Script generated by TTT

Title: Nipkow: Theo (18.07.2019)

Date: Thu Jul 18 14:14:36 CEST 2019

Duration: 69:58 min

Pages: 86

Satz 5.45 (PR = LOOP)

Die primitiv rekursiven sind genau die LOOP-berechenbaren Funktionen.

Satz 5.45 (PR = LOOP)

Die primitiv rekursiven sind genau die LOOP-berechenbaren Funktionen.

Beweis:
LOOP $\rightarrow$ PR:
Sei $f : \mathbb{N}^m \rightarrow \mathbb{N}$ LOOP-berechenbar.

Beweis (Forts.):

- P ist x_i := $x_j \pm c$:

Beweis (Forts.): PR $\rightarrow$ LOOP

Sei $f : \mathbb{N}^k \rightarrow \mathbb{N}$ PR.
Mit Induktion über die Konstruktion von f konstruieren wir ein LOOP-Programm P_f, das f berechnet:

Beweis (Forts.):

- Primitive Rekursion:

$$\begin{aligned} f(0, \overline{x}) &= g(\overline{x}) \\ f(y+1, \overline{x}) &= h(f(y, \overline{x}), y, \overline{x}) \end{aligned}$$

Folgendes LOOP-Programm berechnet $f(y, \overline{x})$:

r := $g(\overline{x})$;
k := 0;
LOOP y DO
  r := $h(r, k, \overline{x})$
  k := $k+1$;
END
x_0 := r

Beweis (Forts.):

- Primitive Rekursion:

$$\begin{aligned} f(0, \overline{x}) &= g(\overline{x}) \\ f(y+1, \overline{x}) &= h(f(y, \overline{x}), y, \overline{x}) \end{aligned}$$

Folgendes LOOP-Programm berechnet $f(y, \overline{x})$:

r := $g(\overline{x})$;
k := 0;
LOOP y DO
  r := $h(r, k, \overline{x})$
  k := $k+1$;
END
x_0 := r

wobei wir nach Induktionsannahme LOOP-Programme zur Berechnung von g und h konstruieren können.

□

5.7 Die μ-rekursiven Funktionen

- Mit PR kann man nur beschränkt suchen: $n, \ldots, 0$.

5.7 Die μ-rekursiven Funktionen

- Mit PR kann man nur beschränkt suchen: $n, \ldots, 0$.
- Die *unbschränkte* Suche $0, \ldots$ erfordert so etwas wie

$$f(n) \;=\; \ldots f(n+1) \ldots$$

5.7 Die μ-rekursiven Funktionen

- Mit PR kann man nur beschränkt suchen: $n, \ldots, 0$.
- Die *unbschränkte* Suche $0, \ldots$ erfordert so etwas wie

$$f(n) \;=\; \ldots f(n+1) \ldots$$

- Der μ-Operator formalisiert diese Art der Suche.

5.7 Die μ-rekursiven Funktionen

- Mit PR kann man nur beschränkt suchen: $n, \ldots, 0$.
- Die *unbschränkte* Suche $0, \ldots$ erfordert so etwas wie

$$f(n) \;=\; \ldots f(n+1) \ldots$$

- Der μ-Operator formalisiert diese Art der Suche.
- Damit erhält man alle berechenbaren Funktionen.

5.7 Die μ-rekursiven Funktionen

- Mit PR kann man nur beschränkt suchen: $n, \ldots, 0$.
- Die *unbschränkte* Suche $0, \ldots$ erfordert so etwas wie

$$f(n) \;=\; \ldots f(n+1) \ldots$$

- Der μ-Operator formalisiert diese Art der Suche.
- Damit erhält man alle berechenbaren Funktionen.
- Andere Such- bzw Rekursionsschemata sind (im Prinzip!) nicht notwendig.

Notation: $f(n) = \bot$ bedeutet „$f(n)$ ist undefiniert.“

Definition 5.46 (μ-Operator)

Sei $f : \mathbb{N}^{k+1} \to \mathbb{N}$ eine (nicht notwendigerweise totale) Funktion.
Die durch Anwendung des μ-Operators entstehende Funktion
$\mu f : \mathbb{N}^k \to \mathbb{N}$ ist definiert durch:

Definition 5.46 (μ-Operator)

Sei $f : \mathbb{N}^{k+1} \to \mathbb{N}$ eine (nicht notwendigerweise totale) Funktion.
Die durch Anwendung des μ-Operators entstehende Funktion
$\mu f : \mathbb{N}^k \to \mathbb{N}$ ist definiert durch:

Intuitiv: $\mu f(\overline{x}) = \mathit{find}(0, \overline{x})$
$\mathit{find}(n, \overline{x}) = \textbf{if } f(n, \overline{x}) = 0 \textbf{ then } n \textbf{ else } \mathit{find}(n{+}1, \overline{x})$

Definition 5.46 (μ-Operator)

Sei $f : \mathbb{N}^{k+1} \to \mathbb{N}$ eine (nicht notwendigerweise totale) Funktion.
Die durch Anwendung des μ-Operators entstehende Funktion
$\mu f : \mathbb{N}^k \to \mathbb{N}$ ist definiert durch:

$$\overline{x} \mapsto \begin{cases} \min\{n \in \mathbb{N} \mid f(n, \overline{x}) = 0\} & \text{falls} \\ \quad \text{ein solches } n \text{ existiert und} & \\ \quad f(m, \overline{x}) \neq \bot \text{ für alle } m \leq n & \\ \bot & \text{sonst} \end{cases}$$

