9a

Primitiv und μ-rekursive Funktionen

Prof. Dr. Jasmin Blanchette

Lehr- und Forschungseinheit für
Theoretische Informatik und Theorembeweisen

Stand: 11. Februar 2025
Basierend auf Folien von PD Dr. David Sabel

Weitere Formalismen

Weitere Formalismen zur Definition der Berechenbarkeit:

- primitiv rekursive Funktionen
- μ-rekursive Funktionen.

Weitere Formalismen

Weitere Formalismen zur Definition der Berechenbarkeit:

- primitiv rekursive Funktionen
- μ-rekursive Funktionen.

Im Skript wird gezeigt:

- Primitiv rekursive Funktionen entsprechen genau den sogenannten LOOP-berechenbaren Funktionen.
- μ-rekursive Funktionen entsprechen genau den turingberechenbaren Funktionen.

Primitiv rekursive Funktionen

Definition

Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ ist primitiv rekursiv, wenn sie der folgenden Definition genügt:

- Jede konstante Funktion $f(x_1, \ldots, x_k) = c \in \mathbb{N}$ ist primitiv rekursiv.
- Die Projektionsfunktionen $\pi_i^k(x_1, \ldots, x_k) = x_i$ sind primitiv rekursiv.
- Die Nachfolgerfunktion $succ(x) = x + 1$ ist primitiv rekursiv.
- Komposition/Einsetzung: Wenn $g : \mathbb{N}^m \to \mathbb{N}$ und für $i = 1, \ldots, m$: $h_i : \mathbb{N}^k \to \mathbb{N}$ primitiv rekursiv sind, dann ist auch f mit $f(x_1, \ldots, x_k) = g(h_1(x_1, \ldots, x_k), \ldots, h_m(x_1, \ldots, x_k))$ primitiv rekursiv.

(Fortsetzung folgt.)

Primitiv rekursive Funktionen

Definition

- Rekursion: Wenn $g : \mathbb{N}^{k-1} \to \mathbb{N}$ und $h : \mathbb{N}^{k+1} \to \mathbb{N}$ primitiv rekursiv sind, dann ist auch f mit

$$f(x_1, \ldots, x_k) = \begin{cases} g(x_2, \ldots, x_k) & \text{falls } x_1 = 0 \\ h(f(x_1 - 1, x_2, \ldots, x_k), x_1 - 1, x_2, \ldots, x_k) & \text{sonst} \end{cases}$$

  primitiv rekursiv.

Additionsfunktion

$add(x_1, x_2) = x_1 + x_2$ ist primitiv rekursiv:

$$add(x_1, x_2) = \begin{cases} x_2 & \text{falls } x_1 = 0 \\ succ(add(x_1 - 1, x_2)) & \text{sonst} \end{cases}$$

Grundgedanke: x_1-mal 1 zu x_2 addieren.

Additionsfunktion

$add(x_1, x_2) = x_1 + x_2$ ist primitiv rekursiv:

$$add(x_1, x_2) = \begin{cases} x_2 & \text{falls } x_1 = 0 \\ succ(add(x_1 - 1, x_2)) & \text{sonst} \end{cases}$$

Grundgedanke: x_1-mal 1 zu x_2 addieren.

Die verwendeten Funktionen g und h aus der Definition der primitiv rekursiven Funktionen sind hier

- $g = \pi_1^1$
- $h(x_1, x_2, x_3) = succ(\pi_1^3(x_1, x_2, x_3))$

Komponenten eines Tupels entfernen/vertauschen/vervielfachen

Wenn $g : \mathbb{N}^4 \to \mathbb{N}$ primitiv rekursiv ist, dann ist auch z.B. $f : \mathbb{N}^3 \to \mathbb{N}$ mit

$$f(n_1, n_2, n_3) = g(n_2, n_3, n_3, n_2)$$

denn

$$f(n_1, n_2, n_3) = g(\pi^3_2(n_1, n_2, n_3), \pi^3_3(n_1, n_2, n_3), \pi^3_3(n_1, n_2, n_3), \pi^3_2(n_1, n_2, n_3))$$

