Script generated by TTT

Title: seidl: Theoretische_Informatik (14.06.2012)

Date: Thu Jun 14 16:10:45 CEST 2012

Duration: 75:59 min

Pages: 73

Definition 4.30 (PR)

Die Menge PR der primitiv rekursiven Funktionen ist die folgende induktiv definierte Teilmenge aller Funktionen $\mathbb{N}^k \to \mathbb{N}$, $k \geq 0$:

- Die Basisfunktionen 0, s, und π_i^k sind primitiv rekursiv.
- Sind g und h_i primitiv rekursiv, dann auch ihre Komposition

$$f(\overline{x}) \;=\; g(h_1(\overline{x}), \ldots, h_k(\overline{x}))$$

- Sind g und h primitiv rekursiv, dann auch die mit primitiver Rekursion definierte Funktion

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

Lemma 4.31

Jede primitiv-rekursive Funktion ist total.

Beweis:

Induktion über den Aufbau der primitiv-rekursiven Funktionen. □

Definition 4.30 (PR)

Die Menge PR der primitiv rekursiven Funktionen ist die folgende induktiv definierte Teilmenge aller Funktionen $\mathbb{N}^k \to \mathbb{N}$, $k \geq 0$:

- Die Basisfunktionen 0, s, und π_i^k sind primitiv rekursiv.
- Sind g und h_i primitiv rekursiv, dann auch ihre Komposition

$$f(\overline{x}) \;=\; g(h_1(\overline{x}), \ldots, h_k(\overline{x}))$$

- Sind g und h primitiv rekursiv, dann auch die mit primitiver Rekursion definierte Funktion

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

Lemma 4.31

Jede primitiv-rekursive Funktion ist total.

Beweis:

Induktion über den Aufbau der primitiv-rekursiven Funktionen. □

Beispiel 4.32

Alle Konstanten $n \in \mathbb{N}$ sind PR: $1 = s(0)$, $2 = s(1)$, ...

Beispiel 4.32

Alle Konstanten $n \in \mathbb{N}$ sind PR: $1 = s(0)$, $2 = s(1)$, $\ldots$

Addition ist PR:

Beispiel 4.32

Alle Konstanten $n \in \mathbb{N}$ sind PR: $1 = s(0)$, $2 = s(1)$, $\ldots$

Addition ist PR:

$$\begin{aligned} h(r,x,y) &= s(\pi_1^3(r,x,y)) \\ add(0,y) &= \pi_1^1(y) \\ add(x+1,y) &= h(add(x,y),x,y) \end{aligned}$$

Beispiel 4.32

Alle Konstanten $n \in \mathbb{N}$ sind PR: $1 = s(0)$, $2 = s(1)$, $\ldots$

Addition ist PR:

$$\begin{aligned} h(r,x,y) &= s(\pi_1^3(r,x,y)) \\ add(0,y) &= \pi_1^1(y) \\ add(x+1,y) &= h(add(x,y),x,y) \end{aligned}$$

Lesbarer, aber nicht dem syntaktischen PR-Format entsprechend:

$$\begin{aligned} add(0,y) &= y \\ add(x+1,y) &= s(add(x,y)) \end{aligned}$$

Beispiel 4.32

Alle Konstanten $n \in \mathbb{N}$ sind PR: $1 = s(0)$, $2 = s(1)$, $\ldots$

Addition ist PR:

$$\begin{aligned} h(r,x,y) &= s(\pi_1^3(r,x,y)) \\ add(0,y) &= \pi_1^1(y) \\ add(x+1,y) &= h(add(x,y),x,y) \end{aligned}$$

Lesbarer, aber nicht dem syntaktischen PR-Format entsprechend:

$$\begin{aligned} add(0,y) &= y \\ add(x+1,y) &= s(add(x,y)) \end{aligned}$$

Streng genommen also kein Nachweis, dass Addition PR ist.

