
8. Rekursive und primitiv rekursive Funktionen

In diesem Abschnitt führen wir eine weitere (letzte) Formalisierung des Berechenbarkeitskonzeptes für Funktionen über den natürlichen Zahlen ein. Hatten wir bei den Registermaschinen schon von der Repräsentation der Zahlen abstrahiert, so gehen wir hier noch einen Schritt weiter und geben eine maschinenunabhängige Charakterisierung der (partiell) berechenbaren Funktionen. Hierzu gehen wir induktiv vor. Wir beginnen mit einfachen berechenbaren Funktionen, den Ausgangsfunktionen, und schließen diese gegen bestimmte effektive Operationen ab, die partiell berechenbare Funktionen in partiell berechenbare Funktionen überführen. Hierzu betrachten wir die Substitution (Einsetzen) von Funktionen, die primitive Rekursion und den μ -Operator. Die Funktionsklasse, die wir auf diese Art erhalten, ist die Klasse $F(\text{REK})$ der partiell rekursiven (manchmal auch μ -rekursiv genannten) Funktionen. Verzichtet man bei dem Abschluss auf den μ -Operator, so erhält man die Klasse $F(\text{PRIM})$ der primitiv rekursiven Funktionen. Von letzteren werden wir zeigen, dass sie gerade die von primitiven Registeroperatoren berechneten Funktionen sind, während die partiell rekursiven Funktionen mit den partiell Turing- oder Registermaschinen-berechenbaren Funktionen zusammenfallen.

Vergleicht man das Vorgehen hier mit den auf Maschinen basierenden Konzepten, so sieht man, dass die Ausgangsfunktionen den elementaren Operationen und Tests der Basismaschinen und die Operatoren, unter denen man abschließt, den Kontrollstrukturen auf der Programmebene entsprechen. Der hier verfolgte Ansatz, Funktionen mittels Substitution und Rekursion zu definieren, hat auch Eingang in die Programmierung gefunden. Z.B. basiert die Programmiersprache LISP auf diesen Ideen. Man spricht hier dann von *Funktionaler Programmierung* im Gegensatz zu der in den vorhergehenden Abschnitten behandelten *Imperativen Programmierung*.

Als Ausgangsfunktionen wählen wir

$$\begin{aligned} S(x) &= x + 1 && \text{(Nachfolger)} \\ U_i^n(x_1, \dots, x_n) &= x_i && \text{(n-stellige Projektion auf die } i\text{-te Komponente;} \\ &&& n \geq 1, 1 \leq i \leq n) \\ C_i^n(x_1, \dots, x_n) &= i && \text{(n-stellige konstante Funktion mit Wert } i; \\ &&& n, i \geq 0) \end{aligned}$$

Hierbei lassen wir auch die 0-stelligen Konstanten C_i^0 zu, die man mit der Zahl i identifizieren kann. Als Einsetzungsschema betrachten wir die simultane Substitution.

8.1 DEFINITION. Seien $g : \mathbb{N}^m \rightarrow \mathbb{N}$ und $h_1, \dots, h_m : \mathbb{N}^n \rightarrow \mathbb{N}$ m - bzw. n -stellige (partielle) Funktionen. Die aus g durch *simultane Substitution* von h_1, \dots, h_m entstehende n -stellige (partielle) Funktion $f = g(h_1, \dots, h_m)$ ist definiert durch

$$\forall \vec{x} \in \mathbb{N}^n (f(\vec{x}) = g(h_1(\vec{x}), \dots, h_m(\vec{x})))$$

Um $f(\vec{x})$, für berechenbare Funktionen g, h_1, \dots, h_m zu berechnen, berechnet man also zunächst die Funktionswerte y_1, \dots, y_m von $h_1(\vec{x}), \dots, h_m(\vec{x})$ und schließlich den Funktionswert von $g(y_1, \dots, y_m)$. Bei partiellen Funktionen legt man entsprechend fest, dass $f(\vec{x})$ genau dann definiert ist, wenn die zur Berechnung erforderlichen „Teilfunktionen“ alle definiert sind:

$$f(\vec{x}) \downarrow \Leftrightarrow \exists y_1, \dots, y_m (h_1(\vec{x}) \downarrow = y_1 \& \dots \& h_m(\vec{x}) \downarrow = y_m \& g(y_1, \dots, y_m) \downarrow)$$

Hiermit sieht man leicht, dass für (partiell) berechenbare g, h_1, \dots, h_m auch $g(h_1, \dots, h_m)$ wiederum (partiell) berechenbar ist.

