6 Berechenbarkeit

Programmierbare/Berechenbare Funktionen

- Imperative Programmiersprache: while-Programme
- Funktionale Programmiersprache: μ-rekursive Ausdrücke
- Logische Programmiersprachen: Prolog ...
  Deklarativ vs. Prozedural
- Abstrakte Maschinen Modelle
  Turing-Maschine, Register-Maschine;
- Techniken: Simulation von Berechnungen, Übersetzung, Interpretation
- Universelle Modelle: Compiler

Aufbau Kapitel 6:

- Primitiv rekursive Funktionen
- μ-rekursive Funktionen (partiell rekursive Funktionen)
- Universalität
- Rekursionstheorie
- Churchsche These
- Wortfunktionen

6.1 Primitiv rekursive Funktionen $\mathcal{P}(\mathbb{N})$

Funktionen: $f : \mathbb{N}^n \to \mathbb{N} \quad n \geq 1$. **Arithmetische Funktionen.**

Verwende „effektive"- Operatoren auf Funktionen, um aus Ausgangsfunktionen + Operatoren neue Funktionen zu definieren.

Erinnerung: Gleichheit von Funktionen $f : A \to B$, $g : A \to B$

$f \sqsubseteq g$ gdw $dom(f) \subseteq dom(g) \wedge f(x) = g(x)$
für $x \in dom(f)$

$f = g$ gdw $dom(f) = dom(g) \wedge f(x) = g(x)$
für $x \in dom(f)$ gdw $(f \sqsubseteq g \wedge g \sqsubseteq f)$.

6.1 Definition Komposition - Primitive Rekursion

a) Seien $g : \mathbb{N}^n \to \mathbb{N}$, $h_1, \ldots, h_n : \mathbb{N}^m \to \mathbb{N}$ Funktionen $n, m \geq 1$
$f : \mathbb{N}^m \to \mathbb{N}$ entsteht aus g und $h_1, \ldots, h_n$ durch **Komposition**, falls gilt
$f(\vec{x}) \downarrow$ gdw $h_1(\vec{x}) \downarrow, \ldots, h_n(\vec{x}) \downarrow$ und $g(h_1(\vec{x}), \ldots, h_n(\vec{x})) \downarrow$ und in diesem Fall ist
$f(\vec{x}) = g(h_1(\vec{x}), \ldots, h_n(\vec{x}))$
Schreibe dafür $f = g \circ (h_1, \ldots, h_n)$
Beachte Stelligkeiten der Funktionen.
Sind $g, h_1, \ldots, h_n$ total, so auch f.
Gilt die Umkehrung?

Operatoren auf Funktionen

b) Seien $g : \mathbb{N}^{n+1} \to \mathbb{N}$, $h : \mathbb{N}^{n+2} \to \mathbb{N}$ Funktionen.
$f : \mathbb{N}^{n+1} \to \mathbb{N}$ entsteht aus g und h durch **primitive Rekursion**, falls gilt:
$f(\vec{x}, 0) \downarrow$ gdw $g(\vec{x}, 0) \downarrow$ $(\vec{x} \in \mathbb{N}^n)$ und in diesem Fall ist
$f(\vec{x}, 0) = g(\vec{x}, 0)$ und
$f(\vec{x}, y+1) \downarrow$ gdw $f(\vec{x}, y) \downarrow$ und $h(\vec{x}, f(\vec{x}, y), y) \downarrow$
$(\vec{x} \in \mathbb{N}^n)$ und in diesem Fall ist
$f(\vec{x}, y+1) = h(\vec{x}, f(\vec{x}, y), y)$
Schreibe dafür $f = R(g, h)$.
Beachte Stelligkeiten der Funktionen.

6.2 Bemerkung - Beispiele: Betrachtet man die Gleichung

$$F(\vec{x}, z) = \begin{cases} g(\vec{x}, 0) & z = 0 \\ h(\vec{x}, F(\vec{x}, y), y) & z = y + 1 \end{cases}$$

so ist $f = R(g, h)$ die kleinste (bzgl. $\sqsubseteq$) Funktion, die diese Gleichung erfüllt.

Sind $g(\cdot, 0) : \mathbb{N}^n \to \mathbb{N}$ und $h : \mathbb{N}^{n+2} \to \mathbb{N}$ total, so ist auch f total.

Gilt die Umkehrung?

Die primitive Rekursion folgt einem sehr strengen Schema und entspricht der Berechnung des Funktionswerts $f(\vec{x}, n+1)$ aus dem Funktionswert $f(\vec{x}, n)$, wobei die Verankerung bei $f(\vec{x}, 0)$ erfolgt.