Intuitiv: $\mu f(\overline{x}) = \mathit{find}(0, \overline{x})$
$\mathit{find}(n, \overline{x}) = \textbf{if } f(n, \overline{x}) = 0 \textbf{ then } n \textbf{ else } \mathit{find}(n{+}1, \overline{x})$

Definition 5.46 (μ-Operator)

Sei $f : \mathbb{N}^{k+1} \to \mathbb{N}$ eine (nicht notwendigerweise totale) Funktion.
Die durch Anwendung des μ-Operators entstehende Funktion
$\mu f : \mathbb{N}^k \to \mathbb{N}$ ist definiert durch:

$$\overline{x} \mapsto \begin{cases} \min\{n \in \mathbb{N} \mid f(n, \overline{x}) = 0\} & \text{falls} \\ \quad \text{ein solches } n \text{ existiert und} & \\ \quad f(m, \overline{x}) \neq \bot \text{ für alle } m \leq n & \\ \bot & \text{sonst} \end{cases}$$

Intuitiv: $\mu f(\overline{x}) = \mathit{find}(0, \overline{x})$
$\mathit{find}(n, \overline{x}) = \textbf{if } f(n, \overline{x}) = 0 \textbf{ then } n \textbf{ else } \mathit{find}(n{+}1, \overline{x})$

Beispiel 5.47

Ist	$f(n, x)$	$=$	$x \dot{-} (n + n)$
dann ist	$\mu f(x)$	$=$	

Definition 5.46 (μ-Operator)

Sei $f : \mathbb{N}^{k+1} \to \mathbb{N}$ eine (nicht notwendigerweise totale) Funktion. Die durch Anwendung des μ-Operators entstehende Funktion $\mu f : \mathbb{N}^k \to \mathbb{N}$ ist definiert durch:

$$\overline{x} \mapsto \begin{cases} \min\{n \in \mathbb{N} \mid f(n, \overline{x}) = 0\} & \text{falls} \\ & \text{ein solches } n \text{ existiert und} \\ & f(m, \overline{x}) \neq \perp \text{ für alle } m \leq n \\ \perp & \text{sonst} \end{cases}$$

Intuitiv: $\mu f(\overline{x}) = \mathit{find}(0, \overline{x})$
$\mathit{find}(n, \overline{x}) = \textbf{if } f(n, \overline{x}) = 0 \textbf{ then } n \textbf{ else } \mathit{find}(n+1, \overline{x})$

Beispiel 5.47

Ist $f(n, x) = x \dot{-} (n + n)$
dann ist $\mu f(x) = \lceil x/2 \rceil$

Definition 5.48 (μR)

Die Menge der μ-rekursiven Funktionen ist induktiv wie folgt definiert:

Definition 5.46 (μ-Operator)

Sei $f : \mathbb{N}^{k+1} \to \mathbb{N}$ eine (nicht notwendigerweise totale) Funktion. Die durch Anwendung des μ-Operators entstehende Funktion $\mu f : \mathbb{N}^k \to \mathbb{N}$ ist definiert durch:

$$\overline{x} \mapsto \begin{cases} \min\{n \in \mathbb{N} \mid f(n, \overline{x}) = 0\} & \text{falls} \\ & \text{ein solches } n \text{ existiert und} \\ & f(m, \overline{x}) \neq \perp \text{ für alle } m \leq n \\ \perp & \text{sonst} \end{cases}$$

Intuitiv: $\mu f(\overline{x}) = \mathit{find}(0, \overline{x})$
$\mathit{find}(n, \overline{x}) = \textbf{if } f(n, \overline{x}) = 0 \textbf{ then } n \textbf{ else } \mathit{find}(n+1, \overline{x})$

Beispiel 5.47

Ist $f(n, x) = x \dot{-} (n + n)$
dann ist $\mu f(x) = \lceil x/2 \rceil$

Definition 5.48 (μR)

Die Menge der μ-rekursiven Funktionen ist induktiv wie folgt definiert:

Definition 5.48 (μR)

Die Menge der μ-rekursiven Funktionen ist induktiv wie folgt definiert:

- Die Basisfunktionen 0, $+1$, π_i^k sind μ-rekursiv.
- Wenn eine Funktion durch
  Komposition, primitive Rekursion, oder den μ-Operator
  aus μ-rekursive Funktionen definiert werden kann, ist sie μ-rekursiv.

Satz 5.49 (μR = WHILE)

Die μ-rekursiven sind genau die WHILE-berechenbaren Funktionen.

Definition 5.48 (μR)

Die Menge der μ-rekursiven Funktionen ist induktiv wie folgt definiert:

- Die Basisfunktionen 0, $+1$, π_i^k sind μ-rekursiv.
- Wenn eine Funktion durch
  Komposition, primitive Rekursion, oder den μ-Operator
  aus μ-rekursive Funktionen definiert werden kann, ist sie μ-rekursiv.

