Berechenbarkeit

Vorlesung 5: While-Programme

15. Mai 2025

Termine — Modul Berechenbarkeit

Übungen	Vorlesung
13.5. Übung 3 B-Woche	15.5. While-Programme
20.5. Übung 3 A-Woche	22.5. Rekursion I (Übungsblatt 4)
27.5. Übung 4 B-Woche	29.5. —
3.6. Übung 4 A-Woche	5.6. Rekursion II (Übungsblatt 5)
10.6. Übung 5 B-Woche (Montag Feiertag)	12.6. Entscheidbarkeit

Übungen	Vorlesung
17.6. Übung 5 A-Woche	19.6. Unentscheidbarkeit (Übungsblatt 6)
24.6. Übung 6 B-Woche	26.6. Spez. Probleme
1.7. Übung 6 A-Woche	3.7. Klasse P
8.7. Abschlussübung beide Wochen	10.7. NP-Vollständigkeit

Wiederholung — Berechenbarkeit

Definition (§4.8 Turing-Berechenbarkeit; *Turing-computability*)

Partielle Funktion $f : \mathbb{N}^k \dashrightarrow \mathbb{N}$ **Turing-berechenbar**
falls deterministische TM M mit $\text{bin}(f) = T(M)$ existiert

Definition (§4.15 Loop-Berechenbarkeit; *Loop-computability*)

Funktion $f : \mathbb{N}^k \to \mathbb{N}$ **Loop-berechenbar**
falls Loop-Programm P mit $f = |P|_k$ existiert

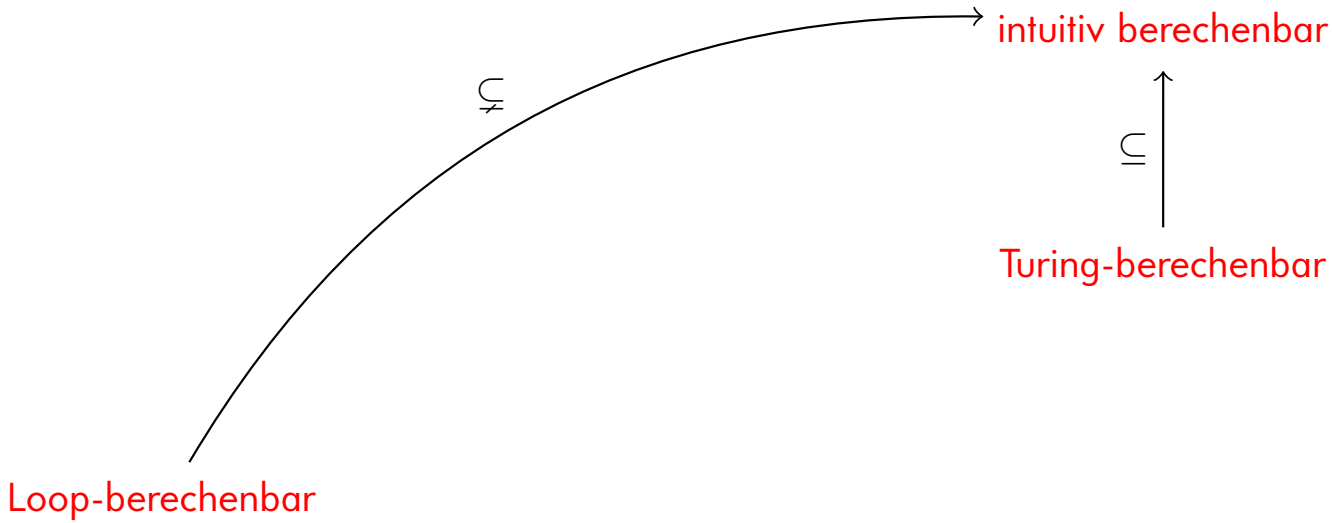
Wiederholung — Berechenbarkeit

intuitiv berechenbar

Turing-berechenbar

Loop-berechenbar

Wiederholung — Berechenbarkeit



Ackermann-Funktion

§5.1 Definition (Ackermann-Funktion; *Ackermann function*)

Für alle $x, y \in \mathbb{N}$ sei

$$a(x,y) = \begin{cases} y+1 & \text{falls } x = 0 \\ a(x-1,1) & \text{falls } x \neq 0 \text{ und } y = 0 \\ a\big(x-1, a(x,y-1)\big) & \text{sonst} \end{cases}$$

Wilhelm Ackermann (* 1896; † 1962)

- Dtsch. Mathematiker
- Student von David Hilbert
- Gymnasiallehrer & Ehrenprofessor Uni Münster

Ackermann-Funktion

§5.2 Theorem

Ackermann-Funktion total; d.h. $a : \mathbb{N}^2 \to \mathbb{N}$

Ackermann-Funktion

§5.2 Theorem

Ackermann-Funktion total; d.h. $a\colon \mathbb{N}^2 \to \mathbb{N}$

Beweis (vollständige Induktion über 1. Argument)

IA: $a(0, y) = y + 1$ für alle $y \in \mathbb{N}$ definiert

Ackermann-Funktion

§5.2 Theorem

Ackermann-Funktion total; d.h. $a\colon \mathbb{N}^2 \to \mathbb{N}$

Beweis (vollständige Induktion über 1. Argument)

IA: $a(0,y) = y+1$ für alle $y \in \mathbb{N}$ definiert

IS: Sei $a(x,y)$ für alle $y \in \mathbb{N}$ definiert. Dann

$$\begin{aligned} a(x+1,y) &= a(x,a(x+1,y-1)) = a(x,a(x,a(x+1,y-2))) \\ &= \cdots = \underbrace{a(x,a(x,\cdots a}_{(y+1)\text{ mal}}(x+1,0)\cdots)) \\ &= \underbrace{a(x,a(x,\cdots a}_{(y+1)\text{ mal}}(x,1)\cdots)) \end{aligned}$$

für alle $y \in \mathbb{N}$ definiert $\square$

Ackermann-Funktion

Problem Ist Ackermann-Funktion Loop-berechenbar?

$x \setminus y$	0	1	2	3	4
0	1	2	3	4	5
1	2	3	4	5	6
2	3	5	7	9	11
3	5	13	29	61	125
4	13	65.533	$\gg 10^{10.000}$	...	...

