

# Theoretische Informatik

## 3. Übung

Richard Weiss

16.5.2021

**Aufgabe 1.** Zeigen Sie dass die folgenden Funktionen primitiv rekursiv sind <sup>1</sup>:

(a) Multiplikation  $(x, y) \mapsto x \cdot y$

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

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

(d) Die charakteristische Funktion von kleiner-gleich  $(x, y) \mapsto \chi_{\leq}(x, y) = \begin{cases} 1 & \text{falls } x \leq y \\ 0 & \text{falls } x > y \end{cases}$

(e) Die charakteristische Funktion der Gleichheit  $(x, y) \mapsto \chi_{=}(x, y) = \begin{cases} 1 & \text{falls } x = y \\ 0 & \text{falls } x \neq y \end{cases}$

(f) Falls  $g, f_0, f_1, \dots, f_n : \mathbb{N}^k \rightarrow \mathbb{N}$  primitiv rekursiv sind, dann ist auch

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

primitiv rekursiv.

*Lösung.*

(a)  $f := \text{Cn}[P_1^1, 0]$  ist, als Verknüpfung von Basisfunktion, primitiv rekursiv.

$$f(x) := 0, \text{ für } x \in \mathbb{N}.$$

Es gilt für alle  $x \in \mathbb{N}$ :

$$x \cdot 0 = 0 = f(x).$$

---

<sup>1</sup>Sie dürfen dafür annehmen dass für alle  $n, k \in \mathbb{N}$  die konstante Funktion  $c_k^n : \mathbb{N}^n \rightarrow \mathbb{N}, (x_1, \dots, x_n) \mapsto k$  primitiv rekursiv ist.


Aus Beispiel 3.1 wissen wir schon, dass die Addition  $(x, y) \mapsto x + y$  zweier natürlicher Zahlen primitiv rekursiv ist.

Wir können  $\cdot$  also als primitive Rekursion der primitiv Rekursiven  $g := \text{Cn}[+, P_1^1, P_1^3]$  ist daher, als Verknüpfung von primitiv rekursiven Funktionen, primitiv rekursiv.

$$g(x, y, z) := z + x, \quad \text{für } x, y, z \in \mathbb{N}.$$

Es gilt für alle  $x, y \in \mathbb{N}$ :

$$x \cdot (y + 1) = x \cdot y + x = g(x, y, x \cdot y).$$

Schließlich ist  $\cdot = \text{Pr}[f, g]$ , als primitive Rekursion von primitiv rekursiven Funktionen, primitiv rekursiv. 

(b)  $f := 0$  ist, als Basisfunktion, primitiv rekursiv. Es gilt


$$p(0) = 0 = f.$$

$g = P_1^2$  ist, als Basisfunktion, primitiv rekursiv.

$$g(y, z) = y, \quad \text{für } y, z \in \mathbb{N}.$$

Es gilt für alle  $y \in \mathbb{N}$ :

$$p(y + 1) = y = g(y, z).$$

Schließlich ist  $p = \text{Pr}[f, g]$ , als primitive Rekursion von primitiv rekursiven (Basis-)Funktionen, primitiv rekursiv. 

(c)  $f := P_1^1$  ist, als Basisfunktion, primitiv rekursiv.

$$f(x) = x, \quad \text{für } x \in \mathbb{N}.$$

Es gilt für alle  $x \in \mathbb{N}$ :

$$x \div 0 = x = f(x)$$

$g = \text{Cn}[p, P_3^3]$  ist, als Verknüpfung von primitiv rekursiven Funktionen, primitiv rekursiv.

$$g(x, y, z) = p(z), \quad \text{für } x, y, z \in \mathbb{N}.$$

Wir machen eine Fallunterscheidung, um zu zeigen, dass für alle  $x, y \in \mathbb{N}$ :

$$x \div (y + 1) = p(x \div y) = g(x, y, x \div y), \quad \text{für } x, y \in \mathbb{N}.$$

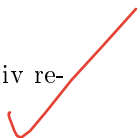
Seien also  $x, y \in \mathbb{N}$ .

1. Fall  $(x \leq y + 1)$ :

$$x \dot{-} (y + 1) = 0 = p(0) = p(x \dot{-} y)$$

2. Fall  $(x > y + 1)$ :

$$x \dot{-} (y + 1) = x - (y + 1) = \underbrace{x - y}_{>0} - 1 = \underbrace{x \dot{-} y}_{>0} - 1 = p(x \dot{-} y)$$

Schließlich ist  $\dot{-} = \text{Pr}[f, g]$ , als primitive Rekursion von primitiv rekursiven Funktionen, primitiv rekursiv. 