Beispiele

- $+$ Addition auf $\mathbb{N}^2$: Mit $g(u,v) = u$ und $h(u,v,w) = v+1$ gilt $x + y = R(g,h)$

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

  oder

  $$x + y = \begin{cases} x & \text{falls } y = 0 \\ (x + z) + 1 & \text{falls } y = z + 1 \end{cases}$$

- $\cdot$ Multiplikation auf $\mathbb{N}^2$: Mit $g(u,v) = 0$ und $h(u,v,w) = v + u$ gilt $x \cdot y = R(g,h)$

  $$x \cdot y = \begin{cases} 0 & \text{falls } y = 0 \\ (x \cdot z) + x & \text{falls } y = z + 1 \end{cases}$$

- Fakultät fac auf $\mathbb{N}$: Mit $g(u) = 1$ und $h(u,v) = u \cdot (v+1)$ gilt $fac(y) = R(g,h)$

  $$fac(y) = \begin{cases} 1 & \text{falls } y = 0 \\ fac(z) \cdot (z+1) & \text{falls } y = z + 1 \end{cases}$$

- Welche Funktion f wird durch folgende Festlegung definiert. Sei

  $$h(u,v) = \begin{cases} u & u \text{ gerade} \\ \uparrow & \text{sonst} \end{cases}$$

  $$f(y) = \begin{cases} 0 & \text{falls } y = 0 \\ h(f(z), z) & \text{falls } y = z + 1 \end{cases}$$

Primitiv rekursive Ausdrücke

6.3 Definition

Syntax: Die Menge der primitiv rekursiven Ausdrücke sind die Zeichenreihen, die durch den folgenden Kalkül erzeugt werden:

$$\overline{NULL} \qquad \overline{SUCC} \qquad \overline{PROJ(i)} \text{ für } i \geq 1$$

$$\frac{G, H_1, \ldots, H_m}{KOMP(G, H_1, \ldots, H_m)} \text{ für } m \geq 1 \qquad \frac{G, H}{REK(G, H)}$$

Semantik: Jeder primitiv rekursive Ausdruck π repräsentiert für beliebige Stelligkeit $n \geq 1$ eine Funktion $f_\pi^{(n)} : \mathbb{N}^n \to \mathbb{N}$, die induktiv über den Aufbau von π wie folgt definiert ist:

$$\left.\begin{array}{l} f_{NULL}^{(n)}(x_1, \ldots, x_n) = 0 \\ f_{SUCC}^{(n)}(x_1, \ldots, x_n) = x_1 + 1 \\ f_{PROJ(i)}^{(n)}(x_1, \ldots, x_n) = \begin{cases} x_i & \text{falls } 1 \leq i \leq n \\ 0 & \text{sonst} \end{cases} \end{array}\right\} \text{Grundfunktionen}$$

$$f_{KOMP(G,H_1,\ldots,H_m)}^{(n)}(x_1, \ldots, x_n) = f_G^{(m)} \circ (f_{H_1}^{(n)}, \ldots, f_{H_m}^{(n)})(x_1, \ldots, x_n)$$

$f_{REK(G,H)}^{(n)} = R(f_G^{(n)}, f_H^{(n+1)})$, d. h.

$f_{REK(G,H)}^{(n)}(x_1, \ldots, x_{n-1}, 0) = f_G^{(n)}(x_1, \ldots, x_{n-1}, 0)$ und

$$f_{REK(G,H)}^{(n)}(x_1, \ldots, x_{n-1}, y+1) = f_H^{(n+1)}(x_1, \ldots, x_{n-1}, f_{REK(G,H)}^{(n)}(x_1, \ldots, x_{n-1}, y), y)$$

Primitiv rekursive Funktionen

Die Menge aller Funktionen $f : \mathbb{N}^n \to \mathbb{N}$ $(n > 0)$, für die $f = f_\pi^{(n)}$ mit einem primitiv rekursiven Ausdruck π gilt, heißt die Menge der **primitiv rekursiven Funktionen**.

Bezeichnung $\mathcal{P}(\mathbb{N})$

6.4 Beispiel Folgende Funktionen sind primitiv rekursiv.

- Konstante Funktionen beliebiger Stelligkeit:
  $a \in \mathbb{N}$ $\quad c_a^{(n)} : \mathbb{N}^n \to \mathbb{N}$ $\quad c_a^{(n)}(x_1, \ldots, x_n) = a$ (total)

Zeige $c_a^{(n)} = f_\pi^{(n)}$ für geeignetes π:

$a = 0$ so klar, wähle $\pi = NULL$

$a = 1$ $\quad f^{(n)}_{KOMP(SUCC,NULL)}(\vec{x}) = f^{(1)}_{SUCC}(f^{(n)}_{NULL}(\vec{x})) = 1$

Ind. Schritt:

$a = m$ $\quad$ sei $f^{(n)}_{\pi_a}(\vec{x}) = a$

$a = m+1$ $f^{(n)}_{KOMP(SUCC,\pi_a)}(\vec{x}) = f^{(1)}_{SUCC}(f^{(n)}_{\pi_a}(\vec{x})) = m+1$

d. h. $\pi_a =$ KOMP(SUCC,KOMP(SUCC,...KOMP(SUCC,NULL) ...))
a-mal $KOMP(SUCC, \ldots)$

Beispiel (Forts.)