Ackermann-Funktion

§5.3 Lemma

$a(x, y) > y$ für alle $x, y \in \mathbb{N}$

Ackermann-Funktion

§5.3 Lemma

$a(x, y) > y$ für alle $x, y \in \mathbb{N}$

Beweis (vollständige Induktion über 1. Argument)

IA: $a(0, y) = y + 1 > y$ für alle $y \in \mathbb{N}$

Ackermann-Funktion

§5.3 Lemma

$a(x, y) > y$ für alle $x, y \in \mathbb{N}$

Beweis (vollständige Induktion über 1. Argument)

IA: $a(0, y) = y + 1 > y$ für alle $y \in \mathbb{N}$

IS: Sei $a(x, y) > y$ für alle $y \in \mathbb{N}$. Vollständige Induktion über y

Ackermann-Funktion

§5.3 Lemma

$a(x, y) > y$ für alle $x, y \in \mathbb{N}$

Beweis (vollständige Induktion über 1. Argument)

IA: $a(0, y) = y + 1 > y$ für alle $y \in \mathbb{N}$

IS: Sei $a(x, y) > y$ für alle $y \in \mathbb{N}$. Vollständige Induktion über y

- IA: $a(x + 1, 0) = a(x, 1) \overset{\text{IH}}{>} 1 > 0$ nach IH $a(x, y) > y$

Ackermann-Funktion

§5.3 Lemma

$a(x, y) > y$ für alle $x, y \in \mathbb{N}$

Beweis (vollständige Induktion über 1. Argument)

IA: $a(0, y) = y + 1 > y$ für alle $y \in \mathbb{N}$

IS: Sei $a(x, y) > y$ für alle $y \in \mathbb{N}$. Vollständige Induktion über y

- IA: $a(x+1, 0) = a(x, 1) \overset{\text{IH}}{>} 1 > 0$ nach IH $a(x, y) > y$
- IS: Sei $a(x+1, y) > y$

$$a(x+1, y+1) = a\big(x, a(x+1, y)\big) \overset{\text{IH}}{>} a(x+1, y) \overset{\text{IH}}{\geq} y + 1$$

nach äußerer und danach innerer IH $\square$

Ackermann-Funktion

§5.4 Lemma

$a(x, y+1) > a(x, y)$ für alle $x, y \in \mathbb{N}$

Ackermann-Funktion

§5.4 Lemma

$a(x, y+1) > a(x, y)$ für alle $x, y \in \mathbb{N}$

Beweis (mit Hilfe von §5.3)

- Sei $x = 0$

$$a(0, y+1) = y+2 > y+1 = a(0, y)$$

Ackermann-Funktion

§5.4 Lemma

$a(x, y+1) > a(x, y)$ für alle $x, y \in \mathbb{N}$

Beweis (mit Hilfe von §5.3)

- Sei $x = 0$

$$a(0, y+1) = y+2 > y+1 = a(0, y)$$

- Sei $x > 0$

$$a(x, y+1) = a\big(x-1, a(x, y)\big) \overset{\text{§5.3}}{>} a(x, y)$$ □

Ackermann-Funktion

§5.5 Lemma

$a(x, y') > a(x, y)$ für alle $x, y, y' \in \mathbb{N}$ mit $y' > y$

Ackermann-Funktion

§5.5 Lemma

$a(x, y') > a(x, y)$ für alle $x, y, y' \in \mathbb{N}$ mit $y' > y$

Beweis

Leichte Übung mit Hilfe von §5.4 □

Ackermann-Funktion

§5.6 Lemma

$a(x+1, y) \geq a(x, y+1)$ für alle $x, y \in \mathbb{N}$

Ackermann-Funktion

§5.6 Lemma

$a(x+1, y) \geq a(x, y+1)$ für alle $x, y \in \mathbb{N}$

Beweis (vollständige Induktion über 2. Argument)

IA: $a(x+1, 0) = a(x, 1)$ für alle $x \in \mathbb{N}$

Ackermann-Funktion

§5.6 Lemma

$a(x+1, y) \geq a(x, y+1)$ für alle $x, y \in \mathbb{N}$

Beweis (vollständige Induktion über 2. Argument)

IA: $a(x+1, 0) = a(x, 1)$ für alle $x \in \mathbb{N}$
IS: Sei $a(x+1, y) \geq a(x, y+1)$ für alle $x \in \mathbb{N}$

$$a(x+1, y+1) = a\big(x, a(x+1, y)\big) \overset{\text{§5.5}}{\geq} a\big(x, a(x, y+1)\big) \overset{\text{§5.5}}{\geq} a(x, y+2)$$

unter Nutzung der IH und §5.4 □

Ackermann-Funktion

§5.7 Lemma

$a(x+1, y) > a(x, y)$ für alle $x, y \in \mathbb{N}$

Ackermann-Funktion

§5.7 Lemma

$a(x+1, y) > a(x, y)$ für alle $x, y \in \mathbb{N}$

Beweis

$$a(x+1, y) \overset{\text{§5.6}}{\geq} a(x, y+1) \overset{\text{§5.4}}{>} a(x, y)$$ □

Ackermann-Funktion

§5.7 Lemma

$a(x+1, y) > a(x, y)$ für alle $x, y \in \mathbb{N}$

Beweis

$$a(x+1, y) \overset{\text{§5.6}}{\geq} a(x, y+1) \overset{\text{§5.4}}{>} a(x, y)$$ □

§5.8 Theorem (Monotonie der Ackermann-Funktion)

$a(x', y') \geq a(x, y)$ für alle $x, x', y, y' \in \mathbb{N}$ mit $x' \geq x$ und $y' \geq y$

Ackermann-Funktion

§5.7 Lemma

$a(x+1, y) > a(x, y)$ für alle $x, y \in \mathbb{N}$

Beweis

$$a(x+1, y) \overset{\text{§5.6}}{\geq} a(x, y+1) \overset{\text{§5.4}}{>} a(x, y)$$ □

§5.8 Theorem (Monotonie der Ackermann-Funktion)

$a(x', y') \geq a(x, y)$ für alle $x, x', y, y' \in \mathbb{N}$ mit $x' \geq x$ und $y' \geq y$

Beweis

$$a(x', y') \overset{\text{§5.7}}{\geq} a(x, y') \overset{\text{§5.5}}{\geq} a(x, y)$$ □

Loop-Berechenbarkeit

§5.9 Definition (norm. Loop-Programm; *unitary Loop program*)

Normiertes Loop-Programm P entweder

- $P = x_i = x_\ell + z$ mit $i, \ell \geq 1$ und $z \in \{-1, 0, 1\}$

Loop-Berechenbarkeit

§5.9 Definition (norm. Loop-Programm; *unitary Loop program*)

Normiertes Loop-Programm P entweder

- $P = x_i = x_\ell + z$ mit $i, \ell \geq 1$ und $z \in \{-1, 0, 1\}$
- $P = P_1 \,;\, P_2$ für normierte Loop-Programme P_1 und P_2

