

## 6. Rekursive und primitiv rekursive Funktionen

Ein maschinenunabhängiges formales Berechnungsmodell auf  
den natürlichen Zahlen

## IDEE:

Definiere eine Klasse von (partiell) berechenbaren Funktionen über  $\mathbb{N}$  *induktiv* durch Festlegung

- einer Ausgangsmenge einfacher berechenbarer Funktion
- von Operatoren, die (partiell) berechenbare Funktionen in (partiell) berechenbare Funktionen überführen.

## Ausgangsfunktionen

$S(x) = x + 1$       *Nachfolger (= Inkrementieren)*

$U_i^n(x_1, \dots, x_n) = x_i$       *n-stellige Projektion auf die  
i-te Komponente  
( $n \geq 1, 1 \leq i \leq n$ )*

$C_i^n(x_1, \dots, x_n) = i$       *n-stellige konstante Funktion  
mit Wert  $i$   
( $n, i \geq 0$ )*

## Abschlussoperationen

- (1) Simultane Substitution (Explizite Definitionen)
- (2) Primitive Rekursion (Implizite Definitionen)
- (3) Minimalisierungsoperator (Unbeschränkte Suche)

Wir betrachten zunächst nur den Abschluss unter (1) und (2) und erhalten so die **primitiv rekursiven Funktionen**.

## 6.1 Primitiv rekursive Funktionen

## Simultane Substitution

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

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

Bemerkungen:

1.  $g, h_1, \dots, h_m$  (partiell) berechenbar  $\Rightarrow g(h_1, \dots, h_m)$  (partiell) berechenbar
2. Für partielle  $g, h_1, \dots, h_m$  und  $f = g(h_1, \dots, h_m)$  gilt hierbei:

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

## BEISPIELE

1. Die Komposition 1-stelliger Funktionen ist ein Spezialfall der simultanen Substitution:

$$(g \circ h)(x) = g(h(x)) = g(h)(x)$$

2. Jedes Polynom über  $\mathbb{N}$  mit Koeffizienten aus  $\mathbb{N}$  lässt sich mit Hilfe einer endlichen Folge von simultanen Substitutionen über den Ausgangsfunktionen erweitert um die (2-stellige) Addition und Multiplikation darstellen.

Z.B. gilt für das Polynom  $p(x) = 2x^2 + 3x + 4$ :

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

Die Simultane Substitution ist ein Beispiel für ein explizites Definitionsschema. Bei einer *expliziten* Definition wird eine (neue) Funktion  $f$  durch Rückgriff auf (alte) Funktionen  $g_1, \dots, g_m$  definiert:

$$f(\vec{x}) = \text{Funktionsterm über } g_1, \dots, g_m$$

Im Gegensatz hierzu wird bei der *impliziten (rekursiven)* Definition einer Funktion  $f$  bei der Definition von  $f(\vec{x})$  auf  $f$  selbst zurückgegriffen. Hierbei ist sicherzustellen, dass die Definition nicht zirkelhaft ist. Dies kann unter anderem dadurch erreicht werden, dass nur auf den Wert von  $f(\vec{y})$  für “kleinere”  $\vec{y}$  zugegriffen wird. Ein besonders einfaches Rekursionsschema ist das der primitiven Rekursion.

## Beispiele Primitiver Rekursionen

1. Rekursive Definition der Addition mit Hilfe der Nachfolgerfunktion:

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

2. Rekursive Definition der Multiplikation mit Hilfe der Addition:

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

3. Rekursive Definition der Fakultät mit Hilfe der Multiplikation:

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

Schema:

- Die Rekursion erfolgt nach der letzten Variablen  $y$ .
- Der Funktionswert an der Stelle  $(\vec{x}, 0)$  wird explizit definiert.
- Der Funktionswert an der Stelle  $(\vec{x}, y + 1)$  wird durch Rückgriff auf den Wert an der Stelle  $(\vec{x}, y)$  definiert.