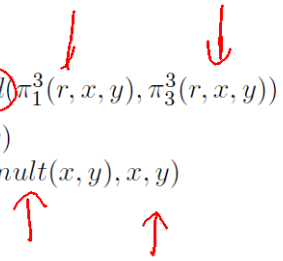
Beispiel 4.32

Alle Konstanten $n \in \mathbb{N}$ sind PR: $1 = s(0)$, $2 = s(1)$, $\ldots$

Beispiel 4.33

Multiplikation ist PR:

$$\begin{aligned} k(y) &= 0 \\ h(r,x,y) &= add(\pi^3_1(r,x,y), \pi^3_3(r,x,y)) \\ mult(0,y) &= k(y) \\ mult(x+1,y) &= h(mult(x,y),x,y) \end{aligned}$$



Beispiel 4.33

Multiplikation ist PR:

$$\begin{aligned} k(y) &= 0 \\ h(r,x,y) &= add(\pi^3_1(r,x,y), \pi^3_3(r,x,y)) \\ mult(0,y) &= k(y) \\ mult(x+1,y) &= h(mult(x,y),x,y) \end{aligned}$$

Lesbarer, aber nicht dem syntaktischen PR-Format entsprechend:

$$\begin{aligned} mult(0,y) &= 0 \\ mult(x+1,y) &= add(mult(x,y),y) \end{aligned}$$

Beispiel 4.33

Multiplikation ist PR:

$$\begin{aligned} k(y) &= 0 \\ h(r,x,y) &= add(\pi^3_1(r,x,y), \pi^3_3(r,x,y)) \\ mult(0,y) &= k(y) \\ mult(x+1,y) &= h(mult(x,y),x,y) \end{aligned}$$

Lesbarer, aber nicht dem syntaktischen PR-Format entsprechend:

$$\begin{aligned} mult(0,y) &= 0 \\ mult(x+1,y) &= add(mult(x,y),y) \end{aligned}$$

In Zukunft: $+$ und $*$ statt add und $mult$.

Beispiel 4.33

Multiplikation ist PR:

$$\begin{aligned} k(y) &= 0 \\ h(r,x,y) &= add(\pi_1^3(r,x,y), \pi_3^3(r,x,y)) \\ mult(0,y) &= k(y) \\ mult(x+1,y) &= h(mult(x,y),x,y) \end{aligned}$$

Lesbarer, aber nicht dem syntaktischen PR-Format entsprechend:

$$\begin{aligned} mult(0,y) &= 0 \\ mult(x+1,y) &= add(mult(x,y),y) \end{aligned}$$

In Zukunft: $+$ und $*$ statt add und $mult$.

Definition 4.34

Wir bezeichen f als eine erweiterte Komposition der Funktionen $g_1, \ldots, g_k$ falls

$$f(x_1, \ldots, x_n) = t$$

Definition 4.34

Wir bezeichen f als eine erweiterte Komposition der Funktionen $g_1, \ldots, g_k$ falls

$$f(x_1, \ldots, x_n) = t$$

so dass t ein Ausdruck ist, der nur aus den Funktionen $g_1, \ldots, g_k$ und den Variablen $x_1, \ldots, x_n$ besteht.

Definition 4.34

Wir bezeichen f als eine erweiterte Komposition der Funktionen $g_1, \ldots, g_k$ falls

$$f(x_1, \ldots, x_n) = t$$

so dass t ein Ausdruck ist, der nur aus den Funktionen $g_1, \ldots, g_k$ und den Variablen $x_1, \ldots, x_n$ besteht.

Beispiel: $f(x,y) = g_1(x, g_2(y, g_3(x)))$

Definition 4.34
Wir bezeichen f als eine erweiterte Komposition der Funktionen $g_1, \ldots, g_k$ falls

$$f(x_1, \ldots, x_n) \;=\; t$$

so dass t ein Ausdruck ist, der nur aus den Funktionen $g_1, \ldots, g_k$ und den Variablen $x_1, \ldots, x_n$ besteht.