Loop-Berechenbarkeit

§5.9 Definition (norm. Loop-Programm; *unitary Loop program*)

Normiertes Loop-Programm P entweder

- $P = x_i = x_\ell + z$ mit $i, \ell \geq 1$ und $z \in \{-1, 0, 1\}$
- $P = P_1 \,;\, P_2$ für normierte Loop-Programme P_1 und P_2
- $P = \mathbf{LOOP}(x_i)\,\{P'\}$ für normiertes Loop-Programm P', $i \notin \mathrm{var}(P')$

Loop-Berechenbarkeit

§5.9 Definition (norm. Loop-Programm; *unitary Loop program*)

Normiertes Loop-Programm P entweder

- $P = x_i = x_\ell + z$ mit $i, \ell \geq 1$ und $z \in \{-1, 0, 1\}$
- $P = P_1 \,;\, P_2$ für normierte Loop-Programme P_1 und P_2
- $P = \mathbf{LOOP}(x_i)\,\{P'\}$ für normiertes Loop-Programm P', $i \notin \mathrm{var}(P')$

Notizen

- Zuweisungen nur mit Addition von $\{-1, 0, 1\}$
- Schleifenvariable nicht in Schleifenkörper

Loop-Berechenbarkeit

§5.10 Theorem

Jedes Loop-Programm P hat äquiv. normiertes Loop-Programm

Loop-Berechenbarkeit

§5.10 Theorem

Jedes Loop-Programm P hat äquiv. normiertes Loop-Programm

Beweisskizze

- Ersetze Zuweisung $x_i = x_\ell + n$ mit $n \in \mathbb{N}$ durch

$$x_i = x_\ell + 1 \,;\, x_i = x_i + 1 \,;\, \ldots \,;\, x_i = x_i + 1$$

- (analog für $n < 0$)

Loop-Berechenbarkeit

§5.10 Theorem

Jedes Loop-Programm P hat äquiv. normiertes Loop-Programm

Beweisskizze

- Ersetze Zuweisung $x_i = x_\ell + n$ mit $n \in \mathbb{N}$ durch

$$x_i = x_\ell + 1 \,;\, x_i = x_i + 1 \,;\, \ldots \,;\, x_i = x_i + 1$$

- (analog für $n < 0$)
- Ersetze $\mathbf{LOOP}(x_i)\,\{P'\}$ durch

$$x_\ell = x_i \,;\, \mathbf{LOOP}(x_\ell)\,\{P'\}$$

wobei $x_\ell \notin \mathrm{var}(P)$ im Gesamtprogramm P nicht vorkommt □

Loop-Berechenbarkeit

§5.11 Definition (Größenbegrenzung)

Sei P normiertes Loop-Programm mit $\max \operatorname{var}(P) \leq n$.

Definiere $f_P^{(n)} : \mathbb{N} \to \mathbb{N}$ für alle $s \in \mathbb{N}$ durch

$$f_P^{(n)}(s) = \max \left\{ \sum_{i=1}^{n} r_i \;\middle|\; s_1, \ldots, s_n \in \mathbb{N}, \sum_{i=1}^{n} s_i \leq s, \right.$$
$$\left. (r_1, \ldots, r_n) = \|P\|_n(s_1, \ldots, s_n) \right\}$$

Loop-Berechenbarkeit

§5.11 Definition (Größenbegrenzung)

Sei P normiertes Loop-Programm mit $\max \operatorname{var}(P) \leq n$.
Definiere $f_P^{(n)} : \mathbb{N} \to \mathbb{N}$ für alle $s \in \mathbb{N}$ durch

$$f_P^{(n)}(s) = \max \left\{ \sum_{i=1}^{n} r_i \;\middle|\; s_1, \ldots, s_n \in \mathbb{N}, \sum_{i=1}^{n} s_i \leq s, \; (r_1, \ldots, r_n) = \|P\|_n(s_1, \ldots, s_n) \right\}$$

Notizen

- $f_P^{(n)}(s)$ maximale Summe Variablenendwerte bei Eingaben $(s_1, \ldots, s_n)$ deren Summe $\sum_{i=1}^{n} s_i$ höchstens s ist
- Kein Variablenendwert oder Funktionsergebnis größer als $f_P(s)$ (bei Eingaben, die sich auf höchstens s summieren)

Loop-Berechenbarkeit

§5.12 Theorem

Für jedes normierte Loop-Programm P mit $\max \operatorname{var}(P) \leq n$
existiert $k \in \mathbb{N}$ mit $f_P^{(n)}(s) < a(k, s)$ für alle $s \in \mathbb{N}$

Loop-Berechenbarkeit

§5.12 Theorem

Für jedes normierte Loop-Programm P mit $\max \operatorname{var}(P) \leq n$ existiert $k \in \mathbb{N}$ mit $f_P^{(n)}(s) < a(k, s)$ für alle $s \in \mathbb{N}$

Beweis (Induktion über Struktur normierter Programme; 1/3)

1. Sei $P = x_i = x_\ell + z$ mit $z \in \{-1, 0, 1\}$. Dann $f_P^{(n)}(s) \leq 2s + 1$.
   Wir wählen $k = 2$

$$f_P^{(n)}(s) \leq 2s + 1 < 2s + 3 = a(2, s)$$

   mit $2s + 3 = a(2, s)$ unbewiesen (nette Übung)

Beweis (Induktion über Struktur normierter Programme; 2/3)

2. Seien P_1 und P_2 normierte Loop-Programme und $k_1, k_2 \in \mathbb{N}$ mit $f_{P_1}^{(n)}(s) < a(k_1, s)$ und $f_{P_2}^{(n)}(s) < a(k_2, s)$ für alle $s \in \mathbb{N}$.