Satz 5.49 (μR = WHILE)

Die μ-rekursiven sind genau die WHILE-berechenbaren Funktionen.

Beweis: als Ergänzung des Beweises von PR = LOOP.

Definition 5.48 (μR)

Die Menge der μ-rekursiven Funktionen ist induktiv wie folgt definiert:

- Die Basisfunktionen 0, $+1$, π_i^k sind μ-rekursiv.
- Wenn eine Funktion durch
  Komposition, primitive Rekursion, oder den μ-Operator
  aus μ-rekursive Funktionen definiert werden kann, ist sie μ-rekursiv.

Satz 5.49 (μR = WHILE)

Die μ-rekursiven sind genau die WHILE-berechenbaren Funktionen.

Beweis: als Ergänzung des Beweises von PR = LOOP.

μR $\rightarrow$ WHILE:
Fall μf: Nach IA gibt es ein WHILE-Programm für f.
Dann ist μf auch WHILE-berechenbar:

x_0 := 0; y := $f(0, \overline{x})$;
WHILE $y \neq 0$ DO x_0 := x_0 + 1; y := $f(x_0, \overline{x})$ END

Beweis (Forts.):

WHILE $\rightarrow$ μR:

Beweis (Forts.):

WHILE $\rightarrow \mu$R:
Fall P ist `WHILE` $x_i \neq 0$ `DO` Q `END`

Beweis (Forts.):

WHILE $\rightarrow \mu$R:
Fall P ist `WHILE` $x_i \neq 0$ `DO` Q `END`
Nach IA gibt es eine Funktion g_Q die Q berechnet.

Beweis (Forts.):

WHILE $\rightarrow \mu$R:
Fall P ist `WHILE` $x_i \neq 0$ `DO` Q `END`
Nach IA gibt es eine Funktion g_Q die Q berechnet.
Dann ist g_P wie folgt definierbar:

Beweis (Forts.):

WHILE $\rightarrow \mu$R:
Fall P ist `WHILE` $x_i \neq 0$ `DO` Q `END`
Nach IA gibt es eine Funktion g_Q die Q berechnet.
Dann ist g_P wie folgt definierbar:

$$\begin{aligned} h(0, x) &= x \\ h(n+1, x) &= g_Q(h(n, x)) \end{aligned}$$

Beweis (Forts.):

WHILE $\to \mu$R:
Fall P ist `WHILE` $x_i \neq 0$ `DO` Q `END`
Nach IA gibt es eine Funktion g_Q die Q berechnet.
Dann ist g_P wie folgt definierbar:

$$\begin{aligned} h(0,x) &= x \\ h(n+1,x) &= g_Q(h(n,x)) \\ h_i(n,x) &= d_i(h(n,x)) \end{aligned}$$

Beweis (Forts.):

WHILE $\to \mu$R:
Fall P ist `WHILE` $x_i \neq 0$ `DO` Q `END`
Nach IA gibt es eine Funktion g_Q die Q berechnet.
Dann ist g_P wie folgt definierbar:

$$\begin{aligned} h(0,x) &= x \\ h(n+1,x) &= g_Q(h(n,x)) \\ h_i(n,x) &= d_i(h(n,x)) \end{aligned}$$

$h_i(n,x)$ ist der Wert von x_i nach n Iterationen von Q.

Beweis (Forts.):

WHILE $\to \mu$R:
Fall P ist `WHILE` $x_i \neq 0$ `DO` Q `END`
Nach IA gibt es eine Funktion g_Q die Q berechnet.
Dann ist g_P wie folgt definierbar:

$$\begin{aligned} h(0,x) &= x \\ h(n+1,x) &= g_Q(h(n,x)) \\ h_i(n,x) &= d_i(h(n,x)) \end{aligned}$$

$h_i(n,x)$ ist der Wert von x_i nach n Iterationen von Q.
Dh $\mu h_i(x)$ ist die Anzahl der Iterationen von Q bis $x_i = 0$,
dh bis P terminiert.

Durch die Transformation

$$\mu\text{R} \to \text{WHILE mit einer Schleife (Kleene)} \to \mu\text{R}$$

genügt also immer ein μ-Operator:

Durch die Transformation

$$\mu\text{R} \rightarrow \text{WHILE mit einer Schleife (Kleene)} \rightarrow \mu\text{R}$$

genügt also immer ein μ-Operator:

Korollar 5.50 (Kleene)

Für jede n-stellige μ-rekursive Funktionen f gibt es zwei $n+1$-stellige PR Funktionen h und h', so dass

$$f(\overline{x}) = h(\mu h'(\overline{x}), \overline{x})$$

5.8 Die Ackermann-Funktion

$$\begin{aligned} a(0, n) &= n+1 \\ a(m+1, 0) &= a(m, 1) \\ a(m+1, n+1) &= a(m, a(m+1, n)) \end{aligned}$$