Beispiel: $f(x, y) = g_1(x, g_2(y, g_3(x)))$

Lemma 4.35
Eine erweiterte Komposition von PR Funktionen ist wieder PR.

Definition 4.34
Wir bezeichen f als eine erweiterte Komposition der Funktionen $g_1, \ldots, g_k$ falls

$$f(x_1, \ldots, x_n) \;=\; t$$

so dass t ein Ausdruck ist, der nur aus den Funktionen $g_1, \ldots, g_k$ und den Variablen $x_1, \ldots, x_n$ besteht.

Beispiel: $f(x, y) = g_1(x, g_2(y, g_3(x)))$

Lemma 4.35
Eine erweiterte Komposition von PR Funktionen ist wieder PR.

Beweis:
Mit Induktion über Aufbau/Größe von t. □

Definition 4.34
Wir bezeichen f als eine erweiterte Komposition der Funktionen $g_1, \ldots, g_k$ falls

$$f(x_1, \ldots, x_n) \;=\; t$$

so dass t ein Ausdruck ist, der nur aus den Funktionen $g_1, \ldots, g_k$ und den Variablen $x_1, \ldots, x_n$ besteht.

Beispiel: $f(x, y) = g_1(x, g_2(y, g_3(x)))$

Lemma 4.35
Eine erweiterte Komposition von PR Funktionen ist wieder PR.

Beweis:
Mit Induktion über Aufbau/Größe von t. □

Beispiel:

$$\begin{aligned} h_1(x, y) &= g_3(\pi_1^2(x, y)) \\ h_2(x, y) &= g_2(\pi_2^2(x, y), h_1(x, y)) \\ f(x, y) &= g_1(\pi_1^2(x, y), h_2(x, y)) \end{aligned}$$

Das erweiterte Schema der primitiven Rekursion erlaubt

$$\begin{aligned} f(0, \overline{x}) &= t_0 \\ f(m+1, \overline{x}) &= t \end{aligned}$$

Das erweiterte Schema der primitiven Rekursion erlaubt

$$\begin{aligned} f(0, \overline{x}) &= t_0 \\ f(m+1, \overline{x}) &= t \end{aligned}$$

wobei

- t_0 enthält nur PR Funktionen und die x_i,

Das erweiterte Schema der primitiven Rekursion erlaubt

$$\begin{aligned} f(0, \overline{x}) &= t_0 \\ f(m+1, \overline{x}) &= t \end{aligned}$$

wobei

- t_0 enthält nur PR Funktionen und die x_i,
- t enthält nur $f(m, \overline{x})$, PR Funktionen, m und die x_i.

Das erweiterte Schema der primitiven Rekursion erlaubt

$$\begin{aligned} f(0, \overline{x}) &= t_0 \\ f(m+1, \overline{x}) &= t \end{aligned}$$

wobei

- t_0 enthält nur PR Funktionen und die x_i,
- t enthält nur $f(m, \overline{x})$, PR Funktionen, m und die x_i.

Lemma 4.36
Das erweiterte Schema der primitiven Rekursion führt nicht aus PR hinaus.

Das erweiterte Schema der primitiven Rekursion erlaubt

$$\begin{aligned} f(0, \overline{x}) &= t_0 \\ f(m+1, \overline{x}) &= t \end{aligned}$$

wobei

- t_0 enthält nur PR Funktionen und die x_i,
- t enthält nur $f(m, \overline{x})$, PR Funktionen, m und die x_i.

Lemma 4.36
Das erweiterte Schema der primitiven Rekursion führt nicht aus PR hinaus.

Beweis:
Mit Hilfe der erweiterten Komposition. □

Das erweiterte Schema der primitiven Rekursion erlaubt

$$\begin{aligned} f(0, \overline{x}) &= t_0 \\ f(m+1, \overline{x}) &= t \end{aligned}$$

wobei

- t_0 enthält nur PR Funktionen und die x_i,
- t enthält nur $f(m, \overline{x})$, PR Funktionen, m und die x_i.