Zur Vereinfachung der Notation soll im folgenden $f^{(n)}$ ausdrücken, dass $f: \mathbb{N}^n \rightarrow \mathbb{N}$ eine n -stellige (partielle) Funktion über \mathbb{N} ist.

8.2 BEISPIEL. Die *Komposition* $g \circ f$ von (partiellen) Funktionen $f^{(n)}$ und $g^{(1)}$ ist ein Spezialfall der simultanen Substitution, nämlich

$$(g \circ f)(\vec{x}) = g(f(\vec{x})) = g(f)(\vec{x}).$$

8.3 BEISPIEL. Das Polynom

$$p(x) = 2x^2 + 3x + 4$$

können wir unter Verwendung der Addition $+$ ⁽²⁾ und Multiplikation $*$ ⁽²⁾ sowie der oben angegebenen Ausgangsfunktionen durch folgende Substitutionen beschreiben

$$\begin{aligned} p(x) &= +(2x^2, +(3x, 4)) \\ &= +(* (2, *(x, x)), +(3, x), 4) \\ &= +(* (C_2^1(x), *(U_1^1(x), U_1^1(x))), +(* (C_3^1(x), U_1^1(x)), C_4^1(x))) \\ &= +(* (C_2^1(x), *(U_1^1(x), U_1^1(x))), +(* (C_3^1(x), U_1^1(x)), C_4^1(x))) \\ &= +(* (C_2^1(x), *(U_1^1(x), U_1^1(x))), +(* (C_3^1(x), U_1^1(x)), C_4^1(x))) \\ &= +(* (C_2^1(x), *(U_1^1(x), U_1^1(x))), +(* (C_3^1(x), U_1^1(x)), C_4^1(x))) \end{aligned}$$

d.h.

$$p = +(* (C_2^1, (U_1^1, U_1^1)), +(* (C_3^1, U_1^1), C_4^1)).$$

Hierbei haben wir also die *explizite* Definition von p über $+$ und $*$, d.h. die Definition von p durch einen Ausdruck, der die Funktionen $+$ und $*$ enthält, durch eine Folge von Substitutionen (unter Hinzunahme der Konstanten und Projektionen) beschrieben. Im nächsten Abschnitt werden wir zeigen, dass wir jede explizite Definition so darstellen können.

Als Nächstes führen wir die primitive Rekursion ein. Rekursive (oder – wie man auch sagt – induktive) Definitionen von Funktionen über \mathbb{N} findet man sehr häufig. Hierbei wird eine Funktion f an einer Stelle durch Rückgriff auf ihre Werte an vorhergehenden Stellen definiert. Beispielsweise kann man (wegen $x + (y + 1) = (x + y) + 1$) die Addition mit Hilfe des Nachfolgers rekursiv definieren durch

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

Ähnlich lässt sich (wegen $x * (y + 1) = (x * y) + x$) die Multiplikation mit Hilfe der Addition rekursiv charakterisieren:

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

Auch hier werden, wie bei den expliziten Definitionen (s. letztes Beispiel), Funktionen durch Ausdrücke definiert. Hier darf die zu definierende Funktion jedoch in dem Ausdruck selbst vorkommen, weshalb man hier von einer *impliziten* Definition spricht.

Allgemein ist eine implizite Definition ein Gleichungssystem, in dem die zu definierende Funktion und bekannte Funktionen vorkommen. Während bei den expliziten Definitionen die zu definierende Funktion links und Ausdrücke aus den bekannten Funktionen rechts stehen, kann bei impliziten Definitionen die zu definierende Funktion links und rechts vorkommen.

Man muss jedoch beachten, dass nicht jedes Gleichungssystem eine Funktion, geschweige denn eine totale Funktion definiert. Zum Beispiel ist das Gleichungssystem

$$\begin{aligned} f(0) &= 0 \\ f(2x+1) &= 1 \\ f(2x) &= f(2x+1)+1 \end{aligned}$$

offensichtlich inkonsistent (da es $0 = f(0) = f(1) + 1 = 2$ impliziert) während

$$f(x) = S(f(x+1))$$

bestenfalls als Definition der nirgends definierten Funktion aufgefasst werden kann, da die Rekursion nicht abbricht. ($f(0) = f(1) + 1 = f(2) + 2 = \dots$).

