B1

Tuesday, March 02, 2010
12:15

1. Stunde 2010-03-02 Teil 2

WAS IST MATH. LOGIK (NICHT)?

Math Logik ist:

(·) Metatheorie der Mathematik: z.B.:
- Was ist ein Algorithmus?
- Was ist ein Beweis?

(·) Behandelt diese Fragen mit mathematischen (nicht philosophischen) Methoden:
zB: Algorithmus :≡ Registermaschine

(·) Fragen stellen sich im "Alltag" oft nicht, es reicht, Bew. bzw Algorithmus zu finden.

(·) Klärung der Fragen aber unabdingbar, wenn man beweisen will: Problem ... ist <u>nicht</u> algorithmisch entscheidbar.
Satz ... ist unentscheidbar (weder beweisbar noch widerlegbar)

Logik:

(·) Trennung von Syntax ("Satzstruktur") und Semantik ("Bedeutung").

(·) Ursprünglich (Aristoteles) entwickelt als "universelle Logik des Denkens" bzw Argumentierens.

(·) Die Alltagslogik ist "problematisch": zu viel Kontext (→ nicht monoton etc)

(·) In Mathematik funktioniert es eben ausgezeichnet.

A1

Monday, March 01, 2010
16:59

= Ziegler S. 60 - 61
(einziger Unterschied: wir verwenden nur $\mathbb{N}$, nicht Zeichenketten, als Registerinhalt)

1. REKURSIONSTHEORIE

1.1 Registermaschine (mit R Registern)

("Hardware": R Register $\mathbb{R}_0, \dots \mathbb{R}_{R-1}$ ($\cong$ Variablen), die jeweils eine natürliche Zahl speichern.
Separat gespeichert: (·) Programm und (·) aktuelle Programmzeile)

Exakter: Maschine (="Programm") ist Folge von Befehlen $(b_0, b_1, \dots, b_N)$

Jeder Befehl b_i ist einer der folgenden Tupel:

- (·) 0 — stop
- (·) $(r, +1)$ $(r < R)$ — $\mathbb{R}_r + 1$
- (·) $(r, \dot{-} 1)$ $(r < R)$ — $\mathbb{R}_r \dot{-} 1$
- (·) (r, ℓ, m) $(r < R, \ell, m < N)$ — IF $\mathbb{R}_r = 0$ THEN GOTO b_ℓ ELSE GOTO b_m

OBdA: $b_N = 0 =$ "stop"

Konfiguration $\cong$ aktuelle Programmzeile + Speicherinhalt
exakter: Tupel $y = (\ell, r_0, \dots, r_{R-1})$ $\ell < N$, $r_0, \dots r_{R-1} \in \mathbb{N}$

Anfangskonfiguration zu input $x_1, x_2 \dots x_n$ $(n < R)$

$$y_0^{x_1 \dots x_n} = (\underset{\text{Pr-Z?}}{0}, \underset{\mathbb{R}_0}{0}, \underset{\mathbb{R}_1}{x_1}, \dots, x_n, 0, \dots \underset{\mathbb{R}_{R-1}}{0})$$

A1 (Forts.)

Monday, May 17, 2010
16:45

Berechnungsschritt: Gegeben Maschine $\mathcal{M}$ und Konfiguration $y = (\ell, r_0, \ldots, r_{R-1})$.

Dann ist Nachfolgerkonfiguration $\mathcal{M}(y) = (\ell', r_0', \ldots, r_{R-1}')$ def. durch:

(·) wenn b_ℓ = stop, dann $\mathcal{M}(y) = y$

(·) wenn $b_\ell = R_s := R_s + 1$, dann $\ell' = \ell + 1$, $r_s' = r_s + 1$, $r_t' = r_t$ für $t \neq s$;

(·) wenn $b_\ell = R_s := R_s \dot{-} 1$, dann $\ell' = \ell + 1$, $r_s' = \max(0, r_s - 1)$, $r_t' = r_t$ für $t \neq s$

