Theoretische Informatik
Kapitel 9 – Primitiv rekursive und partiell rekursive Funktionen

Sommersemester 2024

Dozentin: Mareike Mutz

im Wechsel mit

Prof. Dr. M. Leuschel

Prof. Dr. J. Rothe

Primitiv rekursive Funktionen

- Historisch: Die Einführung der primitiven Rekursivität war ein erster (und erfolgloser) Versuch, den Begriff der "'Berechenbarkeit"' (oft synonym mit "'Rekursivität"' verwendet) zu definieren.
- Man beginnt mit einfachsten Basisfunktionen und gibt Prinzipien an, wie aus diesen neue, kompliziertere ("'zusammengesetzte"') Funktionen konstruiert werden können.
- Da diese Prinzipien rekursiver Natur und die Basisfunktionen (intuitiv) berechenbar sind, müssen die so konstruierten komplizierteren Funktionen ebenfalls berechenbar sein.
- Die Hoffnung, so irgendwann *alles Berechenbare* zu erfassen, hat sich nicht nur nicht bestätigt, sondern kann widerlegt werden.

Primitiv rekursive Funktionen

Definition

Die Klasse $\mathbb{P}\mathrm{r}$ der *primitiv rekursiven Funktionen* ist induktiv so definiert:

(1) Induktionsbasis:

(1a) Alle konstanten Funktionen (wie etwa die einstelligen konstanten Funktionen $c \in \{0, 1, 2, \ldots\}$, $c : \mathbb{N} \to \mathbb{N}$ mit $c(n) = c$ für alle $n \in \mathbb{N}$) sind primitiv rekursiv.

(1b) Die Nachfolgerfunktion $s : \mathbb{N} \to \mathbb{N}$, definiert durch $s(n) = n + 1$, ist primitiv rekursiv.

(1c) Alle Identitäten (Projektionen) $\mathrm{id}_k^m : \mathbb{N}^m \to \mathbb{N}$, $m \geq 0$, $k \leq m$, definiert durch $\mathrm{id}_k^m(n_1, n_2, \ldots, n_m) = n_k$, sind primitiv rekursiv.

Primitiv rekursive Funktionen: NS Substitution

Definition (Fortsetzung)

(2) Induktionsschritt:

(2a) Normalschema der Substitution (Komposition von Funktionen):

Sind die Funktionen $f : \mathbb{N}^k \to \mathbb{N}$ und $g_1, g_2, \dots, g_k : \mathbb{N}^m \to \mathbb{N}$ für $k, m \in \mathbb{N}$ in $\mathbb{P}\mathrm{r}$, so ist auch die Funktion $h : \mathbb{N}^m \to \mathbb{N}$ in $\mathbb{P}\mathrm{r}$, die definiert ist durch

$$
\begin{aligned}
h(n_1, n_2, \dots, n_m) = \\
& f(g_1(n_1, n_2, \dots, n_m), g_2(n_1, n_2, \dots, n_m), \dots, g_k(n_1, n_2, \dots, n_m)).
\end{aligned}
$$

Primitiv rekursive Funktionen: NS Primitive Rekursion

Definition (Fortsetzung)

(2b) Normalschema der primitiven Rekursion:

Sind die Funktionen g und h in $\mathbb{P}\mathrm{r}$, wobei $g : \mathbb{N}^m \rightarrow \mathbb{N}$ und $h : \mathbb{N}^{m+2} \rightarrow \mathbb{N}$ für $m \in \mathbb{N}$, so ist auch die Funktion $f : \mathbb{N}^{m+1} \rightarrow \mathbb{N}$ in $\mathbb{P}\mathrm{r}$, die rekursiv definiert ist durch

$$\begin{array}{rcl} f(0, x_1, x_2, \ldots, x_m) & = & g(x_1, x_2, \ldots, x_m) \\ f(n+1, x_1, x_2, \ldots, x_m) & = & h(n, f(n, x_1, x_2, \ldots, x_m), x_1, x_2, \ldots, x_m). \end{array}$$

Dabei ist
- n die Rekursionsvariable, und
- die $x_1, x_2, \ldots, x_m$ heißen Parameter.

„Vollständige Induktion nach dem ersten Parameter von f.“

Primitiv rekursive Funktionen: Addition

Beispiel: Die Additionsfunktion $\mathrm{add} : \mathbb{N}^2 \to \mathbb{N}$, die durch $\mathrm{add}(x, y) = x + y$ definiert ist, ist primitiv rekursiv. Im Normalschema der primitiven Rekursion lässt sie sich so beschreiben:

$$
\begin{array}{lclcl}
\mathrm{add}(0, x) & = & x & = & \mathrm{id}_1^1(x) \\
\mathrm{add}(n+1, x) & = & s(\mathrm{add}(n, x)) & = & s_2^3(n, \mathrm{add}(n, x), x) \\
\text{mit} & & s_2^3(x_1, x_2, x_3) & = & s(\mathrm{id}_2^3(x_1, x_2, x_3))
\end{array}
$$

Die identische Funktion id_1^1 entspricht dabei g aus (2b) in der Definition und die Funktion $s_2^3 = s \circ \mathrm{id}_2^3$ entspricht dabei h

Der Rekursionsanfang beruht auf der Tatsache, dass $x + 0 = x$ für alle $x \in \mathbb{N}$ gilt, und der Rekursionsschritt bedeutet $x + (n + 1) = (x + n) + 1$. Also ist $\mathrm{add} \in \mathbb{P}\mathrm{r}$.

