TI 2 - Ergänzung 2

$\boxed{A1}$

1. $P = \text{LOOP } x_1 \text{ DO}$

$\qquad \text{LOOP } x_1 \text{ DO}$

$\qquad\qquad x_0 := x_0 + 1$

$\qquad \text{END}$

END

$x_0 := x_0 + x_1$

2. $P = x_0 := x_2 + 0;$

$\qquad x_3 := x_1 + 0;$

$\qquad \text{LOOP } x_2 \text{ DO}$

$\qquad\qquad x_3 := x_3 - 1$

$\qquad \text{END};$ $\leftarrow$ Hier gilt: $x_0 = n$ und $x_3 = m \dot{-} n$.

$\qquad \text{LOOP } x_3 \text{ DO}$

$\qquad\qquad x_0 := x_1 + 0$

$\qquad \text{END}$

Auch möglich:

$P = x_0 := x_2 + 0;$

$\qquad \text{LOOP } x_2 \text{ DO}$

$\qquad\qquad x_1 := x_1 - 1$

$\qquad \text{END};$ $\leftarrow$ Hier gilt: $x_0 = n$ und $x_1 = m \dot{-} n$.

$\qquad \text{LOOP } x_1 \text{ DO}$

$\qquad\qquad x_0 := x_0 + 1$

$\qquad \text{END}$

$x_0 := x_0 + x_1$

3. $P =$ $x_0 := x_2 + 0;$

LOOP x_1 DO

  $x_0 := x_0 - 1$

END; ← Hier gilt: $x_0 = n \dot{-} m$, $x_1 = m$ und $x_2 = n$

LOOP x_2 DO

  $x_1 := x_1 - 1$

END; ← Hier gilt: $x_0 = n \dot{-} m$, $x_1 = m \dot{-} n$ und $x_2 = n$

LOOP x_1 DO

  $x_0 := x_1 + 0$

END

} falls $m > n$ (also $m \dot{-} n > 0$), wird hier x_0 auf $m \dot{-} n$ gesetzt.

4. $P =$ $x_0 := x_0 + 1;$

LOOP x_1 DO

  LOOP x_0 DO

    $x_0 := x_0 + 1$

  END

END

} $x_0 := 2 * x_0$

$\boxed{A2}$

1. Simuliere $x_i := x_j * x_k$ durch:

```
x_i := 0;
LOOP x_j DO
  x_i := x_i + x_k
END.
```

$x_i := c$, $x_i := x_j + x_k$ und $x_i := x_j - x_k$ sollten in der Vorlesung besprochen worden sein.

2. Simuliere IF $x_i = 0$ THEN P END durch

```
    x_j := 1;
(*) x_j := x_j - x_i;
    LOOP x_j DO
      P
    END,
```

$x_j = \begin{cases} 1, & \text{falls } x_i = 0 \\ 0 & \text{sonst} \end{cases}$

wobei x_j eine beliebige Variable ist, die sonst nicht benutzt wird.

(*) Auch LOOP x_i DO $x_j := 0$ END statt $x_j := x_j - x_i$ möglich.

A3

Wir geben für jede Funktion ein Programm P an, das sie berechnet und das von einem LOOP-Programm simulierbar ist.

1. P =
```
x0 := 1;
LOOP x1 DO
   x2 := (x0 * x0) - x1
   IF x2 = 0 THEN
      x0 := x0 + 1
   END
END;
x0 := x0 - 1
```

2. P =
```
x2 = 1
LOOP x1 DO
   x3 = 1
   LOOP x1 DO
      x4 := ((x2 * x3) - x1) + (x1 - (x2 * x3));
      IF x4 = 0 THEN
         x0 := x0 + 1
      END;
      x3 := x3 + 1
   END
   x2 := x2 + 1
END
```

$\boxed{A4}$

1. Ja!

Seien P ein LOOP-Programm mit HALT-Anweisungen und x_i und x_j zwei Variablen, die in P nicht verwendet werden.

Sei P' das LOOP-Programm, das dadurch entsteht, dass man in P alle Vorkommen von HALT durch

IF $x_i = 0$ THEN $x_i := x_i + 1$; $x_j := x_0 + 0$ END

ersetzt. Dann wird P durch

P'; LOOP x_i DO $x_0 := x_j + 0$ END

simuliert.

2. Nein!

LOOP-Programme terminieren immer und berechnen somit nur totale Funktionen. Das Programm

ERROR

berechnet die nirgends definierte Funktion Ω. Diese ist aber nicht LOOP-berechenbar.

$\boxed{\text{A5}}$

1. Aus Anzahlgründen. Es gibt (bis auf Isomorphie) abzählbar viele DTMs, aber überabzählbar viele Funktionen von $\mathbb{N}$ nach $\mathbb{N}$.

<u>Warum abzählbar viele DTMs</u> ?

