LOOP-Programme

Variablen:	x_1, x_2, x_3, ...
Konstanten:	0, 1, 2, ...
Trennsymbole:	; :=
Operationszeichen:	+ -
Schlüsselworte:	LOOP DO END

Aufbau von LOOP-Programmen:

- $x_i := c$, $x_i := x_j + c$, $x_i := x_j - c$ sind LOOP-Programme.

  Interpretation: wie üblich, mit der Einschränkung, daß $x_j - c$ als Null gewertet wird, falls $c > x_j$.

- P_1, P_2 LOOP-Programme $\Longrightarrow$ $P_1; P_2$ LOOP-Programm.

  Interpretation: Führe erst P_1 und dann P_2 aus.

- P LOOP-Programm $\Longrightarrow$ LOOP x_i DO P END ist LOOP-Programm.

  Interpretation: Führe P genau $\langle$Wert von $x_i\rangle$-mal aus.

Beobachtung: LOOP-Programme können IF .. THEN .. ELSE .. END Konstrukte simulieren.

Idee: IF $x = 0$ THEN A END

wird realisiert durch

$y := 1;$

LOOP x DO $y := 0$ END;

LOOP y DO A END

Ackermann-Funktion

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

Eigenschaften, die man per Induktion zeigen kann:

Lemma 1: $a(1,y) = y+2 \quad \forall y,$
$a(2,y) = 2y+3 \quad \forall y$

Lemma 2: $y < a(x,y) \quad \forall x,y$

Lemma 3: $a(x,y) < a(x,y+1) \quad \forall x,y$

Lemma 4: $a(x,y+1) \leq a(x+1,y) \quad \forall x,y$

Lemma 5: $a(x,y) < a(x+1,y) \quad \forall x,y$

Einige Werte:

$$\begin{aligned}
a(1,1) &= 3, \quad a(2,1) = 5, \quad a(3,1) = 13, \\
a(4,1) &= a(3,a(3,1)) = a(3,13) = \text{...maple steigt aus...} \\
a(3,k) &= a(2,a(3,k-1)) \geq 2*a(3,k-1) \\
&\geq 2^2*a(3,k-2) \geq ... \geq 2^k*a(3,0) \geq 2^k \\
a(4,1) &\geq 2^{13} \\
a(4,2) &= a(3,a(4,1)) \geq 2^{a(4,1)} \geq 2^{2^{13}} \\
a(4,3) &= a(3,a(4,2)) \geq 2^{a(4,2)} \geq 2^{2^{2^{13}}} \\
&\vdots
\end{aligned}$$

WHILE-Programme

Variablen:	x_1, x_2, x_3, ...
Konstanten:	0, 1, 2, ...
Trennsymbole:	; := $\neq$
Operationszeichen:	+ -
Schlüsselworte:	WHILE DO END

Aufbau von WHILE-Programmen:

- $x_i := c,\;\; x_i := x_j + c,\;\; x_i := x_j - c$ sind WHILE-Programme.

  Interpretation: wie üblich, mit der Einschränkung, daß $x_j - c$ als Null gewertet wird, falls $c > x_j$.

- P_1, P_2 WHILE-Programme $\Longrightarrow$ $P_1; P_2$ WHILE-Programm.

  Interpretation: Führe erst P_1 und dann P_2 aus.

- P WHILE-Programm $\Longrightarrow$ WHILE $x_i \neq 0$ DO P END ist WHILE-Programm.

  Interpretation: Führe P solange aus bis x_i den Wert Null hat.

GOTO-Programme

Variablen:	$x_1, x_2, x_3, \ldots$
Konstanten:	0, 1, 2, ...
Trennsymbole:	; :=
Operationszeichen:	+ -
Schlüsselworte:	IF THEN GOTO

Aufbau von GOTO-Programmen:

$$M_1 : A_1;\ M_2 : A_2;\ \ldots\ M_k : A_k$$

Als Anweisungen sind zugelassen:

- Wertzuweisungen: $x_i := x_j \pm c$,
- unbedingter Sprung: GOTO M_i
- bedingter Sprung: IF $x_i = 0$ THEN GOTO M_i
- Stopanweisung: HALT

Satz: Jedes WHILE-Programm kann durch ein GOTO-Programm simuliert werden.

Idee: Ersetze jede WHILE-Schleife

$$\text{WHILE } x_i \neq 0 \text{ DO } P \text{ END}$$

durch

$$
\begin{array}{ll}
M_1 : & \text{IF } x_i = 0 \text{ THEN GOTO } M_2 \\
 & P; \\
 & \text{GOTO } M_1 \\
M_2: & \ldots
\end{array}
$$

Satz: Jedes GOTO-Programm kann durch ein WHILE-Programm simuliert werden.

Idee: Gegeben sei das GOTO-Programm

$$M_1 : A_1;\ M_2 : A_2;\ \dots\ M_k : A_k$$

Wir simulieren dies durch eine WHILE-Programm mit genau einer WHILE-Schleife:

$$
\begin{array}{l}
count := 1; \\
\text{WHILE } count \neq 0 \text{ DO} \\
\quad \text{IF } count = 1 \text{ THEN } A_1' \text{ END;} \\
\quad \text{IF } count = 2 \text{ THEN } A_2' \text{ END;} \\
\quad \vdots \\
\quad \text{IF } count = k \text{ THEN } A_k' \text{ END;} \\
\text{END}
\end{array}
$$

wobei

$$
A_i' := \begin{cases}
x_j := x_l \pm c; count := count+1 & \\
\qquad \text{falls } A_i = x_j := x_l \pm c & \\
count := \ell \qquad \text{falls } A_i = \text{ GOTO } M_\ell & \\
\text{IF } x_j = c \text{ THEN } count := \ell \text{ ELSE } count := count+1 \text{ END} & \\
\qquad \text{falls } A_i = \text{IF } x_j = c \text{ THEN GOTO } M_\ell & \\
count := 0 \qquad \text{falls } A_i = \text{HALT} &
\end{cases}
$$