1.5. Rekursive Funktionen

Im vorangegangenen Abschnitt haben wir gesehen, wie Turingmaschinen zur Berechnung von Funktionen, insbesondere zahlentheoretischer Funktionen (von $\mathbb{N}^k$ nach $\mathbb{N}$) eingesetzt werden können. Wir haben auf diese Weise den Begriff der Turing-berechenbaren Funktion erhalten. Die Äquivalenz von Turingmaschinen und RAMs kann natürlich von der Akzeptierung von Sprachen auf die Berechnung von Funktion ausgedehnt werden, sodaß jede Turing-berechenbare Funktion auch von einer RAM berechnet werden kann. Wir sprechen also in Hinkunft nur mehr von berechenbaren Funktionen.

In diesem Abschnitt wollen wir einen anderen, rein mathematisch-logischen Zugang zum Begriff der berechenbaren Funktion behandeln, nämlich den Begriff der rekursiven Funktion. Wiederum kann man feststellen, daß der Begriff der berechenbaren Funktion mit demjenigen der rekursiven Funktion übereinstimmt. Somit haben wir eine weitere Bestätigung der Church'schen These vorliegen. Wir werden hier weitgehend auf Beweise der Äquivalenz von Berechenbarkeit und Rekursivität verzichten. Der Leser ist aufgefordert, sich mithilfe der Literaturliste jeweils Rechenschaft über diese Äquivalenz zu geben.

Immer, wenn wir in diesem Abschnitt von einer Funktion sprechen, so meinen wir eine zahlentheoretische Funktion im obigen Sinne.

Definition 1.5.1: Sei f eine Funktion von k Variablen, und seien $g_1, \ldots, g_k$ Funktionen von n Variablen. Sei

$$h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n)).$$

Dann entsteht h aus f und $g_1, \ldots, g_k$ durch *Komposition.* ■

Natürlich müssen die Funktionen $f, g_1, \ldots, g_k$ nicht total sein, sondern können auch partiell sein. $h(x_1, \ldots, x_n)$ ist definiert, wenn alle $z_i = g_i(x_1, \ldots, x_n)$ definiert sind, und auch $f(z_1, \ldots, z_k)$ definiert ist.

Definition 1.5.2: Sei f eine Funktion von n Variablen und g eine Funktion von $n+2$ Variablen. Sei

$$h(m, x_1, \ldots, x_n) = \begin{cases} f(x_1, \ldots, x_n), & \text{falls } m = 0, \\ g(m-1, h(m-1, x_1, \ldots, x_n), x_1, \ldots, x_n), & \text{falls } m > 0. \end{cases}$$

Dann entsteht h aus f und g durch *Rekursion.* ■

Um die Klasse der rekursiven Funktionen einzuführen, werden wir Funktionen mittels Komposition und Rekursion (und der noch einzuführenden Minimalisierung) kombinieren. Wir brauchen aber auch eine Basis von Grundfunktionen, von denen wir ausgehen können.

Definition 1.5.3: Wir betrachten folgende Funktionen:

Nullfunktion:	$\mathrm{null}() = 0$,
Nachfolgerfunktion (successor):	$\mathrm{succ}(x) = x + 1$,
Projektionsfunktionen:	$\mathrm{proj}_i^n(x_1, \ldots, x_n) = x_i,\ 1 \leq i \leq n.$

Die Funktionen $\mathrm{null}, \mathrm{succ}, \mathrm{proj}_i^n$ heißen *Grundfunktionen.* ■

Damit haben wir nun die Voraussetzungen geschaffen, um zunächst den Begriff der primitiv rekursiven Funktion einzuführen.

Definition 1.5.4: Die Klasse $\mathcal{PR}$ der *primitiv rekursiven Funktionen* ist induktiv wie folgt definiert:

(i) die Grundfunktionen gehören zu $\mathcal{PR}$,

(ii) alle Funktionen, welche aus Funktionen in $\mathcal{PR}$ durch Komposition und Rekursion erzeugt werden, gehören zu $\mathcal{PR}$.