- Die Vorgänger Funktion auf $\mathbb{N}$ $pred : \mathbb{N} \to \mathbb{N}$ ist primitiv rekursiv:
  $pred(0) = 0$
  $pred(y+1) = y$ (total)
  $pred(0) = f^{(1)}_{NULL}(0)$
  $pred(y+1) = f^{(2)}_{PROJ(2)}(pred(y), y) = y$,
  d. h. mit $PRED = REK(NULL, PROJ(2))$ gilt
  $pred = f^{(1)}_{PRED} = f^{(1)}_{REK(NULL,PROJ(2))}$.

6.5 Lemma

Die Menge der primitiv rekursiven Funktionen ist abgeschlossen gegenüber Komposition und primitiver Rekursion.

Sind $g : \mathbb{N}^m \to \mathbb{N}, h_1, \ldots, h_m : \mathbb{N}^n \to \mathbb{N} \in \mathcal{P}(\mathbb{N})$, so auch $g \circ (h_1, \ldots, h_m) : \mathbb{N}^n \to \mathbb{N} \in \mathcal{P}(\mathbb{N})$.

Sind $g : \mathbb{N}^{n+1} \to \mathbb{N}, h : \mathbb{N}^{n+2} \to \mathbb{N} \in \mathcal{P}(\mathbb{N})$, so auch $R(g, h)$.

Beweis: Seien G und $H_1, \ldots, H_m$ primitiv rekursive Ausdrücke für g und $h_1, \ldots, h_m$. Dann ist $KOMP(G, H_1, \ldots, H_m)$ ein primitiv rekursiver Ausdruck für $g \circ (h_1, \ldots, h_m)$.

Analog ist $REK(G, H)$ primitiv rekursiver Ausdruck für $R(g, h)$, falls G, H primitiv rekursive Ausdrucke für g bzw. h sind.

Primitiv rekursive Funktionen (Fort.)

Die Menge der primitiv rekursiven Funktionen $\mathcal{P}(\mathbb{N})$ ist also charakterisiert als die kleinste Menge von Funktionen $f : \mathbb{N}^n \to \mathbb{N}$ $(n > 0)$ die die Grundfunktionen enthält und abgeschlossen ist gegenüber Komposition und primitiver Rekursion.

In der Literatur findet man oft die Betrachtung von Funktionen $f : \mathbb{N}^n \to \mathbb{N}^m$ $(n, m > 0)$. Diese lassen sich über die **Parallelausführung** $f := \langle g, h \rangle$ von $g : \mathbb{N}^n \to \mathbb{N}^s, h : \mathbb{N}^n \to \mathbb{N}^t$ die erklärt ist durch

$$f : \mathbb{N}^n \to \mathbb{N}^{s+t} \qquad \langle g, h \rangle(x) = (g(x), h(x))$$

aus den obigen Funktionen gewinnen.

Offenbar sind die primitiv rekursiven Ausdrücke sehr einfache Programme. Sie sind aufgebaut aus $NULL, SUCC, PROJ(i)$ und den variadischen Operator $KOMP$ und den binären Operator REK. Die Interpretation der atomaren Ausdrücke durch die Grundfunktionen und der Operatoren durch die offensichtlich „effektiven“ Operationen Komposition und primitive Rekursion machen deutlich, dass diese Programme effektive Berechnungen darstellen. Jedes Programm erlaubt es für jedes $n > 0$ eine Funktion der Stelligkeit n zu berechnen. D. h. ein Programm berechnet unendlich viele Funktionen.

Beachte: Ein primitiv rekursiver Ausdruck stellt stets für jedes n eine Funktion dar. Diese können recht unterschiedlich sein. Siehe z. B. $f^{(n)}_{PROJ(i)}$. Welche Funktion ist $f^{(2)}_{REK(NULL,PROJ(2))}$?

Nachweis von Eigenschaften primitiv rekursiver Ausdrücke oder primitiv rekursiver Funktionen

Erneut: Induktion über Aufbau der primitiv rekursiven Ausdrücke (strukturelle Induktion) bzw. für die Menge $\mathcal{P}(\mathbb{N})$ die sogenannte **Induktion über den Aufbau**: Zeige die Eigenschaft gilt für die Grundfunktionen und die Eigenschaft bleibt erhalten bei Komposition und primitiver Rekursion.

6.6 Lemma

Jede primitiv rekursive Funktion ist total.

6.7 Beispiel Weitere primitiv rekursive Funktionen

$add : \mathbb{N}^2 \to \mathbb{N}$ $\qquad$ $add(x, y) = x + y$ primitiv rekursiv

$add(x, 0) = f^{(2)}_{PROJ(1)}(x, 0)$

$add(x, y + 1) = f^{(1)}_{SUCC}(f^{(3)}_{PROJ(2)}(x, add(x, y), y))$

$ADD :: REK(PROJ(1), KOMP(SUCC, PROJ(2))$

ist ein primitiv rekursiver Ausdruck für add, d. h. $add = f^{(2)}_{ADD}$.