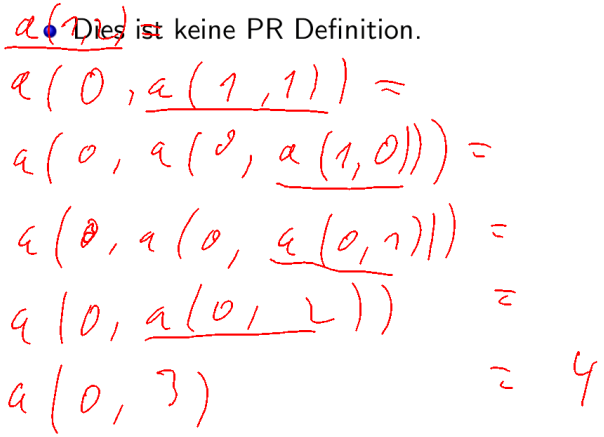
5.8 Die Ackermann-Funktion

$$\begin{aligned} a(0, n) &= n+1 \\ a(m+1, 0) &= a(m, 1) \\ a(m+1, n+1) &= a(m, a(m+1, n)) \end{aligned}$$

5.8 Die Ackermann-Funktion

$$\begin{aligned} a(0, n) &= n+1 \\ a(m+1, 0) &= a(m, 1) \\ a(m+1, n+1) &= a(m, a(m+1, n)) \end{aligned}$$

- Dies ist keine PR Definition.



5.8 Die Ackermann-Funktion

$$\begin{aligned} a(\quad 0, \quad n) &= n+1 \\ a(m+1, \quad 0) &= a(m,1) \\ a(m+1, n+1) &= a(m, a(m+1, n)) \end{aligned}$$

- Dies ist keine PR Definition.

5.8 Die Ackermann-Funktion

$$\begin{aligned} a(\quad 0, \quad n) &= n+1 \\ a(m+1, \quad 0) &= a(m,1) \\ a(m+1, n+1) &= a(m, a(m+1, n)) \end{aligned}$$

- Dies ist keine PR Definition.
- Aber: $\not\Rightarrow$ a ist nicht PR
- Ziel: a ist berechenbar, total, aber nicht PR

5.8 Die Ackermann-Funktion

$$\begin{aligned} a(\quad 0, \quad n) &= n+1 \\ a(m+1, \quad 0) &= a(m,1) \\ a(m+1, n+1) &= a(m, a(m+1, n)) \end{aligned}$$

- Dies ist keine PR Definition.
- Aber: $\not\Rightarrow$ a ist nicht PR
- Ziel: a ist berechenbar, total, aber nicht PR

Fakt 5.51
Die Ackermann-Funktion ist (OCaml-)berechenbar.

Beispiel:

$a(2,2) =$
$a(1, a(2,1)) =$
$a(1, a(1, a(2,0))) =$
$a(1, a(1, a(1,1))) =$
$a(1, a(1, a(0, a(1,0)))) =$
$a(1, a(1, a(0, a(0,1)))) =$
$a(1, a(1, a(0,2))) =$
$a(1, a(1,3)) =$
$a(1, a(0, a(1,2))) =$
$a(1, a(0, a(0, a(1,1)))) =$
$a(1, a(0, a(0, a(0, a(1,0))))) =$
$a(1, a(0, a(0, a(0, a(0,1))))) =$
$a(1, a(0, a(0, a(0,2)))) =$
$a(1, a(0, a(0,3))) =$
$a(1, a(0,4)) =$
$a(1,5) =$
$a(0, a(1,4)) =$
$a(0, a(0, a(1,3))) =$
$a(0, a(0, a(0, a(1,2)))) =$
$a(0, a(0, a(0, a(0, a(1,1))))) =$
$a(0, a(0, a(0, a(0, a(0, a(1,0)))))) =$
$a(0, a(0, a(0, a(0, a(0, a(0,1)))))) =$
$a(0, a(0, a(0, a(0, a(0,2))))) =$
$a(0, a(0, a(0, a(0,3)))) =$
$a(0, a(0, a(0,4))) =$
$a(0, a(0,5)) =$
$a(0,6) =$
7

Scherzfrage: $a(6,6) = ?$

Der Beginn:

$$a(0,n) = n+1$$

Der Beginn:

$$\begin{aligned} a(0,n) &= n+1 \\ a(1,n) &= 2+(n+3)-3 \end{aligned}$$

Beispiel:

$a(2,2) =$
$a(1,a(2,1)) =$
$a(1,a(1,a(2,0))) =$
$a(1,a(1,a(1,1))) =$
$a(1,a(1,a(0,a(1,0)))) =$
$a(1,a(1,a(0,a(0,1)))) =$
$a(1,a(1,a(0,2))) =$
$a(1,a(1,3)) =$
$a(1,a(0,a(1,2))) =$
$a(1,a(0,a(0,a(1,1)))) =$
$a(1,a(0,a(0,a(0,a(1,0))))) =$
$a(1,a(0,a(0,a(0,a(0,1))))) =$
$a(1,a(0,a(0,a(0,2)))) =$
$a(1,a(0,a(0,3))) =$
$a(1,a(0,4)) =$
$a(1,5) =$
$a(0,a(1,4)) =$
$a(0,a(0,a(1,3))) =$
$a(0,a(0,a(0,a(1,2)))) =$
$a(0,a(0,a(0,a(0,a(1,1))))) =$
$a(0,a(0,a(0,a(0,a(0,a(1,0)))))) =$
$a(0,a(0,a(0,a(0,a(0,a(0,1)))))) =$
$a(0,a(0,a(0,a(0,a(0,2))))) =$
$a(0,a(0,a(0,a(0,3)))) =$
$a(0,a(0,a(0,4))) =$
$a(0,a(0,5)) =$
$a(0,6) =$
7