   Sei $k' = \max(k_1 - 1, k_2)$. Dann für $P = P_1 \,;\, P_2$

$$\begin{aligned} f_P(s) &\leq f_{P_2}^{(n)}\big(f_{P_1}^{(n)}(s)\big) \\ &< a\big(k_2, a(k_1, s)\big) && \text{(Monotonie)} \\ &\leq a\big(k', a(k'+1, s)\big) && \text{(Monotonie)} \\ &= a(k'+1, s+1) \\ &\leq a(k'+2, s) && \text{(§5.6)} \end{aligned}$$

   Aussage gilt für $k = k' + 2$

Loop-Berechenbarkeit

Beweis (Induktion über Struktur normierter Programme; 3/3)

3. Sei P' normiertes Loop-Programm, $k' \in \mathbb{N}$ mit $f_{P'}^{(n)}(s) < a(k', s)$ für alle $s \in \mathbb{N}$. Sei $P = \mathbf{LOOP}(x_i)\{P'\}$ mit $i \notin \mathrm{var}(P')$ und $s \in \mathbb{N}$. Sei m_s Wert von x_i der zum Maximum $f_P^{(n)}(s)$ führt

$$
\begin{aligned}
f_P^{(n)}(s) &\leq \underbrace{f_{P'}^{(n)}(f_{P'}^{(n)}(\cdots f_{P'}^{(n)}}_{m_s \text{ mal}}(s - m_s)\cdots)) + m_s \\
&\leq \cdots \leq \underbrace{a(k', a(k', \cdots a(k'}_{m_s \text{ mal}}, s - m_s)\cdots)) && \text{(§5.4)} \\
&< \underbrace{a(k', a(k', \cdots a(k'}_{m_s \text{ mal}} + 1, s - m_s)\cdots)) && \text{(§5.7 + §5.4)} \\
&= a(k' + 1, s - 1) < a(k' + 1, s) && \text{(§5.4)}
\end{aligned}
$$

Aussage gilt für $k = k' + 1$ □

Loop-Berechenbarkeit der Ackermann-Funktion

§5.13 Theorem

Ackermann-Funktion nicht Loop-berechenbar

Beweis

Angenommen $a\colon \mathbb{N}^2 \to \mathbb{N}$ wäre Loop-berechenbar mit Programm P und $n = \max \operatorname{var}(P)$. Dann ist $g\colon \mathbb{N} \to \mathbb{N}$ mit $g(s) = a(s, s)$ für alle $s \in \mathbb{N}$ Loop-berechenbar via $P' = x_2 = x_1 \,;\, P$.

Loop-Berechenbarkeit der Ackermann-Funktion

§5.13 Theorem

Ackermann-Funktion nicht Loop-berechenbar

Beweis

Angenommen $a\colon \mathbb{N}^2 \to \mathbb{N}$ wäre Loop-berechenbar mit Programm P und $n = \max \operatorname{var}(P)$. Dann ist $g\colon \mathbb{N} \to \mathbb{N}$ mit $g(s) = a(s, s)$ für alle $s \in \mathbb{N}$ Loop-berechenbar via $P' = x_2 = x_1 \,;\, P$. Gemäß §5.12 existiert $k \in \mathbb{N}$ mit

$$a(s, s) = g(s) \leq f_{P'}^{(n)}(s) < a(k, s)$$

für alle $s \in \mathbb{N}$.

Loop-Berechenbarkeit der Ackermann-Funktion

§5.13 Theorem

Ackermann-Funktion nicht Loop-berechenbar

Beweis

Angenommen $a\colon \mathbb{N}^2 \to \mathbb{N}$ wäre Loop-berechenbar mit Programm P und $n = \max \operatorname{var}(P)$. Dann ist $g\colon \mathbb{N} \to \mathbb{N}$ mit $g(s) = a(s,s)$ für alle $s \in \mathbb{N}$ Loop-berechenbar via $P' = x_2 = x_1 \,;\, P$. Gemäß §5.12 existiert $k \in \mathbb{N}$ mit

$$a(s,s) = g(s) \leq f_{P'}^{(n)}(s) < a(k,s)$$

für alle $s \in \mathbb{N}$. Für $s = k$ entsteht Widerspruch

$$a(k,k) = g(k) \leq f_{P'}^{(n)}(k) < a(k,k) \qquad \lightning$$

Also Ackermann-Funktion a nicht Loop-berechenbar $\square$

Loop-Berechenbarkeit der Ackermann-Funktion

Konsequenz

Nicht jede intuitiv berechenbare (totale) Funktion Loop-berechenbar

Exkurs: Originale Ackermann-Funktion

Originaldefinition

$$\varphi(x, y, 0) = x + y$$
$$\varphi(x, y, 1) = x \cdot y$$
$$\varphi(x, y, 2) = x^y$$
$$\dots$$
$$\varphi(x, y, z) = x \uparrow^{z-1} y$$

- Iteriert jeweilig vorherige Operation
- Verschachtelungstiefe Schleifen abhängig von Eingabe z
- Nicht Loop-berechenbar

While-Programme

Konventionen

- Alle Variablen $x_1, x_2, \ldots$ vom Typ $\mathbb{N}$ (beliebige Größe)
- Addition auf $\mathbb{N}$ begrenzt

$$n \oplus z = \max(0, n + z) \qquad n \in \mathbb{N}, z \in \mathbb{Z}$$

- Wir schreiben einfach $+$ statt $\oplus$

While-Programme

Konventionen

- Alle Variablen $x_1, x_2, \ldots$ vom Typ $\mathbb{N}$ (beliebige Größe)
- Addition auf $\mathbb{N}$ begrenzt

$$n \oplus z = \max(0, n + z) \qquad n \in \mathbb{N}, z \in \mathbb{Z}$$

- Wir schreiben einfach $+$ statt $\oplus$

Definition (§4.9 Zuweisung; *assignment*)

Zuweisung ist Anweisung der Form $x_i = x_\ell + z$ mit $i, \ell \geq 1$ und $z \in \mathbb{Z}$

While-Programme

§5.14 Definition (While-Programm; *While program*)

While-Programm P entweder

- **Zuweisung** $P = x_i = x_\ell + z$ für $i, \ell \geq 1$ und $z \in \mathbb{Z}$

While-Programme

§5.14 Definition (While-Programm; *While program*)

While-Programm P entweder

- **Zuweisung** $P = x_i = x_\ell + z$ für $i, \ell \geq 1$ und $z \in \mathbb{Z}$
- **Sequenz** $P = P_1 \,;\, P_2$ für While-Programme P_1 und P_2

While-Programme

§5.14 Definition (While-Programm; *While program*)

While-Programm P entweder

- **Zuweisung** $P = x_i = x_\ell + z$ für $i, \ell \geq 1$ und $z \in \mathbb{Z}$
- **Sequenz** $P = P_1 \,;\, P_2$ für While-Programme P_1 und P_2
- **While-Schleife** $P = \textbf{WHILE}(x_i \neq 0)\,\{P'\}$ für While-Programm P', $i \in \mathbb{N}$

While-Programme

§5.14 Definition (While-Programm; *While program*)

While-Programm P entweder

- **Zuweisung** $P = x_i = x_\ell + z$ für $i, \ell \geq 1$ und $z \in \mathbb{Z}$
- **Sequenz** $P = P_1 \,;\, P_2$ für While-Programme P_1 und P_2
- **While-Schleife** $P = \textbf{WHILE}(x_i \neq 0)\,\{P'\}$ für While-Programm P', $i \in \mathbb{N}$

