

Berechenbarkeit

Vorlesung 7: Rekursive Funktionen II

5. Juni 2025

Termine — Modul Berechenbarkeit

ÜBUNGEN	PRÜFUNG	VORLESUNG
3.6. Übung 4 A-Woche	4.6. _____	5.6. Rekursion II (Übungsblatt 5)
10.6. Übung 5 B-Woche (Montag Feiertag)	11.6. _____	12.6. Entscheidbarkeit
17.6. Übung 5 A-Woche	18.6. _____	19.6. Unentscheidbarkeit (Übungsblatt 6)
24.6. Übung 6 B-Woche	25.6. _____	26.6. Spez. Probleme
1.7. Übung 6 A-Woche	2.7. _____	3.7. Klasse P
8.7. Abschlussübung beide Wochen	9.7. _____	10.7. NP-Vollständigkeit
15.7. _____	16.7. Prüfung ab 13:30 Uhr in AudiMax & Hs. 9	17.7. _____

Wiederholung: primitive Rekursion

Definition (§6.7 primitiv rek. Fkt.; *primitive recursive function*)

Genau folgende Funktionen sind **primitiv rekursiv**

- **Rekursive Basisfunktionen** (Konstanten, Projektionen, Nachfolger)

Wiederholung: primitive Rekursion

Definition (§6.7 primitiv rek. Fkt.; *primitive recursive function*)

Genau folgende Funktionen sind **primitiv rekursiv**

- **Rekursive Basisfunktionen** (Konstanten, Projektionen, Nachfolger)
- **Komposition** $f\langle g_1, \dots, g_m \rangle$ primitiv rekursiver Funktionen f, g_1, \dots, g_m
- **Schema primitiver Rekursion** definierte Funktionen $\text{pr}[f, g]$ für primitiv rekursive Funktionen f, g

Keine weiteren primitiv rekursiven Funktionen

Primitiv rekursive Funktionen

Falsche Variante der Subtraktion

$$\text{sub} = \mathbf{pr}[0^{(1)}, \text{nf}\langle \pi_1^{(3)} \rangle]$$

wesentliche Rekursion $(a + 1) - b = (a - b) + 1$

$$\text{sub}(0, b) = 0$$

$$\text{sub}(a + 1, b) = \text{nf}(\text{sub}(a, b))$$

Warum nicht gewünschte Funktion?

Primitiv rekursive Funktionen

Falsche Variante der Subtraktion

$$\text{sub} = \mathbf{pr}[0^{(1)}, \text{nf}\langle \pi_1^{(3)} \rangle]$$

wesentliche Rekursion $(a + 1) - b = (a - b) + 1$

$$\text{sub}(0, b) = 0$$

$$\text{sub}(a + 1, b) = \text{nf}(\text{sub}(a, b))$$

Warum nicht gewünschte Funktion?

Denn $\text{sub}(a, b) = a$ für alle $a, b \in \mathbb{N}$

(trivialer Induktionsbeweis)

Primitiv rekursive Funktionen

Berechnung für $\text{sub}(2, 1)$

$$\text{sub}(2, 1) = \mathbf{pr}[0^{(1)}, \text{nf}\langle \pi_1^{(3)} \rangle](2, 1)$$

Primitiv rekursive Funktionen

Berechnung für $\text{sub}(2, 1)$

$$\begin{aligned}\text{sub}(2, 1) &= \mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](2, 1) \\ &= \text{nf}\langle\pi_1^{(3)}\rangle\left(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](1, 1), 1, 1\right)\end{aligned}$$

Primitiv rekursive Funktionen

Berechnung für $\text{sub}(2, 1)$

$$\begin{aligned}\text{sub}(2, 1) &= \mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](2, 1) \\ &= \text{nf}\langle\pi_1^{(3)}\rangle\left(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](1, 1), 1, 1\right) \\ &= \text{nf}\left(\pi_1^{(3)}\left(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](1, 1), 1, 1\right)\right)\end{aligned}$$

Primitiv rekursive Funktionen

Berechnung für $\text{sub}(2, 1)$

$$\begin{aligned}\text{sub}(2, 1) &= \mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](2, 1) \\ &= \text{nf}\langle\pi_1^{(3)}\rangle(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](1, 1), 1, 1) \\ &= \text{nf}\left(\pi_1^{(3)}(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](1, 1), 1, 1)\right) \\ &= \mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](1, 1) + 1\end{aligned}$$

Primitiv rekursive Funktionen

Berechnung für $\text{sub}(2, 1)$

$$\begin{aligned}\text{sub}(2, 1) &= \mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](2, 1) \\ &= \text{nf}\langle\pi_1^{(3)}\rangle\left(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](1, 1), 1, 1\right) \\ &= \text{nf}\left(\pi_1^{(3)}\left(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](1, 1), 1, 1\right)\right) \\ &= \mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](1, 1) + 1 \\ &= \text{nf}\langle\pi_1^{(3)}\rangle\left(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](0, 1), 0, 1\right) + 1\end{aligned}$$

Primitiv rekursive Funktionen

Berechnung für $\text{sub}(2, 1)$

$$\begin{aligned}\text{sub}(2, 1) &= \mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](2, 1) \\ &= \text{nf}\langle\pi_1^{(3)}\rangle\left(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](1, 1), 1, 1\right) \\ &= \text{nf}\left(\pi_1^{(3)}\left(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](1, 1), 1, 1\right)\right) \\ &= \mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](1, 1) + 1 \\ &= \text{nf}\langle\pi_1^{(3)}\rangle\left(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](0, 1), 0, 1\right) + 1 \\ &= \text{nf}\left(\pi_1^{(3)}\left(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](0, 1), 0, 1\right)\right) + 1\end{aligned}$$

Primitiv rekursive Funktionen

Berechnung für $\text{sub}(2, 1)$

$$\begin{aligned}\text{sub}(2, 1) &= \mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](2, 1) \\ &= \text{nf}\langle\pi_1^{(3)}\rangle\left(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](1, 1), 1, 1\right) \\ &= \text{nf}\left(\pi_1^{(3)}\left(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](1, 1), 1, 1\right)\right) \\ &= \mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](1, 1) + 1 \\ &= \text{nf}\langle\pi_1^{(3)}\rangle\left(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](0, 1), 0, 1\right) + 1 \\ &= \text{nf}\left(\pi_1^{(3)}\left(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](0, 1), 0, 1\right)\right) + 1 \\ &= \mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](0, 1) + 2\end{aligned}$$

Primitiv rekursive Funktionen

Berechnung für $\text{sub}(2, 1)$

$$\begin{aligned}\text{sub}(2, 1) &= \mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](2, 1) \\ &= \text{nf}\langle\pi_1^{(3)}\rangle\left(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](1, 1), 1, 1\right) \\ &= \text{nf}\left(\pi_1^{(3)}\left(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](1, 1), 1, 1\right)\right) \\ &= \mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](1, 1) + 1 \\ &= \text{nf}\langle\pi_1^{(3)}\rangle\left(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](0, 1), 0, 1\right) + 1 \\ &= \text{nf}\left(\pi_1^{(3)}\left(\mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](0, 1), 0, 1\right)\right) + 1 \\ &= \mathbf{pr}[0^{(1)}, \text{nf}\langle\pi_1^{(3)}\rangle](0, 1) + 2 \\ &= 0^{(1)}(1) + 2 = 0 + 2 = 2\end{aligned}$$

Loop-Berechenbarkeit vs. primitive Rekursion

Ansatz Loop-berechenbar impliziert primitiv rekursiv

- Semantik $\|P\|_n: \mathbb{N}^n \rightarrow \mathbb{N}^n$ Loop-Programm P
- Primitiv rekursive Funktion $f: \mathbb{N}^n \rightarrow \mathbb{N}$
- Primitiv rekursive Variante $\|P\|_n$ benötigt Kodierung von \mathbb{N}^n in \mathbb{N}
(z.B. Kellerspeicher)