Es sei angemerkt, dass man mit Projektionen die Argumente immer umstrukturieren kann; z.b.

$$(a, b, c, x, y, z) \rightsquigarrow (z, x, x, c, a, z)$$

durch

$$(P_6^6, P_4^6, P_4^6, P_3^6, P_1^6, P_6^6).$$

Nachdem die Projektionen allesamt primitiv rekursive (Basis-)Funktionen sind, ist diese Umstrukturierung auch „primitiv rekursiv“.

(d) Es gilt für alle  $x, y \in \mathbb{N}$  :

$$\begin{aligned} x \vee y &:= \max(x, y) \\ &= \begin{cases} x, & x \geq y, \\ y, & x < y \end{cases} \\ &= x + (y \dot{-} x), \end{aligned}$$

$$\begin{aligned} x \wedge y &:= \min(x, y) \\ &= x + y - \max(x, y), \end{aligned}$$

und

$$\chi_{\leq}(x, y) = \min(1, (y + 1) \dot{-} x).$$


Die  $\vee$ - $\wedge$ -Notation verwenden wir später noch. 

(e) Es gilt für alle  $x, y \in \mathbb{N}$  :

$$\chi_{=}(x, y) = \chi_{\leq}(x, y) \cdot \chi_{\leq}(y, x).$$

(f) Mittels Induktion nach  $n$  sieht man, dass für alle  $\bar{x} \in \mathbb{N}^k$  :

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

und  $h$ , als endliche Verknüpfung primitiv rekursiver Funktionen, primitiv ist. 

□

**Aufgabe 2.** Sei  $f : \mathbb{N}^{k+1} \rightarrow \mathbb{N}$  primitiv rekursiv. Zeigen Sie dass die folgenden Funktionen ebenfalls primitiv rekursiv sind:

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

(b)  $(\bar{x}, z) \mapsto \prod_{y=0}^z f(\bar{x}, y)$

Sei nun  $f : \mathbb{N}^{k+1} \rightarrow \{0, 1\}$  primitiv rekursiv. Zeigen Sie dass die folgenden Funktionen ebenfalls primitiv rekursiv sind:

(c)  $(\bar{x}, z) \mapsto \forall y \leq z f(\bar{x}, y) = \begin{cases} 1 & \text{falls für alle } y \in \{0, \dots, z\} \text{ gilt: } f(\bar{x}, y) = 1 \\ 0 & \text{falls ein } y \in \{0, \dots, z\} \text{ existiert so dass: } f(\bar{x}, y) = 0 \end{cases}$

(d)  $(\bar{x}, z) \mapsto \exists y \leq z f(\bar{x}, y) = \begin{cases} 1 & \text{falls ein } y \in \{0, \dots, z\} \text{ existiert so dass } f(\bar{x}, y) = 1 \\ 0 & \text{falls für alle } y \in \{0, \dots, z\} \text{ gilt: } f(\bar{x}, y) = 0 \end{cases}$

*Lösung.*

(a) Wir schreiben (nur) hier  $\varphi$  statt  $f$ , um das Symbol „ $f$ “ zu reservieren. Seien

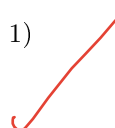
$$h(\bar{x}, y) := \sum_{\eta=0}^y \varphi(\bar{x}, \eta), \quad \text{für } \bar{x} \in \mathbb{N}^k, y \in \mathbb{N}$$

und

$$\begin{aligned} f &:= \text{Cn}[\varphi, P_1^k, \dots, P_k^k, 0], \\ g &:= \text{Cn}[+, P_{k+2}^{\text{red } k+2}, \text{Cn}[\varphi, P_1^{k+2}, \dots, P_k^{k+2}, \text{Cn}[+, P_{k+1}^{k+2}, 1]]], \end{aligned}$$

d. h. für  $\bar{x} \in \mathbb{N}^k$ , und  $y, z \in \mathbb{N}$  ist

$$\begin{aligned} f(\bar{x}) &:= \varphi(P_1^k(\bar{x}), \dots, P_k^k(\bar{x}), 0) \\ &= \varphi(\bar{x}, 0), \\ g(\bar{x}, y, z) &:= P_{k+2}^{k+2}(\bar{x}, y, z) + \varphi(P_1^{k+2}(\bar{x}, y, z), \dots, P_k^{k+2}(\bar{x}, y, z), P_{k+1}^{k+2}(\bar{x}, y, z) + 1) \\ &= z + \varphi(\bar{x}, y + 1). \end{aligned}$$