## Primitive Rekursion (formale 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

$$\forall \vec{x} \in \mathbb{N}^n (f(\vec{x}, 0) = g(\vec{x}))$$

$$\forall \vec{x} \in \mathbb{N}^n \forall y \in \mathbb{N} (f(\vec{x}, y + 1) = h(\vec{x}, y, f(\vec{x}, y)))$$

Bemerkungen:

1.  $g, h$  (partiell) berechenbar  $\Rightarrow f = \text{PR}(g, h)$  (partiell) berechenbar
2. Für partielle  $g, h$  und  $f = \text{PR}(g, h)$  gilt:

$$f(\vec{x}, 0) \downarrow \Leftrightarrow g(\vec{x}) \downarrow$$

$$f(\vec{x}, y + 1) \downarrow \Leftrightarrow [ f(\vec{x}, 0) \downarrow, \dots, f(\vec{x}, y) \downarrow \ \& \ h(\vec{x}, y, f(\vec{x}, y)) \downarrow ]$$

BEISPIEL: Addition  $f(x, y) = x + y$

Es gilt  $f^{(2)} = \text{PR}(g^{(1)}, h^{(3)})$  wobei

$$f(x, 0) = g(x) = x \Rightarrow g(x) = x$$

$$f(x, y + 1) = h(x, y, f(x, y)) = S(x + y) \Rightarrow h(x, y, z) = S(z)$$

Hierbei lassen sich  $g$  und  $h$  mit Hilfe der Ausgangsfunktionen (und einer simultanen Substitution) wie folgt darstellen:

- $g(x) = U_1^1(x)$ , d.h.  $g = U_1^1$
- $h(x, y, z) = S(z) = S(U_3^3(x, y, z)) = S(U_3^3)(x, y, z)$ , d.h.  $h = S(U_3^3)$

Es gilt also

$$+ = f = \text{PR}(U_1^1, S(U_3^3))$$

BEISPIEL: Multiplikation  $f(x, y) = x * y$

Es gilt  $f^{(2)} = \text{PR}(g^{(1)}, h^{(3)})$  wobei

$$f(x, 0) = g(x) = 0 \Rightarrow g(x) = 0$$

$$f(x, y + 1) = h(x, y, f(x, y)) = (x * y) + x \Rightarrow h(x, y, z) = z + x = +(z, x)$$

Hierbei lassen sich  $g$  und  $h$  mit Hilfe der Ausgangsfunktionen und  $+$  (und simultaner Substitutionen) wie folgt darstellen:

- $g(x) = C_0^1(x)$ , d.h.  $g = C_0^1$
- $h(x, y, z) = +(z, x) = +(U_3^3(x, y, z), U_1^3(x, y, z)) = +(U_3^3, U_1^3)(x, y, z)$ ,  
d.h.  $h = +(U_3^3, U_1^3)$

Es gilt also

$$* = f = \text{PR}(C_0^1, +(U_3^3, U_1^3))$$

## Primitiv Rekursive Funktionen

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  $\text{PR}(g, h)$ .

D.h.  $F(\text{PRIM})$  ist die kleinste Funktionsklasse, die die Ausgangsfunktionen enthält und gegen simultane Substitution und primitive Rekursion abgeschlossen ist.

Die primitiv rekursiven Funktionen wurden von Gödel 1931 in seiner berühmten Arbeit über die Unvollständigkeit formaler Systeme der Arithmetik eingeführt.



Kurt Gödel (1906 (Brünn/Brno) - 1954 (Princeton) )

<http://www.time.com/time/time100/scientist/profile/godel.html>

## Bemerkungen

1. Die letzten Beispiele zeigen, dass  $+$ ,  $*$  und  $!$  primitiv rekursiv sind.
2. Primitiv rekursive Funktionen sind total! (Induktion nach dem Aufbau)
3. Wir werden zeigen, dass die primitiv rekursiven Funktionen genau diejenigen Funktionen sind, die von *primitiven* Registeroperatoren berechnet werden.
4. Es gibt totale berechenbare Funktionen, die sich mit Hilfe komplexerer Rekursionsschemata berechnen lassen, die nicht primitiv rekursiv sind. Ein Beispiel für solch eine Funktion ist die Ackermann-Funktion.

