μ-rekursive Funktionen

Sei $k \in \mathbb{N}_0$

Gibt es μ-rekursive Funktionen

Then : $\mathbb{N}_0^k \to \mathbb{N}_0$

und Else : $\mathbb{N}_0^k \to \mathbb{N}_0$

so ist auch die Funktion

isZero : $\mathbb{N}_0^{k+1} \to \mathbb{N}_0$ μ-rekursiv.

„isZero" soll überprüfen, ob der erste Parameter gleich 0 ist. Ist dies der Fall, so gibt die Funktion den Wert von $Then(z_1, \dots, z_k)$ zurück. Ansonsten den Wert von $Else(z_1, \dots, z_k)$.

$$\underline{isZero}(x, z_1, z_2, \dots z_k) :=$$

5. Induktion →

$$\begin{cases} Then(z_1, \dots, z_k) & \text{falls } x=0 \\ Else \circ (\pi_3^{k+2}, \dots, \pi_{k+2}^{k+2})(\underline{isZero}(& \text{falls } x>0 \end{cases}$$

4. Komposition →

$$x-1, z_1, \dots, z_k), \underline{x-1, z_1, \dots, z_k})$$

Diese 2 Argumente werden von der Funktion „ignoriert."

Diese Argumente werden der Else-Funktion übergeben.

Sei $k \geq 2$

isEqual : $\mathbb{N}_0^k \to \mathbb{N}_0$

Voraussetzungen:

- isZero ist definiert : $\mathbb{N}_0^{k+1} \to \mathbb{N}_0$
- Then / Else sind μ-rekursiv definiert : $\mathbb{N}_0^k \to \mathbb{N}_0$

$$\underline{isEqual} := isZero \circ (Max \circ (Monus \circ (\pi_1^k, \pi_2^k), Monus \circ (\pi_2^k, \pi_1^k)), \pi_1^k, \dots, \pi_k^k)$$

Übersetzung in Pseudocode: $\dot{-} \hat{=}$ Monus

```
int isEqual(x, y, z3, ..., zk):
    if (Max((x ∸ y), (y ∸ x)) == 0):
        return Then(x, y, z3, ..., zk)
    else:
        return Else(x, y, z3, ..., zk)
```

Sei $k \geq 2$

isGreater: $\mathbb{N}_0^k \to \mathbb{N}_0$

gleiche Vorraussetzungen wie isEqual.

1. Nachfolger

$$\text{isGreater} := \text{isZero} \circ (\text{Monus} \circ (\sigma \circ (\pi_2^k), \pi_1^k), \pi_1^k, \ldots, \pi_k^k)$$

Übersetzung in Pseudocode:

```
int isGreater(x, y, z3, ..., zk):
    if (((y+1) ∸ x) == 0):
        return Then(x, y, z3, ..., zk)
    else:
        return Else(x, y, z3, ..., zk)
```

Wenn man diese Funktionen noch um ein paar mehr erweitert, so erhält man einen Funktionssatz der einer Programmier-sprache ähnlich ist.
Somit lässt sich die Aufgabe der Althlauser auch auf eine andere - unschöne - Weise lösen, als in der Musterlösung gezeigt:

Pseudocode:

```
int Max3(x, y, z):
    if (x > y):
        if(x>z):
            return x
        else:
            return z
    else:
        if(y>z):
            return y
        else:
            return z
```

$Max3 := isGreater$
mit $Then := isGreater \circ (\pi^3_1, \pi^3_3, \pi^3_2)$
mit $Then := \pi^3_1$
und $Else := \pi^3_2$
und $Else := isGreater \circ (\pi^3_2, \pi^3_3, \pi^3_1)$
mit $Then := \pi^3_1$
und $Else := \pi^3_2$

Natürlich sollen alle isGreater-Funktionen unterschiedliche Then und Else Funktionen benutzen.