Der Beginn:

$$\begin{aligned} a(0,n) &= n+1 \\ a(1,n) &= 2+(n+3)-3 \\ a(2,n) &= 2(n+3)-3 \end{aligned}$$

Der Beginn:

$$\begin{aligned} a(0,n) &= n+1 \\ a(1,n) &= 2+(n+3)-3 \\ a(2,n) &= 2(n+3)-3 \\ a(3,n) &= 2^{n+3}-3 \end{aligned}$$

Der Beginn:

$$\begin{aligned} a(0,n) &= n+1 \\ a(1,n) &= 2+(n+3)-3 \\ a(2,n) &= 2(n+3)-3 \\ a(3,n) &= 2^{n+3}-3 \\ a(4,n) &= \underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{n+3}-3 \end{aligned}$$

Der Beginn:

$$\begin{aligned} a(0,n) &= n+1 \\ a(1,n) &= 2+(n+3)-3 \\ a(2,n) &= 2(n+3)-3 \\ a(3,n) &= 2^{n+3}-3 \\ a(4,n) &= \underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{n+3}-3 \\ a(5,n) &= ??? \end{aligned}$$

Der Beginn:

$a(m+1,0) = a(m,1)$

$a(m+1,n+1) = a(m, a(m+1,n))$ *

$$\begin{aligned} a(0,n) &= n+1 \\ a(1,n) &= 2+(n+3)-3 \\ a(2,n) &= 2(n+3)-3 \\ a(3,n) &= 2^{n+3}-3 \\ a(4,n) &= \underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{n+3}-3 \\ a(5,n) &= ??? \end{aligned}$$

Mit Induktion über n:

$$a(m+1,n) = \underbrace{a(m,a(m,\ldots a(m,1)\ldots)}_{n+1}$$

Der Beginn:

$$\begin{aligned} a(0,n) &= n+1 \\ a(1,n) &= 2+(n+3)-3 \\ a(2,n) &= 2(n+3)-3 \\ a(3,n) &= 2^{n+3}-3 \\ a(4,n) &= \underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{n+3}-3 \\ a(5,n) &= ??? \end{aligned}$$

Mit Induktion über n:

$$a(m+1,n) = \underbrace{a(m,a(m,\dots a(m}_{n+1},1)\dots)$$

Bsp: $a(1,n) =$

Der Beginn:

$$\begin{aligned} a(0,n) &= n+1 \\ a(1,n) &= 2+(n+3)-3 \\ a(2,n) &= 2(n+3)-3 \\ a(3,n) &= 2^{n+3}-3 \\ a(4,n) &= \underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{n+3}-3 \\ a(5,n) &= ??? \end{aligned}$$

Mit Induktion über n:

$$a(m+1,n) = \underbrace{a(m,a(m,\dots a(m}_{n+1},1)\dots)$$

Bsp: $a(1,n) = a(0,\dots a(0,1)\dots) = (+1)^{n+1}(1) = n+2$

Ackermann als Familie von Funktionen A_m: $A_m(n) := a(m,n)$.

Ackermann als Familie von Funktionen A_m: $A_m(n) := a(m,n)$.
Nun gilt:

$$\begin{aligned} A_0(n) &= s(n) \\ A_{m+1}(n) &= A_m^{n+1}(1) \end{aligned}$$

Der Beginn:

$$\begin{aligned} a(0,n) &= n+1 \\ a(1,n) &= 2+(n+3)-3 \\ a(2,n) &= 2(n+3)-3 \\ a(3,n) &= 2^{n+3}-3 \\ a(4,n) &= \underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{n+3}-3 \\ a(5,n) &= ??? \end{aligned}$$

Mit Induktion über n:

$$a(m+1,n) = \underbrace{a(m,a(m,\ldots a(m,1)\ldots)}_{n+1}$$

Bsp: $a(1,n) =$

Beispiel:

$a(2,2) =$
$a(1,a(2,1)) =$
$a(1,a(1,a(2,0))) =$
$a(1,a(1,a(1,1))) =$
$a(1,a(1,a(0,a(1,0)))) =$
$a(1,a(1,a(0,a(0,1)))) =$
$a(1,a(1,a(0,2))) =$
$a(1,a(1,3)) =$
$a(1,a(0,a(1,2))) =$
$a(1,a(0,a(0,a(1,1)))) =$
$a(1,a(0,a(0,a(0,a(1,0))))) =$
$a(1,a(0,a(0,a(0,a(0,1))))) =$
$a(1,a(0,a(0,a(0,2)))) =$
$a(1,a(0,a(0,3))) =$
$a(1,a(0,4)) =$
$a(1,5) =$
$a(0,a(1,4)) =$
$a(0,a(0,a(1,3))) =$
$a(0,a(0,a(0,a(1,2)))) =$
$a(0,a(0,a(0,a(0,a(1,1))))) =$
$a(0,a(0,a(0,a(0,a(0,a(1,0)))))) =$
$a(0,a(0,a(0,a(0,a(0,a(0,1)))))) =$
$a(0,a(0,a(0,a(0,a(0,2))))) =$
$a(0,a(0,a(0,a(0,3)))) =$
$a(0,a(0,a(0,4))) =$
$a(0,a(0,5)) =$
$a(0,6) =$
7