Dann gilt 

$$\begin{aligned}
h(\bar{x}, 0) &:= \sum_{\eta=0}^0 \varphi(\bar{x}, \eta) \\
&= \varphi(\bar{x}, 0) \\
&= f(\bar{x}), \\
h(\bar{x}, y+1) &= \sum_{\eta=0}^{y+1} \varphi(\bar{x}, \eta) \\
&= \sum_{\eta=0}^y \varphi(\bar{x}, \eta) + \varphi(\bar{x}, y+1) \\
&= h(\bar{x}, y) + \varphi(y+1) \\
&= g(\bar{x}, y, h(\bar{x}, y)).
\end{aligned}$$

(b) Das sieht man analog zu (a), nur mit „ $\prod$ “ statt „ $\sum$ “, und „ $\cdot$ “ statt „ $+$ “.

(c)

$$(\forall y \leq z f(\bar{x}, y)) = \prod_{y=1}^z f(\bar{x}, y).$$

(d) Wir treiben den Spaß weiter mit  $\neg g := 1 \div g$ , für  $g : \mathbb{N}^{k+1} \rightarrow \{0, 1\}$ .

$$(\exists y \leq z f(\bar{x}, y)) = (\neg \forall y \leq z \neg f(\bar{x}, y))$$

□

**Aufgabe 3.** Zeigen Sie dass die folgenden Relationen und Funktionen primitiv rekursiv sind:

1. Teilbarkeit:  $(x, y) \mapsto \begin{cases} 1 & \text{falls } x \text{ ein Teiler von } y \text{ ist} \\ 0 & \text{sonst} \end{cases}$

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

3. Die Eulersche  $\varphi$ -Funktion:  $\varphi : \mathbb{N} \rightarrow \mathbb{N}, n \mapsto |\{i \in \mathbb{N} \mid 1 \leq i \leq n, \text{ggT}(i, n) = 1\}|$ .

*Lösung.*

(a)

$$x \mid y = (\exists z \leq y \chi_=(x \cdot z, y))$$

(b) Sei  $\chi_{\neq} := 1 \div \chi_ =$ .

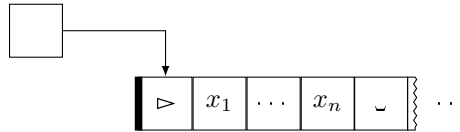
$$\chi_=(\text{ggT}(x, y), 1) = (\neg \exists z \leq x + y (z \mid x \wedge z \mid y \wedge \chi_{\neq}(z, 1)))$$

(c) Es gilt für alle  $n \in \mathbb{N}$ :

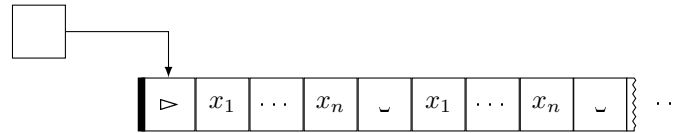
$$\begin{aligned}
\varphi(n) &= |\{i \in \mathbb{N} \mid 1 \leq i \leq n, \text{ggT}(i, n) = 1\}| \\
&= \sum_{i=1}^n |\{i \mid \text{ggT}(i, n) = 1\}| \\
&= \sum_{i=1}^n \chi = (\text{ggT}(i, n), 1).
\end{aligned}$$