Loop-Berechenbarkeit vs. primitive Rekursion

Binomialkoeffizient

$$\text{bk2} = \mathbf{pr}[0^{(0)}, \text{add}\langle \pi_1^{(2)}, \pi_2^{(2)} \rangle]$$

$$\text{wesentliche Rekursion } \binom{a+1}{2} = \binom{a}{1} + \binom{a}{2} = a + \binom{a}{2}$$

$$\text{bk2}(0) = 0$$

$$\text{bk2}(a+1) = a + \text{bk2}(a)$$

Loop-Berechenbarkeit vs. primitive Rekursion

Binomialkoeffizient

$$\text{bk2} = \mathbf{pr}[0^{(0)}, \text{add}\langle \pi_1^{(2)}, \pi_2^{(2)} \rangle]$$

$$\text{wesentliche Rekursion } \binom{a+1}{2} = \binom{a}{1} + \binom{a}{2} = a + \binom{a}{2}$$

$$\text{bk2}(0) = 0$$

$$\text{bk2}(a+1) = a + \text{bk2}(a)$$

Paarung

$$c = \text{add}\langle \pi_1^{(2)}, \text{bk2}\langle \text{nf}\langle \text{add}\langle \pi_1^{(2)}, \pi_2^{(2)} \rangle \rangle \rangle \rangle$$

$$c(a, b) = a + \text{bk2}(a + b + 1) = a + \binom{a + b + 1}{2}$$

Loop-Berechenbarkeit vs. primitive Rekursion

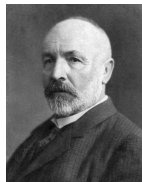
§7.1 Theorem (Cantorsche Paarungsfunktion)

$c: \mathbb{N}^2 \rightarrow \mathbb{N}$ bijektiv

$a \setminus b$	0	1	2	3	4	5
0	0	1	3	6	10	15
1	2	4	7	11	16	22
2	5	8	12	17	23	30
3	9	13	18	24	31	39
4	14	19	25	32	40	49
5	20	26	33	41	50	60

Georg Cantor (* 1845; † 1918)

- Dtsch. Mathematiker
- Begründer moderner Mengenlehre
- Kardinal- & Ordinalzahlen



Loop-Berechenbarkeit vs. primitive Rekursion

Notizen

- Offenbar $a \leq c(a, b)$ und $b \leq c(a, b)$ für alle $a, b \in \mathbb{N}$
- Kodierung Paare natürlicher Zahlen möglich

$$\begin{pmatrix} a \\ b \end{pmatrix} = c(a, b)$$

Loop-Berechenbarkeit vs. primitive Rekursion

Notizen

- Offenbar $a \leq c(a, b)$ und $b \leq c(a, b)$ für alle $a, b \in \mathbb{N}$
- Kodierung Paare natürlicher Zahlen möglich

$$\begin{pmatrix} a \\ b \end{pmatrix} = c(a, b)$$

- Erweiterbar auf beliebige n -Tupel

$$\begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} = c\left(a_1, \begin{pmatrix} a_2 \\ \vdots \\ a_n \end{pmatrix}\right) = c\left(a_1, c\left(a_2, \dots, c(a_{n-1}, a_n) \dots\right)\right)$$

- Primitiv rekursiv für alle $n \in \mathbb{N}$

Loop-Berechenbarkeit vs. primitive Rekursion

Dekodierung

- Funktionen $\Pi_1, \Pi_2: \mathbb{N} \rightarrow \mathbb{N}$

$$\Pi_1(c(a, b)) = \Pi_1 \begin{pmatrix} a \\ b \end{pmatrix} = a$$

$$\Pi_2(c(a, b)) = \Pi_2 \begin{pmatrix} a \\ b \end{pmatrix} = b$$

Loop-Berechenbarkeit vs. primitive Rekursion

Dekodierung

- Funktionen $\Pi_1, \Pi_2: \mathbb{N} \rightarrow \mathbb{N}$

$$\Pi_1(c(a, b)) = \Pi_1 \begin{pmatrix} a \\ b \end{pmatrix} = a$$

$$\Pi_2(c(a, b)) = \Pi_2 \begin{pmatrix} a \\ b \end{pmatrix} = b$$

- Längere Tupel $\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} a_1 \\ \begin{pmatrix} a_2 \\ a_3 \end{pmatrix} \end{pmatrix}$ ebenso dekodierbar

$$a_1 = \Pi_1 \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \quad a_2 = \Pi_1 \left(\Pi_2 \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \right) \quad a_3 = \Pi_2 \left(\Pi_2 \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \right)$$

Loop-Berechenbarkeit vs. primitive Rekursion

§7.2 Definition (beschränkter max-Operator; *bounded maximum*)

Sei $P: \mathbb{N}^{n+1} \rightarrow \{0, 1\}$ Prädikat und $\max \emptyset = 0$.

$$\max_P(a, a_1, \dots, a_n) = \max \{b \leq a \mid P(b, a_1, \dots, a_n) = 1\}$$

Loop-Berechenbarkeit vs. primitive Rekursion

§7.2 Definition (beschränkter max-Operator; *bounded maximum*)

Sei $P: \mathbb{N}^{n+1} \rightarrow \{0, 1\}$ Prädikat und $\max \emptyset = 0$.

$$\max_P(a, a_1, \dots, a_n) = \max \{b \leq a \mid P(b, a_1, \dots, a_n) = 1\}$$

Notizen

- $\max_P(a, a_1, \dots, a_n)$ maximaler Wert $b \leq a$ mit $P(b, a_1, \dots, a_n) = 1$
- Liefert 0 falls kein Wert $b \leq a$ Prädikat $P(b, a_1, \dots, a_n)$ erfüllt

Loop-Berechenbarkeit vs. primitive Rekursion

§7.3 Theorem

$\max_P: \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ primitiv rek. falls $P: \mathbb{N}^{n+1} \rightarrow \{0,1\}$ primitiv rek.

Loop-Berechenbarkeit vs. primitive Rekursion

§7.3 Theorem

$\max_P: \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ primitiv rek. falls $P: \mathbb{N}^{n+1} \rightarrow \{0,1\}$ primitiv rek.

Beweis

Offenbar $\max_P(0, a_1, \dots, a_n) = 0$ und

$$\max_P(a+1, a_1, \dots, a_n) = \begin{cases} a+1 & \text{falls } P(a+1, a_1, \dots, a_n) = 1 \\ \max_P(a, a_1, \dots, a_n) & \text{sonst} \end{cases}$$

Loop-Berechenbarkeit vs. primitive Rekursion

§7.3 Theorem

$\max_P: \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ primitiv rek. falls $P: \mathbb{N}^{n+1} \rightarrow \{0, 1\}$ primitiv rek.

Beweis

Offenbar $\max_P(0, a_1, \dots, a_n) = 0$ und

$$\max_P(a+1, a_1, \dots, a_n) = \begin{cases} a+1 & \text{falls } P(a+1, a_1, \dots, a_n) = 1 \\ \max_P(a, a_1, \dots, a_n) & \text{sonst} \end{cases}$$

Fallunterscheidung äquivalent zu

$$\max_P(a, a_1, \dots, a_n) + P(a+1, a_1, \dots, a_n) \cdot (a+1 - \max_P(a, a_1, \dots, a_n))$$

□

Loop-Berechenbarkeit vs. primitive Rekursion

§7.3 Theorem

$\max_P: \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ primitiv rek. falls $P: \mathbb{N}^{n+1} \rightarrow \{0, 1\}$ primitiv rek.