Ackermann als Familie von Funktionen A_m: $A_m(n) := a(m,n)$.
Nun gilt:

$$\begin{aligned} A_0(n) &= s(n) \\ A_{m+1}(n) &= A_m^{n+1}(1) = \underbrace{A_m(\ldots A_m}_{n+1}(1)\ldots) \end{aligned}$$

Ackermann als Familie von Funktionen A_m: $A_m(n) := a(m,n)$.
Nun gilt:

$$\begin{aligned} A_0(n) &= s(n) \\ A_{m+1}(n) &= A_m^{n+1}(1) = \underbrace{A_m(\ldots A_m}_{n+1}(1)\ldots) \end{aligned}$$

Beobachtung: alle A_m sind total und PR

Ackermann als Familie von Funktionen A_m: $A_m(n) := a(m,n)$.
Nun gilt:

$$\begin{aligned} A_0(n) &= s(n) \\ A_{m+1}(n) &= A_m^{n+1}(1) = \underbrace{A_m(\ldots A_m}_{n+1}(1)\ldots) \end{aligned}$$

Beobachtung: alle A_m sind total und PR (mit Ind. über m)

Ackermann als Familie von Funktionen A_m: $A_m(n) := a(m,n)$.
Nun gilt:

$$\begin{aligned} A_0(n) &= s(n) \\ A_{m+1}(n) &= A_m^{n+1}(1) = \underbrace{A_m(\ldots A_m}_{n+1}(1)\ldots) \end{aligned}$$

Beobachtung: alle A_m sind total und PR (mit Ind. über m)

Mit Currying (z.B. in Haskell):

```
a 0      n  =  n+1
a (m+1)  n  =  iter (n+1) (a m) 1
```


Ackermann als Familie von Funktionen A_m: $A_m(n) := a(m,n)$.
Nun gilt:

$$\begin{aligned} A_0(n) &= s(n) \\ A_{m+1}(n) &= A_m^{n+1}(1) = \underbrace{A_m(\ldots A_m}_{n+1}(1)\ldots) \end{aligned}$$

Beobachtung: alle A_m sind total und PR (mit Ind. über m)

Mit Currying (z.B. in Haskell):

```
a 0      n  =  n+1
a (m+1)  n  =  iter (n+1) (a m) 1
```

```
iter 0      f x  =  x
iter (n+1)  f x  =  f (iter n f x)
```


Ackermann als Familie von Funktionen A_m: $A_m(n) := a(m,n)$.
Nun gilt:

$$\begin{aligned} A_0(n) &= s(n) \\ A_{m+1}(n) &= A_m^{n+1}(1) = \underbrace{A_m(\ldots A_m}_{n+1}(1)\ldots) \end{aligned}$$

Beobachtung: alle A_m sind total und PR (mit Ind. über m)

Mit Currying (z.B. in Haskell):

```
a 0      n  =  n+1
a (m+1)  n  =  iter (n+1) (a m) 1

iter 0      f x  =  x
iter (n+1)  f x  =  f (iter n f x)
```

Ackermann ist mit primitiver Rekursion höherer Stufe definierbar.

Man kann auch direkt zeigen:

Lemma 5.52
Die Ackermann-Funktion ist total.

Man kann auch direkt zeigen:

Lemma 5.52
Die Ackermann-Funktion ist total.

Beweis:
Mit Induktion über m: $\forall n \in \mathbb{N}.\ a(m,n) \neq \bot$ (1)

Man kann auch direkt zeigen:

Lemma 5.52
Die Ackermann-Funktion ist total.

Beweis:
Mit Induktion über m: $\forall n \in \mathbb{N}.\ a(m,n) \neq \bot$ (1)

- Basis: $a(0,n) = n+1 \neq \bot$

Man kann auch direkt zeigen:

Lemma 5.52
Die Ackermann-Funktion ist total.

Beweis:
Mit Induktion über m: $\forall n \in \mathbb{N}.\ a(m,n) \neq \bot$ (1)

- Basis: $a(0,n) = n+1 \neq \bot$

Ackermann als Familie von Funktionen A_m: $A_m(n) := a(m,n)$.
Nun gilt:

$$\begin{aligned} A_0(n) &= s(n) \\ A_{m+1}(n) &= A_m^{n+1}(1) = \underbrace{A_m(\ldots A_m}_{n+1}(1)\ldots) \end{aligned}$$

Beobachtung: alle A_m sind total und PR (mit Ind. über m)

Mit Currying (z.B. in Haskell):

```
a 0      n  =  n+1
a (m+1)  n  =  iter (n+1) (a m) 1
```


Kompakter: die lexikographische Ordnung auf den Argumenten von a nimmt bei jedem rekursiven Aufruf ab:

$$\begin{aligned} (m+1,0) &> (m,1) \\ (m+1,n+1) &> (m,.) \\ (m+1,n+1) &> (m+1,n) \end{aligned}$$

Definition 5.53 (lexikographische Ordnung)

$$(m,n) > (m',n') \quad :\Leftrightarrow \quad m > m' \vee (m = m' \wedge n > n')$$