Dieses Problem tritt bei den obigen Definitionen von $+$ und $*$ jedoch nicht auf, da bei dem rekursiven Schritt die letzte Variable y dekrementiert und die anderen Variablen nicht verändert werden. Der Rekursionsschritt wird also gerade y -fach iteriert und dann die durch die 1. Gleichung beschriebene explizite Definition erreicht.

Rekursionen von diesem Typ nennt man primitive Rekursionen. Komplexere Rekursionen sind z.B. geschachtelte Rekursionen, bei denen die Rekursion über mehrere Variable läuft.

8.4 DEFINITION. Seien $g : \mathbb{N}^n \rightarrow \mathbb{N}$ und $h : \mathbb{N}^{n+2} \rightarrow \mathbb{N}$ (partielle) Funktionen. Die durch *primitive Rekursion* über g und h definierte (partielle) Funktion $f^{(n+1)} = \text{PR}(g, h)$ ist definiert durch

$$\begin{aligned} \forall \vec{x} \in \mathbb{N}^n (f(\vec{x}, 0) &= g(\vec{x})) \\ \forall \vec{x} \in \mathbb{N}^n \forall y \in \mathbb{N} (f(\vec{x}, y+1) &= h(\vec{x}, y, f(\vec{x}, y))) \end{aligned}$$

Aus obigen Bemerkungen sieht man leicht, dass für berechenbare Funktionen g und h die Funktion $f = \text{PR}(g, h)$ wiederum berechenbar ist. Beschreiben kann man die primitive Rekursion durch eine for-Schleife (Eingabe (\vec{x}, y) , Ausgabe z):

$$\begin{aligned} z &:= g(\vec{x}); \\ \text{for } i &= 1, \dots, y \text{ do } z := h(\vec{x}, y, z) \end{aligned}$$

wobei für $y = 0$ die Schleife nicht durchlaufen wird. Für partielles g und h legt man (wie bei der Substitution) fest, dass

$$\begin{aligned} f(\vec{x}, 0) \downarrow &\Leftrightarrow g(\vec{x}) \downarrow \\ f(\vec{x}, y+1) \downarrow &\Leftrightarrow \exists z (f(\vec{x}, y) \downarrow = z \ \& \ h(\vec{x}, y, z) \downarrow) \end{aligned}$$

(Insbesondere impliziert also $f(\vec{x}, y) \downarrow$, dass $f(\vec{x}, y') \downarrow$ für alle $y' < y$.) Hiermit sieht man leicht, dass die primitive Rekursion auch die partielle Berechenbarkeit erhält.

8.5 BEISPIEL. Die oben angegebenen Rekursionsgleichungen für $+$ und $*$ zeigen, dass

$$\begin{aligned} + &= \text{PR}(U_1^1, S(U_3^3)) \\ * &= \text{PR}(C_0^1, +(U_3^3, U_1^3)) \end{aligned}$$

Die Fakultätsfunktion $n!$, die den Rekursionsgleichungen $0! = 1$ und $(n+1)! = n! * (n+1)$ genügt, wird durch

$$! = \text{PR}(C_1^0, *(S(U_1^2), U_2^2))$$

beschrieben.

8.6 DEFINITION. Die Klasse $F(\text{PRIM})$ der *primitiv rekursiven Funktionen* ist induktiv definiert durch

- (i) $S, U_i^n, C_j^m \in F(\text{PRIM})$ (für $n \geq 1, 1 \leq i \leq n, m, j \geq 0$)
- (ii) Sind $g^{(m)}, h_1^{(n)}, \dots, h_m^{(n)} \in F(\text{PRIM})$ so auch $g(h_1, \dots, h_m)$.
- (iii) Sind $g^{(n)}, h^{(n+2)} \in F(\text{PRIM})$, so auch $\text{PR}(g, h)$.

Wie Beispiel 8.5 zeigt, sind $+$, $*$ und $!$ primitiv rekursiv. Im nächsten Abschnitt werden wir noch zahlreiche weitere Beispiele für primitiv rekursive Funktionen angeben.

Man beachte, dass primitiv rekursive Funktionen total sind. Die letzte Abschlusseigenschaft, die wir hier betrachten wollen, nämlich der μ -Operator, kann dagegen totale Funktionen in partielle Funktionen überführen. Grob gesprochen beschreibt der μ -Operator die kleinste Nullstelle einer Funktion.