Primitiv rekursive Funktionen: Addition - Auswertung

Beispiel: $\mathrm{add}(0, x) = \mathrm{id}_1^1(x)$, $\mathrm{add}(n + 1, x) = s_2^3(n, \mathrm{add}(n, x), x)$, $s_2^3(x_1, x_2, x_3) = s(\mathrm{id}_2^3(x_1, x_2, x_3))$.

- $\mathrm{add}(2, 3) =$
- $s_2^3(1, \mathrm{add}(1, 3), 3) =$
- $s_2^3(1, s_2^3(0, \mathrm{add}(0, 3), 3), 3) =$
- $s_2^3(1, s_2^3(0, \mathrm{id}_1^1(3), 3), 3) =$
- $s_2^3(1, s_2^3(0, 3, 3), 3) =$
- $s_2^3(1, s(\mathrm{id}_2^3(0, 3, 3)), 3) =$
- $s_2^3(1, s(3), 3) =$
- $s_2^3(1, 4, 3) =$
- $s(\mathrm{id}_2^3(1, 4, 3)) =$
- $s(4) = 5$

Primitiv rekursive Funktionen: Haskell

Man kann diese Definition auch so in einer funktionalen Programmiersprache wie Haskell eingeben und ausführen.

```haskell
{-# LANGUAGE NPlusKPatterns #-}
add(0,y) = y
add(x+1,y) = suc(add(x,y))
suc(x) = 1+x
```

```
$ ghci PrimRec.hs
GHCi, version 8.10.1: https://www.haskell.org/ghc/  :? for help
[1 of 1] Compiling Main             ( PrimRec.hs, interpreted )
Ok, one module loaded.
*Main> add(2,3)
5
*Main> add(100,200)
300
```

Primitiv rekursive Funktionen: Haskell

Diese Version entspricht komplett unserem Schema:

```
{-# LANGUAGE NPlusKPatterns #-}
id_1_1(x) = x
id_3_2(x,y,z) = y // Projektion auf 2. Element von einem Tripel
s_id_3_2(n,y,z) = s(id_3_2(n,y,z)) -- Komposition
s(x) = x+1

add_pr(0,x) = id_1_1(x)
add_pr(n+1,x) = s_id_3_2(n,add_pr(n,x),x)
```

```
*Main> add_pr(110,200)
310
```

Primitiv rekursive Funktionen: Multiplikation

Beispiel: Die Multiplikationsfunktion $\mathrm{mult} : \mathbb{N}^2 \to \mathbb{N}$, die definiert ist durch $\mathrm{mult}(x, y) = x \cdot y$, ist primitiv rekursiv. Im Normalschema der primitiven Rekursion lässt sie sich so beschreiben:

$$
\begin{aligned}
\mathrm{mult}(0, x) &= 0 \\
\mathrm{mult}(n + 1, x) &= \mathrm{add}(\mathrm{mult}(n, x), x) \\
&= f'(n, \mathrm{mult}(n, x), x),
\end{aligned}
$$

wobei $f'(a, b, c) = add(\mathrm{id}_2^3(a, b, c), \mathrm{id}_3^3(a, b, c))$.

Die konstante Funktion 0 (mit einem Argument x) entspricht dabei der Funktion g aus (2b) in der Definition und die Funktion f' entspricht dabei der Funktion h.

Primitiv rekursive Funktionen: Multiplikation

- Die Funktion add ist primitiv rekursiv nach dem ersten Beispiel.
- Nach dem Normalschema der Substitution (2a) ist die Funktion f' somit primitiv rekursiv.
- Der Rekursionsanfang beruht auf der Tatsache, dass

  $$x \cdot 0 = 0$$

  für alle $x \in \mathbb{N}$ gilt, und der Rekursionsschritt bedeutet

  $$x \cdot (n+1) = x \cdot n + x.$$

- Also ist mult eine primitiv rekursive Funktion.

Primitiv rekursive Funktionen: Haskell

In Haskell kann man dies so schreiben. Um genau dem Schema zu entsprechen führen wir noch die Funktion zero ein:

```
zero(x) = 0
mult(0,x) = zero(x)
mult(n+1,x) = f'(n,mult(n,x),x)
f'(n,m,x) = add(id_3_2(n,m,x),id_3_3(n,m,x))
```

```
*Main> mult(20,30)
600
```

Primitiv rekursive Funktionen: Exponentialfunktion

Beispiel: Die Exponentialfunktion $\exp : \mathbb{N}^2 \to \mathbb{N}$ ist durch

$$\exp(y, x) = x^y$$

definiert, wobei wir in Abweichung vom mathematischen Standard

$$x^0 = 1$$

auch für $x = 0$ setzen, um die Totalität der Funktion zu erreichen.