Die Multiplikation $mult : \mathbb{N}^2 \to \mathbb{N}$ mit $mult(x, y) = x \cdot y$ ist primitiv rekursiv:

$mult(x, 0) = f^{(2)}_{NULL}(x, 0)$

$mult(x, y + 1) = add(f^{(3)}_{PROJ(1)}(x, mult(x, y), y),$
$\qquad f^{(3)}_{PROJ(2)}(x, mult(x, y), y))$

d. h. $REK(NULL, KOMP(ADD, PROJ(1), PROJ(2)))$

repräsentiert folglich $mult$.

Beispiele und Vereinfachungen

Die Funktion $sgn : \mathbb{N} \longrightarrow \mathbb{N}$ mit

$$sgn(x) = \begin{cases} 0 & x = 0 \\ 1 & \text{sonst} \end{cases}$$

$sgn(0) = f^{(1)}_{NULL}(0)$

$sgn(y+1) = f^{(2)}_{KOMP(SUCC,NULL)}(sgn(y), y)$

d. h. $REK(NULL, KOMP(SUCC, NULL))$ repräsentiert sgn.

Analog die Funktion $\overline{sgn} : \mathbb{N} \longrightarrow \mathbb{N}$ mit

$$\overline{sgn}(x) = \begin{cases} 1 & x = 0 \\ 0 & \text{sonst} \end{cases}$$

Vereinfachungen: Auflockerung des strengen Schemas der primitiven Rekursion.

- Variablen permutieren, mehrfache Verwendung, oder Nicht-Verwendung von Variablen.
- Weitere Abschlusseigenschaften.
- Verwendung bereits als primitiv rekursiv nachgewiesener Funktionen.

Vereinfachungen

6.8 Lemma

Sei $g : \mathbb{N}^m \to \mathbb{N}$ primitiv rekursiv, $n \geq m$ und seien $1 \leq i_1 \leq n, \ldots, 1 \leq i_m \leq n$ Indizes. Dann ist auch die Funktion $h : \mathbb{N}^n \to \mathbb{N}$ mit

$$h(x_1, \ldots, x_n) = g(x_{i_1}, \ldots, x_{i_m})$$

primitiv rekursiv.

Beweis: Es gilt

$h(x_1, \ldots, x_n) =$
$g(f^{(n)}_{PROJ(i_1)}(x_1, \ldots, x_n), \ldots, f^{(n)}_{PROJ(i_m)}(x_1, \ldots, x_n))$

6.9 Beispiel Es genügt in Zukunft, den Nachweis der primitiven Rekursion einer Funktion auf der Basis einer Rekursionsgleichung wie bei den folgenden Funktionen zu führen.

- Nicht negative Differenz: $- : \mathbb{N}^2 \to \mathbb{N}$
  $x - 0 \quad = x$
  $x - (y + 1) = pred(x - y)$
- Fakultät $fac : \mathbb{N} \to \mathbb{N}$
  $fac(0) \quad = 1 \; (= f^{(1)}_{KOMP(SUCC,NULL)}(0))$
  $fac(y + 1) = fac(y) \cdot (y + 1)$
  $(= f^{(2)}_{KOMP(MULT(PROJ(1),KOMP(SUCC,PROJ(2))))}(fac(y), y))$

Weitere Abschlusseigenschaften

- Insbesondere ist die Funktion $|x - y| : \mathbb{N}^2 \longrightarrow \mathbb{N}$, mit

$$|x - y| \;=\; (x - y) + (y - x)$$

  primitiv rekursiv.

- Einfache Fallunterscheidung. Oft werden Funktionen nur in bestimmten Bereichen benötigt. Sei etwa $h : \mathbb{N}^2 \longrightarrow \mathbb{N}$ primitiv rekursiv, dann ist auch die Funktion

  $F(x, y) = \begin{cases} h(x, y) & \text{falls } x > y \\ 0 & \text{sonst} \end{cases}$ primitiv rekursiv.

  Es ist $F(x, y) = sgn(x - y) \cdot h(x, y)$

  Später werden wir allgemeinere Formen der Fallunterscheidung kennenlernen.

6.10 Lemma Abschluss von $\mathcal{P}(\mathbb{N})$ gegenüber Iteration.

Sei $f : \mathbb{N} \longrightarrow \mathbb{N}$ primitiv rekursiv, dann ist auch $g : \mathbb{N}^2 \longrightarrow \mathbb{N}$ mit $g(x, t) = f^t(x)$ primitiv rekursiv.

Beweis: Es ist

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

Frage: Lässt sich jede „Rekursionsgleichung“ durch primitive Rekursion simulieren?

Andere Rekursionsformate

- Das folgende Format ist erlaubt:
  $$\begin{aligned} f(0, y) &= g(y) \\ f(x+1, y) &= h(x, f(x, y), y) \end{aligned}$$
  Mit primitiv rekursiven Funktionen g, h. Dann ist f auch primitiv rekursiv (Argumente vertauscht).
- Hingegen ist die alternative Definition der **Iteration**:
  $$\begin{aligned} g(x, 0) &= x \\ g(x, t+1) &= g(f(x), t) \end{aligned}$$
  nicht vom Format einer primitiven Rekursion, da der Parameter $f(x)$ statt x in der Rekursion verwandt wird.
  Wir werden gleich zeigen, dass auch dieses Format erlaubt ist.
- Allerdings ist die **Ackermannfunktion** $A : \mathbb{N}^2 \longrightarrow \mathbb{N}$, die durch folgende Rekursionsgleichung definiert wird
  $$\begin{aligned} A(0, y) &= y + 1 \\ A(x+1, 0) &= A(x, 1) \\ A(x+1, y+1) &= A(x, A(x+1, y)) \end{aligned}$$
  keine primitiv rekursive Funktion (Beweis später).

