

Theoretische Informatik

3. Übungsblatt

Paul Winkler
11818749

Aufgabe 1. Wir zeigen, dass folgende Funktionen primitiv rekursiv sind:

- (a) $(x, y) \mapsto x \cdot y$:

Dass die Addition primitiv rekursiv ist, wissen wir bereits aus der Vorlesung. Es gilt $x \cdot 0 = 0$ und $x \cdot (y + 1) = x \cdot y + x$. Wir definieren also

$$f(x) = 0 = c_0^1$$

$$g(x, y, z) = z + x = \text{Cn}[+, P_3^3, P_1^3]$$

und es gilt $\cdot = \text{Pr}[f, g] = \text{Pr}[c_0^1, \text{Cn}[+, P_3^3, P_1^3]]$.

- (b) $(x, y) \mapsto p(x) = \begin{cases} 0 & \text{falls } x = 0 \\ x - 1 & \text{falls } x > 0 \end{cases}$:

Es gilt $p(0) = 0$ und $p(y + 1) = y$ und somit $p = \text{Pr}[f, g]$ mit

$$f(0) = 0 = c_0^1$$

$$g(y, z) = y = P_1^2.$$

- (c) $(x, y) \mapsto x \dot{-} y = \begin{cases} 0 & \text{falls } x \leq y \\ x - y & \text{falls } x > y \end{cases}$:

Durch Fallunterscheidung sieht man leicht, dass $x \dot{-} (y + 1) = p(x \dot{-} y)$. Nach (b) ist p primitiv rekursiv. Also ist $\dot{-} = \text{Pr}[f, g]$ mit

$$f(x) = x \dot{-} 0 = x = P_1^1$$

$$g(x, y, z) = p(z) = \text{Cn}[p, P_3^3].$$

- (d) $(x, y) \mapsto \chi_{\leq}(x, y) = \begin{cases} 1 & \text{falls } x \leq y \\ 0 & \text{falls } x > y \end{cases}$:

Falls $x \leq y$, dann ist $x \dot{-} y = 0$; sonst ist $x \dot{-} y = x - y \geq 1$. Folglich gilt $\chi_{\leq}(x, y) = 1 \dot{-} (x \dot{-} y)$, was nach (c) p. r. ist.

- (e) $(x, y) \mapsto \chi_{=}(x, y) = \begin{cases} 1 & \text{falls } x = y \\ 0 & \text{falls } x \neq y \end{cases}$:

Das folgt mit $\chi_{=}(x, y) = \chi_{\leq}(x, y) \cdot \chi_{\leq}(y, x)$ unmittelbar aus (d).

- (f) $h: \mathbb{N}^k \rightarrow \mathbb{N}, \bar{x} \mapsto \begin{cases} f_0(\bar{x}) & \text{falls } g(\bar{x}) = 0 \\ f_1(\bar{x}) & \text{falls } g(\bar{x}) = 1 \\ \vdots \\ f_{n-1}(\bar{x}) & \text{falls } g(\bar{x}) = n - 1 \\ f_n(\bar{x}) & \text{falls } g(\bar{x}) \geq n, \end{cases}$ wobei $g, f_0, \dots, f_n: \mathbb{N}^k \rightarrow \mathbb{N}$ p. r. sind:

Das folgt aus dem bisher Gezeigten wegen

$$h(\bar{x}) = f_n(\bar{x}) \cdot \chi_{\leq}(n, g(\bar{x})) + \sum_{i=0}^{n-1} f_i(\bar{x}) \cdot \chi_{=}(g(\bar{x}), i).$$

Aufgabe 2. Sei $f: \mathbb{N}^{k+1} \rightarrow \mathbb{N}$ p. r., dann sind auch folgende Funktionen p. r.:

(a) $h: (\bar{x}, z) \mapsto \sum_{y=0}^z f(\bar{x}, y):$

Wir wissen schon, dass Addition p. r. ist. Nun stellen wir fest, dass

$$\begin{aligned} h(\bar{x}, 0) &= f(\bar{x}, 0) \\ h(\bar{x}, y+1) &= h(\bar{x}, y) + f(\bar{x}, y+1), \end{aligned}$$

also $h = \text{Pr}[g, j]$ mit

$$\begin{aligned} g(\bar{x}) &= f(\bar{x}, 0) \\ j(\bar{x}, y, z) &= z + f(\bar{x}, y+1). \end{aligned}$$

(b) $h: (\bar{x}, z) \mapsto \prod_{y=0}^z f(\bar{x}, y):$ Genauso wie (a), nur mit \cdot statt $+$.