Diese Funktion ist primitiv rekursiv. Im Normalschema der primitiven Rekursion lässt sie sich so beschreiben:

$$\begin{aligned} \exp(0, x) &= 1 \\ \exp(n + 1, x) &= \mathrm{mult}(\exp(n, x), x) \\ &= f'(n, \exp(n, x), x), \end{aligned}$$

wobei $f'(a, b, c) = \mathrm{mult}(\mathrm{id}_2^3(a, b, c), \mathrm{id}_3^3(a, b, c))$.

Primitiv rekursive Funktionen: Exponentialfunktion

- Die konstante Funktion 1 entspricht dabei der Funktion g aus (2b) in der Definition und die Funktion f' entspricht dabei der Funktion h.
- Die Funktion mult ist primitiv rekursiv nach dem zweiten Beispiel.
- Nach dem Normalschema der Substitution (2a) ist die Funktion f' somit primitiv rekursiv.
- Der Rekursionsanfang beruht auf der Tatsache, dass $x^0 = 1$ für alle $x \in \mathbb{N}$ gilt, und der Rekursionsschritt bedeutet

$$x^{n+1} = x^n \cdot x.$$

- Also ist $\exp$ eine primitiv rekursive Funktion.

Primitiv rekursive Funktionen: Haskell

In Haskell kann man dies so schreiben. Um genau dem Schema zu entsprechen führen wir noch die Funktion zero ein:

```
exp_pr(0,x) = 1
exp_pr(n+1,x) = exp_aux(n,exp_pr(n,x),x)

exp_aux(n,e,x) = mult(id_3_2(n,e,x),id_3_3(n,e,x))
```

```
*Main> exp_pr(10,2)
1024
```

Man kann auch einfach Argumente vertauschen:

```
my_exp(basis,e) = exp_pr(id_2_2(basis,e),id_2_1(basis,e))
```

Primitiv rekursive Funktionen: Fakultätsfunktion

Beispiel: Die Fakultätsfunktion $\mathrm{fa} : \mathbb{N} \rightarrow \mathbb{N}$ ist definiert durch

$$\mathrm{fa}(x) = \begin{cases} 1 & \text{falls } x = 0 \\ 1 \cdot 2 \cdot \ldots \cdot x & \text{falls } x \geq 1. \end{cases}$$

Diese Funktion ist primitiv rekursiv. Im Normalschema der primitiven Rekursion lässt sie sich so beschreiben:

$$\begin{aligned} \mathrm{fa}(0) &= 1 \\ \mathrm{fa}(n+1) &= \mathrm{mult}(\mathrm{fa}(n), s(n)) \\ &= f'(n, \mathrm{fa}(n)), \end{aligned}$$

wobei $f'(a, b) = \mathrm{mult}(\mathrm{id}_2^2(a, b), s(\mathrm{id}_1^2(a, b)))$.

Primitiv rekursive Funktionen: Fakultätsfunktion

- Die konstante Funktion 1 entspricht dabei der Funktion g aus (2b) in der Definition und die Funktion f' entspricht dabei der Funktion h.
- Die Funktion mult ist primitiv rekursiv nach dem zweiten Beispiel.
- Nach dem Normalschema der Substitution (2a) ist die Funktion f' somit primitiv rekursiv.
- Der Rekursionsanfang beruht auf der Tatsache, dass $\mathrm{fa}(0) = 1$ gilt, und der Rekursionsschritt bedeutet

$$(x + 1)! = x! \cdot (x + 1).$$

- Also ist fa eine primitiv rekursive Funktion.

Primitiv rekursive Funktionen: Weitere Beispiele

Die folgenden Funktionen sind ebenfalls primitiv rekursiv:

- Die Vorgängerfunktion $V : \mathbb{N} \to \mathbb{N}$ ist definiert durch

$$V(x) = \begin{cases} 0 & \text{falls } x = 0 \\ x - 1 & \text{falls } x \geq 1. \end{cases}$$

- Die modifizierte Differenz $\mathrm{md} : \mathbb{N}^2 \to \mathbb{N}$ ist definiert durch

$$\mathrm{md}(x, y) = x \dot{-} y = \begin{cases} 0 & \text{falls } x < y \\ x - y & \text{falls } x \geq y. \end{cases}$$

Primitiv rekursive Funktionen: Weitere Beispiele

Die folgenden Funktionen sind ebenfalls primitiv rekursiv:

- Die Abstandsfunktion $A : \mathbb{N}^2 \rightarrow \mathbb{N}$ ist definiert durch

$$A(x, y) = |x - y| = \begin{cases} x - y & \text{falls } y \leq x \\ y - x & \text{falls } x < y. \end{cases}$$