A ist eine totale Funktion, die sehr schnell wächst. Überzeugen Sie sich!

Andere Rekursionsformate (Fort.)

6.11 Lemma

Seien $g : \mathbb{N} \to \mathbb{N}$, $h : \mathbb{N}^3 \to \mathbb{N}$, $w : \mathbb{N} \to \mathbb{N}$ primitiv rekursiv und $f : \mathbb{N}^2 \to \mathbb{N}$ durch die Gleichungen

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

definiert.

Dann ist auch f primitiv rekursiv.

Beweis:

Sei $F(t,x,y) = \begin{cases} f(w^{t-y}(x), y) & \text{falls } t \geq y \\ 0 & \text{sonst} \end{cases}$

Es gilt $F(t,x,0) = f(w^t(x),0) = g(w^t(x))$ und
für $t \geq y+1$

$$\begin{aligned} F(t,x,y+1) &= f(w^{t-y-1}(x), y+1) \\ &= h(w^{t-y-1}(x), f(w(w^{t-y-1}(x)), y), y) \\ &= h(w^{t-y-1}(x), f(w^{t-y}(x), y), y) \\ &= h(w^{t-y-1}(x), F(t,x,y), y) \end{aligned}$$

Für $t < y+1$ gilt

$$F(t,x,y+1) = 0$$

F ist also primitiv rekursiv.
Wegen $f(x,y) = F(y,x,y)$ ist auch f primitiv rekursiv.

Fallunterscheidung

6.12 Lemma Abschluss gegenüber Fallunterscheidung

Seien $g_i : \mathbb{N}^n \to \mathbb{N}$, $h_i : \mathbb{N}^n \to \mathbb{N}$ mit h_i totale Funktionen für $i = 1, \ldots, r$, so dass es zu jedem $\vec{x} \in \mathbb{N}^n$ es genau ein i gibt mit $h_i(\vec{x}) = 0$. Die Fallunterscheidung mit den Funktionen g_i, h_i ist die Funktion

$f = FU(g_i, h_i \;\; i = 1, \ldots, k)$, mit

$$f(\vec{x}) = \begin{cases} g_1(\vec{x}) & \text{falls } h_1(\vec{x}) = 0 \\ \vdots & \\ g_k(\vec{x}) & \text{falls } h_k(\vec{x}) = 0 \end{cases}$$

Offenbar gilt:
$f(\vec{x}) = \overline{sgn}(h_1(\vec{x})) \cdot g_1(\vec{x}) + \cdots + \overline{sgn}(h_k(\vec{x})) \cdot g_k(\vec{x})$,
d. h. sind $g_i, h_i \in \mathcal{P}(\mathbb{N})$, so ist auch $f \in \mathcal{P}(\mathbb{N})$.

Die Fallunterscheidung wird meistens mit $k = 2$ angewendet. Sei h gegeben und $h_1(x) = h(x), h_2(x) = \overline{sgn}(h(x))$. Dann gilt

$$f(x) = \begin{cases} g_1(x) & h_1(x) = 0 \\ g_2(x) & h_2(x) = 0 \end{cases} = \begin{cases} g_1(x) & h(x) = 0 \\ g_2(x) & \text{sonst} \end{cases}$$

Primitiv rekursive Relationen

6.13 Definition

Eine **Relation $R \subseteq \mathbb{N}^n$ heißt primitiv rekursiv**, falls ihre charakteristische Funktion $\chi_R \in \mathcal{P}(\mathbb{N})$.

Erinnerung für $\vec{x} \in \mathbb{N}^n$ gilt:

$$\chi_R(\vec{x}) = \begin{cases} 1 & \text{falls } \vec{x} \in R \\ 0 & \text{falls } \vec{x} \notin R \end{cases}$$

6.14 Beispiel

Primitiv rekursive Relationen sind:

- Die Gleichheitsrelation: $= \quad \subseteq \mathbb{N} \times \mathbb{N}$
  $\chi_=(x, y) = 1 - |x - y|$

- Kleinerrelation: $< \quad \subseteq \mathbb{N} \times \mathbb{N}$
  $\chi_<(x, y) = sgn(y - x)$

- Kleinergleichrelation: $\leq \quad \subseteq \mathbb{N} \times \mathbb{N}$
  $\chi_\leq(x, y) = \chi_=(x, y) + \chi_<(x, y)$
- Analog Größerrelation und Größergleichrelation.

Abschlusseigenschaften primitiv rekursiver Relationen

Erinnerung:

Seien $R, S \subseteq \mathbb{N}^n$ Relationen.