Sei $f: \mathbb{N}^{k+1} \rightarrow \{0, 1\}$ p. r., dann sind auch folgende Funktionen p. r.:

(c) $h: (\bar{x}, z) \mapsto \forall y \leq z (f(\bar{x}, y) = 1):$

$$h(\bar{x}, z) = \prod_{y=0}^z \chi_{=}(f(\bar{x}, y), 1).$$

$= f(\bar{x}, y)$

(d) $h: (\bar{x}, z) \mapsto \exists y \leq z (f(\bar{x}, y) = 1):$


$$\begin{aligned} h(\bar{x}, 0) &= \chi_{=}(f(\bar{x}, 0), 1) \\ h(\bar{x}, z+1) &= h(\bar{x}, z) + \chi_{=}(h(\bar{x}, z), 0) \cdot \chi_{=}(f(\bar{x}, z+1), 1). \end{aligned}$$

Aufgabe 3. Folgende Funktionen sind p. r.:

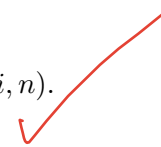
(a) $(x, y) \mapsto \begin{cases} 1 & \text{falls } x \mid y \\ 0 & \text{sonst} \end{cases} :$

Es gilt $x \mid y \equiv \exists z \leq y (x \cdot z = y)$, was als beschränkte Formel mit primitiv rekursiver Matrix nach Aufgabe 2 p. r. ist.

$$(b) \ (x, y) \mapsto \begin{cases} 1 & \text{falls } \text{ggT}(x, y) = 1 \\ 0 & \text{sonst} \end{cases} :$$

Das gilt wegen $\text{ggT}(x, y) = 1 \iff \neg \exists z \leq y \ (2 \leq z \wedge z \mid x \wedge z \mid y)$. Diese Formel ist p. r. wegen (a) sowie wegen $\neg \varphi = 1 \div \varphi$ und $\varphi \wedge \psi = \varphi \cdot \psi$. 

$$(c) \ \varphi: n \mapsto |\{i \in \mathbb{N}: 1 \leq i \leq n, \text{ggT}(i, n) = 1\}|:$$

Bezeichne h die Funktion aus (b), dann gilt $\varphi(n) = \sum_{i=1}^n h(i, n)$. 

Aufgabe 4. Die Turingmaschine $M = \langle Q, \delta, s \rangle$ führt eine Duplikation eines Wortes $w \in \{0, 1\}^*$ durch, wobei

$$Q = \{s, q_0, k_0, k'_0, k_1, k'_1, r_0, r'_0, r_1, r'_1\}$$

und die Übergangsfunktion δ gegeben ist durch

q	x	q'	x'	r
s	\triangleright	q_0	\triangleright	\rightarrow
q_0	0	k_0	\sim	\rightarrow
k_0	0	k_0	0	\rightarrow
k_0	1	k_0	1	\rightarrow
k_0	\sim	k'_0	\sim	\rightarrow
k'_0	0	k'_0	0	\rightarrow
k'_0	1	k'_0	1	\rightarrow
k'_0	\sim	r_0	0	\leftarrow
r_0	0	r_0	0	\leftarrow
r_0	1	r_0	1	\leftarrow
r_0	\sim	r'_0	\sim	\leftarrow
r'_0	0	r'_0	0	\leftarrow
r'_0	1	r'_0	1	\leftarrow
r'_0	\sim	q_0	0	\rightarrow
q_0	1	k_1	\sim	\rightarrow
k_1	0	k_1	0	\rightarrow
k_1	1	k_1	1	\rightarrow
k_1	\sim	k'_1	\sim	\rightarrow
k'_1	0	k'_1	0	\rightarrow
k'_1	1	k'_1	1	\rightarrow
k'_1	\sim	r_1	1	\leftarrow
r_1	0	r_1	0	\leftarrow
r_1	1	r_1	1	\leftarrow
r_1	\sim	r'_1	\sim	\leftarrow
r'_1	0	r'_1	0	\leftarrow
r'_1	1	r'_1	1	\leftarrow
r'_1	\sim	q_0	1	\rightarrow
q_0	\sim	fertig	\sim	—

