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

$$f(x) = 0 = \mathrm{c}^1_0$$
$$g(x, y, z) = z + x = \mathrm{Cn}[+, \mathrm{P}^3_3, \mathrm{P}^3_1]$$

und es gilt $\cdot = \mathrm{Pr}[f, g] = \mathrm{Pr}[\mathrm{c}^1_0, \mathrm{Cn}[+, \mathrm{P}^3_3, \mathrm{P}^3_1]]$.

(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 $p = \mathrm{Pr}[f, g]$ mit

$$f(0) = 0 = \mathrm{c}^1_0$$
$$g(y, z) = y = \mathrm{P}^2_1.$$

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

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

$$f(x) = x \dot{-} 0 = x = \mathrm{P}^1_1$$
$$g(x, y, z) = p(z) = \mathrm{Cn}[p, \mathrm{P}^3_3].$$

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

Falls $x \leqslant y$, dann ist $x \dot{-} y = 0$; sonst ist $x \dot{-} y = x - y \geqslant 1$. Folglich gilt $\chi_{\leqslant}(x, y) = 1 \dot{-} (x \dot{-} y)$, 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 $\chi_{=}(x, y) = \chi_{\leqslant}(x, y) \cdot \chi_{\leqslant}(y, x)$ unmittelbar aus (d).

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

Das folgt aus dem bisher Gezeigten wegen

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

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

(a) $h\colon (\overline{x}, z) \mapsto \sum\limits_{y=0}^{z} f(\overline{x}, y)$:

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

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

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

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

(b) $h\colon (\overline{x}, z) \mapsto \prod\limits_{y=0}^{z} f(\overline{x}, y)$: Genauso wie (a), nur mit »·« statt »+«.

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

(c) $h\colon (\overline{x}, z) \mapsto \forall y \leqslant z\ (f(\overline{x}, y) = 1)$:

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

(d) $h\colon (\overline{x}, z) \mapsto \exists y \leqslant z\ (f(\overline{x}, y) = 1)$:

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

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

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

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

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

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	$\rhd$	q_0	$\rhd$	$\rightarrow$
q_0	0	k_0	$\llcorner\!\lrcorner$	$\rightarrow$
k_0	0	k_0	0	$\rightarrow$
k_0	1	k_0	1	$\rightarrow$
k_0	$\llcorner\!\lrcorner$	k_0'	$\llcorner\!\lrcorner$	$\rightarrow$
k_0'	0	k_0'	0	$\rightarrow$
k_0'	1	k_0'	1	$\rightarrow$
k_0'	$\llcorner\!\lrcorner$	r_0	0	$\leftarrow$
r_0	0	r_0	0	$\leftarrow$
r_0	1	r_0	1	$\leftarrow$
r_0	$\llcorner\!\lrcorner$	r_0'	$\llcorner\!\lrcorner$	$\leftarrow$
r_0'	0	r_0'	0	$\leftarrow$
r_0'	1	r_0'	1	$\leftarrow$
r_0'	$\llcorner\!\lrcorner$	q_0	0	$\rightarrow$
q_0	1	k_1	$\llcorner\!\lrcorner$	$\rightarrow$
k_1	0	k_1	0	$\rightarrow$
k_1	1	k_1	1	$\rightarrow$
k_1	$\llcorner\!\lrcorner$	k_1'	$\llcorner\!\lrcorner$	$\rightarrow$
k_1'	0	k_1'	0	$\rightarrow$
k_1'	1	k_1'	1	$\rightarrow$
k_1'	$\llcorner\!\lrcorner$	r_1	1	$\leftarrow$
r_1	0	r_1	0	$\leftarrow$
r_1	1	r_1	1	$\leftarrow$
r_1	$\llcorner\!\lrcorner$	r_1'	$\llcorner\!\lrcorner$	$\leftarrow$
r_1'	0	r_1'	0	$\leftarrow$
r_1'	1	r_1'	1	$\leftarrow$
r_1'	$\llcorner\!\lrcorner$	q_0	1	$\rightarrow$
q_0	$\llcorner\!\lrcorner$	`fertig`	$\llcorner\!\lrcorner$	$-$

.