Keine andere Funktion gehört zu $\mathcal{PR}$. ■

Beispiel 1.5.1: Viele der bekannten zahlentheoretischen Funktionen sind primitiv rekursiv. Wir weisen das für einige Funktionen nach.

(a) “$x + y$” ist primitiv rekursiv: die Additionsfunktion $f(x, y) = x + y$ ist rekursiv definiert als

$$f(0, y) = y,$$
$$f(x + 1, y) = f(x, y) + 1.$$

Diese Definition kann geschrieben werden als

$$f(x, y) = \begin{cases} \mathrm{proj}_1^1(y), & \text{falls } x = 0, \\ \mathrm{succ}(\mathrm{proj}_2^3(x - 1, f(x - 1, y), y)), & \text{falls } x > 0. \end{cases}$$

“$x+y$” entsteht also aus den Grundfunktionen durch Rekursion und Komposition.

(b) “$x \cdot y$” ist primitiv rekursiv: die Multiplikationsfunktion $g(x, y) = x \cdot y$ ist rekursiv definiert als

$$g(0, y) = 0,$$
$$g(x + 1, y) = g(x, y) + y.$$

Diese Definition kann geschrieben werden als

$$g(x,y) = \begin{cases} \text{null}(y), & \text{falls } x = 0, \\ r(x-1, g(x-1,y), y), & \text{falls } x > 0. \end{cases}$$

Dabei sei $f(x,y) = x + y$ wie oben, und

$$r(y,v,w) = f(\text{proj}_2^3(u,v,w), \text{proj}_3^3(u,v,w)).$$

Die unäre Nullfunktion $\text{null}(y) = 0$ entsteht aus null() durch Komposition (dabei ist $k = 0$ in Def. 1.5.1). "$x \cdot y$" entsteht also aus den Grundfunktionen durch Rekursion und Komposition.

(c) "$x!$" ist primitiv rekursiv: die Rekursionsgleichungen sind

$$0! = 1,$$
$$(x+1)! = x! \cdot \text{succ}(x).$$

Die Funktion $h(x) = x!$ kann also geschrieben werden als

$$h(x) = \begin{cases} \text{succ}(\text{null}()), & \text{falls } x = 0, \\ s(x-1, h(x-1)), & \text{falls } x > 0, \end{cases}$$

wobei

$$s(x_1, x_2) = \text{succ}(x_1) \cdot x_2.$$

Die Faktoriellenfunktion "$x!$" ist also primitiv rekursiv.

In den folgenden Beispielen primitiv rekursiver Funktionen überlassen wir es dem Leser zu überprüfen, daß die Rekursionsgleichungen in die exakte Gestalt gebracht werden können, wie sie von der Definition gefordert wird.

(d) "x^y": die Rekursionsgleichungen sind

$$x^0 = 1,$$
$$x^{y+1} = x^y \cdot x.$$

(e) "pred(x)": die Vorgängerfunktion pred(x) ist folgendermaßen definiert:

$$\text{pred}(x) = \begin{cases} 0, & \text{falls } x = 0, \\ x - 1, & \text{sonst.} \end{cases}$$

(f) Die modifizierte Differenz "$x \dot{-} y$" ist definiert als

$$x \dot{-} y = \begin{cases} x - y, & \text{falls } x \geq y, \\ 0, & \text{falls } x < y. \end{cases}$$

Die Rekursionsgleichungen sind

$$x \dot{-} 0 = x,$$
$$x \dot{-} (t+1) = \mathrm{pred}(x \dot{-} t).$$

(g) Die Abstandsfunktion $\mid x - y \mid$ ist definiert als Absolutwert der Differenz zwischen x und y. Sie kann dargestellt werden als

$$\mid x - y \mid = (x \dot{-} y) + (y \dot{-} x). \qquad \blacksquare$$

Unter einem *Prädikat* verstehen wir einfach eine Funktion, welche nur die Werte 0 oder 1 annimmt. Interpretieren wir 1 als "wahr" und 0 als "falsch", so erhalten wir die übliche Vorstellung von einem Prädikat als boolescher Funktion. Da also Prädikate nichts anderes als spezielle zahlentheoretische Funktionen sind, können wir problemlos von primitiv rekursiven Prädikaten sprechen.