- Die Signumfunktion $S : \mathbb{N} \rightarrow \mathbb{N}$ ist definiert durch

$$S(x) = \begin{cases} 0 & \text{falls } x = 0 \\ 1 & \text{falls } x \geq 1. \end{cases}$$

Dedekindscher Rechtfertigungssatz

Theorem

Es seien für $m \geq 0$ die Funktionen $g : \mathbb{N}^m \to \mathbb{N}$ und $h : \mathbb{N}^{m+2} \to \mathbb{N}$ gegeben. Dann existiert genau eine *Funktion*

$$f : \mathbb{N}^{m+1} \to \mathbb{N},$$

die Lösung des Normalschemas der primitiven Rekursion (2b) in der obigen Definition ist. **ohne Beweis**

LOOP-Berechenbarkeit versus primitive Rekursion

Da jede primitiv rekursive Funktion total ist, es aber natürlich nicht-totale Funktionen gibt, die (im intuitiven Sinne) berechenbar sind (z.B. $f : \mathbb{N}^2 \to \mathbb{N}, f(n_1, n_2) = n_1 \text{ div } n_2$), umfasst die Klasse $\mathbb{P}\mathrm{r}$ trivialerweise nicht alles intuitiv Berechenbare.

Theorem

Die Klasse der primitiv rekursiven Funktionen stimmt mit der Klasse der LOOP-berechenbaren Funktionen überein. **ohne Beweis**

Gibt es totale berechenbare, nicht primitive rekursive Funktionen?

- Interessanter ist die Frage, ob es *totale* berechenbare Funktionen gibt, die aber außerhalb von $\mathbb{P}\mathrm{r}$ liegen.
- Auch hier ist die Antwort positiv. Um dies zu beweisen, brauchen wir als technische Vorbereitung die so genannte *Gödelisierung* von $\mathbb{P}\mathrm{r}$.
- Eine solche Gödelisierung, auch *Gödelsches Wörterbuch* genannt, lässt sich analog für jede Klasse von Maschinen (TMs, PDAs, NFAs usw.) angeben, die eine syntaktische Beschreibung bzw. Formalisierung bestimmter Klassen von Algorithmen liefern.

Gödelisierung von $\mathbb{P}\mathrm{r}$

Wir suchen eine Funktion $G : \mathbb{P}\mathrm{r} \mapsto N$, wobei $N = \Sigma^*$ oder N die Menge der natürlichen Zahlen ist und

- G injektiv und berechenbar,
- die Bildmenge $G[\mathbb{P}\mathrm{r}]$ „algorithmisch entscheidbar" (formale Definition kommt später) und
- die Umkehrfunktion von G berechenbar ist.

Bemerkung: $\mathbb{P}\mathrm{r}$ steht hier für die Menge aller primitiv rekursiven Funktionsdefinitionen.

Gödelisierung von $\mathbb{P}\mathrm{r}$

1. Über dem Alphabet

$$\Sigma = \{x, |, (,), [,], ,, ;, *, s, 0, \mathrm{id}, \mathrm{SUB}, \mathrm{PR}\}$$

lassen sich die Funktionen in $\mathbb{P}\mathrm{r}$ wie folgt darstellen:

- **Variablen:** $x_1, x_2, \ldots, x_i, \ldots$ werden repräsentiert durch $x|, x||, \ldots, x\underbrace{||\cdots|}_{i\text{-mal}}, \ldots$
- **Trennsymbole:** ( ) [ ] , ; $*$
- **Basisfunktionen:** s und 0. Mit diesen beiden Funktionen lässt sich jede konstante Funktion $c \in \{0, 1, 2, \ldots\}$ darstellen. Zum Beispiel ist $G(2) = s(s(0))$. Wir haben auch $G(s) = s$.
- **Identitäten:** id_k^m wird dargestellt als $\mathrm{id}\underbrace{||\cdots|}_{m\text{-mal}} * \underbrace{||\cdots|}_{k\text{-mal}}$.

Gödelisierung von $\mathbb{P}\mathrm{r}$

- **Substitutionen:** Werden die Funktionen $g_1, g_2, \ldots, g_k : \mathbb{N}^m \to \mathbb{N}$ in die Funktion $f : \mathbb{N}^k \to \mathbb{N}$ substituiert, so wird die resultierende Funktion $h : \mathbb{N}^m \to \mathbb{N}$ dargestellt als

$$G(h) = \mathrm{SUB}[G(f); G(g_1), G(g_2), \ldots, G(g_k)](G(x_1), G(x_2), \ldots, G(x_m)).$$

- **Primitive Rekursion:** Ergibt sich die Funktion $f : \mathbb{N}^{m+1} \to \mathbb{N}$ aus $g : \mathbb{N}^m \to \mathbb{N}$ und $h : \mathbb{N}^{m+2} \to \mathbb{N}$ durch primitive Rekursion, so stellen wir f dar als

$$G(f) = \mathrm{PR}[G(g), G(h)](G(x_1), G(x_2), \ldots, G(x_{m+1})).$$

Gödelisierung von $\mathbb{P}\mathrm{r}$

Jedes $f \in \mathbb{P}\mathrm{r}$ lässt sich als ein Wort über dem Alphabet Σ darstellen. Beispielsweise kann man die Additionsfunktion $\mathrm{add} : \mathbb{N}^2 \rightarrow \mathbb{N}$ so als ein Wort über Σ darstellen.