□

**Aufgabe 4.** Geben Sie eine Turingmaschine an, die eine Duplikation durchführt, d.h. die das Eingabeband

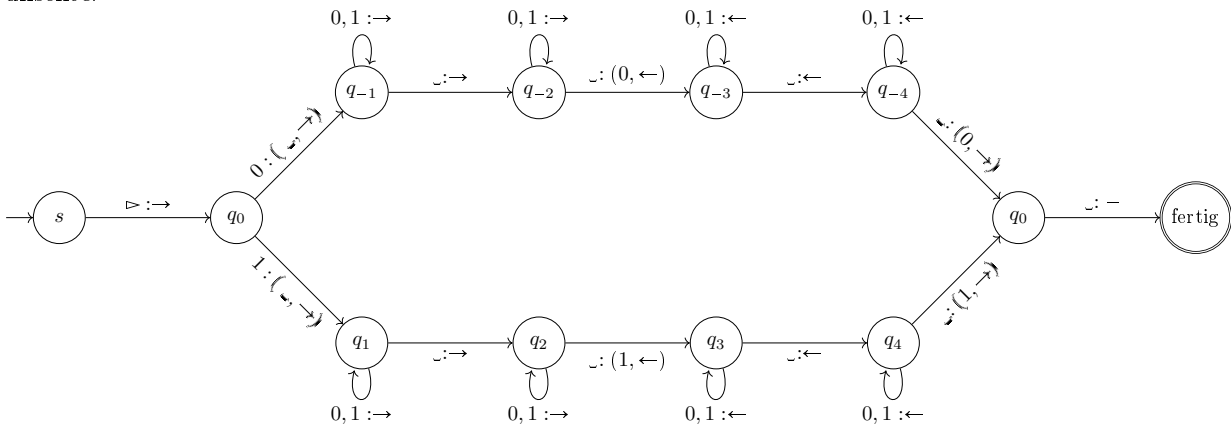


für  $x_1, \dots, x_n \in \{0, 1\}$  in das Ausgabeband



transformiert.

*Lösung.* Eine Turingmaschine, die die Instruktionen aus dem folgenden Diagramm befolgt, leistet das Gewünschte.



Der Ästhetik halber kommt  $q_0$  doppelt vor. Beide Vorkommnisse werden identifiziert und die Mengen der jeweiligen Inputs bzw. Outputs vereinigt.

Die Zustände sind wie folgt zu verstehen.

- $q_0$  ... Wenn ein  $\_$  vorliegt, terminiere; sonst, kopiere das vorliegende Bit, 0 oder 1, und setze den Platzhalter  $\_$ .
- $q_{\pm 1}$  ... Trage den Bit bis zur Barriere, dem nächsten  $\_$ .
- $q_{\pm 2}$  ... Setze den Bit beim nächst-rechten  $\_$  ein.
- $q_{\pm 3}$  ... Geh zur Barriere zurück.
- $q_{\pm 4}$  ... Geh zum Platzhalter zurück.

□

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

Zeigen Sie dass eine Funktion  $f : \mathbb{N} \rightarrow \mathbb{N}$  primitiv rekursiv ist genau dann wenn eine Turingmaschine existiert die  $f$  in primitiv rekursiver Laufzeit berechnet.

*Hinweis: Stützen Sie sich auf die Beweise der Äquivalenz der Begriffe Turing-berechenbar und partiell rekursiv. Sie dürfen die Aussage verwenden dass für alle primitiv rekursiven  $g : \mathbb{N}^{k+1} \rightarrow \{0, 1\}$  auch die Funktion*

$$(\bar{x}, z) \mapsto (\mu y \leq z) g(\bar{x}, y) = \begin{cases} \text{das kleinste } y \leq z \text{ so dass } g(\bar{x}, y) = 1 & \text{falls so ein } y \text{ existiert} \\ 0 & \text{sonst} \end{cases}$$

*primitiv rekursiv ist.*

*Lösung.*

1. Richtung („ $\implies$ “):

Wir zeigen, dass es für jede primitiv rekursive Funktion eine Turingmaschine, und eine primitiv rekursive Funktion gibt, die deren Laufzeit beschränkt, i.Z.

$$\begin{aligned} \forall f : \mathbb{N}^k \rightarrow \mathbb{N} \text{ primitiv rekursiv :} \\ \exists M = \langle Q, \delta, q_0 \rangle \text{ Turingmaschine, } \exists t : \mathbb{N} \rightarrow \mathbb{N} \text{ primitiv rekursiv :} \\ \forall x \in \mathbb{N}^k : \\ \exists (\text{fertig}, u, v) \text{ Konfiguration, } \exists k \leq t(x) : \\ (q_0, \triangleright, x) \xrightarrow{M^k} (\text{fertig}, u, v). \end{aligned}$$

Sei  $h$  primitiv rekursiv. Dann hat  $h$  eine Operator-darstellung. Wir gehen mit Induktion auf dieser Operator-darstellung vor. Dabei werden wir immer die Turingmaschine aus dem Beweis von Satz 3.2 verwenden.