Beweis

Offenbar $\max_P(0, a_1, \dots, a_n) = 0$ und

$$\max_P(a+1, a_1, \dots, a_n) = \begin{cases} a+1 & \text{falls } P(a+1, a_1, \dots, a_n) = 1 \\ \max_P(a, a_1, \dots, a_n) & \text{sonst} \end{cases}$$

Fallunterscheidung äquivalent zu

$$\max_P(a, a_1, \dots, a_n) + P(a+1, a_1, \dots, a_n) \cdot (a+1 - \max_P(a, a_1, \dots, a_n))$$

□

$$\max_P = \mathbf{pr}[0^{(n)}, \text{add}\langle \pi_1^{(n+2)}, \text{mult}\langle P\langle \text{nf}\langle \pi_2^{(n+2)} \rangle, \pi_3^{(n+2)}, \dots, \pi_{n+2}^{(n+2)} \rangle, \\ \text{sub}\langle \text{nf}\langle \pi_2^{(n+2)} \rangle, \pi_1^{(n+2)} \rangle \rangle \rangle]$$

Loop-Berechenbarkeit vs. primitive Rekursion

§7.4 Definition (beschränkter \exists -Quantor; *bounded \exists -quantifier*)

Sei $P: \mathbb{N}^{n+1} \rightarrow \{0, 1\}$ Prädikat

$$\exists_P(a, a_1, \dots, a_n) = \begin{cases} 1 & \text{falls } \exists b \leq a: P(b, a_1, \dots, a_n) = 1 \\ 0 & \text{sonst} \end{cases}$$

Loop-Berechenbarkeit vs. primitive Rekursion

§7.4 Definition (beschränkter \exists -Quantor; *bounded \exists -quantifier*)

Sei $P: \mathbb{N}^{n+1} \rightarrow \{0, 1\}$ Prädikat

$$\exists_P(a, a_1, \dots, a_n) = \begin{cases} 1 & \text{falls } \exists b \leq a: P(b, a_1, \dots, a_n) = 1 \\ 0 & \text{sonst} \end{cases}$$

Notizen

- $\exists_P(a, a_1, \dots, a_n) = 1$, falls $b \leq a$ mit $P(b, a_1, \dots, a_n) = 1$ existiert
- Liefert 0 falls kein Wert $b \leq a$ Prädikat $P(b, a_1, \dots, a_n)$ erfüllt

Loop-Berechenbarkeit vs. primitive Rekursion

Maximum

$$\text{max} = \text{add} \langle \text{sub} \langle \pi_1^{(2)}, \pi_2^{(2)} \rangle, \pi_2^{(2)} \rangle$$

$$\text{max}(a, b) = (a - b) + b$$

2 Fälle: Falls $a \geq b$, dann ist $(a - b) + b = a$ und damit $\text{max}(a, b) = a$.
Sonst ist $a - b = 0$ und damit $\text{max}(a, b) = b$.

Loop-Berechenbarkeit vs. primitive Rekursion

§7.5 Theorem

$\exists p: \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ primitiv rekursiv falls $P: \mathbb{N}^{n+1} \rightarrow \{0,1\}$ primitiv rekursiv

Loop-Berechenbarkeit vs. primitive Rekursion

§7.5 Theorem

$\exists_P: \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ primitiv rekursiv falls $P: \mathbb{N}^{n+1} \rightarrow \{0,1\}$ primitiv rekursiv

Beweis

Offenbar $\exists_P(0, a_1, \dots, a_n) = P(0, a_1, \dots, a_n)$ und

$$\exists_P(\alpha + 1, a_1, \dots, a_n) = \begin{cases} 1 & \text{falls } P(\alpha + 1, a_1, \dots, a_n) = 1 \\ \exists_P(\alpha, a_1, \dots, a_n) & \text{sonst} \end{cases}$$

Loop-Berechenbarkeit vs. primitive Rekursion

§7.5 Theorem

$\exists_P: \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ primitiv rekursiv falls $P: \mathbb{N}^{n+1} \rightarrow \{0,1\}$ primitiv rekursiv

Beweis

Offenbar $\exists_P(0, a_1, \dots, a_n) = P(0, a_1, \dots, a_n)$ und

$$\exists_P(a+1, a_1, \dots, a_n) = \begin{cases} 1 & \text{falls } P(a+1, a_1, \dots, a_n) = 1 \\ \exists_P(a, a_1, \dots, a_n) & \text{sonst} \end{cases}$$

Fallunterscheidung äquivalent zu

$$\max(P(a+1, a_1, \dots, a_n), \exists_P(a, a_1, \dots, a_n))$$



Loop-Berechenbarkeit vs. primitive Rekursion

§7.5 Theorem

$\exists_P: \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ primitiv rekursiv falls $P: \mathbb{N}^{n+1} \rightarrow \{0,1\}$ primitiv rekursiv

Beweis

Offenbar $\exists_P(0, a_1, \dots, a_n) = P(0, a_1, \dots, a_n)$ und

$$\exists_P(a+1, a_1, \dots, a_n) = \begin{cases} 1 & \text{falls } P(a+1, a_1, \dots, a_n) = 1 \\ \exists_P(a, a_1, \dots, a_n) & \text{sonst} \end{cases}$$

Fallunterscheidung äquivalent zu

$$\max(P(a+1, a_1, \dots, a_n), \exists_P(a, a_1, \dots, a_n))$$

□

$$\exists_P = \mathbf{pr} \left[P \langle 0^{(n)}, \pi_1^{(n)}, \dots, \pi_n^{(n)} \rangle, \right. \\ \left. \max \langle P \langle \text{nf} \langle \pi_2^{(n+2)} \rangle, \pi_3^{(n+2)}, \dots, \pi_{n+2}^{(n+2)} \rangle, \pi_1^{(n+2)} \rangle \right]$$

Loop-Berechenbarkeit vs. primitive Rekursion

Dekodierung Paare $\begin{pmatrix} a \\ b \end{pmatrix}$

- Definiere Prädikat $C: \mathbb{N}^3 \rightarrow \{0, 1\}$ mit Hilfe von c

$$C(a, b, d) = \left(1 - (c(a, b) - d)\right) \cdot \left(1 - (d - c(a, b))\right)$$

- C primitiv rekursiv

Loop-Berechenbarkeit vs. primitive Rekursion

Dekodierung Paare $\begin{pmatrix} a \\ b \end{pmatrix}$

- Definiere Prädikat $C: \mathbb{N}^3 \rightarrow \{0, 1\}$ mit Hilfe von c

$$C(a, b, d) = \left(1 - (c(a, b) - d)\right) \cdot \left(1 - (d - c(a, b))\right)$$

- C primitiv rekursiv
- $C(a, b, d) = 0$ falls $c(a, b) > d$

Loop-Berechenbarkeit vs. primitive Rekursion

Dekodierung Paare $\begin{pmatrix} a \\ b \end{pmatrix}$

- Definiere Prädikat $C: \mathbb{N}^3 \rightarrow \{0, 1\}$ mit Hilfe von c

$$C(a, b, d) = \left(1 - (c(a, b) - d)\right) \cdot \left(1 - (d - c(a, b))\right)$$

- C primitiv rekursiv
- $C(a, b, d) = 0$ falls $c(a, b) > d$
- $C(a, b, d) = 0$ falls $c(a, b) < d$

Loop-Berechenbarkeit vs. primitive Rekursion

Dekodierung Paare $\begin{pmatrix} a \\ b \end{pmatrix}$

- Definiere Prädikat $C: \mathbb{N}^3 \rightarrow \{0, 1\}$ mit Hilfe von c

$$C(a, b, d) = \left(1 - (c(a, b) - d)\right) \cdot \left(1 - (d - c(a, b))\right)$$