Beispiele

- $\textbf{WHILE}(x_1 \neq 0)\,\{x_2 = x_1 + 5 \,;\, x_1 = x_3 + 1\} \,;\, x_1 = x_3 + 0$

While-Programme

§5.14 Definition (While-Programm; *While program*)

While-Programm P entweder

- **Zuweisung** $P = x_i = x_\ell + z$ für $i, \ell \geq 1$ und $z \in \mathbb{Z}$
- **Sequenz** $P = P_1 \,;\, P_2$ für While-Programme P_1 und P_2
- **While-Schleife** $P = \textbf{WHILE}(x_i \neq 0)\,\{P'\}$ für While-Programm P', $i \in \mathbb{N}$

Beispiele

- $\textbf{WHILE}(x_1 \neq 0)\,\{x_2 = x_1 + 5 \,;\, x_1 = x_3 + 1\} \,;\, x_1 = x_3 + 0$
- $\textbf{WHILE}(x_1 \neq 0)\,\{$ gleiches Programm, leichter lesbar
  $x_2 = x_1 + 5$
  $x_1 = x_3 + 1\}$
  $x_1 = x_3 + 0$

While-Programme

(Verzicht auf vollständige Quantifikation)

§5.15 Definition (Variablen und maximaler Variablenindex)

Für While-Programm P seien $\mathrm{var}(P) \subseteq \mathbb{N}$ und $\max \mathrm{var}(P) \in \mathbb{N}$ verwendeten Variablenindices und größter verwendeter Variablenindex

$$\mathrm{var}(x_i = x_\ell + z) = \{i, \ell\}$$
$$\mathrm{var}(P_1 \,;\, P_2) = \mathrm{var}(P_1) \cup \mathrm{var}(P_2)$$
$$\mathrm{var}(\textbf{WHILE}(x_i \neq 0)\,\{P'\}) = \{i\} \cup \mathrm{var}(P')$$

While-Programme

(Verzicht auf vollständige Quantifikation)

§5.15 Definition (Variablen und maximaler Variablenindex)

Für While-Programm P seien $\mathrm{var}(P) \subseteq \mathbb{N}$ und $\max \mathrm{var}(P) \in \mathbb{N}$ verwendeten Variablenindices und größter verwendeter Variablenindex

$$\mathrm{var}(x_i = x_\ell + z) = \{i, \ell\}$$
$$\mathrm{var}(P_1 \,;\, P_2) = \mathrm{var}(P_1) \cup \mathrm{var}(P_2)$$
$$\mathrm{var}(\textbf{WHILE}(x_i \neq 0)\,\{P'\}) = \{i\} \cup \mathrm{var}(P')$$

$\mathrm{var}(P) = \{1, 2, 3\}$ und $\max \mathrm{var}(P) = 3$ für folgendes Programm P

$\textbf{WHILE}(x_1 \neq 0)\,\{x_2 = x_1 + 5 \,;\, x_1 = x_3 + 1\} \,;\, x_1 = x_3 + 0$

While-Programme

Überblick

- k Eingaben in Variablen $x_1, \ldots, x_k$ hinterlegt
- Erwartete Semantik für Zuweisung (wie bisher)

While-Programme

Überblick

- k Eingaben in Variablen $x_1, \ldots, x_k$ hinterlegt
- Erwartete Semantik für Zuweisung (wie bisher)
- $P_1 \,;\, P_2$ führt P_1 und danach P_2 aus (wie bisher)

While-Programme

Überblick

- k Eingaben in Variablen $x_1, \ldots, x_k$ hinterlegt
- Erwartete Semantik für Zuweisung (wie bisher)
- $P_1 \,;\, P_2$ führt P_1 und danach P_2 aus (wie bisher)
- **WHILE**$(x_i \neq 0)\,\{P'\}$ wiederholt P' bis $0 =$ aktueller Wert von x_i
  (Änderungen an x_i ändern Anzahl Schleifendurchläufe)

While-Programme

Überblick

- k Eingaben in Variablen $x_1, \ldots, x_k$ hinterlegt
- Erwartete Semantik für Zuweisung (wie bisher)
- $P_1 \,;\, P_2$ führt P_1 und danach P_2 aus (wie bisher)
- **WHILE**$(x_i \neq 0)\,\{P'\}$ wiederholt P' bis $0 =$ aktueller Wert von x_i
  (Änderungen an x_i ändern Anzahl Schleifendurchläufe)
- Funktionswert ist Wert von x_1 nach Ablauf Programms

While-Programme

§5.16 Definition (Programmsemantik; *program semantics*)

Für While-Programm P und $\max \operatorname{var}(P) \leq n$ ist **Semantik** von P partielle Funktion $\|P\|_n \colon \mathbb{N}^n \dashrightarrow \mathbb{N}^n$ für alle $a_1, \ldots, a_n \in \mathbb{N}$

- $\|x_i = x_\ell + z\|_n(a_1, \ldots, a_n) = (a_1, \ldots, a_{i-1}, a_\ell + z, a_{i+1}, \ldots, a_n)$

While-Programme

§5.16 Definition (Programmsemantik; *program semantics*)

Für While-Programm P und $\max \operatorname{var}(P) \leq n$ ist **Semantik** von P partielle Funktion $\|P\|_n \colon \mathbb{N}^n \dashrightarrow \mathbb{N}^n$ für alle $a_1, \ldots, a_n \in \mathbb{N}$

- $\|x_i = x_\ell + z\|_n(a_1, \ldots, a_n) = (a_1, \ldots, a_{i-1}, a_\ell + z, a_{i+1}, \ldots, a_n)$
- $\|P_1 \,;\, P_2\|_n(a_1, \ldots, a_n) = \|P_2\|_n\big(\|P_1\|_n(a_1, \ldots, a_n)\big)$

While-Programme

§5.16 Definition (Programmsemantik; *program semantics*)

Für While-Programm P und $\max \operatorname{var}(P) \leq n$ ist **Semantik** von P partielle Funktion $\|P\|_n \colon \mathbb{N}^n \dashrightarrow \mathbb{N}^n$ für alle $a_1, \ldots, a_n \in \mathbb{N}$