Für die Induktionsbasis ist zu zeigen, dass jede der Basisfunktionen, in primitiv rekursiver Laufzeit, Turing-berechenbar ist.

- Konstante Null:

Der Cursor läuft immer eins nach vorne, und dann wieder zurück. Die Turingmaschine hat also Konstante Laufzeit. Wir wählen also

$$t_0 := 2 \geq 2.$$

- Nachfolgerfunktion:

Sei  $x \in \mathbb{N}$  die Eingabe Der Cursor läuft in beiden Fällen,  $x = 2^n - 1$  für ein  $n \in \mathbb{N}$ , und sonst, vom Anfang bis eine Stelle nach dem Ende der besetzten Stellen, und dann wieder (komplett) zurück. (In ersterem Fall wird am Ende noch, in konstanter Laufzeit, eine 1 an die erste Stelle geschrieben.) Die Turingmaschine hat also lineare Laufzeit. Wir wählen also

$$t_s(x) := 2x + 2 \geq 2[\log_2 x] + 2.$$

- Projektionen:

Für  $k \geq 1$  und  $1 \leq i \leq k$ , wird die Projektion  $P_i^k$ , auf der Eingabe  $\bar{x} = (x_1, \dots, x_k)$ , durch 4 Schritte berechnet. Diese haben allesamt primitiv rekursive Laufzeit.

1. „Überschreibe  $x_1, \dots, x_{i-1}, x_i, \dots, x_k$  mit  $\sqsubset$ “  
Der Cursor muss vom Anfang bis zur  $k$ -ten Stelle mit  $\sqsubset$  laufen. Wir wählen also

$$t_1(\bar{x}) := \sum_{j=1}^k (x_j + 1) \geq \sum_{j=1}^k (\lfloor \log_2 x_j \rfloor + 1).$$

2. „Bewege den Cursor zurück an den Anfang von  $x_i$ .“  
Wir wählen also

$$t_2(\bar{x}) := \sum_{j=k}^i (x_j + 1) \geq \sum_{j=k}^i (\lfloor \log_2 x_j \rfloor + 1).$$

3. „Verschiebe  $x_i$  Zeichen für Zeichen an den Anfang des Bands.“  
Das funktioniert ähnlich wie in Aufgabe 4, nur dass die Bits nicht kopiert (d.h. ausgeschnitten und dann wieder aufgefüllt), sondern ausgeschnitten werden. Insgesamt muss der Cursor  $(\log_2 x_i)$ -mal, in linearer Laufzeit, nach hinten und nach vorne fahren. Eventuell muss dann noch ein Marker gelöscht werden, damit der Cursor nicht ewig lang nach (nicht) noch nicht verschobenen  $x_i$ -Bits sucht, nachdem alle  $x_i$ -Bits von den alten Stellen zu den neuen verschoben wurden. Das geht aber in konstanter Laufzeit. Insgesamt finden wir ein passendes  $t_3(\bar{x})$
4. „Bewege den Cursor auf das Startsymbol.“  
Der Cursor ist höchstens an der Stelle des ursprünglich  $k$ -ten  $\sqsubset$  (von links). Wir wählen also

$$t_4(\bar{x}) := t_1(\bar{x}).$$

Insgesamt wählen wir also

$$t_{P_i^k}(\bar{x}) := \sum_{j=1}^4 t_j(\bar{x}).$$



Kommen wir nun zum Induktionsschritt.

- Komposition:

Seien  $f : \mathbb{N}^n \rightarrow \mathbb{N}$  und  $g_1, \dots, g_n : \mathbb{N}^k \rightarrow \mathbb{N}$  primitiv Rekursiv, sodass  $h = \text{Cn}[f, g_1, \dots, g_n] : \mathbb{N}^k \rightarrow \mathbb{N}$ . Die Komposition  $h$  wird, auf der Eingabe  $\bar{x} = (x_1, \dots, x_k)$ , auf einem der  $n$  Bänder (und bei den anderen  $n - 1$  Bändern mit leerer Eingabe  $\varepsilon$ ), wieder durch 4 Schritte berechnet.

