1.5 LOOP-Berechenbarkeit

LOOP-Programme sind wie folgt definiert:

Variablen: $x_1, x_2, x_3, \ldots$

Konstanten: 0, 1, 2, . . .

Trennsymbole: ;

Operatoren: $+$ - $:=$

Schlüsselwörter: LOOP DO END

Der Aufbau von LOOP-Programmen:

- $x_i := c$, $x_i := x_j + c$, $x_i := x_j - c$ sind LOOP-Programme.
  Die Interpretation dieser Ausdrücke erfolgt, wie üblich, mit der Einschränkung, dass $x_j - c$ als 0 gewertet wird, falls $c > x_j$.
- Sind P_1 und P_2 LOOP-Programme, so ist auch
  $$P_1; P_2$$
  ein LOOP-Programm.
  Interpretation: Führe zuerst P_1 und dann P_2 aus.
- Ist P ein LOOP-Programm, so ist auch
  $$\text{LOOP } x_i \text{ DO } P \text{ END}$$
  ein LOOP-Programm.
  Interpretation: Führe P genau x_i-mal aus.
  Achtung: Zuweisungen an x_i im Innern von P haben keinen Einfluss auf die Anzahl der Schleifendurchläufe!

Definition 143

Eine Funktion f heißt LOOP-berechenbar genau dann, wenn es ein LOOP-Programm gibt, das f berechnet.

LOOP-Programme können IF ... THEN ... ELSE ... END Konstrukte simulieren. Der Ausdruck IF $x = 0$ THEN A END kann durch folgendes Programm nachgebildet werden:

$y := 1$;
LOOP x DO $y := 0$ END;
LOOP y DO A END;

LOOP-berechenbare Funktion sind immer total, denn: LOOP-Programme stoppen immer. Damit stellt sich natürlich die Frage, ob alle totalen Funktionen LOOP-berechenbar sind.

Satz 144

f ist primitiv-rekursiv $\Longleftrightarrow$ f ist LOOP-berechenbar.

Beweis:

Wir zeigen zunächst „$\Leftarrow$“: Sei also P ein LOOP-Programm, das $f : \mathbb{N}_0^n \to \mathbb{N}_0$ berechnet. P verwende die Variablen $x_1, \ldots, x_k$, $k \geq n$.

Zu zeigen: f ist primitiv-rekursiv.

Der Beweis erfolgt durch strukturelle Induktion über den Aufbau von P.

Beweis:

Induktionsanfang: $P : x_i := x_j \pm c$

Wir zeigen: Es gibt eine primitiv-rekursive Funktion

$$\underbrace{g_P(\langle a_1, \ldots, a_k \rangle)}_{\text{Belegung der Variablen beim Start von } P} = \underbrace{\langle b_1, \ldots, b_k \rangle}_{\text{Belegung der Variablen am Ende von } P}$$

Für $P : x_i := x_j \pm c$ erhält man:

$$g_P(\langle a_1, \ldots, a_k \rangle) = \langle a_1, \ldots, a_{i-1}, a_j \pm c, a_{i+1}, \ldots, a_k \rangle$$

Beweis:

Induktionsschritt: Hier unterscheiden wir 2 Fälle:

1. Sei $P: Q; R$. Dann ist $g_P(x) = g_R(g_Q(x))$.
2. Sei nun P : LOOP x_i DO Q END.
   Idee: Definiere Funktion $h(n, x)$, die die Belegung der Variablen berechnet, wenn man mit Belegung x startet und dann Q genau n mal ausführt. Dann ist:

$$h(0, x) = x$$
$$h(n+1, x) = g_Q(h(n, x))$$

   und damit $g_P(x) = h(d_i(x), x)$, wobei d_i die i-te Umkehrfunktion von $\langle x_1, \ldots, x_k \rangle$ ist, also $d_i(\langle x_1, \ldots, x_k \rangle) = x_i$.

Die Richtung „⇒“ wird durch strukturelle Induktion über den Aufbau von f gezeigt (Übungsaufgabe). $\square$

Definition 145

Die Ackermann-Funktion $a : \mathbb{N}_0^2 \to \mathbb{N}_0$:

$$a(x,y) := \begin{cases} y+1 & \text{falls } x = 0 \\ a(x-1,1) & \text{falls } x \geq 1, y = 0 \\ a(x-1, a(x, y-1)) & \text{falls } x, y \geq 1 \end{cases}$$

Einige Eigenschaften der Ackermann-Funktion, die man per Induktion zeigen kann:

1. $a(1,y) = y+2$, $a(2,y) = 2y+3$ für alle y
2. $y < a(x,y) \qquad \forall x, y$
3. $a(x,y) < a(x, y+1) \qquad \forall x, y$
4. $a(x, y+1) \leq a(x+1, y) \qquad \forall x, y$
5. $a(x,y) < a(x+1, y) \qquad \forall x, y$