Dann

- $\neg R$ **Komplement** von R $\quad \mathbb{N}^n - R$
- $R \wedge S$ **Durchschnitt** von R und S $\quad R \cap S$
- $R \vee S$ **Vereinigung** von R und S $\quad R \cup S$

6.15 Lemma

Sind $R, S \subseteq \mathbb{N}^n$ primitiv rekursiv, so auch $\neg R$, $R \wedge S$, $R \vee S$.

Beweis: Es ist

$\chi_{\neg R} = 1 - \chi_R$, $\chi_{R \wedge S} = \chi_R \cdot \chi_S$ und $R \vee S = \neg(\neg R \wedge \neg S)$.

Frage: Gilt $\chi_{R \vee S} = \chi_R + \chi_S$?

Insbesondere sind $\neq, \leq, >, \geq$ primitiv rekursiv.

Reduzierbarkeit

6.16 Lemma

Seien $S \subseteq \mathbb{N}^n$, $h_i : \mathbb{N}^n \to \mathbb{N}$ $(1 \leq i \leq n)$ primitiv rekursiv. Dann ist auch die Relation $R \subseteq \mathbb{N}^m$ mit

$$R\vec{x} \quad \text{gdw} \quad Sh_1(\vec{x}) \cdots h_n(\vec{x})$$

primitiv rekursiv.

R ist auf S **primitiv rekursiv reduzierbar**, falls es primitiv rekursive Funktionen $h_i : \mathbb{N}^m \to \mathbb{N}$ $(1 \leq i \leq n)$ gibt mit obiger Eigenschaft.

Beweis:

$$\begin{aligned}\chi_R(\vec{x}) &= \chi_S(h_1(\vec{x}), \ldots, h_n(\vec{x})) \\ &= \chi_S \circ (h_1, \ldots, h_n)(\vec{x})\end{aligned}$$

6.17 Beispiel Sei $R \subseteq \mathbb{N}^2$ mit

Rxy gdw x ist ganzzahliger Anteil der Quadratwurzel von y.

Dann ist R primitiv rekursiv

$$Rxy \quad \text{gdw} \quad x \cdot x \leq y \wedge (x+1) \cdot (x+1) > y$$

Wende Lemma an mit

$$Suvw \text{ gdw } u \leq v \wedge w > v$$

$$h_1(x,y) = x \cdot x, h_2(x,y) = y, h_3(x,y) = (x+1) \cdot (x+1)$$

Fallunterscheidung mit primitiv rekursiven Relationen

6.18 Lemma

$R_i \subseteq \mathbb{N}^n \; 1 \leq i \leq m$ sind paarweise disjunkte primitiv rekursive Relationen und $h_1, \ldots, h_{m+1}$ n-stellige Funktionen mit $h_i \in \mathcal{P}(\mathbb{N})$ $i = 1, \ldots, m+1$.

Dann gilt für $f : \mathbb{N}^n \longrightarrow \mathbb{N}$ mit

$$f(\vec{x}) = \begin{cases} h_1(x) & \text{falls } R_1\vec{x} \\ \vdots & \\ h_m(\vec{x}) & \text{falls } R_m\vec{x} \\ h_{m+1}(\vec{x}) & \text{sonst} \end{cases}$$

$f \in \mathcal{P}(\mathbb{N})$.

Beweis:

$$f(\vec{x}) = \chi_{R_1}(\vec{x}) \cdot h_1(\vec{x}) + \cdots + \chi_{R_m}(x) \cdot h_m(\vec{x}) + (1 - (\chi_{R_1 \vee \cdots \vee R_m}(\vec{x})) \cdot h_{m+1}(\vec{x})$$

6.19 Beispiel Maximum und Minimum von (n)-zwei Zahlen

$$\max(x, y) = \begin{cases} x & \text{falls } x > y \\ y & \text{sonst} \end{cases}$$

Weitere Abschlusseigenschaften von $\mathcal{P}(\mathbb{N})$

6.20 Definition

1. Die **beschränkte Summation** und **beschränkte Multiplikation** mit der Funktion $g : \mathbb{N}^{n+1} \longrightarrow \mathbb{N}$ sind erklärt durch

$$f, h : \mathbb{N}^{n+1} \longrightarrow \mathbb{N}$$

$$f(\vec{x}, y) = \sum_{z \leq y} g(\vec{x}, z) \qquad h(\vec{x}, y) = \prod_{z \leq y} g(\vec{x}, z)$$

2. Die **beschränkte Minimierung** mit der Funktion $g : \mathbb{N}^{n+1} \longrightarrow \mathbb{N}$ ist erklärt durch die Funktion $f : \mathbb{N}^{n+1} \longrightarrow \mathbb{N}$ mit

$$f(\vec{x}, y) = \begin{cases} u & u \leq y,\ g(\vec{x}, u) = 0 \text{ und } g(\vec{x}, z) > 0 \text{ für } z < u \\ 0 & g(\vec{x}, z) > 0 \text{ für alle } z \leq y \\ \uparrow & \text{es gibt } u \leq y \text{ mit } g(\vec{x}, u)\uparrow,\ g(\vec{x}, z) > 0 \text{ für } z < u \end{cases}$$