- C primitiv rekursiv
- $C(a, b, d) = 0$ falls $c(a, b) > d$
- $C(a, b, d) = 0$ falls $c(a, b) < d$
- $C(a, b, d) = 1$ falls $c(a, b) = d$

Loop-Berechenbarkeit vs. primitive Rekursion

Dekodierung Paare $\begin{pmatrix} a \\ b \end{pmatrix}$

- Definiere Prädikat $C: \mathbb{N}^3 \rightarrow \{0, 1\}$ mit Hilfe von c

$$C(a, b, d) = \left(1 - (c(a, b) - d)\right) \cdot \left(1 - (d - c(a, b))\right)$$

- C primitiv rekursiv
- $C(a, b, d) = 0$ falls $c(a, b) > d$
- $C(a, b, d) = 0$ falls $c(a, b) < d$
- $C(a, b, d) = 1$ falls $c(a, b) = d$
- Also $C(a, b, d) = 1$ gdw. $c(a, b) = d$

Loop-Berechenbarkeit vs. primitive Rekursion

Dekodierfunktionen

$$C'(b, a, d) = C(a, b, d) \qquad C' = C\langle \pi_2^{(3)}, \pi_1^{(3)}, \pi_3^{(3)} \rangle$$

$$E'(a, b, d) = \exists_{C'}(b, a, d) = \begin{cases} 1 & \text{falls } \exists y \leq b: C(a, y, d) = 1 \\ 0 & \text{sonst} \end{cases}$$

$$\begin{aligned} \Pi'_1(a, b, d) &= \max_{E'}(a, b, d) \\ &= \max\{x \leq a \mid E'(x, b, d) = 1\} \\ &= \max\{x \leq a \mid \exists y \leq b: C(x, y, d) = 1\} \\ &= \max\{x \leq a \mid \exists y \leq b: c(x, y) = d\} \end{aligned}$$

Loop-Berechenbarkeit vs. primitive Rekursion

Dekodierfunktionen

$$E(b, a, d) = \exists_C(a, b, d) = \begin{cases} 1 & \text{falls } \exists x \leq a: C(x, b, d) = 1 \\ 0 & \text{sonst} \end{cases}$$

$$\begin{aligned} \Pi'_2(a, b, d) &= \max_E(b, a, d) \\ &= \max\{y \leq b \mid E(y, a, d) = 1\} \\ &= \max\{y \leq b \mid \exists x \leq a: C(x, y, d) = 1\} \\ &= \max\{y \leq b \mid \exists x \leq a: c(x, y) = d\} \end{aligned}$$

Loop-Berechenbarkeit vs. primitive Rekursion

§7.6 Theorem (Dekodierung)

$\Pi_1: \mathbb{N} \rightarrow \mathbb{N}$ und $\Pi_2: \mathbb{N} \rightarrow \mathbb{N}$ primitiv rekursiv

Loop-Berechenbarkeit vs. primitive Rekursion

§7.6 Theorem (Dekodierung)

$\Pi_1: \mathbb{N} \rightarrow \mathbb{N}$ und $\Pi_2: \mathbb{N} \rightarrow \mathbb{N}$ primitiv rekursiv

Beweis

Beide Funktionen sind primitiv rekursiv da

$$\Pi_1(d) = \Pi'_1(d, d, d) \quad \text{und} \quad \Pi_2(d) = \Pi'_2(d, d, d) \quad \text{für alle } d \in \mathbb{N}$$

Loop-Berechenbarkeit vs. primitive Rekursion

§7.6 Theorem (Dekodierung)

$\Pi_1: \mathbb{N} \rightarrow \mathbb{N}$ und $\Pi_2: \mathbb{N} \rightarrow \mathbb{N}$ primitiv rekursiv

Beweis

Beide Funktionen sind primitiv rekursiv da

$$\Pi_1(d) = \Pi'_1(d, d, d) \quad \text{und} \quad \Pi_2(d) = \Pi'_2(d, d, d) \quad \text{für alle } d \in \mathbb{N}$$

Da $a \leq c(a, b)$ und $b \leq c(a, b)$ wird d geeignet dekodiert. \square

Loop-Berechenbarkeit vs. primitive Rekursion

§7.6 Theorem (Dekodierung)

$\Pi_1: \mathbb{N} \rightarrow \mathbb{N}$ und $\Pi_2: \mathbb{N} \rightarrow \mathbb{N}$ primitiv rekursiv

Beweis

Beide Funktionen sind primitiv rekursiv da

$$\Pi_1(d) = \Pi'_1(d, d, d) \quad \text{und} \quad \Pi_2(d) = \Pi'_2(d, d, d) \quad \text{für alle } d \in \mathbb{N}$$

Da $a \leq c(a, b)$ und $b \leq c(a, b)$ wird d geeignet dekodiert. \square

Notiz

- Verwenden Vektornotation und greifen direkt auf Komponenten zu

Loop-Berechenbarkeit vs. primitive Rekursion

§7.7 Theorem

Jede Loop-berechenbare Funktion ist primitiv rekursiv

Loop-Berechenbarkeit vs. primitive Rekursion

§7.7 Theorem

Jede Loop-berechenbare Funktion ist primitiv rekursiv

Beweis (1/4)

Zeigen $\text{sem}_P: \mathbb{N}^n \rightarrow \mathbb{N}$ primitiv rekursiv für jedes Loop-Programm P mit $\max \text{var}(P) = n$ per Induktion über P . Für alle $a_1, \dots, a_n \in \mathbb{N}$

$$\begin{aligned} \text{sem}_P(a_1, \dots, a_n) &= c(b_1, \dots, b_n) \\ \iff \|P\|_n(a_1, \dots, a_n) &= (b_1, \dots, b_n) \end{aligned}$$

$\text{sem}_P = c\langle \text{nf}\langle \pi_2^{(2)} \rangle, \pi_2^{(2)} \rangle$ für $n = 2$ und $x_1 = x_2 + 1$

Loop-Berechenbarkeit vs. primitive Rekursion

§7.7 Theorem

Jede Loop-berechenbare Funktion ist primitiv rekursiv

Beweis (1/4)

Zeigen $\text{sem}_P: \mathbb{N}^n \rightarrow \mathbb{N}$ primitiv rekursiv für jedes Loop-Programm P mit $\max \text{var}(P) = n$ per Induktion über P . Für alle $a_1, \dots, a_n \in \mathbb{N}$

$$\begin{aligned} \text{sem}_P(a_1, \dots, a_n) &= c(b_1, \dots, b_n) \\ \iff \|P\|_n(a_1, \dots, a_n) &= (b_1, \dots, b_n) \end{aligned}$$

- Sei P Zuweisung $x_i = x_\ell + z$. Dann

$$\text{sem}_P(a_1, \dots, a_n) = c(a_1, \dots, a_{i-1}, a_\ell + z, a_{i+1}, \dots, a_n)$$

$$\text{sem}_P = c\langle \text{nf}\langle \pi_2^{(2)} \rangle, \pi_2^{(2)} \rangle \text{ für } n = 2 \text{ und } x_1 = x_2 + 1$$

Loop-Berechenbarkeit vs. primitive Rekursion

Beweis (2/4)

- Sei $P = P_1 ; P_2$. Dann sem_{P_1} und sem_{P_2} primitiv rekursiv gemäß IH

$$\begin{aligned}\text{sem}_P(a_1, \dots, a_n) &= \text{sem}_{P_2} \left(\Pi_1(\text{sem}_{P_1}(a_1, \dots, a_n)), \right. \\ &\quad \dots \\ &\quad \left. \Pi_2(\dots \Pi_2(\text{sem}_{P_1}(a_1, \dots, a_n)) \dots) \right) \\ &= \text{sem}_{P_2}(b_1, \dots, b_n)\end{aligned}$$

$$\text{mit } \text{sem}_{P_1}(a_1, \dots, a_n) = c(b_1, \dots, b_n)$$

$$\text{sem}_P = (\text{sem}_{P_2} \langle \Pi_1, \Pi_2 \rangle) \langle \text{sem}_{P_1} \rangle \text{ für } n = 2$$