Genauer:

$$
\begin{array}{cc|cccccc}
 & & & & \mathbf{y} \rightarrow & & & \\
 & & 0 & 1 & 2 & 3 & 4 & 5 \\
\hline
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\mathbf{x} & 2 & 3 & 5 & 7 & 9 & 11 & 13 \\
\downarrow & 3 & 5 & 13 & 29 & 61 & 125 & 253 \\
 & 4 & 13 & 65533 & 2^{65536}-3 & 2^{2^{65536}}-3 & 2^{2^{2^{65536}}}-3 & \ldots \\
 & 5 & 65533 & \ldots & & & & \\
\end{array}
$$

Lemma 146

Sei P ein LOOP-Programm mit den Variablen $x_1, \ldots, x_k$, und sei

$$f_P(n) = \max\{\sum_i n'_i;\ \sum_i n_i \leq n\},$$

wobei, für $i = 1, \ldots, k$, n'_i der Wert von x_i nach Beendigung von P und n_i der Startwert von x_i vor Beginn von P ist. Dann gibt es ein $t \in \mathbb{N}$, so dass

$$\forall n \in \mathbb{N}_0 : f_P(n) < a(t, n).$$

Beweis:

durch Induktion über den Aufbau von P.

Induktionsanfang:

$P = \quad x_i := x_j \pm c$, o.E. $c \in \{0, 1\}$. Es ist klar, dass

$$f_P(n) \leq 2n + 1 < 2n + 3 = a(2, n) \text{ für alle } n$$

$\Rightarrow t = 2$ tut es!

Beweis:

durch Induktion über den Aufbau von P.

Induktionsschritt:

1. Fall: $P = P_1; P_2$. Nach Induktionsannahme gibt es $k_1, k_2 \in \mathbb{N}$, so dass für $i = 1, 2$ und für alle $n \in \mathbb{N}_0$ $f_{P_i} < a(k_i, n)$.

Damit

$$
\begin{aligned}
f_P(n) &\leq f_{P_2}(f_{P_1}(n)) & \\
&\leq f_{P_2}(a(k_1, n)) & \text{Monotonie von } f_{P_2} \\
&< a(k_2, a(k_1, n)) & \text{I.A., setze } k_3 := \max\{k_2, k_1 - 1\}. \\
&\leq a(k_3, a(k_3 + 1, n)) & \text{Monotonie!} \\
&= a(k_3 + 1, n + 1) & \\
&\leq a(k_3 + 2, n) & \text{Eigenschaft 4 der Ackermannfkt.}
\end{aligned}
$$

und $t = k_3 + 2$ tut es hier!

Beweis:

durch Induktion über den Aufbau von P.

Induktionsschritt:

2. Fall: $P =$ LOOP x_i DO Q END.

Die Abschätzung erfolgt analog zum 1. Fall und wird als Übungsaufgabe überlassen.

$\square$

Satz 147

Die Ackermann-Funktion ist nicht LOOP-berechenbar.

Beweis:

Angenommen doch. Sei P ein zugehöriges LOOP-Programm, das

$$g(n) := a(n, n)$$

berechnet.
Nach Definition von f_P gilt $g(n) \leq f_P(n)$ für alle $n \in \mathbb{N}_0$.

Wähle gemäß dem obigen Lemma t mit $f_P(\cdot) < a(t, \cdot)$ und setze $n = t$:

$$f_P(t) < a(t, t) = g(t) \leq f_P(t)$$

$\Rightarrow$ Widerspruch!

Korollar 148

Die primitiv-rekursiven Funktionen sind eine echte Teilklasse der berechenbaren totalen Funktionen.

Beweis:

Die Ackermannfunktion ist total, berechenbar und nicht primitiv-rekursiv.

1.6 μ-rekursive Funktionen

Für eine Funktion f liefert der so genannte μ-Operator das kleinste Argument, für das f gleich 0 wird (falls ein solches existiert). Der μ-Operator gestattet so z.B., vorab die Anzahl der Durchläufe einer WHILE-Schleife zu bestimmen, bis die Laufvariable gleich 0 wird.

Definition 149

Sei f eine (nicht notwendigerweise totale) $k+1$-stellige Funktion. Die durch Anwendung des μ-Operators entstehende Funktion f_μ ist definiert durch:

$$
\begin{aligned}
f_\mu : \mathbb{N}_0^k &\to \mathbb{N}_0 \\
(x_1, \ldots, x_k) &\mapsto \begin{cases} \min\{n \in \mathbb{N}_0;\ f(n, x_1, \ldots, x_k) = 0\} & \text{falls} \\ \qquad \text{dieses } n \text{ existiert und } f(m, x_1, \ldots, x_k) & \\ \qquad \text{für alle } m \le n \text{ definiert ist;} & \\ \perp \text{ (undefiniert)} & \text{sonst} \end{cases}
\end{aligned}
$$

Definition 150

Die Klasse der μ-rekursiven Funktionen ist die kleinste Klasse von (nicht notwendigerweise totalen) Funktionen, die die Basisfunktionen (konstante Funktionen, Nachfolgerfunktion, Projektionen) enthält und alle Funktionen, die man hieraus durch (evtl. wiederholte) Anwendung von Komposition, primitiver Rekursion und/oder des μ-Operators gewinnen kann.

Satz 151

f μ-rekursiv $\Longleftrightarrow$ f WHILE-berechenbar.

Beweis:

Der Beweis ist elementar.