   Bezeichnung $f(\vec{x}, y) = \mu_{z \leq y}[g(\vec{x}, z) = 0]$

3. Die **beschränkte Minimierung** mit einer Relation $R \subseteq \mathbb{N}^{n+1}$ ist erklärt durch die Funktion $f : \mathbb{N}^{n+1} \longrightarrow \mathbb{N}$ mit

$$f(\vec{x}, y) = \begin{cases} \text{kleinstes } z \text{ mit } z \leq y \wedge R\vec{x}z & \text{falls } z \text{ existiert} \\ y & \text{sonst} \end{cases}$$

   Schreibe $f(\vec{x}, y) = \mu z \leq y.R\vec{x}z$ f **ist total.**

Beispiele

6.21 Beispiel

1. Mit $g(y) = y + 1$ ergibt sich

$$\begin{aligned} f(y) &= \sum_{z \le y} g(z) = \frac{(y+1) \cdot (y+2)}{2} \\ h(y) &= \prod_{z \le y} g(z) = fac(y+1) \end{aligned}$$

   Mit $g(x, y) = x$ ergibt sich

$$\begin{aligned} f(x, y) &= \sum_{z \le y} g(x, z) = x \cdot (y+1) \\ h(x, y) &= \prod_{z \le y} g(x, z) = x^{y+1} \end{aligned}$$

2. Sei $f(x, y) = \mu_{z \le y}[g(x, z) = 0]$.
   Wir betrachten zwei Funktionen g:

$$g(x, y) = x - y \qquad \text{liefert} \quad f(x, y) = \begin{cases} 0 & x > y \\ x & x \le y \end{cases}$$

$$g(x, y) = \begin{cases} x \cdot y & x + y > 0 \\ \uparrow & \text{sonst} \end{cases} \quad \text{liefert} \quad f(x, y) = \begin{cases} 0 & x > 0 \\ \uparrow & x = 0 \end{cases}$$

3. Beschränkte Minimierung mit einer Relation
   - $f(x) =$ ganzzahliger Anteil der Quadratwurzel von x
     $= \mu y \leq x.(y \cdot y \leq x \wedge (y+1) \cdot (y+1) > x)$
   - $x \operatorname{div} y = \begin{cases} \mu t \leq x.(t+1) \cdot y > x & \text{falls } y > 0 \\ 0 & \text{sonst} \end{cases}$
   - $x \bmod y = \begin{cases} x - (x \operatorname{div} y) \cdot y & \text{falls } y > 0 \\ 0 & \text{sonst} \end{cases}$

6.22 Lemma $\mathcal{P}(\mathbb{N})$ ist abgeschlossen bezüglich beschränkter Summation, beschränkter Multiplikation und den beschränkten Minimierungen.

Beweis

1. Sei $g \in \mathcal{P}(\mathbb{N})$; zu zeigen ist $f, h \in \mathcal{P}(\mathbb{N})$, wobei

$$\begin{aligned} f(x,y) &= \sum_{z \leq y} g(x,z) \\ h(x,y) &= \prod_{z \leq y} g(x,z) \end{aligned}$$

   Es gilt

$$\begin{array}{lllllll} f(x,0) & = & g(x,0) & f(x,y+1) & = & f(x,y) + g(x,y+1) \\ h(x,0) & = & g(x,0) & h(x,y+1) & = & h(x,y) \cdot g(x,y+1) \end{array}$$

also

$$f = R(g, h_1) \quad \text{mit} \quad h_1(x, y, z) = y + g(x, z + 1)$$
$$h = R(g, h_2) \quad \text{mit} \quad h_2(x, y, z) = y \cdot g(x, z + 1)$$

Es gilt $g_i, h_i \in \mathcal{P}(\mathbb{N})$, also $f, h \in \mathcal{P}(\mathbb{N})$

2. Sei $g \in \mathcal{P}(\mathbb{N})$ und $f(x, y) = \mu_{z \leq y}[g(x, z) = 0]$. Für den Nachweis, dass $f \in \mathcal{P}(\mathbb{N})$ gilt, muss f in der funktionalen Programmiersprache zu $\mathcal{P}(\mathbb{N})$ programmiert werden. Die Idee hierzu spiegelt die natürliche Berechnung von $f(x, y)$ wider: Wir bilden die Funktion

$$f_0(x, z) = \begin{cases} 1 & g(x, u) > 0 \text{ für alle } u \leq z \\ 0 & \text{sonst} \end{cases}$$

und summieren die Werte $f_0(x, z)$ für $z = 0, 1, \ldots, y$ auf. Setze also

$$f_0(x, z) = sgn\left(\prod_{u \leq z} g(x, u)\right)$$

dann gilt $f_0 \in \mathcal{P}(\mathbb{N})$ nach 1. und eine kurze Überlegung zeigt

$$f(x, y) = \begin{cases} \sum_{z \leq y} f_0(x, z) & f_0(x, y) = 0 \\ 0 & \overline{sgn}(f_0(x, y)) = 0 \end{cases}$$

Also gilt $f \in \mathcal{P}(\mathbb{N})$ nach 1. und Lemma 6.12.