Lemma 4.36

Das erweiterte Schema der primitiven Rekursion führt nicht aus PR hinaus.

Beweis:

Mit Hilfe der erweiterten Komposition. □

Moral:

Primitive Rekursion erlaubt

$f(m+1, \overline{x}) = \ldots f(m, \overline{x}) \ldots$

Beispiel 4.37

Die Vorgängerfunktion ist PR:

$$\begin{aligned} pred(0) &= 0 \\ pred(x+1) &= x \end{aligned}$$

Beispiel 4.37

Die Vorgängerfunktion ist PR:

$$\begin{aligned} pred(0) &= 0 \\ pred(x+1) &= x \end{aligned}$$

Die modifizierte Differenz ist PR:

$$\begin{aligned} x \dot{-} 0 &= x \\ x \dot{-} (y+1) &= pred(x \dot{-} y) \end{aligned}$$

Definition 4.38

Sei $P(x)$ ein Prädikat, d.h. ein logischer Ausdruck, der in Abhängigkeit von $x \in \mathbb{N}_0$ den Wert **true** oder **false** liefert.

Definition 4.38
Sei $P(x)$ ein Prädikat, d.h. ein logischer Ausdruck, der in Abhängigkeit von $x \in \mathbb{N}_0$ den Wert **true** oder **false** liefert. Dann können wir P in natürlicher Weise eine Funktion

$$\hat{P} : \mathbb{N} \to \{0, 1\}$$

zuordnen: $\hat{P}(x) = 1$ gdw $P(x) =$ **true**.

Definition 4.38
Sei $P(x)$ ein Prädikat, d.h. ein logischer Ausdruck, der in Abhängigkeit von $x \in \mathbb{N}_0$ den Wert **true** oder **false** liefert. Dann können wir P in natürlicher Weise eine Funktion

$$\hat{P} : \mathbb{N} \to \{0, 1\}$$

zuordnen: $\hat{P}(x) = 1$ gdw $P(x) =$ **true**.

Wir nennen P primitiv rekursiv genau dann, wenn $\hat{P}$ primitiv rekursiv ist.

Ist P primitiv rekursiv, dann auch der beschränkte max-Operator

$$\max\{x \leq n \mid P(x)\}$$

wobei $\max \emptyset := 0$.

Ist P primitiv rekursiv, dann auch der beschränkte max-Operator

$$\max\{x \leq n \mid P(x)\} \;=:\; q(n)$$

wobei $\max \emptyset := 0$.

$$q(0) \;=\; 0$$

Ist P primitiv rekursiv, dann auch der beschränkte max-Operator

$$\max\{x < n \mid P(x)\} \;=:\; q(n)$$

wobei $\max \emptyset := 0$.

$$\begin{aligned} q(0) &= 0 \\ q(n+1) &= q(n) + \hat{P}(n) * (n \dot{-} q(n)) \end{aligned}$$

Ist P primitiv rekursiv, dann auch der beschränkte max-Operator

$$\max\{x \leq n \mid P(x)\} \;=:\; q(n)$$

wobei $\max \emptyset := 0$.

$$\begin{aligned} q(0) &= 0 \\ q(n+1) &= q(n) + \hat{P}(n) * (n \dot{-} q(n)) \\ &= \begin{cases} n & \text{falls } P(n) \\ q(n) & \text{sonst} \end{cases} \end{aligned}$$

Ist P primitiv rekursiv, dann auch der beschränkte Existenzquantor

$$\exists x \leq n.\ P(x)$$

Ist P primitiv rekursiv, dann auch der beschränkte Existenzquantor

$$\exists x \leq n.\ P(x) \quad =:\; Q(n)$$

denn:

$$\hat{Q}(0) \;=\; \hat{P}(0)$$

Ist P primitiv rekursiv, dann auch der beschränkte Existenzquantor

$$\exists x \leq n.\ P(x) \quad =:\ Q(n)$$

denn:

$$\begin{aligned} \hat{Q}(0) &= \hat{P}(0) \\ \hat{Q}(n+1) &= \hat{Q}(n) + \hat{P}(n+1) * (1 \dot{-} \hat{Q}(n)) \end{aligned}$$