8.7 DEFINITION. Sei $g: \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ eine (möglicherweise partielle) Funktion. Die aus g durch Anwendung des μ -Operators entstehende partielle Funktion $f^{(n)} = \mu(g)$ ist definiert durch

$$\begin{aligned} \forall \vec{x} \in \mathbb{N}^n (f(\vec{x}) = \mu y (g(\vec{x}, y) = 0 \ \& \ \forall z < y (g(\vec{x}, z) \downarrow))) \\ = \min\{y : g(\vec{x}, y) = 0 \ \& \ \forall z < y (g(\vec{x}, z) \downarrow)\}, \end{aligned}$$

wobei $\min \emptyset \equiv \uparrow$.

Der μ -Operator wird mitunter auch *Minimalisierungs-Operator* genannt. Man spricht auch von einem *Suchoperator*, da ja die kleinste Nullstelle gesucht wird. Die Berechnung von $\mu(g)$ lässt sich durch folgende while-Schleife beschreiben:

$$\begin{aligned} z &:= 0; \\ \text{while } g(\vec{x}, z) \neq 0 \text{ do } z &:= z + 1; \\ \mu(g)(\vec{x}) &:= z \end{aligned}$$

Dies zeigt, dass für partiell berechenbares g die partielle Funktion $\mu(g)$ ebenfalls partiell berechenbar ist. Zugleich verdeutlicht dies, warum wir für partielles g fordern, dass $g(\vec{x}, z) \downarrow$ für z unterhalb der Nullstelle y gelten muss. Für z mit $g(\vec{x}, z) \uparrow$ wird der zugehörige Schleifendurchgang nämlich nicht beendet, weshalb $\mu(g)(\vec{x}) \uparrow$ in diesem Fall. (Später werden wir zeigen, dass ohne diese Definiertheitsforderung der partielle μ -Operator partiell berechenbare g auf nichtberechenbare partielle $\mu(g)$ abbilden kann.) Für totales g ist $\mu(g)$ an einer Stelle \vec{x} genau dann undefiniert, wenn die Funktion $g_{\vec{x}}(y) = g(\vec{x}, y)$ keine Nullstellen besitzt.

8.8 DEFINITION. Die Klasse $F(\text{REK})$ der *partiell rekursiven Funktionen* ist induktiv definiert durch

- (i) $S, U_i^n, C_j^m \in F(\text{REK})$ (für $n \geq 1, 1 \leq i \leq n, m, j \geq 0$)
- (ii) Sind $g^{(m)}, h_1^{(n)}, \dots, h_m^{(n)} \in F(\text{REK})$ so auch $g(h_1, \dots, h_m)$.
- (iii) Sind $g^{(n)}, h^{(n+2)} \in F(\text{REK})$, so auch $\text{PR}(g, h)$.
- (iv) Ist $g^{(n+1)} \in F(\text{REK})$, so auch $\mu(g)$.

Ist $f \in F(\text{REK})$ total, so nennt man f (*total*) *rekursiv*.

Im Gegensatz zu $F(\text{PRIM})$ enthält $F(\text{REK})$ auch partielle Funktionen, es gilt also $F(\text{PRIM}) \subset F(\text{REK})$. Z.B. ist für $g = C_1^2$ die Funktion $f^{(1)} = \mu(g)$ nirgends definiert, da g keine Nullstellen besitzt. Später werden wir zeigen, dass es auch total rekursive Funktionen gibt, die nicht primitiv rekursiv sind.

Zum Abschluss dieses Abschnitts zeigen wir, dass die partiell rekursiven Funktionen von Registeroperatoren berechnet werden, wobei für primitiv rekursive Funktionen primitive Registeroperatoren ausreichen.

8.9 SATZ. $F(\text{REK}) \subseteq F(\text{RO})$ und $F(\text{PRIM}) \subseteq F(\text{PRO})$.

Da $F(\text{PRIM})$ die kleinste Funktionsklasse ist, die die Ausgangsfunktionen S, U_i^n und C_j^m enthält und gegen simultane Substitution und primitive Rekursion abgeschlossen ist, genügt es zum Beweis von $F(\text{PRIM}) \subseteq F(\text{PRO})$ zu zeigen, dass $F(\text{PRO})$ die Ausgangsfunktion enthält und ebenfalls diese Abschlusseigenschaften hat. Zum Nachweis von $F(\text{REK}) \subseteq F(\text{RO})$ zeigt man entsprechendes für $F(\text{RO})$ und zusätzlich den Abschluss gegen den μ -Operator. Satz 8.9 folgt also aus den nachstehenden Lemmata 8.10 – 8.13.