Beispiel 1.5.2: Die folgenden Prädikate sind primitiv rekursiv.

(a) Das Prädikat istnull(x) ist definiert als

$$\mathrm{istnull}(x) = \begin{cases} 1, & \text{falls } x = 0, \\ 0, & \text{falls } x \neq 0. \end{cases}$$

Wir können auch schreiben istnull(x) $= 1 \dot{-} x$.

(b) Das Gleichheitsprädikat "$x = y$" entspricht der Funktion

$$d(x,y) = \begin{cases} 1, & \text{falls } x = y, \\ 0, & \text{falls } x \neq y. \end{cases}$$

Das Gleichheitsprädikat ist also primitiv rekursiv, da wir schreiben können

$$d(x,y) = \mathrm{istnull}(\mid x - y \mid).$$

(c) Das Prädikat "$x \leq y$" ist einfach die Funktion istnull($x \dot{-} y$). ■

Satz 1.5.1: *Sind P, Q primitiv rekursive Prädikate, dann sind auch $\neg P, P \vee Q, P \wedge Q$ primitiv rekursiv.*

Beweis: $\neg P = \mathrm{istnull}(P)$, somit ist $\neg P$ primitiv rekursiv.

Die Konjunktion ist primitiv rekursiv wegen $P \wedge Q = P \cdot Q$.

Schließlich sieht man aus der Regel von De Morgan,

$$P \vee Q \iff \neg(\neg P \wedge \neg Q),$$

daß auch die Disjunktion primitiv rekursiv ist. ∎

Satz 1.5.2: *Seien g und h primitiv rekursive Funktionen und P ein primitiv rekursives Prädikat. Sei f die folgendermaßen definierte Funktion*

$$f(x_1, \ldots, x_n) = \begin{cases} g(x_1, \ldots, x_n) & \text{falls } P(x_1, \ldots, x_n) \\ h(x_1, \ldots, x_n) & \text{sonst.} \end{cases}$$

Dann ist auch f primitiv rekursiv.

Beweis: Das sieht man sofort aus der Beziehung

$$\begin{aligned} &f(x_1, \ldots, x_n) = \\ &g(x_1, \ldots, x_n) \cdot P(x_1, \ldots, x_n) + h(x_1, \ldots, x_n) \cdot (\neg P(x_1, \ldots, x_n)). \end{aligned}$$

∎

Satz 1.5.3: *Sei $f(t, x_1, \ldots, x_n)$ eine primitiv rekursive Funktion. Dann sind auch die folgendermaßen definierten Funktionen g und h primitiv rekursiv.*

$$\begin{aligned} g(y, x_1, \ldots, x_n) &= \sum_{t=0}^{y} f(t, x_1, \ldots, x_n), \\ h(y, x_1, \ldots, x_n) &= \prod_{t=0}^{y} f(t, x_1, \ldots, x_n). \end{aligned}$$

Beweis: Es gelten die Rekursionsgleichungen

$$\begin{aligned} g(0, x_1, \ldots, x_n) &= f(0, x_1, \ldots, x_n), \\ g(t+1, x_1, \ldots, x_n) &= g(t, x_1, \ldots, x_n) + f(t+1, x_1, \ldots, x_n). \end{aligned}$$

Da die Addition primitiv rekursiv ist, ist auch g primitiv rekursiv.

