Semantik:

- Jede Variable kann eine natürliche Zahl aufnehmen (idealisiert: beliebig groß).
- Zuweisung: wie üblich.
- ";": wie üblich (Nacheinanderausführung).
- **loop** x **do** P **enddo** : P wird so oft nacheinander ausgeführt, wie der Wert der Variablen x zu Beginn der Schleife angibt.

Bemerkungen

1. Die Programmiersprache LOOP ist sehr einfach, mächtigere Konstruktionen sind programmierbar, z.B. (x, y, z Variablen, $c \in \mathbb{N}_0$):
   - $x_1 := c$: $\underbrace{x_1 := 0;\ x_1 := x_1 + 1;\ \ldots;\ x_1 := x_1 + 1}_{c-\text{mal}}$
   - $x_1 := x_1 \dot{-} 1$: $y := 0$; $z := 0$; **loop** x_1 **do** $y := z$; $z := z + 1$ **enddo**; $x_1 := y$
   - $x_1 := x_2 \dot{-} c$: $x_3 := c$; $x_1 := x_2$; **loop** x_3 **do** $x_1 := x_1 \dot{-} 1$ **enddo**
   - $x_1 := x_2 + x_3$: $x_1 := x_2$; **loop** x_3 **do** $x_1 := x_1 + 1$ **enddo**
   - **if** $y = 0$ **then** P_1 **else** P_2 **endif** :
     $z_1 := 1 \dot{-} y$; $z_2 := 1 \dot{-} z_1$; **loop** z_1 **do** P_1 **enddo**; **loop** z_2 **do** P_2 **enddo**
2. Jedes LOOP-Programm stoppt nach endlicher Zeit.

Definition. Eine Funktion $f : \mathbb{N}_0^k \to \mathbb{N}_0$ heißt LOOP-***berechenbar***, wenn es ein LOOP-Programm P gibt, das f in folgendem Sinn berechnet: Wird P in einem Zustand gestartet, in dem die Variablen $x_1, \ldots, x_k$ mit $n_1, \ldots, n_k$ und alle übrigen Variablen mit 0 belegt sind, so hat die Variable x_0 nach Beendigung des Programms den Wert $f(n_1, \ldots, n_k)$.

Bemerkungen

1. Alle "üblichen" Funktionen (Prädikate) sind LOOP-berechenbar:
   $x + y$, $x * y$, x^y, $max(x, y)$, $min(x, y)$, $x\ div\ y$, $x\ mod\ y$, $x!$, ...,
   $x = y$, $x < y$, $x \mid y$, ...
2. Alle LOOP-berechenbaren Funktionen sind total.

Die Paarfunktion

Sei $paar^2 : \mathbb{N}_0^2 \to \mathbb{N}_0$ definiert durch: $paar^2(x, y) = x + \sum_{i=1}^{x+y} i$.

Es gilt: $paar^2$ ist bijektiv.

Umkehrfunktionen von $paar^2$: $\pi_1^2(paar^2(x, y)) = x$,
$\pi_2^2(paar^2(x, y)) = y$.

Allgemeiner (für $k \geq 1$): $paar^k : \mathbb{N}_0^k \to \mathbb{N}_0$ (bijektiv) definiert durch:

$$paar^1(x) = x,$$
$$paar^k(x_1, \ldots, x_k) = paar^2(x_1, paar^2(x_2, paar^2(x_3, \ldots, paar^2(x_{k-1}, x_k) \ldots))) \quad \text{für } k > 1.$$

Umkehrfunktionen von $paar^k$: $\pi_1^k, \ldots, \pi_k^k$.

Es gilt: $paar^k, \pi_1^k, \ldots, \pi_k^k$ sind LOOP-berechenbar (für $k \geq 1$).

Die Ackermannfunktion

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

Lemma 2.1.1 Die Ackermannfunktion ack hat folgende Eigenschaften (für alle $x, y \in \mathbb{N}_0$):

a) $y < ack(x, y)$.

b) $ack(x, y) < ack(x, y + 1)$.

c) $ack(x, y + 1) \leq ack(x + 1, y)$.

d) $ack(x, y) < ack(x + 1, y)$.

Satz 2.1.2 Zu jeder LOOP-berechenbaren Funktion $f : \mathbb{N}_0^k \to \mathbb{N}_0$ gibt es $l \in \mathbb{N}_0$ mit $f(z_1, \ldots, z_k) < ack(l, max(z_1, \ldots, z_k))$.

Satz 2.1.3 Die Ackermannfunktion ist nicht LOOP-berechenbar.

Programmiersprache WHILE

⟨Programm⟩	::=	⟨Anweisung⟩ | ⟨Anweisung⟩;⟨Programm⟩
⟨Anweisung⟩	::=	⟨Zuweisung⟩ | ⟨Schleife⟩
⟨Zuweisung⟩	::=	⟨Variable⟩ := ⟨Ausdruck⟩
⟨Ausdruck⟩	::=	0 | ⟨Variable⟩ | ⟨Variable⟩ + 1
⟨Schleife⟩	::=	**loop** ⟨Variable⟩ **do** ⟨Programm⟩ **enddo** |
		while ⟨Variable⟩ $\neq$ 0 **do** ⟨Programm⟩ **enddo**
⟨Variable⟩	::=	$x_0 \mid x_1 \mid x_2 \mid x_3 \mid \ldots \mid y_0 \mid \ldots \mid z_0 \mid \ldots$

Semantik der **while**-Schleife:

⟨Programm⟩ wird so lange wiederholt, wie der Wert von ⟨Variable⟩ nicht 0 ist.