- $\|x_i = x_\ell + z\|_n(a_1, \ldots, a_n) = (a_1, \ldots, a_{i-1}, a_\ell + z, a_{i+1}, \ldots, a_n)$
- $\|P_1 \,;\, P_2\|_n(a_1, \ldots, a_n) = \|P_2\|_n\big(\|P_1\|_n(a_1, \ldots, a_n)\big)$
- $\|\textbf{WHILE}(x_i \neq 0)\,\{P'\}\|_n(a_1, \ldots, a_n)$

$$= \begin{cases} \|P'\|_n^t(a_1, \ldots, a_n) & \text{falls } t \in \mathbb{N} \text{ existiert und für alle } s < t \\ & \pi_i^{(n)}(\|P'\|_n^t(a_1, \ldots, a_n)) = 0 \\ & \pi_i^{(n)}(\|P'\|_n^s(a_1, \ldots, a_n)) \neq 0 \\ \text{undef} & \text{sonst} \end{cases}$$

While-Programme

Semantik der While-Schleife

- Finde Iterationsanzahl t mit
  - x_i enthält 0 nach t Iterationen
  - x_i enthält nicht 0 nach $s < t$ Iterationen

While-Programme

Semantik der While-Schleife

- Finde Iterationsanzahl t mit
  - x_i enthält 0 nach t Iterationen
  - x_i enthält nicht 0 nach $s < t$ Iterationen
- Gesamtberechnung *undefiniert* falls Teilberechnung *undefiniert*
  (undef = Endlosschleife)

While-Programme

Semantik der While-Schleife

- Finde Iterationsanzahl t mit
  - x_i enthält 0 nach t Iterationen
  - x_i enthält nicht 0 nach $s < t$ Iterationen
- Gesamtberechnung *undefiniert* falls Teilberechnung *undefiniert*
  (undef = Endlosschleife)

Beispiele

- $\|x_2 = x_1 + 5 \; ; \; x_1 = x_3 + 1\|_3(0, 3, 7) = (8, 5, 7)$

While-Programme

Semantik der While-Schleife

- Finde Iterationsanzahl t mit
  - x_i enthält 0 nach t Iterationen
  - x_i enthält nicht 0 nach $s < t$ Iterationen
- Gesamtberechnung *undefiniert* falls Teilberechnung *undefiniert*
  (undef = Endlosschleife)

Beispiele

- $\|x_2 = x_1 + 5 \,;\, x_1 = x_3 + 1\|_3(0,3,7) = (8,5,7)$
- $\|\textbf{WHILE}(x_1 \neq 0)\,\{x_2 = x_1 + 5 \,;\, x_1 = x_3 + 1\}\|_3(0,3,7) = (0,3,7)$

While-Programme

Semantik der While-Schleife

- Finde Iterationsanzahl t mit
  - x_i enthält 0 nach t Iterationen
  - x_i enthält nicht 0 nach $s < t$ Iterationen
- Gesamtberechnung *undefiniert* falls Teilberechnung *undefiniert*
  (undef = Endlosschleife)

Beispiele

- $\|x_2 = x_1 + 5 \,;\, x_1 = x_3 + 1\|_3(0,3,7) = (8,5,7)$
- $\|\textbf{WHILE}(x_1 \neq 0)\,\{x_2 = x_1 + 5 \,;\, x_1 = x_3 + 1\}\|_3(0,3,7) = (0,3,7)$
- $\|\textbf{WHILE}(x_1 \neq 0)\,\{x_2 = x_1 + 5 \,;\, x_1 = x_3 + 1\}\|_3(1,3,7) = \text{undef}$

While-Programme

§5.17 Definition (berechnete Funktion; *computed function*)

While-Programm P mit $\max \operatorname{var}(P) = n$ **berechnet** k-stellige partielle Funktion $|P|_k : \mathbb{N}^k \dashrightarrow \mathbb{N}$ mit $k \leq n$ gegeben für alle $a_1, \ldots, a_k \in \mathbb{N}$

$$|P|_k(a_1, \ldots, a_k) = \pi_1^{(n)}\big(\|P\|_n(a_1, \ldots, a_k, \underbrace{0, \ldots, 0}_{(n-k) \text{ mal}})\big)$$

While-Programme

§5.17 Definition (berechnete Funktion; *computed function*)

While-Programm P mit $\max \operatorname{var}(P) = n$ **berechnet** k-stellige partielle Funktion $|P|_k \colon \mathbb{N}^k \dashrightarrow \mathbb{N}$ mit $k \leq n$ gegeben für alle $a_1, \ldots, a_k \in \mathbb{N}$

$$|P|_k(a_1, \ldots, a_k) = \pi_1^{(n)}\big(\|P\|_n(a_1, \ldots, a_k, \underbrace{0, \ldots, 0}_{(n-k) \text{ mal}})\big)$$

Notizen

- Eingaben $a_1, \ldots, a_k$ in ersten k Variablen $x_1, \ldots, x_k$
- Weitere Variablen $x_{k+1}, \ldots, x_n$ initial 0
- Auswertung Programm mit dieser initialen Variablenbelegung
- Ergebnis ist Inhalt erster Variable x_1 nach Ablauf

While-Berechenbarkeit

§5.18 Definition (While-Berechenbarkeit; *While-computability*)

Partielle Funktion $f : \mathbb{N}^k \dashrightarrow \mathbb{N}$ **While-berechenbar**
falls While-Programm P mit $f = |P|_k$ existiert

While-Berechenbarkeit

Vollständig undefinierte partielle Funktion

$x_1 = x_1 + 1$ $(x_1 > 0)$

WHILE$(x_1 \neq 0)\,\{x_1 = x_1 + 1\}$ $(x_1 > 0)$

While-Berechenbarkeit

Vollständig undefinierte partielle Funktion

$x_1 = x_1 + 1$ $\qquad (x_1 > 0)$

$\mathbf{WHILE}(x_1 \neq 0)\,\{x_1 = x_1 + 1\}$ $\qquad (x_1 > 0)$

Auswertung für $a \in \mathbb{N}$

$$\begin{aligned} &\|x_1 = x_1 + 1\,;\, \mathbf{WHILE}(x_1 \neq 0)\,\{x_1 = x_1 + 1\}\|_1(a) \\ &= \|\mathbf{WHILE}(x_1 \neq 0)\,\{x_1 = x_1 + 1\}\|_1(a+1) \\ &= \text{undef} \end{aligned}$$

da $\|x_1 = x_1 + 1\|_1^t(a+1) = (a+1+t)$ für alle $t \in \mathbb{N}$

While-Berechenbarkeit

Iteration (Simulation von **LOOP**) (x_ℓ unbenutzt)

$x_\ell = x_i$; **WHILE**$(x_\ell \neq 0)\,\{P'\,;\, x_\ell = x_\ell - 1\}$ Schreibweise: **LOOP**$(x_i)\,\{P'\}$

