Tutorium Theoretische Informatik II 19. Mai 2010

Note Title 5/3/2010

- Ackermann Funktion
  - warum ist A nicht p.r. obwohl jedes einzelne A_x p.r.
- Begriffsbildung Y Kombinator etc.
  + Beispiele

$$A(0, y) = y + 1 \qquad f_0(y) = y + 1$$

$$A(x+1, 0) = A(x, 1) \qquad f_{x+1}(0) = f_x(1)$$

$$A(x+1, y+1) = A(x, A(x+1, y)) \qquad f_{x+1}(y+1) = f_x(f_{x+1}(y))$$

$f_x(y) := A(x, y)$ erster Parameter wird eingefroren

jedes f_x ist p.r. aber A ist nicht p.r.

Induktionsbeweis

Anfang f_0 p.r. denn $f_0(y) = y+1 = S(y)$ also $f_0 = S$

Ann sei f_x als p.r. bewiesen

dann ist $f_{x+1}(0) = f_x(1) = f_x \circ c_1^0()$

$f_{x+1}(y+1) = f_x(f_{x+1}(y))$

$= f_x \circ pr_2^2 \qquad (y, f_{x+1}(y))$

also

$$f_{x+1} = Pr\left[f_x \circ c_1^0,\ f_x \circ pr_2^2\right]$$

wegen Annahme ist f_{x+1} p.r.

Warum ist A nicht p.r.???

Argument: Index der Funktion f_x und Eingabe y sind eingabe für A

Um f_x zu haben benötigt man x primitive Rekursion

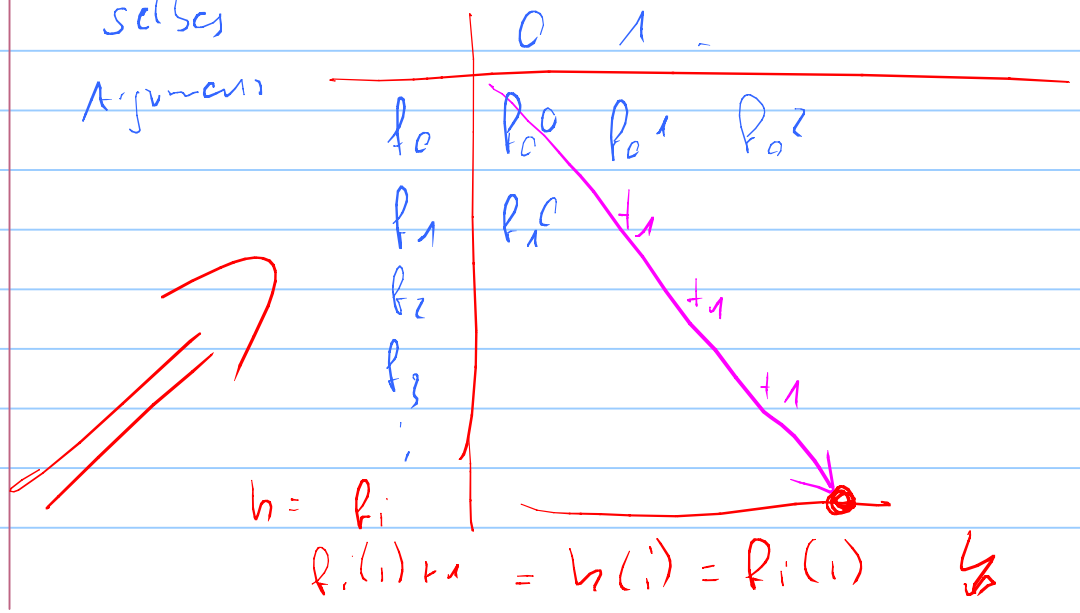
um $A(x,y)$ zu berechnen, benötigt man $\geq$ x p. Rekursionen

um A zu programmieren, benötigt man unbegrenzte Anzahl von p. Rekursion

Programm kann nicht erzeugt werden !

Methode A geht 'diagonal' durch Anzahl der Rekursionen

selbst Argument



Anzahl der Funktionen auf $\mathbb{N}$ ist nicht abzählbar

Definiere $h(x) = f_x(x) + 1$

dann ist h Funktion auf $\mathbb{N}$ aber eines der f_i

λ-Theorie: $t = s$? gleiche Bedeutung

Falls 1) t identisch mit s $\quad \lambda x.x = \lambda x.x$

2) t ist identisch mit s bis auf Umbenennung gebundener Variablen

$\lambda x.x = \lambda y.y$

$4+3 = 2+5$

↓ ↓

7

3) $t \to u$ und $s \to v$ und u, v identisch bis auf Umbenennung

$\lambda y.(\lambda x.\ x+3)(4) = \lambda z.\ 7$

↓

$\lambda y.\ 4+3$

↓

$\lambda y.\ 7$

//

Fixpunktkombinator R ("~~Rekursor~~")