Multiplikationsfunktion

$mult(x_1, x_2) = x_1 \cdot x_2$ ist primitiv rekursiv:

$$mult(x_1, x_2) = \begin{cases} 0 & \text{falls } x_1 = 0 \\ add(mult(x_1 - 1, x_2), x_2) & \text{sonst} \end{cases}$$

Grundgedanke: x_1-mal x_2 zu 0 addieren.

Rekursion durch das *i*-te Argument

Für $1 \leq i \leq k$ kann man

$$f(x_1, \ldots, x_k) = \begin{cases} g(x_1, \ldots, \boldsymbol{x_{i-1}}, \boldsymbol{x_{i+1}}, x_k), & \text{falls } x_i = 0 \\ h(f(x_1, \ldots, \boldsymbol{x_{i-1}}, \boldsymbol{x_i} - 1, \boldsymbol{x_{i+1}}, \ldots, x_k), x_1, \ldots, \boldsymbol{x_{i-1}}, \boldsymbol{x_i} - 1, \boldsymbol{x_{i+1}}, \ldots, x_k), & \text{sonst} \end{cases}$$

durch $f(x_1, \ldots, x_k) = f'(x_i, x_1 \ldots, x_{i-1}, x_{i+1}, \ldots, x_k)$ darstellen, wobei

$$\begin{aligned} f'(y_1, \ldots, y_k) &= \begin{cases} g(y_2, \ldots, y_k) & \text{falls } y_1 = 0 \\ h'(f'(y_1 - 1, y_2 \ldots, y_k), y_1 - 1, y_2, \ldots, y_k) & \text{sonst} \end{cases} \\ h'(y_0, \ldots, y_k) &= h(y_0, y_2, y_3, \ldots, y_i, y_1, y_{i+1}, y_{i+2}, \ldots, y_k) \end{aligned}$$

Angepasste Differenz

Im Allgemeinen ist $x_1 - x_2$ nicht primitiv rekursiv, weil der undefinierte Fall $x_1 < x_2$ nicht darstellbar ist.

Angepasste Differenz

Im Allgemeinen ist $x_1 - x_2$ nicht primitiv rekursiv, weil der undefinierte Fall $x_1 < x_2$ nicht darstellbar ist.

Hingegen ist die angepasste Differenz, die 0 liefert falls $x_1 < x_2$, primitiv rekursiv:

$$sub(x_1, x_2) = \begin{cases} x_1 & \text{falls } x_2 = 0 \\ pred(sub(x_1, x_2 - 1)) & \text{sonst} \end{cases}$$

wobei

$$pred(x_1) = \begin{cases} 0 & \text{falls } x_1 = 0 \\ x_1 - 1 & \text{sonst} \end{cases}$$

μ-Operator

Definition

Sei $h : \mathbb{N}^{k+1} \to \mathbb{N}$ eine (partielle oder totale) Funktion.
Dann ist $\mu h : \mathbb{N}^k \to \mathbb{N}$ definiert als

$$(\mu h)(x_1, \ldots, x_k) = \begin{cases} n & \text{falls } h(n, x_1, \ldots, x_k) = 0 \text{ und für} \\ & \text{alle } m < n\text{: } h(m, x_1, \ldots, x_k) \text{ ist definiert} \\ & \text{und } h(m, x_1, \ldots, x_k) > 0 \\ \text{undefiniert} & \text{sonst} \end{cases}$$

μ-Operator

Definition

Sei $h : \mathbb{N}^{k+1} \to \mathbb{N}$ eine (partielle oder totale) Funktion.
Dann ist $\mu h : \mathbb{N}^k \to \mathbb{N}$ definiert als

$$(\mu h)(x_1, \dots, x_k) = \begin{cases} n & \text{falls } h(n, x_1, \dots, x_k) = 0 \text{ und für} \\ & \text{alle } m < n\text{: } h(m, x_1, \dots, x_k) \text{ ist definiert} \\ & \text{und } h(m, x_1, \dots, x_k) > 0 \\ \text{undefiniert} & \text{sonst} \end{cases}$$

Der μ-Operator „sucht" nach der ersten Nullstelle von h.