# Die Ackermann-Funktion

Die Ackermann-Funktion  $\alpha : \mathbb{N}^2 \rightarrow \mathbb{N}$  ist durch folgende geschachtelte Rekursion definiert:

- $\alpha(0, y) = y + 1$
- $\alpha(x + 1, 0) = \alpha(x, 1)$
- $\alpha(x + 1, y + 1) = \alpha(x, \alpha(x + 1, y))$

Während bei einer primitiven Rekursion, die letzte Variable die Rekursionsvariable ist, sind hier beide Variablen  $x$  und  $y$  Rekursionsvariablen.

LEMMA. Die Ackermann-Funktion ist wohldefiniert, total und berechenbar.

Dies zeigt man durch Hauptinduktion nach  $x$  und Nebeninduktion nach  $y$ .

## Die Ackermann-Funktion (Beispiel)

Wir berechnen  $\alpha(2, 1)$  mit Hilfe der Ersetzungsregeln

(i)  $\alpha(0, y) \rightarrow y + 1$

(ii)  $\alpha(x + 1, 0) \rightarrow \alpha(x, 1)$

(iii)  $\alpha(x + 1, y + 1) \rightarrow \alpha(x, \alpha(x + 1, y))$

$$\begin{aligned} (2, 1) &= (1, (2, 0)) && (iii) \\ &= (1, (1, 1)) && (ii) \\ &= (1, (0, (1, 0))) && (iii) \\ &= (1, (0, (0, 1))) && (ii) \\ &= (1, (0, 2)) && (i) \\ &= (1, 3) && (i) \\ &= (0, (1, 2)) && (iii) \\ &= (0, (0, (1, 1))) && (iii) \\ &= (0, (0, (0, (1, 0)))) && (iii) \\ &= (0, (0, (0, (0, 1)))) && (ii) \\ &= (0, (0, (0, 2))) && (i) \\ &= (0, (0, 3)) && (i) \\ &= (0, 4) && (i) \\ &= 5 && (i) \end{aligned}$$

## Die Ackermann-Funktion (Fortsetzung)

Jeder Zweig  $\alpha_x : \mathbb{N} \rightarrow \mathbb{N}$  der Ackermann-Funktion, wobei  $\alpha_x(y) := \alpha(x, y)$ , ist primitiv rekursiv. (Beweis durch Induktion nach  $x$ .) Die Anzahl der primitiven Rekursionen, die man zur Darstellung von  $\alpha_x$  benötigt, wächst jedoch mit  $x$ .

Mit wachsendem  $x$  machen die Zweige  $\alpha_x$  der Ackermann-Funktion jeweils einen "Wachstumssprung":

- $\alpha_1(y) = y + 2$
- $\alpha_2(y) = 2y + 3$
- $\alpha_3(y) = 2^{y+3} - 3$
- ...

Dies benutzt man, um zu zeigen, dass die Diagonale  $\hat{\alpha}(y) := \alpha(y, y)$  von  $\alpha$  schneller wächst als jede primitiv rekursive Funktion. (Siehe z.B. A. Oberschelp: Rekursionstheorie, BI Wissenschaftsverlag, 1993)

Die Klasse der primitiv rekursiven Funktionen enthält sämtliche berechenbaren Funktionen, auf die man in den üblichen Anwendungen stösst. Das Beispiel der Ackermann-Funktion zeigt jedoch, dass die Klasse der primitiv rekursiven Funktionen nicht alle berechenbaren Funktionen umfasst. (Man kann aber zeigen, dass für Funktionen, die berechenbar aber nicht primitiv rekursiv sind, wie bei der Ackermann-Funktion die Berechnung so zeit-aufwändig ist, dass diese für grössere Eingaben praktisch nicht mehr ausführbar ist.)

Um alle berechenbaren Funktionen zu erhalten, benötigen wir eine weitere Abschlussoperation, die partiell berechenbare Funktionen wiederum auf partiell berechenbare Funktionen abbildet: die Minimalisierung. Durch diese können jedoch totale (d.h. berechenbare) Funktionen auf partielle (d.h. partiell berechenbare) Funktionen abgebildet werden.

## Der Minimalisierungsoperator ( $\mu$ -Operator)

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

$$f^{(n)} = \mu(g)$$

ist definiert durch

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

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

Der  $\mu$ -Operator wird auch *Minimalisierungs-Operator* genannt. Man spricht auch von einem *Suchoperator*, da ja die kleinste Nullstelle gesucht wird.