hat Eigenschaft $Rt = t(Rt) = t(t(Rt)) = t(t(t(Rt)))\ldots$

Beispiel $Y = (\lambda f.((\lambda x.(f(xx)))(\lambda x.(f(xx)))))$

$Yt = (\lambda f(\quad)(\quad))t$

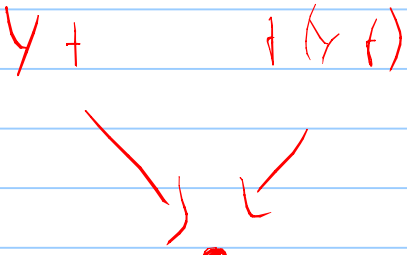
$\Rightarrow ((\lambda x.\ t(xx))(\lambda x.\ t(xx)))$

$\rightarrow t((\lambda x.\ t(xx))(\lambda x.\ t(xx)))$

terminiert nicht !!

$t(Yt) = t((\lambda f(\quad)(\quad))t)$

$\Rightarrow t((\lambda x(t(xx))(\lambda x.(t(xx)))))$



$Wt \xrightarrow{*} t(Wt)$

Fixpunktkombinatoren sind "Selbstbezugserzeuger"

wenn t keinen Ausdruck auswertet, dann terminiert Y t nicht

$t = \lambda f.\ \lambda x.\ \text{if zero } x \text{ then } \bar{1} \text{ else mul } \bar{2}\ (f\ (p\ x))$

$Y\ t = t\ (Y\ t)$

$= (\lambda f\ \lambda x.\ \text{if zero } x \text{ then } \bar{1} \text{ else mul } \bar{2}\ (f\ (p\ x)))\ (Y\ t)$

$= \lambda x.\ \text{if zero } x \text{ then } \bar{1} \text{ else mul } \bar{2}\ (Y\ t\ (p\ x))$

$(Y\ t)\ z = (t\ (Y\ t))\ z$

$(Y\ t)\ \bar{2} = (\lambda x.\ \text{if zero } x \text{ then } \bar{1} \text{ else mul } \bar{2}\ (Y\ t\ (p\ x)))\ \bar{2}$

$\Rightarrow$ if zero $\bar{2}$ then $\bar{1}$ else mul $\bar{2}$ $(Y\ t\ (p\ \bar{2}))$

$\rightarrow$ if $\bar{F}$ then $\bar{1}$ else mul $\bar{2}$ $(Y\ t\ \bar{1})$

$\rightarrow$ mul $\bar{2}$ $(Y\ t\ \bar{1})$

$\xrightarrow{\zeta}$ $\quad mul\ \bar{2}\ (mul\ \bar{2}\ (y + \bar{c}))$

$\rightarrow \quad mul\ \bar{2}\ (mul\ \bar{2}\ (\bar{1}))$

Reduktionsreihenfolge ist entscheidend für Terminierung

Wenn, dann hilft "call by name"

zuerst links außen

$(\lambda x.\lambda y.\ y)((\lambda x.\ xx)(\lambda x.\ xx))\ \bar{2}$

$(\lambda y.\ y)\ \bar{2}$

$\downarrow$

$\bar{2}$

$(\lambda x \lambda y.\ y)\ ((\lambda x.\ xx)(\lambda x.\ xx))\ \bar{2}$

$\downarrow \infty$

Alle terminierenden Reduktionen liefern dasselbe Ergebnis !!

('Konfluenz')

Begriffsbildung:

Rekursive Funktionen definiert man über Gleichungen

"Sei f x definiert durch

$$f\,x = \text{program}\,[\text{mit Vorkommen von } f \text{ und } x]"$$

im λ-Kalkül Demgemäß

$$f\,x = t\,[f, x]$$

Gleichung ist kein Programm!!

mit Y Kombinator kann man Gleichung in Programm umsetzen

$$f \equiv Y\,(\lambda f.\lambda x.\, t\,[f, x])$$

Notation aus funktionale Sprachen:

Sei f x definiert (rekursiv) durch t,

ML let rec f x = t

≡ Y (λf. λx . t)

max 5 Zeilen

Klausur: eine einfache λ Berechnung / Programmierung (z

→ eine/zwei generische Quizfragen

$\forall x : \mathbb{N} \;.\; \exists y : \mathbb{N} \;.\; x < y$

$\exists y : \mathbb{N} \;.\; \forall x : \mathbb{N} \;.\; x < y$ ⚡

$\forall x \;.\; \exists\, p$ A_x mit p prog

$\exists p \;\; \forall x$ A_x mit p läuft