Beispiel: Betrachte

$$\begin{aligned} \mathrm{add}(0, x) &= \mathrm{id}_1^1(x) \\ \mathrm{add}(n+1, x) &= h(n, \mathrm{add}(n, x), x) \end{aligned}$$

mit $h(n, y, z) = s(\mathrm{id}_2^3(n, y, z))$. Dann erhalten wir:

$$G(\mathrm{add}) = \mathrm{PR}[\mathrm{id}| * |, \mathrm{SUB}[s; \mathrm{id}||| * ||](x|, x||, x|||)](x|, x||).$$

Umgekehrt ist nicht jedes Wort $w \in \Sigma^*$ eine syntaktisch korrekte Funktion in $\mathbb{P}\mathrm{r}$.

Gödelisierung von $\mathbb{P}\mathrm{r}$

2. Legen wir eine lineare Ordnung der Symbole in Σ fest, ergibt sich eine quasilexikographische Ordnung aller Wörter in Σ^*.

   Entfernen wir alle syntaktisch inkorrekten Wörter, so erhalten wir die Gödelisierung von $\mathbb{P}\mathrm{r}$:

$$\psi_0, \psi_1, \psi_2, \ldots$$

Bemerkung: Jedes $f \in \mathbb{P}\mathrm{r}$ hat unendlich viele Gödelnummern, d.h., es gibt unendlich viele verschiedene $i \in \mathbb{N}$, so dass $f = \psi_i$. So gilt etwa:

$$f = f + 0 = f + 0 + 0 = \cdots ,$$

und alle diese Funktionen sind syntaktisch verschieden und haben daher verschiedene Gödelnummern.

Wichtige Eigenschaften der Gödelisierung von $\mathbb{P}\mathrm{r}$

1. Es gibt ein algorithmisches Verfahren, das zu gegebener Gödelnummer i die Funktion ψ_i (besser: das Wort, das diese Funktion beschreibt) bestimmt.
2. Es gibt ein algorithmisches Verfahren, das zu gegebenem Wort $w \in \Sigma^*$ feststellt, ob w eine syntaktisch korrekte Funktion f aus $\mathbb{P}\mathrm{r}$ beschreibt, und wenn ja, die Nr. i bestimmt, so dass $\psi_i = f$.

(Man braucht nämlich nur entsprechend obiger Vorschrift das Gödelsche Wörterbuch bis zur gegebenen Nr. i bzw. bis zum gegebenen Wort w aufzubauen.)

$u(i, j) = \psi_i(j)$ wird auch universelle Funktion genannt.

Wichtige Eigenschaften der Gödelisierung von $\mathbb{P}\mathrm{r}$

Beispielhafte Aufzählung mit einem Prolog Programm:

Nr	Prolog AST	Wort	ψ
1	cst(0)	0	$\psi_1(x) = 0$
2	s	s	$\psi_2(x) = x + 1$
3	cst(1)	s(0)	$\psi_3(x) = 1$
4	cst(0)	0	
5	id(1)	$\mid * \mid$	$\psi_5(x) = x$
7	cst(2)	s(s(0))	$\psi_7(x) = 2$
13	sub(cst(1),[cst(1)])	$SUB[s(0); s(0)](x\mid)$	$\psi_{13}(x) = 1$
26	sub(s,[cst(1)])	$SUB[s; s(0)](x\mid)$	$\psi_{26}(x) = 2$
31	pr(cst(1),cst(0))	$PR[s(0), 0](x\mid)$	$\psi_{31}(x) = 1$ für x=0, 0 sonst

Totale berechenbare Funktionen außerhalb $\mathbb{P}\mathrm{r}$

Theorem

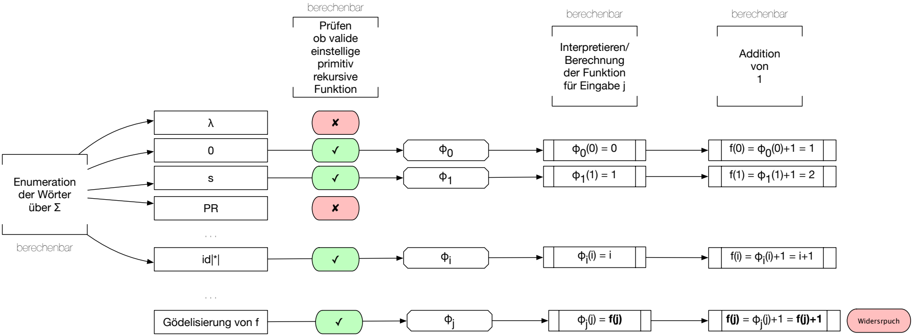
Es gibt eine totale, (intuitiv) berechenbare Funktion f mit $f \notin \mathbb{P}\mathrm{r}$.