Ist P primitiv rekursiv, dann auch der beschränkte Existenzquantor

$$\exists x \leq n.\ P(x) \quad =:\ Q(n)$$

denn:

$$\begin{aligned} \hat{Q}(0) &= \hat{P}(0) \\ \hat{Q}(n+1) &= \hat{Q}(n) + \hat{P}(n+1) * (1 \dot{-} \hat{Q}(n)) \end{aligned}$$

$$= \begin{cases} 1 & \text{falls } P(n+1) \\ \hat{Q}(n) & \text{sonst} \end{cases}$$

4.6 PR = LOOP

Hauptproblem bei LOOP $\rightarrow$ PR:
Kodierung *aller* Variablen eines LOOP Programms in *einer* Zahl.

4.6 PR = LOOP

Hauptproblem bei LOOP $\rightarrow$ PR:
Kodierung *aller* Variablen eines LOOP Programms in *einer* Zahl.

Satz 4.39
Die Cantorsche Paarungsfunktion

$$c(x,y) := \binom{x+y+1}{2} + x = (x+y)(x+y+1)/2 + x$$

ist eine Bijektion zwischen $\mathbb{N}^2$ *und* $\mathbb{N}$.

	0	1	2	3	...
0	0	2	5	9	...
1	1	4	8	13	...
2	3	7	12	18	...
3	6	11	17	24	...
⋮	⋮	⋮	⋮	⋮	⋱

4.6 PR = LOOP

Hauptproblem bei LOOP $\to$ PR:
Kodierung *aller* Variablen eines LOOP Programms in *einer* Zahl.

Satz 4.39
Die Cantorsche Paarungsfunktion

$$c(x,y) := \binom{x+y+1}{2} + x = (x+y)(x+y+1)/2 + x$$

ist eine Bijektion zwischen $\mathbb{N}^2$ *und* $\mathbb{N}$.

	0	1	2	3	...
0	0	2	5	9	...
1	1	4	8	13	...
2	3	7	12	18	...
3	6	11	17	24	...
⋮	⋮	⋮	⋮	⋮	⋱

Die Funktion $x \mapsto \binom{x}{2}$ ist PR:

$$\begin{aligned} \binom{0}{2} &= 0 \\ \binom{n+1}{2} &= \binom{n}{2} + n \end{aligned}$$

Die Funktion $x \mapsto \binom{x}{2}$ ist PR:

$$\begin{aligned} \binom{0}{2} &= 0 \\ \binom{n+1}{2} &= \binom{n}{2} + n \end{aligned}$$

Mit Komposition ist auch c PR:

$$c(x,y) = \binom{x+y+1}{2} + x$$

Mit c kodiert man $k+1$ Tupel:

$$\langle n_0, n_1, \ldots, n_k \rangle := c(n_0, c(n_1, \ldots, c(n_k, 0)\ldots))$$

Mit c kodiert man $k+1$ Tupel:

$$\langle n_0, n_1, \ldots, n_k \rangle \;:=\; c(n_0, c(n_1, \ldots, c(n_k, 0)\ldots))$$

Wir brauchen die Umkehrfunktionen p_1 und p_2 von c:

$$p_1(c(x,y)) = x \qquad p_2(c(x,y)) = y$$

Mit c kodiert man $k+1$ Tupel:

$$\langle n_0, n_1, \ldots, n_k \rangle \;:=\; c(n_0, c(n_1, \ldots, c(n_k, 0)\ldots))$$

Wir brauchen die Umkehrfunktionen p_1 und p_2 von c:

$$p_1(c(x,y)) = x \qquad p_2(c(x,y)) = y$$

Damit kann man Projektionsfunktionen auf Tupeln definieren:

$$\begin{aligned} d_0(n) &:= p_1(n) \\ d_1(n) &:= p_1(p_2(n)) \\ &\vdots \\ d_k(n) &:= p_1(\underbrace{p_2 \ldots p_2}_{k}(n)\ldots) \end{aligned}$$