8.10 LEMMA. $S, U_i^n, C_j^m \in F(\text{PRO})$ ($n \geq 1; 1 \leq i \leq n; m, j \geq 0$).

BEWEIS. Die Funktionen S, U_i^n und C_j^m werden von den primitiven Registeroperatoren $TL_{1 \rightarrow 2} a_2, TL_{i \rightarrow n+1}$ bzw. $(a_{n+1})^i$ berechnet. \square

8.11 LEMMA. $F(\text{PRO})$ und $F(\text{RO})$ sind gegen simultane Substitution abgeschlossen.

BEWEIS. Seien $g^{(m)}, h_1^{(n)}, \dots, h_m^{(n)} \in F((\text{P})\text{RO})$. Wir haben

$$f^{(n)} = g(h_1, \dots, h_m) \in F((\text{P})\text{RO})$$

zu zeigen. Nach Annahme gibt es (P)ROs $P_g, P_{h_1}, \dots, P_{h_m}$ die g, h_1, \dots, h_m berechnen. Wegen Lemma 7.7 können wir annehmen, dass diese Berechnungen konservativ sind und durch Umbenennen der Register können wir folgende Registerbelegung voraussetzen:

	Eingaberegister	Ausgaberegister	Hilfsregister
P_{h_1}	$1, \dots, n$	$n+2$	$> n+m+1$
\cdot	\cdot	\cdot	\cdot
\cdot	\cdot	\cdot	\cdot
\cdot	\cdot	\cdot	\cdot
P_{h_m}	$1, \dots, n$	$n+m+1$	$> n+m+1$
P_g	$n+2, \dots, n+m+1$	$n+1$	$> n+m+1$

Um f (nicht konservativ) zu berechnen, genügt es also, diese Operatoren nacheinander anzuwenden:

$$P_f = P_{h1} \dots P_{hm} P_g.$$

Also $f \in F((P)RO)$. □

8.12 LEMMA. $F(PRO)$ und $F(RO)$ sind gegen primitive Rekursion abgeschlossen.

BEWEIS. Es gelte $g^{(n)}, h^{(n+2)} \in F((P)RO)$. Zu zeigen: $f^{(n+1)} = PR(g, h) \in F((P)RO)$. Seien P_g und P_h (primitive) Registeroperatoren die o.B.d.A. g und h mit folgenden Registerbelegungen konservativ berechnen:

	Eingaberegister	Ausgaberegister	Hilfsregister
P_g	$1, \dots, n$	$n + 2$	$> n + 5$
P_h	$1, \dots, n, n + 3, n + 2$	$n + 4$	$> n + 5$

Wie oben bereits gezeigt, lässt sich $f = PR(g, h)$ durch eine for-Schleife beschreiben, die wir durch folgenden (primitiven) RO simulieren können:

$$P_f = T_{n+1 \rightarrow n+5, n+6} P_g [s_{n+5} P_h a_{n+3} T L_{n+4 \rightarrow n+2}]_{n+5}$$

Hierbei geht der Operator P_f zur Berechnung von $f(\vec{x}, y)$ wie folgt vor. Register $n + 5$ wird zum Zählen der Schleifendurchläufe (d.h. der Rekursionsschritte mit Anwendungen von h) benutzt. Hierzu wird y zunächst in Register $n + 5$ als Zählvariable z kopiert und dann mittels P_g $f(\vec{x}, 0) = g(\vec{x})$ berechnet. So lange $z > 0$ wird dann h iteriert an der Stelle $(\vec{x}, y - z, f(\vec{x}, y - z))$ berechnet, wobei in jedem Schritt z in Register $n + 5$ dekrementiert, also $y - z$ in Register $n + 3$ inkrementiert wird. □

8.13 LEMMA. $F(RO)$ ist gegen den μ -Operator abgeschlossen.

BEWEIS. Für $g^{(n+1)} \in F(RO)$ haben wir $f^{(n)} = \mu(g) \in F(RO)$ zu zeigen. Ist P_g ein RO, der g konservativ berechnet, so berechnet

$$P_f = P_g [[s_{n+2}]_{n+2} a_{n+1} P_g]_{n+2}$$

$f = \mu(g)$, indem P_f den Operator P_g so lange für $y = 0, 1, \dots$ iteriert, bis dieser erstmals das Ergebnis 0 liefert. □

Die Umkehrung von Satz 8.9 zeigen wir in Abschnitt 10. Hierzu müssen wir zunächst im nächsten Abschnitt von einer Reihe von Hilfsfunktionen zeigen, dass diese primitiv rekursiv sind.