Beweis:

- Sei $\psi_0, \psi_1, \psi_2, \ldots$ eine Gödelisierung aller einstelligen Funktionen in $\mathbb{P}\mathrm{r}$. Setze
$$f(i) = \psi_i(i) + 1.$$
- Aus der Annahme, dass $f \in \mathbb{P}\mathrm{r}$ ist, folgt, dass $f = \psi_j$ für ein festes j gilt. Daraus ergibt sich der Widerspruch:
$$\psi_j(j) = f(j) = \psi_j(j) + 1.$$

❑

Totale berechenbare Funktionen außerhalb $\mathbb{P}r$



Die Ackermann-Funktion

Beispiel: Die *Ackermann-Funktion* $\alpha : \mathbb{N}^2 \to \mathbb{N}$ ist definiert durch

$$\alpha(m,n) = \begin{cases} n+1 & \text{falls } m = 0 \text{ und } n \geq 0 \quad (1) \\ \alpha(m-1,1) & \text{falls } m > 0 \text{ und } n = 0 \quad (2) \\ \alpha(m-1,\alpha(m,n-1)) & \text{falls } m > 0 \text{ und } n > 0. \quad (3) \end{cases}$$

Die Ackermann-Funktion ist eine

- totale und
- berechenbare Funktion,
- die nicht primitiv rekursiv ist.

Sie war historisch die erste bekannte solche Funktion.

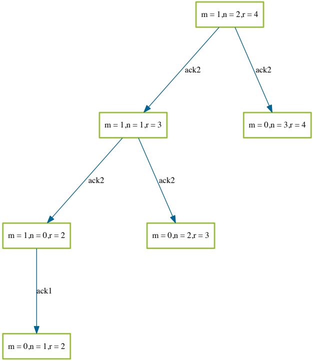
Die Ackermann-Funktion

Der Beweis, dass α nicht primitiv rekursiv ist, beruht darauf, dass man zeigen kann, dass α schneller wächst als jede primitiv rekursive Funktion. Beispielsweise ist

$$
\begin{aligned}
\alpha(1,2) &= \alpha(0,\alpha(1,1)) && (3)\\
&= \alpha(1,1)+1 && (1)\\
&= \alpha(0,\alpha(1,0))+1 && (3)\\
&= \alpha(1,0)+1+1 && (1)\\
&= \alpha(0,1)+2 && (2)\\
&= 1+1+2 && (1)\\
&= 4.
\end{aligned}
$$

Die Ackermann-Funktion $\alpha(1, 2) = 4$

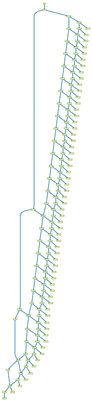
$$\alpha(m,n) = \begin{cases} n+1 & \text{falls } m = 0 \text{ und } n \geq 0 \quad (1) \\ \alpha(m-1, 1) & \text{falls } m > 0 \text{ und } n = 0 \quad (2) \\ \alpha(m-1, \alpha(m, n-1)) & \text{falls } m > 0 \text{ und } n > 0. \quad (3) \end{cases}$$



Die Ackermann-Funktion

Aufrufgraph (mit Memoisierung):

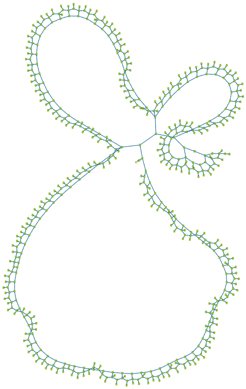
$$\alpha(3,3) \quad = \quad 61$$



Die Ackermann-Funktion

Aufrufgraph (mit Memoisierung):

$$\alpha(3,5) \quad = \quad 253$$



Die Ackermann-Funktion

Aber schon

$$\begin{aligned}\alpha(3,3) &= 61 \quad \text{und} \\ \alpha(4,4) &= 2^{2^{2^{2^{16}}}} - 3.\end{aligned}$$

Für alle $n \in \mathbb{N}$ gilt:

1. $\alpha(0, n) = n + 1$
2. $\alpha(1, n) = n + 2$
3. $\alpha(2, n) = 2 \cdot n + 3$
4. $\alpha(3, n) = 2^{n+3} - 3$

Die Ackermann-Funktion

Übung: Füllen Sie die folgende Tabelle mit den Werten der Ackermannfunktion $\alpha(m, n)$, $0 \leq m \leq 3$, $0 \leq n \leq 4$:

m \ n	0	1	2	3	4
0					
1					
2					
3					

Der μ-Operator

Frage: Was fehlt den primitiv rekursiven Funktionen, um das intuitiv Berechenbare vollständig zu umfassen?