Loop-Berechenbarkeit vs. primitive Rekursion

Beweis (3/4)

- Sei $P = \text{LOOP}(x_i) \{P'\}$. Dann $\text{sem}_{P'}$ primitiv rekursiv gemäß IH. Definiere Funktion

$$\begin{aligned}f(0, a_1, \dots, a_n) &= c(a_1, \dots, a_n) \\f(a+1, a_1, \dots, a_n) &= \text{sem}_{P'}(b_1, \dots, b_n)\end{aligned}$$

wobei $f(a, a_1, \dots, a_n) = c(b_1, \dots, b_n)$. Dann ist

$$\text{sem}_P(a_1, \dots, a_n) = f(a_i, a_1, \dots, a_n)$$

$$\text{sem}_P = \left(\text{pr} \left[c, (\text{sem}_{P'} \langle \Pi_1, \Pi_2 \rangle) \langle \pi_1^{(4)} \rangle \right] \right) \langle \pi_2^{(2)}, \pi_1^{(2)}, \pi_2^{(2)} \rangle \text{ für } n = i = 2$$

Loop-Berechenbarkeit vs. primitive Rekursion

Beweis (4/4)

Sei $f: \mathbb{N}^k \rightarrow \mathbb{N}$ Loop-berechenbar via P ($k \leq n$)

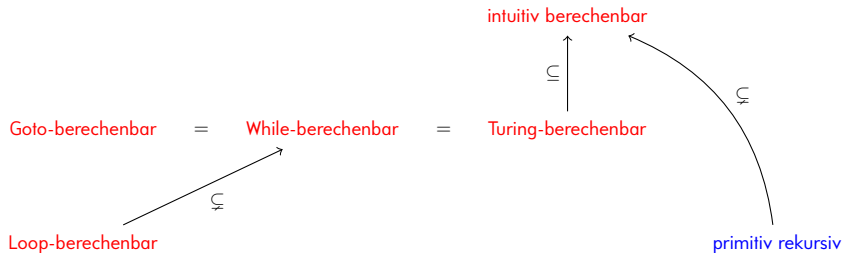
Für alle $a_1, \dots, a_k \in \mathbb{N}$

$$\begin{aligned} f(a_1, \dots, a_k) &= |P|_k(a_1, \dots, a_k) \\ &= \pi_1^{(n)}(\|P\|(a_1, \dots, a_k, 0, \dots, 0)) \\ &= \Pi_1(\text{sem}_P(a_1, \dots, a_k, 0, \dots, 0)) \end{aligned}$$

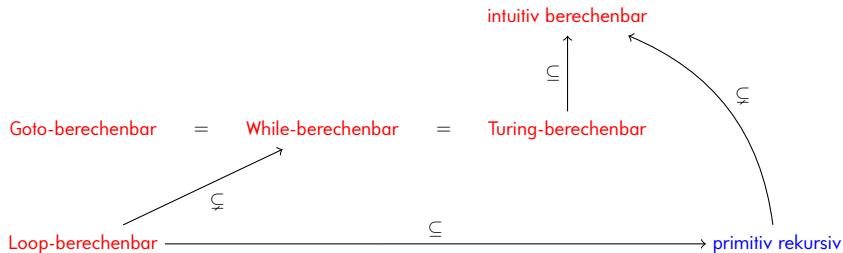
Damit f primitiv rekursiv



Loop-Berechenbarkeit vs. primitive Rekursion



Loop-Berechenbarkeit vs. primitive Rekursion



Loop-Berechenbarkeit vs. primitive Rekursion

§7.8 Lemma

Für alle $k, k' \in \mathbb{N}$ und Loop-Programme P existiert Loop-Programm P' mit $\max \text{var}(P') = n$ und

- $|P'|_k = |P|_k$ (gleiche berechnete Funktion)
- $\pi_i^{(n)}(\|P'\|_n(a_1, \dots, a_n)) = a_i$ (überschreibt $x_2, \dots, x_{k'}$ nicht)
für alle $2 \leq i \leq k'$ und $a_1, \dots, a_n \in \mathbb{N}$

Loop-Berechenbarkeit vs. primitive Rekursion

§7.9 Theorem

Jede primitiv rekursive Funktion ist Loop-berechenbar

Beweis (1/3)

Induktion über Struktur primitiv rekursiver Funktionen

- **Basisfunktionen:** Trivial Loop-berechenbar
(Konstanten, Projektionen, Nachfolger)

Loop-Berechenbarkeit vs. primitive Rekursion

Beweis (2/3)

- Sei $f' = f\langle g_1, \dots, g_m \rangle$ **Komposition** primitiv rekursiver Funktionen f, g_1, \dots, g_m

$$f'(a_1, \dots, a_n) = f(g_1(a_1, \dots, a_n), \dots, g_m(a_1, \dots, a_n))$$

Loop-Berechenbarkeit vs. primitive Rekursion

Beweis (2/3)

- Sei $f' = f\langle g_1, \dots, g_m \rangle$ **Komposition** primitiv rekursiver Funktionen f, g_1, \dots, g_m

$$f'(a_1, \dots, a_n) = f(g_1(a_1, \dots, a_n), \dots, g_m(a_1, \dots, a_n))$$

IH und §7.8 liefern äquivalente Loop-Programme P, P_1, \dots, P_m die Variablen x_2, \dots, x_{n+m+1} nicht überschreiben.

Loop-Berechenbarkeit vs. primitive Rekursion

Beweis (2/3)

- Sei $f' = f\langle g_1, \dots, g_m \rangle$ **Komposition** primitiv rekursiver Funktionen f, g_1, \dots, g_m

$$f'(a_1, \dots, a_n) = f(g_1(a_1, \dots, a_n), \dots, g_m(a_1, \dots, a_n))$$

IH und §7.8 liefern äquivalente Loop-Programme P, P_1, \dots, P_m die Variablen x_2, \dots, x_{n+m+1} nicht überschreiben.

Folgendes Programm berechnet f'

$x_{n+m+1} = x_1$ (1. Eingabe sichern)

$P_m ; x_{n+m} = x_1 ; x_1 = x_{n+m+1}$ (Ergebnis sichern; 1. Eingabe setzen)

...

$P_1 ; x_2 = x_{n+2} ; \dots ; x_m = x_{n+m}$ (Eingaben auf Ergebnisse setzen)

P

Loop-Berechenbarkeit vs. primitive Rekursion

Beweis (3/3)

- Sei $f' = \text{pr}[f, g]$ **primitive Rekursion** mit primitiv rekursiven Funktionen f und g

$$f'(0, a_1, \dots, a_n) = f(a_1, \dots, a_n)$$

$$f'(a+1, a_1, \dots, a_n) = g(f'(a, a_1, \dots, a_n), a, a_1, \dots, a_n)$$

Loop-Berechenbarkeit vs. primitive Rekursion

Beweis (3/3)

- Sei $f' = \text{pr}[f, g]$ **primitive Rekursion** mit primitiv rekursiven Funktionen f und g

$$f'(0, a_1, \dots, a_n) = f(a_1, \dots, a_n)$$

$$f'(a+1, a_1, \dots, a_n) = g(f'(a, a_1, \dots, a_n), a, a_1, \dots, a_n)$$

IH und Lemma §7.8 liefern äquivalente Loop-Programme P_f und P_g die Variablen x_2, \dots, x_{n+4} nicht überschreiben.

