Bemerkungen

1. `LOOP`-programmierbar $\Longrightarrow$ `WHILE`-programmierbar.
2. **loop** x **do** P **enddo** $\hat{=}$ $y := x$; **while** $y \neq 0$ **do** $y := y \dot{-} 1$; P **enddo**.
3. `WHILE`-Programme müssen nicht terminieren.

Definition. Eine partielle Funktion $f : \mathbb{N}_0^k \to \mathbb{N}_0$ heißt `WHILE`-***berechenbar***, wenn es ein `WHILE`-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 terminiert P genau dann, wenn $f(n_1, \ldots, n_k)$ definiert ist, und falls P terminiert, hat die Variable x_0 nach Beendigung von P den Wert $f(n_1, \ldots, n_k)$.

Satz 2.1.4 Die Ackermannfunktion ist `WHILE`-berechenbar.

2.2 Maschinenberechenbarkeit

Berechnung von $n+m$ für $n, m \in \mathbb{N}_0$ mit einer Turingmaschine mit Eingabealphabet $\{1\}$:
Eingabewort: $\underbrace{11\ldots1}_{n}\,\Box\,\underbrace{11\ldots1}_{m}$
Gewünschte Bandinschrift am Ende der Rechnung: $\underbrace{11\ldots1}_{n+m}$

Turingmaschine:

$$
\begin{aligned}
&Z = \{z_0, z_1, z_2, z_e\},\\
&E = \{z_e\},\\
&\delta: \ (z_0, 1) \mapsto (z_0, 1, \mathsf{R}), (z_0, \Box) \mapsto (z_1, 1, \mathsf{L}), (z_1, 1) \mapsto (z_1, 1, \mathsf{L}),\\
&\qquad (z_1, \Box) \mapsto (z_2, \Box, \mathsf{R}), (z_2, 1) \mapsto (z_e, \Box, \mathsf{R})
\end{aligned}
$$

Es gilt: $z_0 1^n \Box 1^m \vdash^* \Box\Box z_e 1^{n+m}$.

Definition. Eine partielle Funktion $f : \mathbb{N}_0^k \to \mathbb{N}_0$ heißt ***Turing-berechenbar***, wenn es eine (deterministische) Turingmaschine $M = (Z, \{1\}, \Gamma, \delta, z_0, \Box, E)$ gibt, so dass $\delta(z_e, a)$ undefiniert ist für alle $z_e \in E$, $a \in \Gamma$ und für alle $n_1, \ldots, n_k, m \in \mathbb{N}_0$ gilt:

$$
f(n_1, \ldots, n_k) \text{ ist definiert und } f(n_1, \ldots, n_k) = m \iff
z_0 1^{n_1} \Box 1^{n_2} \Box \ldots \Box 1^{n_k} \vdash_M^* \Box \ldots \Box z_e 1^m \Box \ldots \Box, \quad z_e \in E.
$$

Satz 2.2.1 Jede `WHILE`-berechenbare Funktion ist Turing-berechenbar.

Andere Maschinentypen/-konzepte

1. TM-Varianten
   - Nur einseitig unendliches Band.
   - Statt einem Band: endliche Anzahl von Bändern (***Mehrband-TM***);
     auf jedem Band unabhängig voneinander Positionierung des Schreib-/Lesekopfes, Lesen und Schreiben möglich.
   - Diese Varianten sind gleichmächtig zur “Standard”-TM.
2. ***Registermaschine*** (***RAM***, ***URM***)
   - Realistischere Nachbildung (der Grundprinzipien) heutiger Computer.
   - ***RAM-berechenbar***: Eingabe $x_1, \ldots, x_k$ in Eingabedatei, nach Ende der Rechnung: Ergebnis in Ausgaberegister.
   - Es gilt: Eine Funktion $f : \mathbb{N}_0^k \to \mathbb{N}_0$ ist genau dann RAM-berechenbar, wenn sie Turing-berechenbar ist.

2.3 Primitiv-rekursive und μ-rekursive Funktionen

Bisher: Berechenbarkeit

- durch Programme,
- durch Maschinen.

Jetzt abstrakter Ansatz: Menge der berechenbaren Funktionen induktiv erzeugt.

Grundidee

Der Ansatz basiert auf der Beobachtung, dass man die üblichen arithmetischen Funktionen alle durch Rekursion aus einigen wenigen Grundfunktionen gewinnen kann.

Bezeichnungen

- ***Nullfunktion:*** $C^k : \mathbb{N}_0^k \to \mathbb{N}_0,\ C^k(x_1, \ldots, x_k) = 0.$
- ***Nachfolgerfunktion:*** $S : \mathbb{N}_0 \to \mathbb{N}_0,\ S(x) = x + 1.$
- ***Projektionen:*** $U_i^k : \mathbb{N}_0^k \to \mathbb{N}_0,\ U_i^k(x_1, \ldots, x_k) = x_i \quad (i = 1, \ldots, k).$