Mit c kodiert man $k+1$ Tupel:

$$\langle n_0, n_1, \ldots, n_k \rangle \;:=\; c(n_0, c(n_1, \ldots, c(n_k, 0)\ldots))$$

Wir brauchen die Umkehrfunktionen p_1 und p_2 von c:

$$p_1(c(x,y)) = x \qquad p_2(c(x,y)) = y$$

Damit kann man Projektionsfunktionen auf Tupeln definieren:

$$\begin{aligned} d_0(n) &:= p_1(n) \\ d_1(n) &:= p_1(p_2(n)) \\ &\vdots \\ d_k(n) &:= p_1(\underbrace{p_2 \ldots p_2}_{k}(n)\ldots) \end{aligned}$$

Sind p_1, p_2 PR, so auch $d_0, \ldots, d_k$.

Satz 4.40

Die Umkehrfunktionen von c sind PR definierbar:

$$\begin{aligned} p_1(n) &= max\{x \leq n \mid \exists y \leq n.\, c(x,y) = n\} \\ p_2(n) &= max\{y \leq n \mid \exists x \leq n.\, c(x,y) = n\} \end{aligned}$$

Satz 4.40
Die Umkehrfunktionen von c sind PR definierbar:

$$\begin{aligned} p_1(n) &= max\{x \leq n \mid \exists y \leq n.\ c(x,y) = n\} \\ p_2(n) &= max\{y \leq n \mid \exists x \leq n.\ c(x,y) = n\} \end{aligned}$$

Beweis:
Alle Funktionen in der Definition der p_i sind PR,

Satz 4.40
Die Umkehrfunktionen von c sind PR definierbar:

$$\begin{aligned} p_1(n) &= max\{x \leq n \mid \exists y \leq n.\ c(x,y) = n\} \\ p_2(n) &= max\{y \leq n \mid \exists x \leq n.\ c(x,y) = n\} \end{aligned}$$

Beweis:
Alle Funktionen in der Definition der p_i sind PR,
auch der Gleichheitstest (Übung!).

Satz 4.40
Die Umkehrfunktionen von c sind PR definierbar:

$$\begin{aligned} p_1(n) &= max\{x \leq n \mid \exists y \leq n.\ c(x,y) = n\} \\ p_2(n) &= max\{y \leq n \mid \exists x \leq n.\ c(x,y) = n\} \end{aligned}$$

Beweis:
Alle Funktionen in der Definition der p_i sind PR,
auch der Gleichheitstest (Übung!).
Die p_i sind die Umkehrfunktionen von c denn:

- Es gibt zu jedem n eindeutige x und y mit $c(x,y) = n$.

Satz 4.40
Die Umkehrfunktionen von c sind PR definierbar:

$$\begin{aligned} p_1(n) &= max\{x \leq n \mid \exists y \leq n.\ c(x,y) = n\} \\ p_2(n) &= max\{y \leq n \mid \exists x \leq n.\ c(x,y) = n\} \end{aligned}$$

Beweis:
Alle Funktionen in der Definition der p_i sind PR,
auch der Gleichheitstest (Übung!).
Die p_i sind die Umkehrfunktionen von c denn:

- Es gibt zu jedem n eindeutige x und y mit $c(x,y) = n$.
  (Bijektivität von c)
- $x \leq c(x,y)$ und $y \leq c(x,y)$.

Damit findet die beschränkte Suche immer die gesuchten x und y.
□

Satz 4.41 (PR = LOOP)

Die primitiv rekursiven sind genau die LOOP-berechenbaren Funktionen.

Satz 4.41 (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.

Satz 4.41 (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.
Dann gibt es ein LOOP-Programm P (mit Variablen $x_0, \ldots, x_k$), das f berechnet.