## BEMERKUNGEN

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 Definiiertheitsforderung der derart modifizierte  $\mu$ -Operator  $\hat{\mu}$  eine partiell berechenbare Funktion  $g$  auf eine **nicht**berechenbare partielle Funktion  $\hat{\mu}(g)$  abbilden kann.)

## Die Partiell Rekursiven Funktionen

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*.

## BEMERKUNGEN

1. 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.
2. Später werden wir zeigen, dass es auch total rekursive Funktionen gibt, die nicht primitiv rekursiv sind. Die Ackermann-Funktion ist ein Beispiel für solch eine Funktion.

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  $C_i^n$  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 Ausgangsfunktionen enthält und ebenfalls diese Abschlusseigenschaften besitzt. Zum Nachweis von  $F(\text{REK}) \subseteq F(\text{RO})$  zeigt man Entsprechendes für  $F(\text{RO})$  und zusätzlich den Abschluss gegen den  $\mu$ -Operator.

Um dies zu zeigen, beobachtet man, dass man unter Verwendung von Registertransfers die Ausgangsfunktionen mit Hilfe der elementaren ROs darstellen kann, die simultane Substitution mit Hilfe der Hintereinanderausführung von Operatoren, die primitive Rekursion mit Hilfe der beschränkten Iteration (for-Schleife) und den  $\mu$ -Operator mit Hilfe der allgemeinen Iteration (while-Schleife) von ROs.

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

BEWEIS. Es gilt

$$S = TL_{1 \rightarrow 2} a_2$$

$$U_i^n = TL_{i \rightarrow n+1}$$

$$C_i^n = (a_{n+1})^i$$

LEMMA 2.  $F(\text{PRO})$  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  $(\text{P})\text{ROs}$   $P_g, P_{h_1}, \dots, P_{h_m}$  die  $g, h_1, \dots, h_m$  (o.B.d.A. konservativ) berechnen. Umbenennen der Register liefert:

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

Es folgt:  $P_f = P_{h_1} \dots P_{h_m} P_g$  berechnet  $f$ .

LEMMA 3.  $F(\text{PRO})$  und  $F(\text{RO})$  sind gegen primitive Rekursion abgeschlossen.

BEWEIS. Es gelte  $g^{(n)}, h^{(n+2)} \in F((\text{P})\text{RO})$ . Zu zeigen:  $f^{(n+1)} = \text{PR}(g, h) \in F((\text{P})\text{RO})$ . 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  $f = \text{PR}(g, h)$  durch eine for-Schleife beschreiben, die wir durch folgenden (primitiven) RO simulieren können:

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

(NB: 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.)

LEMMA 4.  $F(\text{RO})$  ist gegen den  $\mu$ -Operator abgeschlossen.

BEWEIS. Für  $g^{(n+1)} \in F(\text{RO})$  haben wir  $f^{(n)} = \mu(g) \in F(\text{RO})$  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 Eingabe  $y = 0, 1, \dots$  iteriert, bis dieser erstmals das Ergebnis 0 liefert.

Damit ist der Satz bewiesen.

Nächstes Ziel

ÄQUIVALENZSATZ:

$$F(\text{REK}) = F(\text{RO}) = F(\text{RM}) = F(\text{TO}) = F(\text{TM}) = F_k(\text{TM})$$

KLEINER ÄQUIVALENZSATZ:

$$F(\text{PRIM}) = F(\text{PRO})$$

Bereits gezeigt:

- (1)  $F(\text{REK}) \subseteq F(\text{RO}) \subseteq F(\text{TO}) \subseteq F(\text{TM}) = F_k(\text{TM})$
- (2)  $F(\text{RO}) \subseteq F(\text{RM}) \subseteq F_k(\text{TM})$   
(hiervon wurde die 2. Inklusion in den Übungen gezeigt)
- (3)  $F(\text{PRIM}) \subseteq F(\text{PRO})$

Zu zeigen bleibt:

$$F(\text{TM}) \subseteq F(\text{REK}) \quad \text{und} \quad F(\text{PRO}) \subseteq F(\text{PRIM})$$

Zur Vorbereitung geben wir als nächstes eine Reihe von Beispielen von primitiv rekursiven Funktionen an und zeigen weitere Abschlusseigenschaften dieser Funktionsklasse.