Programm terminiert und benötigt höchstens 1GB Speicher≫. Der letzte mögliche Fall ist der, dass eine bereits erreichte Konfiguration mit weniger als 1GB erneut erreicht wird (da es nur endlich viele Konfigurationen dieser Art gibt) ohne dass zuvor terminiert wurde oder eine zu große Konfiguration besucht wurde. In diesem Fall würde sich das Programm also in eine Endlosschleife begeben und der Simulator bricht mit ≪das Programm terminiert nicht≫ ab.

(c) Falsch. Zuerst zeigen wir, dass es unentscheidbar ist, ob eine Turing-Maschine auf keiner Eingabe terminiert. Dazu zeigen wir eine Reduktion auf das Komplement des speziellen Halteproblem (das Komplement einer unentscheidbaren Sprache ist wiederum unentscheidbar; folgt aus Satz 26, Folie 479), d.h., wenn man prüfen könnte, ob eine Turing-Maschine auf keiner Eingabe terminiert, so könnte man auch prüfen, ob eine Turing-Maschine nicht auf seiner eigenen Kodierung terminiert. Dazu modifizieren wir die gegebene Turing-Maschine $M_w$ wie folgt: Eine beliebige Eingabe wird zu Beginn durch die Kodierung $w$ der gegebenen Turing-Maschine überschrieben und anschließend arbeitet die modifizierte Turing-Maschine $M'_w$ genau wie die ursprüngliche Turing-Maschine $M_w$. Nun gilt, dass $M'_w$ auf keiner Eingabe terminiert, genau dann, wenn die originale Turing-Maschine $M_w$ auf der eigenen Kodierung nicht terminiert. Somit könnte man das Komplement des speziellen Halteproblems entscheiden, falls man entscheiden könnte ob eine Turing-Maschine auf keiner Eingabe terminiert, was zur Folge hat, dass dieses Problem ebenfalls unentscheidbar ist.
Im zweiten Schritt geben wir eine Reduktion von (c) auf das eben betrachtete unentscheidbare Problem an, welches prüft ob ein Programm

auf keiner Eingabe terminiert. Die Reduktion funktioniert wie folgt: Wir zeigen, dass wenn man prüfen könnte, ob ein gegebenes Programm niemals die Ausgabe 123 produziert, so könnte man auch prüfen ob ein Programm auf keiner Eingabe terminiert. Dazu modifizieren wir das gegebene Programm so, dass wann immer das Programm terminiert (also im Falle einer Turing-Maschine in einen Endzustand übergeht), so wird vorher noch die Ausgabe 123 produziert. Dieses modifizierte Programm produziert nun niemals die Ausgabe 123 genau dann, wenn das ursprüngliche Programm auf keiner Eingabe terminiert (da im Umkehrschluss die Ausgabe 123 immer produziert würde, wenn das Programm terminieren würde). Somit könnte man mit Hilfe von (c) ein unentscheidbares Problem entscheiden, was bedeutet, dass auch (c) unentscheidbar sein muss.

**Aufgabe 4.** Ein bekanntes Problem aus der Mathematik ist Hilberts zehntes Problem: Gegeben ein Polynom $p(x_1, \ldots, x_n)$ mit ganzzahligen Koeffizienten in $n$ Variablen ($n \geq 1$ beliebig), existieren $x_1, \ldots, x_n \in \mathbb{Z}$ mit $p(x_1, \ldots, x_n) = 0$? Erst 1970 wurde bewiesen, dass dieses Problem unentscheidbar ist.

(a) Ist Hilberts zehntes Problem semi-entscheidbar?

(b) Ist das Komplement semi-entscheidbar?

**Lösung zu Aufgabe 4.**

(a) Wahr. Die Menge der möglichen Belegungen für ein Polynom über $\mathbb{Z}$ mit $n$ Variablen ist $\mathbb{Z}^n$, welches eine abzählbar unendliche Menge darstellt.[1] Das bedeutet, dass man alle möglichen Belegungen sukzessive durchprobieren kann. Das Verfahren terminiert, wenn es eine Belegung gibt, die das Polynom zu Null auswertet. Ansonsten läuft das Programm für immer weiter. Dies beschreibt also einen Semi-Entscheidungs-Algorithmus.

(b) Falsch. Aus a) und der Unentscheidbarkeit des Problems folgt, dass b) nicht semi-entscheidbar ist (Satz 26, Folie 479).

[1] Der Beweis ist analog zur Konstruktion der Funktion $\langle n_1, n_2, \ldots, n_k \rangle$ auf Folie 426, welches eine Bijektion von $\mathbb{N}^k$ nach $\mathbb{N}$ ist.