Wenn diese nicht existiert (entweder da h keine Nullstelle hat, oder da h undefiniert ist für Werte, die kleiner als die Nullstelle sind), dann ist auch μh undefiniert.

Beispiel für den μ-Operator

Sei $bus(x_1, x_2) = sub(x_2, x_1)$, wobei sub die angepasste Differenz berechnet.

$(\mu bus)(5) =$?

Beispiel für den μ-Operator

Sei $bus(x_1, x_2) = sub(x_2, x_1)$, wobei sub die angepasste Differenz berechnet.

$(\mu bus)(5) =$ 5 weil
$bus(0, 5) = 5$
$bus(1, 5) = 4$
$bus(2, 5) = 3$
$bus(3, 5) = 2$
$bus(4, 5) = 1$
$bus(5, 5) = 0.$

μ-rekursive Funktionen

Definition

Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ ist μ-rekursiv, wenn sie der folgenden Definition genügt:

- Jede konstante Funktion $f(x_1, \ldots, x_k) = c \in \mathbb{N}$ ist μ-rekursiv.
- Die Projektionsfunktionen $\pi_i^k(x_1, \ldots, x_k) = x_i$ sind μ-rekursiv.
- Die Nachfolgerfunktion $succ(x) = x + 1$ ist μ-rekursiv.
- Komposition/Einsetzung: Wenn $g : \mathbb{N}^m \to \mathbb{N}$ und für $i = 1, \ldots, m$: $h_i : \mathbb{N}^k \to \mathbb{N}$ μ-rekursiv sind, dann ist auch f mit $f(x_1, \ldots, x_k) = g(h_1(x_1, \ldots, x_k), \ldots, h_m(x_1, \ldots, x_k))$ μ-rekursiv.

(Fortsetzung folgt.)

μ-rekursive Funktionen

Definition

- Rekursion: Wenn $g : \mathbb{N}^{k-1} \to \mathbb{N}$ und $h : \mathbb{N}^{k+1} \to \mathbb{N}$ μ-rekursiv sind, dann ist

$$f(x_1, \ldots, x_k) = \begin{cases} g(x_2, \ldots, x_k) & \text{falls } x_1 = 0 \\ h(f(x_1 - 1, x_2, \ldots, x_k), x_1 - 1, x_2, \ldots, x_k) & \text{sonst} \end{cases}$$

  auch μ-rekursiv.
- μ-Operator: Wenn $h : \mathbb{N}^{k+1} \to \mathbb{N}$ μ-rekursiv ist, dann ist auch $f = \mu h$ μ-rekursiv.

Überblick über die Berechenbarkeitsformalismen im Skript

$$
\begin{tikzcd}
\text{turingberechenbar} \arrow[rr, Rightarrow, "\text{Satz 10.5.2}"] & & \text{GOTO-berechenbar} \arrow[dll, Rightarrow, "\text{Theorem 10.4.1}"] \\
\text{WHILE-berechenbar} \arrow[u, Rightarrow, "\text{Theorem 10.2.4}"] \arrow[rr, Leftrightarrow, "\text{Theorem 11.2.4}"] & & \mu\text{-rekursiv} \\
\text{LOOP-berechenbar} \arrow[u, Rightarrow, "\text{Satz 10.2.3}"] \arrow[rr, Leftrightarrow, "\text{Theorem 11.1.8}"] & & \text{primitiv rekursiv}
\end{tikzcd}
$$