While-Berechenbarkeit

Iteration (Simulation von **LOOP**) (x_ℓ unbenutzt)

$x_\ell = x_i$; **WHILE**$(x_\ell \neq 0)\,\{P'\,;\, x_\ell = x_\ell - 1\}$ Schreibweise: **LOOP**$(x_i)\,\{P'\}$

Notizen

- Jedes Loop-Programm damit auch While-Programm
  (für jedes Loop-Programm existiert äquivalentes While-Programm)
- Loop-berechenbare Funktionen sind also While-berechenbar

While-Berechenbarkeit

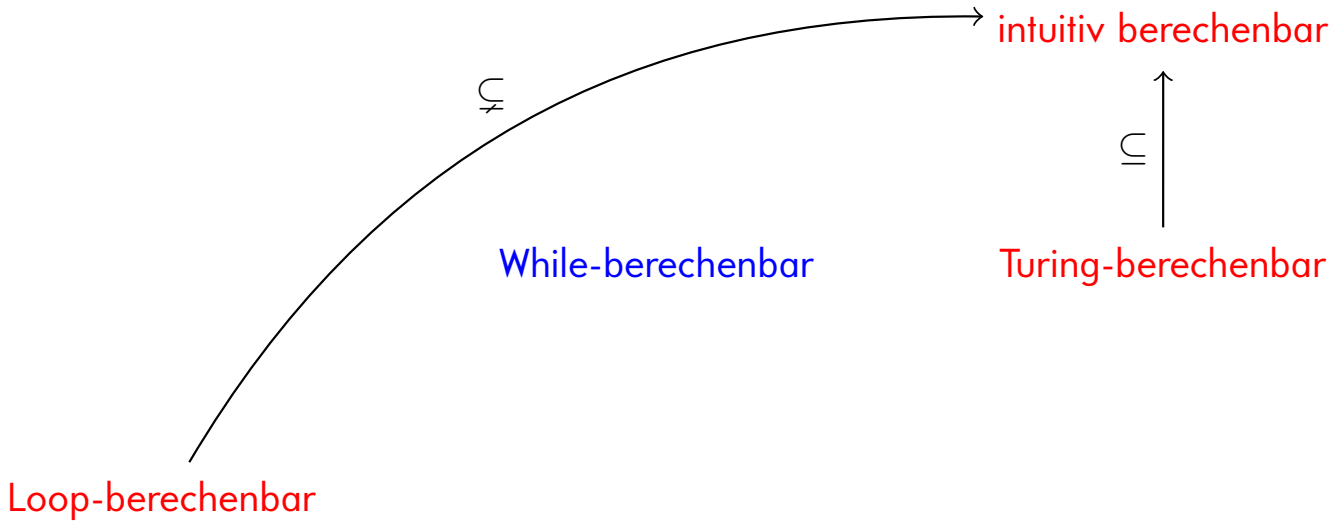
Iteration (Simulation von **LOOP**) (x_ℓ unbenutzt)

$x_\ell = x_i$; **WHILE**$(x_\ell \neq 0)\,\{P'\,;\, x_\ell = x_\ell - 1\}$ Schreibweise: **LOOP**$(x_i)\,\{P'\}$

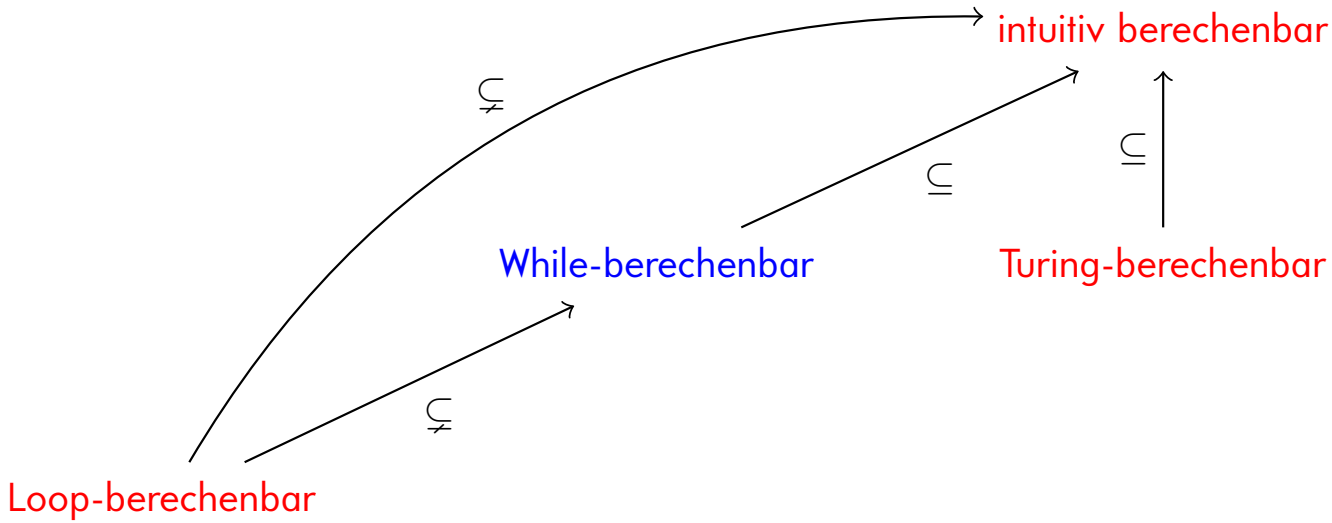
Notizen

- Jedes Loop-Programm damit auch While-Programm
  (für jedes Loop-Programm existiert äquivalentes While-Programm)
- Loop-berechenbare Funktionen sind also While-berechenbar
- Schreibweisen Loop-Programme erlaubt (**IF-THEN-ELSE**, etc.)
- Nicht jede While-berechenbare partielle Funktion
  Loop-berechenbar (z.B. vollständig undefinierte partielle Funktion)