Loop-Berechenbarkeit vs. primitive Rekursion

Beweis (3/3)

- Sei $f' = \text{pr}[f, g]$ **primitive Rekursion** mit primitiv rekursiven Funktionen f und g

$$f'(0, a_1, \dots, a_n) = f(a_1, \dots, a_n)$$

$$f'(a+1, a_1, \dots, a_n) = g(f'(a, a_1, \dots, a_n), a, a_1, \dots, a_n)$$

IH und Lemma §7.8 liefern äquivalente Loop-Programme P_f und P_g die Variablen x_2, \dots, x_{n+4} nicht überschreiben. Folgendes Programm berechnet f'

$x_{n+3} = x_1 ; x_{n+4} = x_2$

$x_1 = x_2 ; x_2 = x_3 ; \dots ; x_n = x_{n+1} ; P_f$

$x_{n+2} = x_n ; \dots ; x_4 = x_2 ; x_3 = x_{n+3} ; x_2 = 0$

LOOP(x_{n+3}) { $P_g ; x_2 = x_2 + 1$ }

(1. & 2. Eingabe sichern)

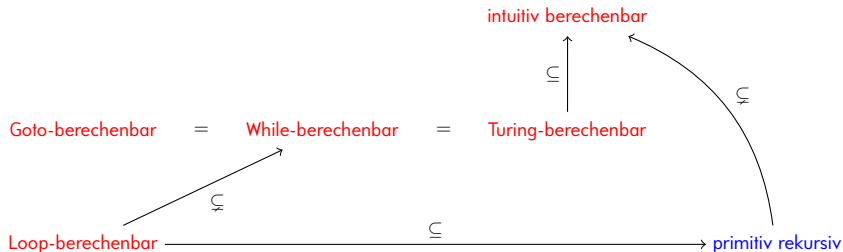
(Eingaben für f)

(Eingaben für g)

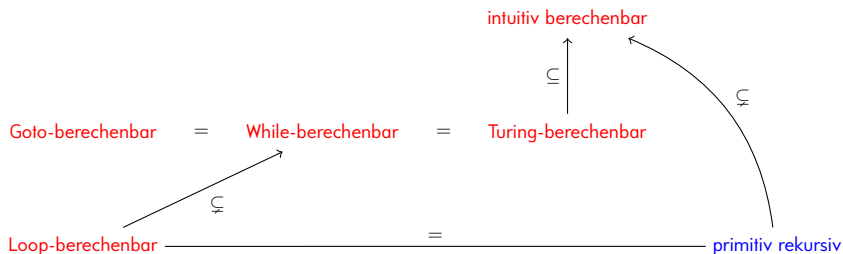
(Iterationen zählen)



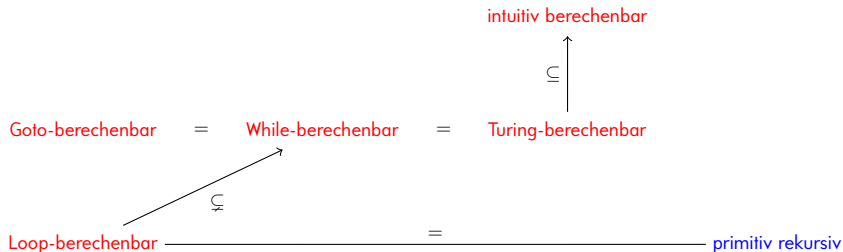
Loop-Berechenbarkeit vs. primitive Rekursion



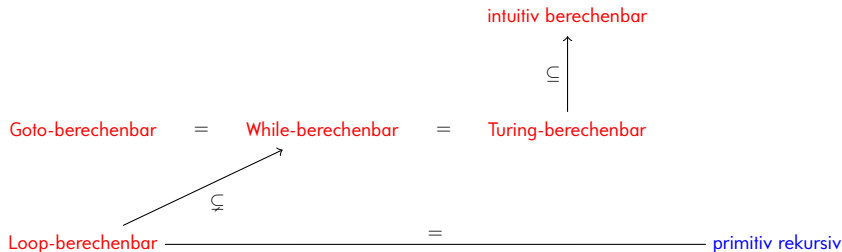
Loop-Berechenbarkeit vs. primitive Rekursion



Loop-Berechenbarkeit vs. primitive Rekursion



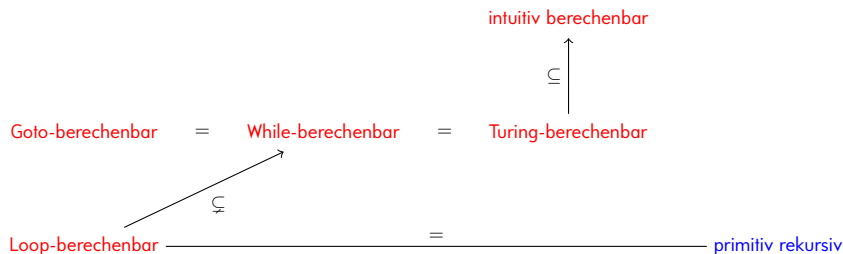
Loop-Berechenbarkeit vs. primitive Rekursion



Notizen

- Primitiv rekursive Funktionen total
- Nicht jede While-berechenbare Funktion primitiv rekursiv
(z.B. Ackermann-Funktion nicht primitiv rekursiv)

Loop-Berechenbarkeit vs. primitive Rekursion



Notizen

- Primitiv rekursive Funktionen total
- Nicht jede While-berechenbare Funktion primitiv rekursiv (z.B. Ackermann-Funktion nicht primitiv rekursiv)
- Allgemeine Rekursion noch nicht erfasst

Rekursive partielle Funktionen

§7.10 Definition (μ -rekursiv; μ -recursive)

Genau folgende partielle Funktionen sind μ -rekursiv

- rekursive Basisfunktionen
- Komposition und primitive Rekursion μ -rekursiver Funktionen

Rekursive partielle Funktionen

§7.10 Definition (μ -rekursiv; μ -recursive)

Genau folgende partielle Funktionen sind **μ -rekursiv**

- **rekursive Basisfunktionen**
- **Komposition** und **primitive Rekursion** μ -rekursiver Funktionen
- **Minimierung** $\mu f: \mathbb{N}^n \dashrightarrow \mathbb{N}$ (μ -Operator) μ -rekursiver Funktion $f: \mathbb{N}^{n+1} \dashrightarrow \mathbb{N}$ gegeben für alle $a_1, \dots, a_n \in \mathbb{N}$ durch

$$\mu f(a_1, \dots, a_n) = \min \{ a \in \mathbb{N} \mid f(a, a_1, \dots, a_n) = 0 \text{ und} \\ \forall b < a: f(b, a_1, \dots, a_n) \text{ definiert} \}$$

mit $\min \emptyset = \text{undef}$

Keine weiteren partiellen Funktionen μ -rekursiv

Rekursive partielle Funktionen

Intuition

Berechnung von $\mu f(a_1, \dots, a_n)$

1. Setze $a \leftarrow 0$
2. Berechne $f(a, a_1, \dots, a_n)$

Rekursive partielle Funktionen

Intuition

Berechnung von $\mu f(a_1, \dots, a_n)$

1. Setze $a \leftarrow 0$
2. Berechne $f(a, a_1, \dots, a_n)$
3. Liefere **undefiniert** falls $f(a, a_1, \dots, a_n)$ undefiniert
(Endlosschleife)

Rekursive partielle Funktionen

Intuition

Berechnung von $\mu f(a_1, \dots, a_n)$

1. Setze $a \leftarrow 0$
2. Berechne $f(a, a_1, \dots, a_n)$
3. Liefere **undefiniert** falls $f(a, a_1, \dots, a_n)$ undefiniert
(Endlosschleife)
4. Erhöhe a und zurück zu 2. falls $f(a, a_1, \dots, a_n) \neq 0$
5. Liefere a (falls $f(a, a_1, \dots, a_n) = 0$)