Erklärung: M liest das erste Zeichen. Dieses ist entweder »0« (gelber Block) oder »1« (türkischer Block). Das Zeichen wird durch »_« überschrieben. Zum Beispiel im ersten Fall wechselt die Maschine in den Zustand k_0 (»kopiere Null«). Dann wandert der Cursor nach rechts, bis zum ersten Mal »_« auftritt (also die Grenze zwischen dem ursprünglichen Wort und dem bereits kopierten Teil), worauf sie in den Zustand k'_0 wechselt. Tritt dann erneut »_« auf, ist das Ende erreicht und »0« wird geschrieben. Der neue Zustand ist dann r_0 (»rückwärts Null«) bzw. nach einem Erreichen von »_« r'_0 . Tritt dann nochmals »_« auf, wird die ausgeschnittene »0« an dieser Stelle wieder in das ursprüngliche Wort eingefügt. Der Cursor wandert einen Schritt nach rechts und der Zustand q_0 wird wiederhergestellt. Ist im Zustand q_0 das nächste Zeichen »_«, ist das Wort vollständig kopiert und der Rechenvorgang terminiert.

Aufgabe 5. Wir sagen dass eine deterministische Turingmaschine $M = \langle Q, \delta, q_0 \rangle$ primitiv rekursive Laufzeit hat, falls eine primitiv rekursive Funktion $t: \mathbb{N}^k \rightarrow \mathbb{N}$ existiert, so dass für alle $x \in \mathbb{N}^k$ eine Konfiguration (**fertig**, u, v) existiert sowie ein $k \leq t(x)$, sodass $(q_0, \triangleright, x) \xrightarrow{M^k} (\text{fertig}, u, v)$. Wir zeigen, dass $f: \mathbb{N}^k \rightarrow \mathbb{N}$ primitiv rekursiv ist genau dann wenn eine Turingmaschine existiert, die f in primitiv rekursiver Laufzeit berechnet.

Für die eine Richtung sei f primitiv rekursiv. Aus der Vorlesung wissen wir, dass f Turing-berechenbar ist. Wir zeigen induktiv nach der Operatordarstellung von f , dass eine Funktion t wie oben existiert:

- Die konstante Nullfunktion kann durch eine Turingmaschine in zwei Schritten berechnet werden, wir wählen daher $t \equiv 2$.
- Die Nachfolgerfunktion kann durch die Turingmaschine aus der Vorlesung berechnet werden; sie durchläuft das Band zweimal in der Länge der Eingabe. Da die Anzahl der Ziffern einer Zahl in Binärdarstellung durch die Zahl selbst beschränkt ist, sollte $t(x) = 3x$ eine passende Schranke sein.
- Die Projektion P_i^n kann durch die Turingmaschine aus der Vorlesung berechnet werden. Diese Berechnung umfasst vier Schritte, für die wir jeweils eine primitiv rekursive Laufzeitfunktion angeben können:

1. »Überschreibe $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_k$ mit »_«:

Wir wählen $t_1(\bar{x}) = n + 2 + \sum_{i=1}^n x_i$, weil die Anzahl der Ziffern einer Zahl in Binärdarstellung kleiner gleich der Zahl selbst ist, wir n Leerzeichen haben und die Maschine nach zwei aufeinanderfolgenden Leerzeichen mit diesem Schritt fertig ist.

2. »Bewege den Cursor zurück an den Anfang von x_i «,
3. »Verschiebe x_i Zeichen für Zeichen an den Anfang des Bands«,
4. »Bewege den Cursor zurück an den Anfang«:

Diese Schritte haben jeweils höchstens den selben Aufwand wie der erste, daher wählen wir $t_2(\bar{x}) = t_3(\bar{x}) = t_4(\bar{x})$ und insgesamt $t(\bar{x}) = 4t_1(\bar{x})$.

- Für den Induktionsschritt seien $f: \mathbb{N}^n \hookrightarrow \mathbb{N}$, $g_1, \dots, g_n: \mathbb{N}^k \hookrightarrow \mathbb{N}$ und $h = \text{Cn}[f, g_1, \dots, g_n]$. Nach IV gibt es für alle $i = 1, \dots, n$ und für f eine primitiv rekursive Laufzeitfunktion t_{g_i} bzw. t_f . Wir verwenden wieder die Turingmaschine aus der Vorlesung.