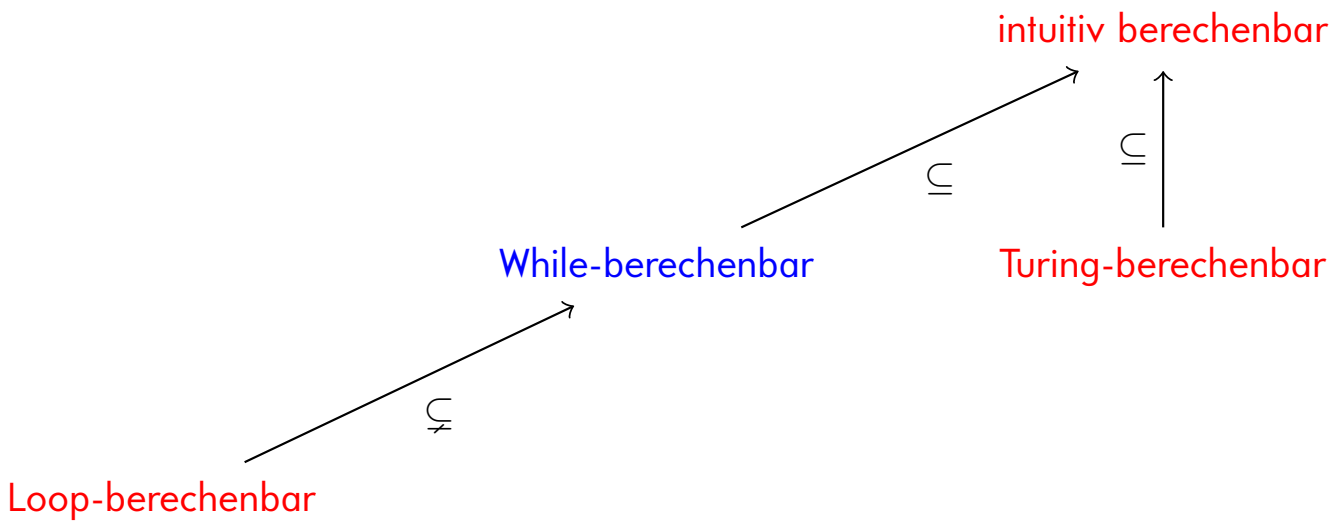
While-Berechenbarkeit



While-Berechenbarkeit



While-Berechenbarkeit



While-Berechenbarkeit

Komplexere Bedingung

(x_k unbenutzt)

$x_k = x_i + 1 \,;\, x_k = x_k - x_\ell$ ($x_k = 0$ gdw. $x_i < x_\ell$)

WHILE$(x_k \neq 0)\,\{$

$\quad P' \,;\, x_k = x_i + 1 \,;\, x_k = x_k - x_\ell$ ($x_k = 0$ gdw. $x_i < x_\ell$)

$\}$

Schreibweise: **WHILE**$(x_i \geq x_\ell)\,\{P'\}$

While-Berechenbarkeit

Komplexere Bedingung

(x_k unbenutzt)

$x_k = x_i + 1 \,;\, x_k = x_k - x_\ell$ ($x_k = 0$ gdw. $x_i < x_\ell$)

WHILE$(x_k \neq 0)\,\{$

$\quad P' \,;\, x_k = x_i + 1 \,;\, x_k = x_k - x_\ell$ ($x_k = 0$ gdw. $x_i < x_\ell$)

$\}$ Schreibweise: **WHILE**$(x_i \geq x_\ell)\,\{P'\}$

Ganzzahlige Division von x_i durch x_m in x_ℓ

(x_k unbenutzt)

$x_\ell = 0 \,;\, x_k = x_i$

WHILE$(x_k \geq x_m)\,\{$

$\quad x_\ell = x_\ell + 1 \,;\, x_k = x_k - x_m$

$\}$ Schreibweise: $x_\ell = x_i$ **DIV** x_m

While-Berechenbarkeit

Ganzzahliger Rest von x_i durch x_m in x_ℓ

$x_\ell = x_i$
WHILE$(x_\ell \geq x_m)\,\{x_\ell = x_\ell - x_m\}$

Schreibweise: $x_\ell = x_i$ **MOD** x_m

While-Berechenbarkeit

Ganzzahliger Rest von x_i durch x_m in x_ℓ

$x_\ell = x_i$
WHILE$(x_\ell \geq x_m)\,\{x_\ell = x_\ell - x_m\}$ Schreibweise: $x_\ell = x_i$ **MOD** x_m

Collatz-Iteration

WHILE$(x_1 > 1)\,\{$
  IF$(x_1$ **MOD** $2 = 0)\,\{x_1 = x_1$ **DIV** $2\}$ (halbiere x_1 falls gerade)
  ELSE $\{x_1 = 3 \cdot x_1 + 1\}$ (sonst verdreifache & addiere 1)
$\}$

While-Berechenbarkeit

Ganzzahliger Rest von x_i durch x_m in x_ℓ

$x_\ell = x_i$
WHILE$(x_\ell \geq x_m)\,\{x_\ell = x_\ell - x_m\}$

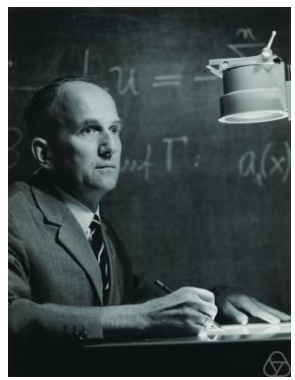
Schreibweise: $x_\ell = x_i$ **MOD** x_m

Collatz-Iteration

WHILE$(x_1 > 1)\,\{$
  IF$(x_1$ **MOD** $2 = 0)\,\{x_1 = x_1$ **DIV** $2\}$ (halbiere x_1 falls gerade)
  ELSE $\{x_1 = 3 \cdot x_1 + 1\}$ (sonst verdreifache & addiere 1)
$\}$

Lothar Collatz (* 1910; † 1990)

- Dtsch. Mathematiker
- Formulierte ungelöste Collatz-Behauptung
- Ehrendoktorwürde TU Dresden



© Konrad Jacobs

While-Berechenbarkeit

Fallunterscheidung $(n \in \mathbb{N})$

$$
\begin{array}{ll}
\textbf{IF}(x_i = 0)\,\{P_0\} & \text{(Fall 0)} \\
\textbf{ELSE}\ \{\textbf{IF}(x_i - 1 = 0)\,\{P_1\} & \text{(Fall 1)} \\
\qquad \textbf{ELSE}\ \{\cdots & \\
\qquad\qquad \textbf{ELSE}\ \{\textbf{IF}(x_i - n = 0)\,\{P_n\} & \text{(Fall } n\text{)} \\
\qquad\qquad\qquad \textbf{ELSE}\ \{P\} & \\
\qquad\qquad \} & \\
\qquad \ldots & \\
\qquad \} & \\
\} &
\end{array}
$$

Schreibweise: $\textbf{CASE}(x_i)\ \textbf{OF}\ 0 : \{P_0\} \ldots n : \{P_n\}\ \textbf{ELSE}\ \{P\}$

Zusammenfassung

- Ackermann-Funktion
- Loop-berechenbar $\subsetneq$ intuitiv berechenbar
- While-Berechenbarkeit

Dritte Übungsserie bereits im Moodle