Rekursive partielle Funktionen

Intuition

Berechnung von $\mu f(a_1, \dots, a_n)$

1. Setze $a \leftarrow 0$
2. Berechne $f(a, a_1, \dots, a_n)$
3. Liefere **undefiniert** falls $f(a, a_1, \dots, a_n)$ undefiniert
(Endlosschleife)
4. Erhöhe a und zurück zu 2. falls $f(a, a_1, \dots, a_n) \neq 0$
5. Liefere a (falls $f(a, a_1, \dots, a_n) = 0$)

While-Schleife erkennbar

Rekursive partielle Funktionen

Überall undefinierte Funktion $\mu 1^{(2)}: \mathbb{N} \rightarrow \mathbb{N}$

- $1^{(2)}(a, b) = 1$ für alle $a, b \in \mathbb{N}$

Rekursive partielle Funktionen

Überall undefinierte Funktion $\mu 1^{(2)}: \mathbb{N} \rightarrow \mathbb{N}$

- $1^{(2)}(a, b) = 1$ für alle $a, b \in \mathbb{N}$
- Also $\mu 1^{(2)}(b) = \text{undef}$ für alle $b \in \mathbb{N}$

Rekursive partielle Funktionen

Überall undefinierte Funktion $\mu 1^{(2)}: \mathbb{N} \rightarrow \mathbb{N}$

- $1^{(2)}(a, b) = 1$ für alle $a, b \in \mathbb{N}$
- Also $\mu 1^{(2)}(b) = \text{undef}$ für alle $b \in \mathbb{N}$

Logarithmus

$$\text{ld}(13) = \lceil 3, 7 \rceil = 4$$

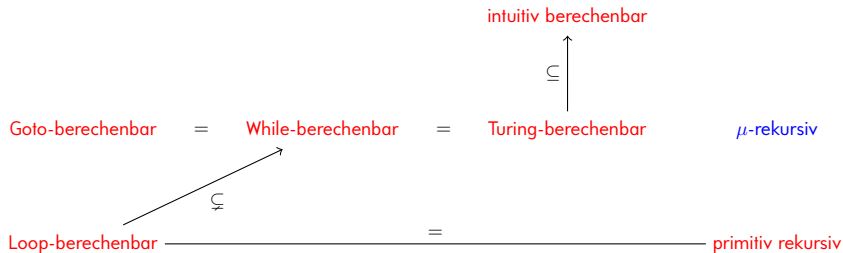
$\text{ld}: \mathbb{N} \rightarrow \mathbb{N}$ μ -rekursiv mit $\text{ld}(0) = 0$ und $\text{ld}(a) = \lceil \log_2(a) \rceil$ für alle $a \in \mathbb{N}_+$

- $f(a, b) = b - 2^a$ primitiv rekursiv
- $\text{ld}(b) = \mu f(b)$

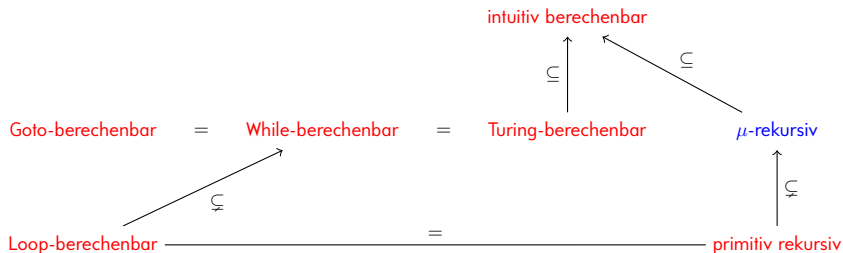
(Loop-berechenbar)
(kleinstes a mit $2^a \geq b$)

ld primitiv rekursiv

Rekursive partielle Funktionen



Rekursive partielle Funktionen



While-Berechenbarkeit vs. Rekursion

§7.11 Theorem

Jede While-berechenbare partielle Funktion ist μ -rekursiv

While-Berechenbarkeit vs. Rekursion

§7.11 Theorem

Jede While-berechenbare partielle Funktion ist μ -rekursiv

Beweis

Analog zu Loop-Programm mit neuem dritten Fall

- Sei $P = \text{WHILE}(x_i \neq 0) \{P'\}$. Dann existiert μ -rekursive Funktion $\text{sem}_{P'}$ gemäß IH. Sei

$$f(0, a_1, \dots, a_n) = c(a_1, \dots, a_n)$$

$$f(a+1, a_1, \dots, a_n) = \text{sem}_{P'}(b_1, \dots, b_n)$$

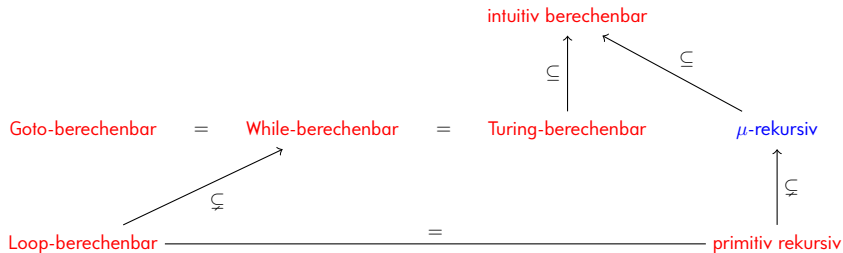
$$g(a, a_1, \dots, a_n) = b_i$$

wobei $f(a, a_1, \dots, a_n) = c(b_1, \dots, b_n)$

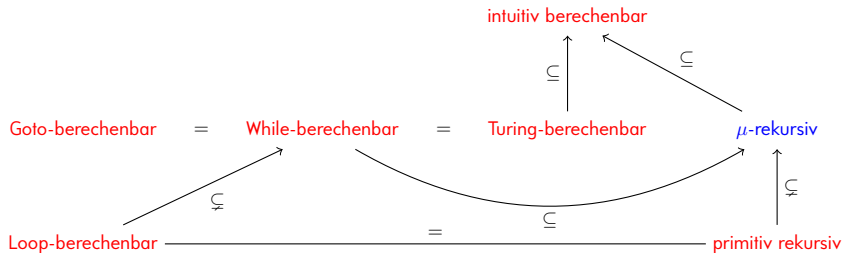
$$\text{sem}_P(a_1, \dots, a_n) = f((\mu g)(a_1, \dots, a_n), a_1, \dots, a_n)$$



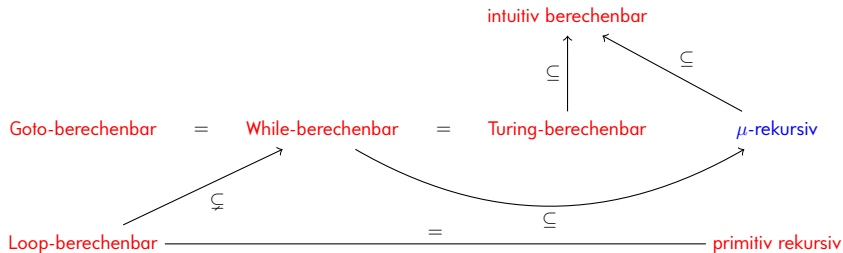
While-Berechenbarkeit vs. Rekursion



While-Berechenbarkeit vs. Rekursion



While-Berechenbarkeit vs. Rekursion



While-Berechenbarkeit vs. Rekursion

§7.12 Theorem

Jede μ -rekursive partielle Funktion ist While-berechenbar

Beweis

Induktion über Struktur μ -rekursiver partieller Funktionen

- Sei $f' = \mu f$ für μ -rekursive Funktion f . Dann existiert äquivalentes While-Programm P ohne Überschreibung Variablen x_2, \dots, x_{n+2} .
Programm für f'