1. »Kopiere die Eingabe \bar{x} auf jedes der n Bänder«:

Wir können die Anzahl der Schritte mit $t_1(\bar{x}) = 2n \sum_{i=1}^k x_i$ abschätzen. Dabei verwenden wir wieder das Argument mit der Zifferndarstellung, müssen aber berücksichtigen, dass wir den Cursor wieder in die Startposition bringen müssen.

2. »Für $i = 1, \dots, n$ berechne $g_i(\bar{x})$ durch die Turingmaschine M_{g_i} auf dem i -ten Band«:

Damit können wir den Aufwand abschätzen durch $t_2(\bar{x}) = \sum_{i=1}^n t_{g_i}(\bar{x})$.

3. »Kopiere $g_2(\bar{x}), \dots, g_n(\bar{x})$ auf das erste Band zu $g_1(\bar{x})$ «:

Für jedes $i = 1, \dots, n$ hat $g_i(\bar{x})$ höchstens $g_i(\bar{x})$ Ziffern in Binärdarstellung. Wir müssen stets ein Leerzeichen einfügen und den Cursor zurückbewegen und verwenden sicherheitshalber die nicht scharfe Abschätzung $t_3(\bar{x}) = 5 \sum_{i=1}^n g_i(\bar{x})$.

4. »Berechne $f(g_1(\bar{x}), \dots, g_n(\bar{x}))$ durch M_f auf dem ersten Band«:

Offenbar funktioniert $t_4(\bar{x}) = t_f(g_1(\bar{x}), \dots, g_n(\bar{x}))$ und insgesamt $t(\bar{x}) = t_1(\bar{x}) + t_2(\bar{x}) + t_3(\bar{x}) + t_4(\bar{x})$.

- Sei $f = \text{Pr}[h, g]$. Nach IV gibt es p.r. Laufzeitfunktionen t_h und t_g für h bzw. g . Wir definieren

$$\begin{aligned}\tilde{t}_f(\bar{x}, 0) &= t_h(\bar{x}), \\ \tilde{t}_f(\bar{x}, y+1) &= t_g(\bar{x}, y, f(\bar{x}, y)).\end{aligned}$$

Die in der Vorlesung angegebene Turingmaschine hat noch ein Band mit einem Zähler, der in jedem Berechnungsschritt inkrementiert wird; wir wissen bereits, dass $t_s(y) = 3y$ den Aufwand der Nachfolgerfunktion abschätzt. Insgesamt können wir den Aufwand also mit

$$\tilde{t}_f(\bar{x}, y) + \sum_{i=0}^{y-1} 3i \leq \tilde{t}_f(\bar{x}, y) + 2y^2 =: t_f(\bar{x}, y)$$

abschätzen.

Für die andere Richtung sei M eine Turingmaschine, die f in primitiv rekursiver Laufzeit berechnet. Sei t_f die dazu gehörige Laufzeit-Funktion. Wir wissen aus der Vorlesung, dass f als Turing-berechenbare Funktion zumindest partiell rekursiv ist. Nach Voraussetzung terminiert M auf allen Eingaben, daher ist f sogar total rekursiv. f hat eine Operatordarstellung, und in dieser treten endlich viele Minimierungen auf. Wir gehen induktiv nach diesen Minimierungen vor (von innen): Wenn bei der Berechnung von $f(\bar{x})$ die Minimierung $\mu y g(x_1(\bar{x}), y)$ mit primitiv rekursiven x_1 und g auftritt, dann gilt

$$(\mu y) g(x_1(\bar{x}), y) = (\mu y \leq t_f(\bar{x})) g(x_1(\bar{x}), y)$$

Die Funktion $\bar{x} \mapsto (\mu y \leq t_f(\bar{x})) g(x_1(\bar{x}), y)$ ist nach dem Hinweis und weil t_f p.r. ist p.r. und daher insgesamt auch f .

Anzahl der Rechenschritte !

Code der Berechnung

Man bräuhle hier $\mu y \leq b(t_f(\bar{x}))$ wobei $b(n)$ Schranke für Code von n -Schritt-Berechnung von M ist (ist prim. rek.).