Antwort: Der μ-Operator (Minimalisierung).

Der μ-Operator

Definition (μ-Operator)

Sei $k \geq 0$, und sei $f : \mathbb{N}^{k+1} \rightarrow \mathbb{N}$ eine Funktion. Durch Anwendung des *μ-Operators* auf f entsteht die Funktion $g : \mathbb{N}^k \rightarrow \mathbb{N}$, die definiert ist als

$$g(x_1, x_2, \ldots, x_k) = \min \left\{ n \in \mathbb{N} \;\middle|\; \begin{array}{l} f(n, x_1, x_2, \ldots, x_k) = 0 \text{ und für alle} \\ m < n \text{ ist } f(m, x_1, x_2, \ldots, x_k) \text{ definiert} \end{array} \right\}.$$

Hier ist $\min \emptyset$ nicht definiert, d.h., ist die Menge, über die minimiert wird, leer, so ist g an dieser Stelle nicht definiert.

Schreibweise: $\mu f = g$ oder ausführlicher

$$\mu n[f(n, \ldots)] = g(\cdots).$$

Partiell und allgemein rekursive Funktionen

Definition (Partiell und allgemein rekursive Funktionen)

- Die Klasse $\mathbb{P}$ der *partiell rekursiven Funktionen* (der μ-rekursiven Funkionen) ist definiert als die kleinste Klasse von Funktionen, die
  - die Basisfunktionen aus der Definition von $\mathbb{P}\mathrm{r}$ (die Konstanten, die Nachfolgerfunktion und die Identitäten) enthält und
  - abgeschlossen ist unter Substitution, primitiver Rekursion und dem μ-Operator.
- Die Klasse $\mathbb{R}$ der *allgemein rekursiven Funktionen* ist definiert als

$$\mathbb{R} = \{f \mid f \in \mathbb{P} \text{ und } f \text{ ist total}\}.$$

Partiell und allgemein rekursive Funktionen

Beispiel: Definiere die Funktion f wie folgt:

$$\begin{aligned} f(0,0) &= 1 \\ f(1,0) &= 1 \\ f(2,0) &= \text{nicht definiert} \\ f(3,0) &= 0 \\ f(m,0) &= 1 \quad \text{für alle } m > 3. \end{aligned}$$

Ist $g = \mu f$, so ist $g(0)$ nicht etwa gleich 3, sondern $g(0)$ ist nicht definiert, weil $f(2,0)$ nicht definiert ist und die entsprechende Menge

$$\left\{ n \in \mathbb{N} \;\middle|\; \begin{array}{l} f(n, x_1, x_2, \ldots, x_k) = 0 \text{ und für alle} \\ m < n \text{ ist } f(m, x_1, x_2, \ldots, x_k) \text{ definiert} \end{array} \right\}.$$

somit leer ist.

Partiell und allgemein rekursive Funktionen

Beispiel: Ist

$$f(x, y) = x + y + 1,$$

so gilt für alle $x, y \in \mathbb{N}$: $f(x, y) \neq 0$. Also ist $g(y)$ für kein $y \in \mathbb{N}$ definiert, also ist $g = \mu f$ die *nirgends definierte Funktion* Ω.

Somit können durch Anwendung des μ-Operators partielle Funktionen entstehen.

Ist dagegen

$$f(x, y) = x + y,$$

so ist $g = \mu f$ definiert durch

$$g(y) = \begin{cases} 0 & \text{falls } y = 0 \\ \textit{undefiniert} & \text{falls } y > 0. \end{cases}$$

Ganzzahlige Division ist partiell rekursiv

Beispiel: Die ganzzahlige Divisionsfunktion $g : \mathbb{N}^2 \to \mathbb{N}$ mit

$$g(y, z) = \begin{cases} \frac{z}{y} & \text{falls } y \neq 0 \text{ und } y \text{ teilt } z \\ \textit{undefiniert} & \text{sonst} \end{cases}$$

ist partiell rekursiv. Dies kann man zeigen, indem man eine geeignete Funktion $f : \mathbb{N}^3 \to \mathbb{N}$ angibt, so dass $\mu f = g$ gilt.

Wie wir wissen, sind die folgenden Funktionen primitiv rekursiv:

- die Multiplikationsfunktion mult,
- die Signumfunktion S,
- die Exponentialfunktion $\exp$ und
- die Abstandsfunktion A.

Ganzzahlige Division ist partiell rekursiv

Somit ist die Funktion $f : \mathbb{N}^3 \to \mathbb{N}$ mit

$$f(x, y, z) = |x \cdot y - z|^{S(y)} = \begin{cases} |x \cdot y - z| & \text{falls } y \neq 0 \\ 1 & \text{falls } y = 0 \end{cases}$$

ebenfalls primitiv rekursiv.

Wann nimmt f den Wert 0 an?
Dies ist genau dann der Fall, wenn

$$y \neq 0 \quad \text{und} \quad x \cdot y = z$$

gilt.