$x_{n+2} = 0 ; x_{n+1} = x_n ; \dots ; x_2 = x_1 ; x_1 = 0 ; P$ (Eingaben setzen)

WHILE($x_1 \neq 0$) { $x_{n+2} = x_{n+2} + 1 ; x_1 = x_{n+2} ; P$ }

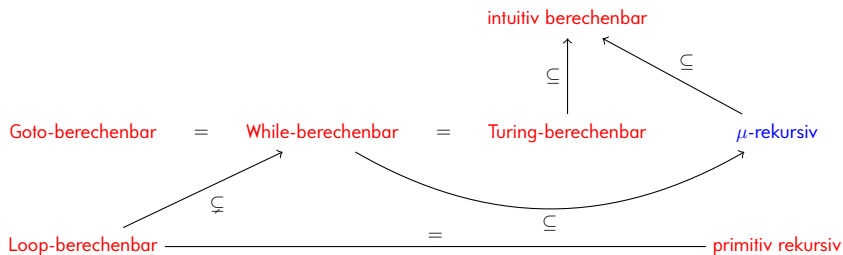
(Iteration erhöhen, Eingaben vorbereiten, weiterer Aufruf)

$x_1 = x_{n+2}$

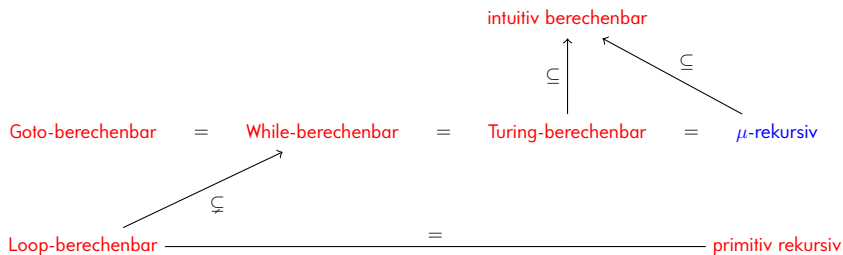
(Ergebnis ist Anzahl Iterationen)



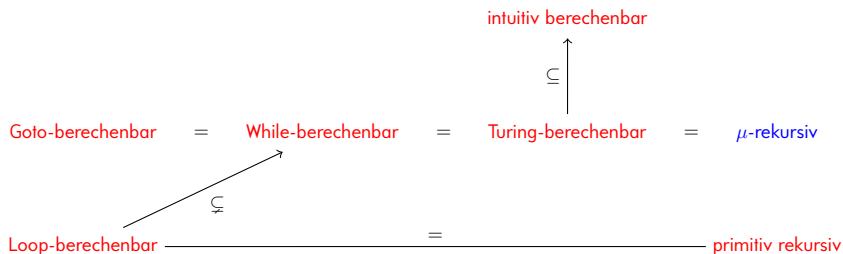
While-Berechenbarkeit vs. Rekursion



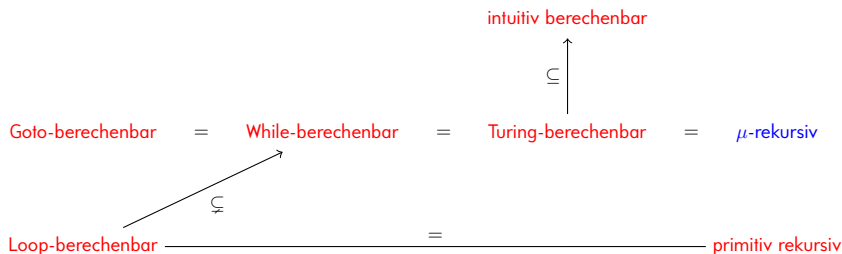
While-Berechenbarkeit vs. Rekursion



While-Berechenbarkeit vs. Rekursion



While-Berechenbarkeit vs. Rekursion



Notizen

- Verschiedene Berechnungsmodelle & viele weitere existieren
- Alle höchstens Turing-Berechenbarkeit

These von Church

§7.13 Hypothese (These von Church; *Church's conjecture*)

Jede intuitiv berechenbare Funktion ist Turing-berechenbar

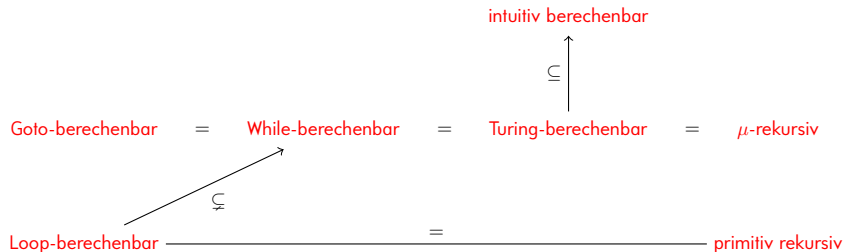
Alonzo Church (* 1903; † 1995)

- Amer. Mathematiker und Logiker
- Entwickelte λ -Kalkül (nicht vorgestellt)
- Doktorvater von Stephen Kleene & Alan Turing

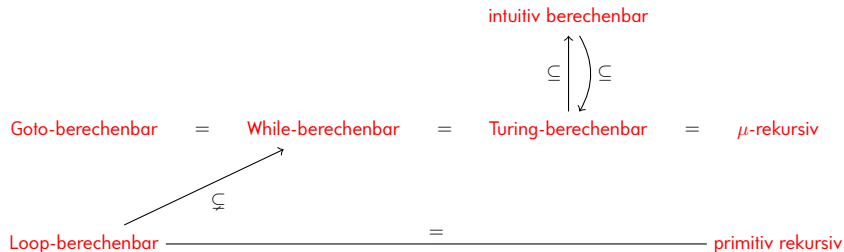


© Princeton University

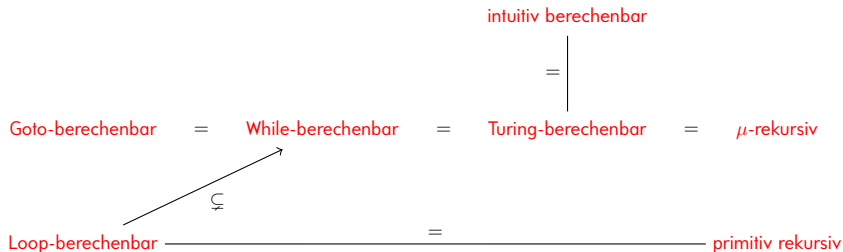
These von Church



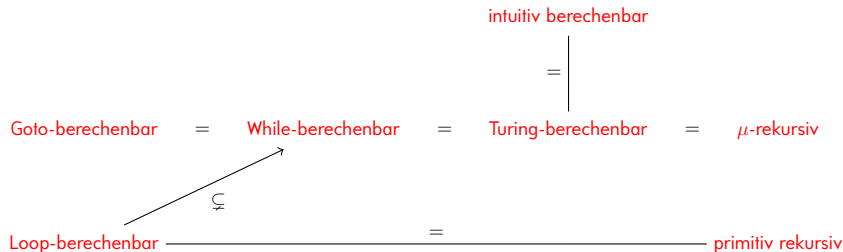
These von Church



These von Church



These von Church



Hauptannahme Berechenbarkeit

berechenbar = Turing-berechenbar = While-berechenbar
= intuitiv berechenbar = ...

Zusammenfassung

- Loop-berechenbar = primitiv rekursiv
- μ -rekursive partielle Funktionen
- μ -rekursiv = While-berechenbar

Fünfte Übungsserie bereits im Moodle