Analog verfährt man mit

$$\begin{aligned} h(0, x_1, \ldots, x_n) &= f(0, x_1, \ldots, x_n), \\ h(t+1, x_1, \ldots, x_n) &= h(t, x_1, \ldots, x_n) \cdot f(t+1, x_1, \ldots, x_n). \end{aligned}$$

(Warum ist es nicht zielführend, den Beweis durch Induktion über y zu versuchen?) ∎

Satz 1.5.4: *Sei $P(t, x_1, \ldots, x_n)$ ein primitiv rekursives Prädikat. Dann sind auch die folgenden Prädikate primitiv rekursiv,*

$$(\forall t)_{\leq y} P(t, x_1, \ldots, x_n) \quad \text{und} \quad (\exists t)_{\leq y} P(t, x_1, \ldots, x_n).$$

Beweis: Das folgt sofort aus der Beobachtung

$$(\forall t)_{\leq y} P(t, x_1, \ldots, x_n) \iff \Big[\prod_{t=0}^{y} P(t, x_1, \ldots, x_n)\Big] = 1$$

und

$$(\exists t)_{\leq y} P(t, x_1, \ldots, x_n) \iff \Big[\sum_{t=0}^{y} P(t, x_1, \ldots, x_n)\Big] \neq 0. \qquad \blacksquare$$

Beispiel 1.5.3: Die folgenden Prädikate sind primitiv rekursiv.

(a) "$y \mid x$", also das Prädikat "y teilt x".

$$y \mid x \iff (\exists t)_{\leq x}(y \cdot t = x).$$

(b) "Prime(x)", also das Prädikat "x ist eine Primzahl".

$$\text{Prime}(x) \iff (x \geq 2) \wedge (\forall t)_{\leq x}\{t = 1 \vee t = x \vee \neg(t \mid x)\}. \qquad \blacksquare$$

Wie wir also gesehen haben, umfaßt die Klasse der primitiv rekursiven Funktionen eine große Anzahl der typischen zahlentheoretischen Funktionen und Prädikate. Alle diese primitiv rekursiven Funktionen sind (Turing-)berechenbar. Es gibt aber andererseits berechenbare Funktionen, welche nicht primitiv rekursiv sind. Ein klassisches Beispiel dafür ist die Ackermannfunktion

$$\begin{aligned} \text{ack}(0, y) &= y + 1, \\ \text{ack}(x, 0) &= \text{ack}(x - 1, 1), \\ \text{ack}(x, y) &= \text{ack}(x - 1, \text{ack}(x, y - 1)), \quad \text{sonst.} \end{aligned}$$

Einen Beweis dieser Tatsache findet man etwa in [DW]. Um auch solche Funktionen behandeln zu können, muß man die Klasse $\mathcal{PR}$ erweitern um das Konstruktionsprinzip der Minimalisierung. Dadurch erhält man dann die Klasse der rekursiven Funktionen, welche dann tatsächlich mit den berechenbaren Funktionen übereinstimmt.

Definition 1.5.5: Sei $P(y, x_1, \ldots, x_n)$ ein Prädikat und sei $\min_y P(y, x_1, \ldots, x_n)$ die (partielle) Funktion, welche den kleinsten Wert y ergibt, sodaß das Prädikat $P(y, x_1, \ldots, x_n)$ wahr ist, falls ein solches y existiert. Andernfalls ist $\min_y P(y, x_1, \ldots, x_n)$ undefiniert. $\blacksquare$

Definition 1.5.6: Die Klasse $\mathcal{R}$ der *rekursiven Funktionen* ist induktiv wie folgt definiert:

(i) die Grundfunktionen gehören zu $\mathcal{R}$,

(ii) alle Funktionen, welche aus Funktionen (und Prädikaten) in $\mathcal{R}$ durch Komposition, Rekursion und Minimalisierung erzeugt werden, gehören zu $\mathcal{R}$.

Keine andere Funktion gehört zu $\mathcal{R}$. ∎