Kompakter: die lexikographische Ordnung auf den Argumenten von a nimmt bei jedem rekursiven Aufruf ab:

$$\begin{aligned} (m+1,0) &> (m,1) \\ (m+1,n+1) &> (m,.) \\ (m+1,n+1) &> (m+1,n) \end{aligned}$$

Definition 5.53 (lexikographische Ordnung)

$$(m,n) > (m',n') \quad :\Leftrightarrow \quad m > m' \vee (m = m' \wedge n > n')$$

Kompakter: die lexikographische Ordnung auf den Argumenten von a nimmt bei jedem rekursiven Aufruf ab:

$$\begin{aligned} (m+1,0) &> (m,1) \\ (m+1,n+1) &> (m,.) \\ (m+1,n+1) &> (m+1,n) \end{aligned}$$

Definition 5.53 (lexikographische Ordnung)

$$(m,n) > (m',n') \quad :\Leftrightarrow \quad m > m' \vee (m = m' \wedge n > n')$$

Kompakter: die lexikographische Ordnung auf den Argumenten von a nimmt bei jedem rekursiven Aufruf ab:

$$\begin{array}{rcl}(m+1,0) & > & (m,1)\\ (m+1,n+1) & > & (m,.)\\ (m+1,n+1) & > & (m+1,n)\end{array}$$

Definition 5.53 (lexikographische Ordnung)

$$(m,n) > (m',n') \quad :\Leftrightarrow \quad m > m' \vee (m = m' \wedge n > n')$$

Beispiel: $(2,3) > (2,2) > (1,42) > \cdots$

Kompakter: die lexikographische Ordnung auf den Argumenten von a nimmt bei jedem rekursiven Aufruf ab:

$$\begin{array}{rcl}(m+1,0) & > & (m,1)\\ (m+1,n+1) & > & (m,.)\\ (m+1,n+1) & > & (m+1,n)\end{array}$$

Definition 5.53 (lexikographische Ordnung)

$$(m,n) > (m',n') \quad :\Leftrightarrow \quad m > m' \vee (m = m' \wedge n > n')$$

Beispiel: $(2,3) > (2,2) > (1,42) > \cdots$

Satz 5.54
Die lexikographische Ordnung auf $\mathbb{N} \times \mathbb{N}$ terminiert.

Kompakter: die lexikographische Ordnung auf den Argumenten von a nimmt bei jedem rekursiven Aufruf ab:

$$\begin{array}{rcl}(m+1,0) & > & (m,1)\\ (m+1,n+1) & > & (m,.)\\ (m+1,n+1) & > & (m+1,n)\end{array}$$

Definition 5.53 (lexikographische Ordnung)

$$(m,n) > (m',n') \quad :\Leftrightarrow \quad m > m' \vee (m = m' \wedge n > n')$$

Beispiel: $(2,3) > (2,2) > (1,42) > \cdots$

Satz 5.54
Die lexikographische Ordnung auf $\mathbb{N} \times \mathbb{N}$ terminiert.

Beweis:
mit geschachtelter Induktion, wie bei der Totalität von a. □

Warnung: Kein Beweis der Totalität einer rekursiven Funktion:

„denn in jedem rekursiven Aufruf wird eines der Argumente kleiner“

Warnung: Kein Beweis der Totalität einer rekursiven Funktion:

„denn in jedem rekursiven Aufruf wird eines der Argumente kleiner“

$$\begin{aligned} f(m+1,0) &= f(m,1) \\ f(0,n+1) &= f(1,n) \end{aligned}$$

Lemma 5.55
Für jede PR Funktion $f : \mathbb{N}^k \to \mathbb{N}$ gibt es ein $t \in \mathbb{N}$, so dass

$$\forall \overline{x} \in \mathbb{N}^k.\ f(\overline{x}) < a(t, \max \overline{x}).$$

Lemma 5.55
Für jede PR Funktion $f : \mathbb{N}^k \to \mathbb{N}$ gibt es ein $t \in \mathbb{N}$, so dass

$$\forall \overline{x} \in \mathbb{N}^k.\ f(\overline{x}) < a(t, \max \overline{x}).$$

Beweis:
Mit Induktion über den Aufbau der Definition von f. □

Lemma 5.55
Für jede PR Funktion $f : \mathbb{N}^k \to \mathbb{N}$ gibt es ein $t \in \mathbb{N}$, so dass

$$\forall \overline{x} \in \mathbb{N}^k.\ f(\overline{x}) < a(t, \max \overline{x}).$$

Beweis:
Mit Induktion über den Aufbau der Definition von f. □

Prinzip: $t \approx$ Länge der Definition von f.
Details: [Felscher. *Berechenbarkeit*]

Lemma 5.55
Für jede PR Funktion $f : \mathbb{N}^k \to \mathbb{N}$ gibt es ein $t \in \mathbb{N}$, so dass

$$\forall \overline{x} \in \mathbb{N}^k.\ f(\overline{x}) < a(t, \max \overline{x}).$$

Beweis:
Mit Induktion über den Aufbau der Definition von f. □

Prinzip: $t \approx$ Länge der Definition von f.
Details: [Felscher. *Berechenbarkeit*]

Satz 5.56
Die Ackermann-Funktion ist nicht PR.