Satz 4.41 (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.
Dann gibt es ein LOOP-Programm P (mit Variablen $x_0, \ldots, x_k$), das f berechnet.
Mit Induktion über die Struktur von P
konstruieren wir eine PR Funktion g_P mit

$$\underbrace{g_P(\underbrace{\langle a_0, \ldots, a_k \rangle}_{\substack{\text{Belegung der Variablen} \\ \text{beim Start von } P}})}_{\text{Belegung der Variablen am Ende von } P} = \underbrace{\langle b_0, \ldots, b_k \rangle}_{\text{Belegung der Variablen am Ende von } P}$$

Satz 4.41 (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.
Dann gibt es ein LOOP-Programm P (mit Variablen $x_0, \ldots, x_k$), das f berechnet.
Mit Induktion über die Struktur von P
konstruieren wir eine PR Funktion g_P mit

$$\underbrace{g_P(\langle a_0, \ldots, a_k \rangle)}_{\substack{\text{Belegung der Variablen} \\ \text{beim Start von } P}} = \underbrace{\langle b_0, \ldots, b_k \rangle}_{\substack{\text{Belegung der Variablen} \\ \text{am Ende von } P}}$$

Beweis (Forts.):

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

Beweis (Forts.):

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

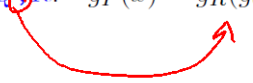
$$g_P(x) = \langle d_0(x), \ldots, d_{i-1}(x), d_j(x) \pm c, d_{i+1}(x), \ldots, d_k(x) \rangle$$

Beweis (Forts.):

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

$$g_P(x) = \langle d_0(x), \ldots, d_{i-1}(x), d_j(x) \pm c, d_{i+1}(x), \ldots, d_k(x) \rangle$$

- P ist $Q;R$: $g_P(x) = g_R(g_Q(x))$



Beweis (Forts.):

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

$$g_P(x) = \langle d_0(x), \ldots, d_{i-1}(x), d_j(x) \pm c, d_{i+1}(x), \ldots, d_k(x) \rangle$$

- P ist $Q;R$: $g_P(x) = g_R(g_Q(x))$
- P ist LOOP x_i DO Q END:

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

$g_P(x)$ berechnet den Zustand nach $d_i(x)$ Iterationen von Q.

Beweis (Forts.):

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

$$g_P(x) = \langle d_0(x), \ldots, d_{i-1}(x), d_j(x) \pm c, d_{i+1}(x), \ldots, d_k(x) \rangle$$

- P ist $Q;R$: $g_P(x) = g_R(g_Q(x))$
- P ist LOOP x_i DO Q END:

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

$g_P(x)$ berechnet den Zustand nach $d_i(x)$ Iterationen von Q.

Berechnet P die Funktion $f : \mathbb{N}^m \to \mathbb{N}$, dann ist f PR denn

$$f(x_1, \ldots, x_m) = d_0(g_P(\langle 0, x_1, \ldots x_m, \underbrace{0, \ldots, 0}_{k-m} \rangle)).$$

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

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

Sei $f : \mathbb{N}^k \to \mathbb{N}$ PR.

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

Sei $f : \mathbb{N}^k \rightarrow \mathbb{N}$ PR.

Mit Induktion über die Erzeugung von f konstruieren wir ein LOOP-Programm P_f, das f berechnet:

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

Sei $f : \mathbb{N}^k \rightarrow \mathbb{N}$ PR.

Mit Induktion über die Erzeugung von f konstruieren wir ein LOOP-Programm P_f, das f berechnet:

- 0:

  x_0 := 0

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

Sei $f : \mathbb{N}^k \rightarrow \mathbb{N}$ PR.

Mit Induktion über die Erzeugung von f konstruieren wir ein LOOP-Programm P_f, das f berechnet:

- 0:

  x_0 := 0
- Nachfolger-Funktion:

  x_0 := x_1 + 1

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

Sei $f : \mathbb{N}^k \rightarrow \mathbb{N}$ PR.