Ganzzahlige Division ist partiell rekursiv

Wir wenden nun den μ-Operator auf f an und erhalten:

$$
\begin{aligned}
\mu x[f(x,y,z)] &= \begin{cases} \text{das kleinste } x \in \mathbb{N} \text{ mit } f(x,y,z) = 0 & \text{falls es existiert} \\ \textit{undefiniert} & \text{sonst} \end{cases} \\
&= \begin{cases} \text{das kleinste } x \in \mathbb{N} \text{ mit } x \cdot y = z \; (y \neq 0) & \text{falls es existiert} \\ \textit{undefiniert} & \text{sonst} \end{cases} \\
&= \begin{cases} \frac{z}{y} & \text{falls } y \neq 0 \text{ und } y \text{ teilt } z \\ \textit{undefiniert} & \text{sonst} \end{cases} \\
&= g(y,z)
\end{aligned}
$$

Ganzzahlige Wurzelfunktion ist partiell rekursiv

Beispiel: Die Funktion $g : \mathbb{N} \to \mathbb{N}$ mit

$$g(m) = \begin{cases} \sqrt{m} & \text{falls } \sqrt{m} \in \mathbb{N} \\ \textit{undefiniert} & \text{sonst} \end{cases}$$

ist partiell rekursiv mittels $f : \mathbb{N}^2 \to \mathbb{N}$ mit

$$f(n, m) = |n^2 - m|,$$

denn es gilt $\mu n[f(n, m)] = g(m)$ (s. Übungen).

$\mathbb{P}\mathrm{r}$ versus $\mathbb{R}$ versus $\mathbb{P}$

Theorem

$\mathbb{P}\mathrm{r} \subset \mathbb{R} \subset \mathbb{P}$.

Beweis: Der Beweis von diesem Theorem 9 ist trivial bis auf $\mathbb{P}\mathrm{r} \neq \mathbb{R}$, was unmittelbar aus dem letzten Satz folgt ("es gibt eine totale, (intuitiv) berechenbare Funktion f mit $f \notin \mathbb{P}\mathrm{r}$"). ❑

Bemerkung:

1. Wir werden jetzt zeigen, weshalb der Widerspruchsbeweis von

$$\mathbb{P}\mathrm{r} \neq \mathbb{R}$$

für $\mathbb{P}$ scheitert (d.h. man kann nicht zeigen, dass es berechenbare Funktionen außerhalb von $\mathbb{P}$ gibt).

$\mathbb{Pr}$ versus $\mathbb{R}$ versus $\mathbb{P}$

- Sei φ_i die i-te einstellige Funktion in einer Gödelisierung von $\mathbb{P}$.
- Definiere die Funktion $f : \mathbb{N} \rightarrow \mathbb{N}$ durch $f(i) = \varphi_i(i) + 1$.
- Angenommen, f ist in $\mathbb{P}$.
- Dann existiert ein j, so dass $f = \varphi_j$. Bezeichnet D_h den Definitionsbereich einer Funktion h, so gilt also:
  - $\mathrm{D}_f = \mathrm{D}_{\varphi_j}$ und
  - $f(n) = \varphi_j(n)$ für alle $n \in \mathrm{D}_f$.

  Nun ist aber die Aussage

  $$\varphi_j(j) = f(j) = \varphi_j(j) + 1 \tag{1}$$

  kein Widerspruch mehr, denn an der Stelle j können f und φ_j undefiniert sein; dann ist natürlich auch $\varphi_j(j) + 1$ nicht definiert.

$\mathbb{P}\mathrm{r}$ versus $\mathbb{R}$ versus $\mathbb{P}$

- Die Gleichung (1) sagt also:

$$\text{„nicht definiert} = \text{nicht definiert“}.$$

2. Man überlege sich, weshalb der Widerspruchsbeweis von

$$\mathbb{P}\mathrm{r} \neq \mathbb{R}$$

auch für $\mathbb{R}$ scheitert.

Kann man $\mathbb{R}$ gödelisieren?

Der Hauptsatz der Berechenbarkeitstheorie

Theorem

Die folgenden Algorithmenbegriffe sind äquivalent in dem Sinne, dass die Klasse der von ihnen berechneten Funktionen mit der Klasse $\mathbb{P}$ der partiell rekursiven Funktionen übereinstimmt:

- *Turingmaschinen,*
- *WHILE-Programme,*
- *GOTO-Programme,*
- $\cdots$.

Der Hauptsatz der Berechenbarkeitstheorie

Bemerkung:

- Wir nennen nun Turing-, WHILE-, GOTO-, …-berechenbare und partiell rekursive Funktionen im Folgenden kurz *berechenbare* Funktionen.

GOTO-berechenbar =

WHILE-berechenbar =

Turing-berechenbar =

partiell rekursiv

LOOP-berechenbar =

primitiv rekursiv

Abbildung: Turing-, WHILE-, GOTO-, LOOP-Berechenbarkeit und primitiv rekursive bzw. partiell rekursive Funktionen