Lemma 5.55
Für jede PR Funktion $f : \mathbb{N}^k \to \mathbb{N}$ gibt es ein $t \in \mathbb{N}$, so dass

$$\forall \overline{x} \in \mathbb{N}^k.\ f(\overline{x}) < a(t, \max \overline{x}).$$

Beweis:
Mit Induktion über den Aufbau der Definition von f. □

Prinzip: $t \approx$ Länge der Definition von f.
Details: [Felscher. *Berechenbarkeit*]

Satz 5.56
Die Ackermann-Funktion ist nicht PR.

Beweis:
Indirekt. Angenommen a wäre doch PR. Dann ist auch f PR:

$$f(n) \ := \ a(n, n)$$

Lemma 5.55
Für jede PR Funktion $f : \mathbb{N}^k \to \mathbb{N}$ gibt es ein $t \in \mathbb{N}$, so dass

$$\forall \overline{x} \in \mathbb{N}^k.\ f(\overline{x}) < a(t, \max \overline{x}).$$

Beweis:
Mit Induktion über den Aufbau der Definition von f. □

Prinzip: $t \approx$ Länge der Definition von f.
Details: [Felscher. *Berechenbarkeit*]

Satz 5.56
Die Ackermann-Funktion ist nicht PR.

Beweis:
Indirekt. Angenommen a wäre doch PR. Dann ist auch f PR:

$$f(n) \ := \ a(n, n)$$

Nach obigem Lemma gibt es t mit $f(n) < a(t, n)$ für alle n.

Lemma 5.55
Für jede PR Funktion $f : \mathbb{N}^k \to \mathbb{N}$ gibt es ein $t \in \mathbb{N}$, so dass

$$\forall \overline{x} \in \mathbb{N}^k.\ f(\overline{x}) < a(t, \max \overline{x}).$$

Beweis:
Mit Induktion über den Aufbau der Definition von f. □

Prinzip: $t \approx$ Länge der Definition von f.
Details: [Felscher. *Berechenbarkeit*]

Satz 5.56
Die Ackermann-Funktion ist nicht PR.

Beweis:
Indirekt. Angenommen a wäre doch PR. Dann ist auch f PR:

$$f(n) \ := \ a(n, n)$$

Nach obigem Lemma gibt es t mit $f(n) < a(t, n)$ für alle n.

$$f(t) < a(t, t) = f(t)$$

Lemma 5.55
Für jede PR Funktion $f : \mathbb{N}^k \to \mathbb{N}$ gibt es ein $t \in \mathbb{N}$, so dass

$$\forall \overline{x} \in \mathbb{N}^k.\ f(\overline{x}) < a(t, \max \overline{x}).$$

Beweis:
Mit Induktion über den Aufbau der Definition von f. □

Prinzip: $t \approx$ Länge der Definition von f.
Details: [Felscher. *Berechenbarkeit*]

Satz 5.56
Die Ackermann-Funktion ist nicht PR.

Beweis:
Indirekt. Angenommen a wäre doch PR. Dann ist auch f PR:

$$f(n) \ := \ a(n, n)$$

Nach obigem Lemma gibt es t mit $f(n) < a(t, n)$ für alle n.

$$f(t) < a(t, t) = f(t)$$

□

Oberflächlich intuitiv:

Die Ackermann-Funktion wächst schneller als alle PR Funktionen.

Oberflächlich intuitiv:

Die Ackermann-Funktion wächst schneller als alle PR Funktionen.

Genauer:

Die Funktion $n \mapsto a(n, n)$ wächst schneller als alle PR Funktionen.

Oberflächlich intuitiv:

Die Ackermann-Funktion wächst schneller als alle PR Funktionen.

Genauer:

Die Funktion $n \mapsto a(n, n)$ wächst schneller als alle PR Funktionen.

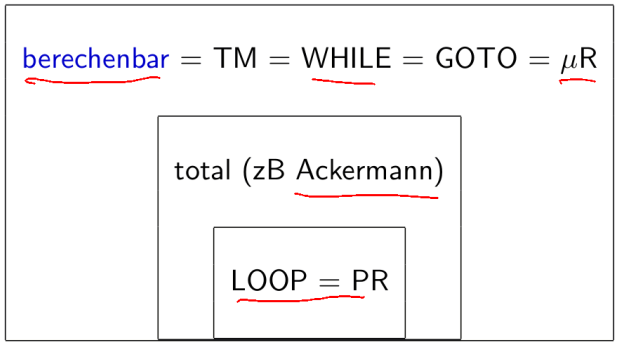
Intuitiver Grund:

- Für fixes t ist $n \mapsto a(t, n)$ PR, denn dies ist A_t.
- A_t braucht aber eine PR Definition der Länge $O(t)$.

Da die Ackermann-Funktion total, berechenbar und nicht PR ist:

Korollar 5.57
Die PR Funktionen sind eine echte Teilklasse der berechenbaren totalen Funktionen.

Die berechenbaren Funktionen



Knobelaufgabe: Sei $b : \mathbb{Z}^3 \to \mathbb{Z}$

$$b(i,j,k) = \textbf{if } j \leq i \textbf{ then } j \textbf{ else } b(b(i+1,j,k), b(j+1,k,i), b(k+1,i,j))$$