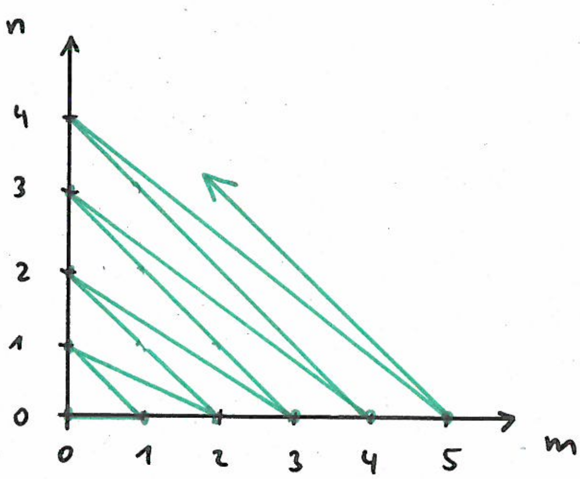
Für feste $m, n \in \mathbb{N}$ gibt es endlich viele DTMs mit Zustandsmenge $Z = \{0, \dots, m\}$, Eingabealphabet $\Sigma = \{0, 1\}$, Bandalphabet $\Gamma = \{0, \dots, n+1, \square\}$ und Leersymbol $\square$.

Γ muss ja Σ enthalten!

Es sind genau:

$$\underbrace{|Z \times \Gamma \times \{L, R, N\}|^{|Z \times \Gamma|}}_{\delta} \cdot \underbrace{|Z|}_{z_0} \cdot \underbrace{2^{|Z|}}_{E} = (3 \cdot |Z| \cdot |\Gamma|)^{|Z| \cdot |\Gamma|} \cdot |Z| \cdot 2^{|Z|}.$$

Diese kann man beliebig durchnummerieren. Zähle dann alle DTMs nach folgendem Schema durch:



Warum überabzählbar viele Funktionen von $\mathbb{N}$ nach $\mathbb{N}$?

Angenommen, man könnte sie alle durchnummerieren:

$$f_0, f_1, f_2, f_3, f_4, \ldots$$

Unabhängig von der Durchnummerierung lässt sich immer eine Funktion f finden mit $f \neq f_i$ für alle $i \in \mathbb{N}$,

z.B. $f: \mathbb{N} \to \mathbb{N}$ mit $f(n) = \begin{cases} f_n(n) + 1, & \text{falls } f_n(n) \text{ definiert} \\ 0 & \text{sonst} \end{cases}$.

2. Ja!

Wir wissen aus der Vorlesung, dass diese drei Berechnungsmodelle gleichmächtig sind. Deshalb reicht es, ein WHILE-Programm, ein GOTO-Programm oder eine DTM für div anzugeben. Wir geben exemplarisch ein WHILE- und ein GOTO-Programm an.

WHILE:

```
x1 := x1 + 1;
WHILE x1 ≠ 0 DO
   x0 := x0 + 1;
   LOOP x2 DO
      x1 := x1 - 1
   END
END;
x0 := x0 - 1
```

$x_1 := x_1 - x_2$ wäre hier auch ok.

GOTO:

```
M1: x1 := x1 + 1;
M2: x0 := x0 + 1;
M3: IF x2 = 0 THEN GOTO M7;
M4: x1 := x1 - 1;
M5: x2 := x2 - 1;
M6: GOTO M3;
M7: IF x1 = 0 THEN GOTO M2;
M8: x0 := x0 - 1
```

3. Nein!

Jede LOOP-berechenbare Funktion ist total und div ist es nicht (Division durch 0 nicht definiert).

4. Nur für $c=0$!

Für $c=0$ ist $c \cdot h$ die konstante Nullfunktion. Diese ist Turing-berechenbar.

Für $c \neq 0$ ist $c \cdot h$ nicht Turing-berechenbar. Wäre $c \cdot h$ Turing-berechenbar, dann könnten wir mittels

$$h(n) = \operatorname{div}(c \cdot h(n), c) = \left\lfloor \frac{c \cdot h(n)}{c} \right\rfloor$$

auch h berechnen, was nicht geht.

5. Nein!

Wähle z.B. $g(n) = 2 \cdot h(n)$. Nach Teilaufgabe 4 ist g nicht Turing-berechenbar. Da $g(n)$ für alle n gerade ist, ist dann f die konstante Nullfunktion, die bekanntlich Turing-berechenbar ist.

Moral: Dass eine nicht Turing-berechenbare Funktion in der Definition einer anderen Funktion vorkommt, heißt nicht, dass diese notwendigerweise auch nicht Turing-berechenbar ist!

A6

Ja!

Sei $P = M_1 : A_1 ; \dots ; M_n : A_n$ ein um Anweisungen der Form GOTO x_i erweitertes GOTO-Programm. Ersetzt man in P jede Anweisung der Form GOTO x_i durch

IF $x_i = 1$ THEN GOTO M_1
⋮
IF $x_i = n$ THEN GOTO M_n
GOTO M_{n+1}

und fügt am Ende die Zeile

M_{n+1} : GOTO M_{n+1}

hinzu, so entsteht ein normales GOTO-Programm, das dieselbe Funktion wie P berechnet.