Mit Induktion über die Erzeugung von f konstruieren wir ein LOOP-Programm P_f, das f berechnet:

- 0:

  x_0 := 0
- Nachfolger-Funktion:

  x_0 := x_1 + 1
- Projektions-Funktionen, zB $\pi^3_2(x_1, x_2, x_3) = x_2$:

  x_0 := x_2

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

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

- 0:
  x_0 := 0
- Nachfolger-Funktion:
  x_0 := x_1 + 1
- Projektions-Funktionen, zB $\pi_2^3(x_1, x_2, x_3) = x_2$:
  x_0 := x_2
- Komposition, zB $f(x_1, x_2) = g(h_1(x_1, x_2), h_2(x_1, x_2))$:

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

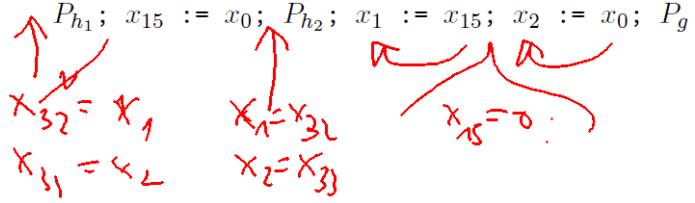
Sei $f : \mathbb{N}^k \rightarrow \mathbb{N}$ PR.
Mit Induktion über die Erzeugung von f konstruieren wir ein LOOP-Programm P_f, das f berechnet:

- 0:
  x_0 := 0
- Nachfolger-Funktion:
  x_0 := x_1 + 1
- Projektions-Funktionen, zB $\pi_2^3(x_1, x_2, x_3) = x_2$:
  x_0 := x_2
- Komposition, zB $f(x_1, x_2) = g(h_1(x_1, x_2), h_2(x_1, x_2))$:
  P_{h_1}; x_{15} := x_0; P_{h_2}; P_g

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

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

- 0:
  x_0 := 0
- Nachfolger-Funktion:
  x_0 := x_1 + 1
- Projektions-Funktionen, zB $\pi_2^3(x_1, x_2, x_3) = x_2$:
  x_0 := x_2
- Komposition, zB $f(x_1, x_2) = g(h_1(x_1, x_2), h_2(x_1, x_2))$:
  P_{h_1}; x_{15} := x_0; P_{h_2}; x_1 := x_{15}; x_2 := x_0; P_g



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

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

- 0:
  x_0 := 0
- Nachfolger-Funktion:
  x_0 := x_1 + 1
- Projektions-Funktionen, zB $\pi_2^3(x_1, x_2, x_3) = x_2$:
  x_0 := x_2
- Komposition, zB $f(x_1, x_2) = g(h_1(x_1, x_2), h_2(x_1, x_2))$:
  P_{h_1}; x_{15} := x_0; P_{h_2}; x_1 := x_{15}; x_2 := x_0; P_g

  Funktions-Komposition wird simuliert durch Hintereinanderausführung (;)

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

Sei $f : \mathbb{N}^k \rightarrow \mathbb{N}$ PR.

Mit Induktion über die Erzeugung von f konstruieren wir ein LOOP-Programm P_f, das f berechnet:

- 0:

  x_0 := 0

- Nachfolger-Funktion:

  x_0 := x_1 + 1

- Projektions-Funktionen, zB $\pi_2^3(x_1, x_2, x_3) = x_2$:

  x_0 := x_2

- Komposition, zB $f(x_1, x_2) = g(h_1(x_1, x_2), h_2(x_1, x_2))$:

  P_{h_1}; x_{15} := x_0; P_{h_2}; x_1 := x_{15}; x_2 := x_0; P_g

  Funktions-Komposition wird simuliert durch Hintereinanderausführung (;) wobei Zwischenergebnisse in separaten Variablen zwischengespeichert werden müssen.

Beweis (Forts.):

- Primitive Rekursion:

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