1. „Kopiere die Eingabe  $\bar{x}$  auf jedes der  $n$  Bänder.“  
Auf dem Band, auf dem die Eingabe  $\bar{x}$  liegt, muss der Cursor den anderen „voranschreiten“, und die übrigen Cursor tragen auf den übrigen Bändern den entsprechenden Input ein. Insgesamt, muss der „Eingabe-Cursor“ zum  $k$ -ten  $\sqsubset$ , und zurück laufen. Nachdem die übrigen Cursor diesem „folgen“, gilt das auch für sie. Wir wählen also

$$t_1(\bar{x}) := 2 \sum_{i=1}^k (x_i + 1) = 2 \sum_{i=1}^k (\lfloor \log_2 x_i \rfloor + 1).$$

2. „Für  $i = 1, \dots, n$  berechne  $g_i(\bar{x})$  durch die Turingmaschine  $M_{g_i}$  auf dem  $i$ -ten Band.“  
Laut der Induktionshypothese, gibt es für jedes  $M_{g_i}$  eine entsprechende primitiv rekursive „Laufzeit-Funktion“  $t_{g_i}$ . Wenn die Turingmaschinen  $M_{g_i}$  hintereinander operieren, so wählen wir

$$t_2(\bar{x}) := \sum_{i=1}^n t_{g_i}(\bar{x}).$$



Falls sie parallel agieren, so wählen wir

$$t_2(\bar{x}) := \max_{i=1}^n t_{g_i}(\bar{x}),$$

wobei wir die „nicht maximierenden“ Turingmaschinen  $M_{g_b}$ , mit  $a := \operatorname{argmax} \max_{i=1}^n t_{g_i}(\bar{x}) \neq b = 1, \dots, n$ , erweitern, sodass sie auf  $M_{g_a}$  „warten“ müssen.

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

Das Kopieren verläuft ähnlich wie bei 1. Der Cursor muss beim 1-ten Band zunächst bis zum Ende (dem ersten  $\_$ ) laufen, die anderen Bänder müssen warten. Dann werden, eines nach dem anderen, das 2-te bis zum  $n$ -ten Band an die passenden freien Stellen des 1-ten Bands verschoben. Danach müssen die Cursor wieder zum Anfang zurückkehren. Wir wählen also, analog zu 1.,

$$t_3(\bar{x}) := 2 \sum_{i=1}^n (g_i(\bar{x}) + 1) = 2 \sum_{i=1}^n (\lfloor \log_2 g_i(\bar{x}) \rfloor + 1).$$

4. „Berechne  $f(g_1(\bar{x}), \dots, g_n(\bar{x}))$  durch die Turingmaschine  $M_f$  auf dem ersten Band.“

Laut der Induktionshypothese, gibt es für  $M_f$  eine entsprechende primitiv rekursive „Laufzeit-Funktion“  $t_f$ . Wir wählen also

$$t_4(\bar{x}) := t_f(\bar{x}).$$

Insgesamt wählen wir also

$$t_h(\bar{x}) := \sum_{j=1}^4 t_j(\bar{x}).$$



• Primitive Rekursion:

Seien  $f : \mathbb{N}^k \rightarrow \mathbb{N}$  und  $g : \mathbb{N}^{k+2} \rightarrow \mathbb{N}$  primitiv Rekursiv, sodass  $h = \operatorname{Pr}[f, g] : \mathbb{N}^{k+1} \rightarrow \mathbb{N}$ .

- „Das erste Band enthält konstant die Eingabe  $\bar{x}, y$  von  $h$ .“  
Dafür ist keine Laufzeit nötig.
- „Das zweite Band enthält einen Zähler  $z$  der mit 0 initialisiert wird.“  
Dafür ist Laufzeit  $t_0$  erforderlich.
- „Das dritte Band wird unter Verwendung von  $M_f$  mit  $f(\bar{x}) = h(\bar{x}, 0)$  initialisiert.“  
Laut der Induktionshypothese, gibt es für  $M_f$  eine entsprechende primitiv rekursive „Laufzeit-Funktion“  $t_f$ . Dieser Schritt hat also Laufzeit  $t_f(x)$ .  
Wir wählen also

$$t_h(\bar{x}, 0) := t_f(\bar{x}),$$

bzw.