Erklärung: M liest das erste Zeichen. Dieses ist entweder »0« (gelber Block) oder »1« (türkiser Block). Das Zeichen wird durch »$\sqcup$« 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 »$\sqcup$« auftritt (also die Grenze zwischen dem ursprünglichen Wort und dem bereits kopierten Teil), worauf sie in den Zustand k_0' wechselt. Tritt dann erneut »$\sqcup$« auf, ist das Ende erreicht und »0« wird geschrieben. Der neue Zustand ist dann r_0 (»rückwärts Null«) bzw. nach einem Erreichen von »$\sqcup$« r_0'. Tritt dann nochmals »$\sqcup$« 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 »$\sqcup$«, 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 $t\colon \mathbb{N}^k \to \mathbb{N}$ existiert, so dass für alle $x \in \mathbb{N}^k$ eine Konfiguration $(\texttt{fertig}, u, v)$ existiert sowie ein $k \leqslant t(x)$, sodass $(q_0, \rhd, x) \overset{M^k}{\to} (\texttt{fertig}, u, v)$. Wir zeigen, dass $f : \mathbb{N}^k \to \mathbb{N}$ 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 $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k$ mit $\sqcup$«:
     Wir wählen $t_1(\overline{x}) = n + 2 + \sum\limits_{i=1}^{n} x_i$, 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 $t_2(\overline{x}) = t_3(\overline{x}) = t_4(\overline{x})$ und insgesamt $t(\overline{x}) = 4t_1(\overline{x})$.
- Für den Induktionsschritt seien $f\colon \mathbb{N}^n \hookrightarrow \mathbb{N}$, $g_1, \ldots, g_n\colon \mathbb{N}^k \hookrightarrow \mathbb{N}$ und $h = \mathrm{Cn}[f, g_1, \ldots, g_n]$. Nach IV gibt es für alle $i = 1, \ldots, n$ 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 $\overline{x}$ auf jedes der n Bänder«:

   Wir können die Anzahl der Schritte mit $t_1(\overline{x}) = 2n \sum_{i=1}^{k} x_i$ 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 $g_i(\overline{x})$ durch die Turingmaschine M_{g_i} auf dem i-ten Band«:

   Damit können wir den Aufwand abschätzen durch $t_2(\overline{x}) = \sum_{i=1}^{n} t_{g_i}(\overline{x})$.
3. »Kopiere $g_2(\overline{x}), \dots, g_n(\overline{x})$ auf das erste Band zu $g_1(\overline{x})$«:

   Für jedes $i = 1, \dots, n$ hat $g_i(\overline{x})$ höchstens $g_i(\overline{x})$ Ziffern in Binärdarstellung. Wir müssen stets ein Leerzeichen einfügen und den Cursor zurückbewegen und verwenden sicherheitshalber die nicht scharfe Abschätzung $t_3(\overline{x}) = 5 \sum_{i=1}^{n} g_i(\overline{x})$.
4. »Berechne $f(g_1(\overline{x}), \dots, g_n(\overline{x})$ durch M_f auf dem ersten Band«:

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

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

$$\tilde{t}_f(\overline{x}, 0) = t_h(\overline{x}),$$
$$\tilde{t}_f(\overline{x}, y + 1) = t_g(\overline{x}, y, f(\overline{x}, y)).$$

  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. Ingesamt können wir den Aufwand also mit

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

  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 $f(\overline{x})$ die Minimierung $\mu y\ g(x_1(\overline{x}), y)$ mit primitiv rekursiven x_1 und g auftritt, dann gilt

$$(\mu y)\ g(x_1(\overline{x}), y) = (\mu y \leqslant t_f(\overline{x}))\ g(x_1(\overline{x}), y).$$

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