Der Beweis zur beschränkten Minimierung soll noch an folgendem Beispiel verdeutlicht werden. Sei

$$g(x, y) = x - y^2$$

Die folgende Tabelle verdeutlicht die Berechnung von $f(5, y)$

y	$g(5, y)$	$f_0(5, y)$	$f(5, y)$
0	5	1	0
1	4	1	0
2	1	1	0
3	0	0	3
4	0	0	3

3. Ist R primitiv rekursiv und f durch beschränkte Minimierung aus R wie eben definiert, so ist $f \in \mathcal{P}(\mathbb{N})$. Es gilt nämlich
   $f(\vec{x}, 0) = 0$
   $$f(\vec{x}, y+1) = \begin{cases} f(\vec{x}, y) & \text{falls } R\vec{x}f(\vec{x}, y) \\ y + 1 & \text{sonst} \end{cases}$$

Beschränkte Quantifizierung

6.23 Definition

Sei $R \subseteq \mathbb{N}^{n+1}$. Definiere Relationen $T \subseteq \mathbb{N}^{n+1}$ und $S \subseteq \mathbb{N}^{n+1}$ durch **beschränkte All-/Existenz-Quantifizierung** durch

$S\vec{x}b$ gdw es gibt $y \leq b$ mit $R\vec{x}y$ (Schreibe $\exists y \leq b.R\vec{x}y$)

$T\vec{x}b$ gdw für alle $y \leq b$ gilt $R\vec{x}y$ (Schreibe $\forall y \leq b.R\vec{x}y$)

6.24 Lemma

Die primitiv rekursiven Relationen sind abgeschlossen gegenüber beschränkter Quantifizierung.

Beweis: Sei $S\vec{x}b$ gdw $\exists y \leq b.R\vec{x}y$

Dann gilt
$$\begin{aligned} \chi_S(\vec{x}, 0) &= \chi_R(\vec{x}, 0) \\ \chi_S(\vec{x}, b+1) &= \max(\chi_R(\vec{x}, b+1), \chi_S(\vec{x}, b)) \end{aligned}$$

Wegen $T\vec{x}b$ gdw $\neg\exists y \leq b.\neg R\vec{x}y$ folgt die Behauptung.

6.25 Beispiel

Die Teilbarkeitsrelation $|$ ist primitiv rekursiv.

- $x \mid y$ gdw $\exists t \leq y.t \cdot x = y$
  $Rxyz$ gdw $z \cdot x = y$
  Dann ist $x \mid y$ gdw $Sxyy$ gdw $\exists t \leq y.Rxyt$ gdw
  $\exists t \leq y.t \cdot x = y$
- $\{p : p$ ist Primzahl$\}$ ist primitiv rekursiv.

Primitiv rekursive Codier- und Decodierfunktionen

Codierung von Zahlenfolgen, Paarungsfunktionen

6.26 Definition

Die Cauchysche Paarungsfunktion $\langle \cdot, \cdot \rangle : \mathbb{N}^2 \to \mathbb{N}$ wird definiert durch

$$\langle x, y \rangle = (x + y)(x + y + 1) \operatorname{div} 2 + y$$

Sie ist primitiv rekursiv und bijektiv.

(x, y)	$(0,0)$	$(1,0)$	$(0,1)$	$(2,0)$	$(1,1)$	$(0,2) \cdots$
$\langle x, y \rangle$	0	1	2	3	4	$5 \cdots$

Abzählen auf geeigneten Diagonalen
$x + y$ konstant (x, y) kommt auf der Diagonalen $x + y$ vor.

$$1 + 2 + \cdots + (x + y) = (x + y)(x + y + 1) \operatorname{div} 2$$

- Komplett gefüllte Diagonale.
- In der Diagonalen in der (x, y) steht, kommen noch n viele Punkte vor dem Punkt (x, y).

$$\langle x, y \rangle \text{ ist bijektiv}$$

Definiere folgende Umkehrfunktionen first, rest: $\mathbb{N} \to \mathbb{N}$ mit $\langle \text{first}(z), \text{rest}(z) \rangle = z$ und

$\text{first}(\langle x, y \rangle) = x$
$\text{rest}(\langle x, y \rangle) = y$

Codierung von Zahlenfolgen, Paarungsfunktionen (Forts.)

6.27 Lemma

Die Funktionen $\langle \cdot, \cdot \rangle : \mathbb{N}^2 \longrightarrow \mathbb{N}$ und first, rest sind primitiv rekursiv.

Beweis:

Ist $\langle x, y \rangle = z$, dann gilt $x \leq z$ und $y \leq z$.

Dann

$$\text{first}(z) = \mu x \leq z . \exists y \leq z . \langle x, y \rangle = z$$

$$\text{rest}(z) = \mu y \leq z . \exists x \leq z . \langle x, y \rangle = z$$