$$\varphi(\bar{x}) := t_f(\bar{x}).$$

- \* „In jedem Schritt wird nun unter Verwendung von  $M_g$  auf dem zweiten Band aus  $h(\bar{x}, z)$  der Wert  $h(\bar{x}, z + 1)$  berechnet [...]“  
Sei  $t_h(\bar{x}, z)$  bereits bekannt und primitiv rekursiv (am Anfang  $z = 0$ ). Laut der Induktionshypothese, gibt es für  $M_g$  eine entsprechende primitiv rekursive „Laufzeit-Funktion“  $t_g$ . Die Turingmaschine berechnet zunächst  $h(\bar{x}, z)$ . Das benötigt die Laufzeit

$$t_h(\bar{x}, z).$$

Danach wird der Wert  $h(\bar{z} + 1)$  berechnet. Das benötigt die Laufzeit

$$t_g(x, z, h(\bar{x}, z)).$$

\* „[...] und dabei  $z$  auf dem zweiten Band entsprechend inkrementiert.“  
Das benötigt die Laufzeit

$$t_s(z).$$

\* „Dieser Schritt wird wiederholt bis  $z$  (vom zweiten Band gleich  $y$  (vom ersten Band) ist.“  
Um zu überprüfen, ob  $z = y$  ist, muss mindestens der ganze Bit-String von  $z$ , nach vorne und zurück, durchlaufen werden. Das benötigt Laufzeit

$$2\lfloor \log_2 z \rfloor.$$

Wir wählen also

$$t_=(z) := 2z \geq 2\lfloor \log_2 z \rfloor.$$

Wir wählen also

$$t_h(\bar{x}, z + 1) := t_h(\bar{x}, z) + t_g(x, z, h(\bar{x}, z)) + t_s(z) + t_=(z),$$

bzw.

$$\psi(\bar{x}, z, u) := u + t_g(x, z, u) + t_s(z) + t_=(z), \quad \text{für } u \in \mathbb{N}.$$

Insgesamt wählen wir also

$$t_h := \Pr[\phi, \psi].$$

2. Richtung („ $\Leftarrow$ “):

Wir zeigen, dass jede Funktion, für die es eine Turingmaschine, und eine primitiv rekursive Funktion gibt, die deren Laufzeit beschränkt, primitiv rekursiv ist.

Sei  $f : \mathbb{N} \rightarrow \mathbb{N}$  eine solche Funktion, und  $M_f$  sowie  $t_f$  die zugehörige Turingmaschine bzw. primitiv rekursive Laufzeit-Funktion. Laut Satz 3.3, ist diese bereits partiell rekursiv. Nachdem  $f$  aber total ist, ist  $f$  schlicht rekursiv.

Wenn  $f$  eine Basisfunktion ist, oder daraus, durch eine endliche Anzahl von Anwendungen der Operatoren Komposition und primitive Rekursion, erhalten wird, ist  $f$  bereits, per definitionem, primitiv rekursiv. Möge  $f$  nun ebenfalls durch eine endliche Anzahl von Minimierungen entstehen; o.B.d.A. bloß eine Anwendung des  $\mu$ -Operators. Für  $x \in \mathbb{N}$ , können wir  $f(x)$  daher schreiben als

$$f(x) = \mu y g(x, y), \quad \text{d.h.} \quad g(x, y) = T_1(\#M_f, x, y) \quad \text{(aus Satz 3.3) } \uparrow$$

mit einer, von  $x$  unabhängigen, primitiv rekursiven Funktion  $g : \mathbb{N}^{1+1} \rightarrow \mathbb{N}$ .

Code einer  
Berechnung

Nachdem  $f$  ja total ist, terminiert  $M_f$ , mit einer Laufzeit von höchstens  $t_f(x)$  Schritten. o.B.d.A. möge  $M_f$  seinen „Minimierungs-Vorgang“ so vollziehen, dass zuerst mit 0 getestet wird, dann 1, dann 2, usw. Dann kann die Anzahl der Tests locker durch  $t(x)$  beschränkt werden. Weil  $M_f$  schließlich terminiert, wurde der passende Kandidat für die Minimierung, unter  $t(x)$  Schritten, gefunden. Daher gilt sogar

$$f(x) = (\mu y \leq \underline{t(x)})g(x, y).$$

Wenn man nun and die, durchaus plausible, Annahme im Hinweis glaubt, dann auch, dass  $f$  tatsächlich, als Komposition von  $t_f$  und dem primitiv rekursiven  $\mu$ -Operator, primitiv rekursiv ist.

Achtung:  $t(x)$  beschränkt Anzahl der Rechenschritte aber i.A. nicht Code der Berechnung.  $\square$

Brechen hier  $\mu y \leq b(t(x))$  wobei  $b(n)$  Schritte auf Code von Berechnung in  $n$  Schritten ist.