(·) wenn b_ℓ = IF $R_s = 0$ THEN GOTO b_a ELSE GOTO b_b

dann $r_t' = r_t$ für alle t, und

$(r_s = 0 \rightarrow \ell' = a \; ; \; r_s \neq 0 \rightarrow \ell' = b)$

Die durch $\mathcal{M}$ berechnete partielle Funktion:

$f^n_{\mathcal{M}} : \mathbb{N}^n \rightarrow \mathbb{N}$: $(n < R)$

Gegeben $x_1 \ldots x_n \in \mathbb{N}^n$.

<u>Wenn</u> für ein k gilt: $\mathcal{M}^k(y_0^{x_1 \ldots x_n}) = (\ell, r_0, \ldots)$

und b_ℓ = stop, <u>dann</u> $f^n_{\mathcal{M}}(x_1 \ldots x_n) = r_0$

<u>Sonst</u> $f^n_{\mathcal{M}}(x_1 \ldots x_n)$ undef.

Def: $f : A \rightarrow B$ <u>part</u> Fkt. $:\Leftrightarrow \exists A' \subseteq A \;\; f : A' \rightarrow B$

Wenn $f : A \rightarrow B$ "klassisch" (d.h. $A = A'$) : f "total"

Def: $f : \mathbb{N}^n \rightarrow \mathbb{N}$ maschinen-berechenbar, wenn $\exists R > n$ und R-Registermaschine $\mathcal{M}$ sd. $f = f^n_{\mathcal{M}}$

A1 (Forts.)

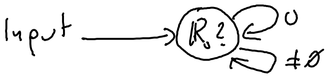
Monday, May 17, 2010
16:45

Bsp 1: $\mathcal{M} = (b_0)$

$b_0 =$ IF $R_0 = 0$ THEN GOTO 0 ELSE GOTO 0

D.h.: formal: $\mathcal{M} = (0)$ (Folge d. Länge 1, einz. El 0)

$\Rightarrow f^n_{\mathcal{M}}$ nirgends def



Bsp 2: $\mathcal{M} = (b_0, b_1, b_2, b_3, b_4)$

0: $b_0 =$ IF $R_1 = 0$ THEN GOTO 4 ELSE GOTO 1

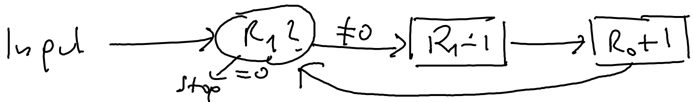
1: $b_1 = R_1 := R_1 \dot{-} 1$

2: $b_2 = R_0 := R_0 + 1$

3: $b_3 =$ GOTO 0 [IF $R_0 = 0$ GOTO 0 ELSE GOTO 0]

4: $b_4 =$ STOP

D.h. formal $\mathcal{M} = ((1,4,1), (1,-1), (0,+1), (0,0,0), 0$



$\Rightarrow f^1_{\mathcal{M}} = \mathrm{Id}$ $\quad f^1_{\mathcal{M}}(n) = n$

Bzw $f^n_{\mathcal{M}} = Pr^n_1$ $\quad f^2_{\mathcal{M}}(a,b) = a$

Def: $f : \mathbb{N}^n \to \mathbb{N}$ heißt (maschinen) berechenb., wenn
$\exists \mathcal{M}$ sd $f = f^n_{\mathcal{M}}$ (d.h., f wird durch $\mathcal{M}$ berechnet, insbesondere: $f(x_1 \dots x_n)$ ist def. gdw $\mathcal{M}$ auf input $(x_1 \dots x_n)$ hält !)

(Typische Prüfungsfrage: Zeige: $x \mapsto {}^{x+2}/_3$ (bel. Rundung) ist berechenbar. [Gib Maschine in beliebiger Notation an, gib Rundung an, z.B $\lfloor \frac{x+2}{3} \rfloor$ oder $\lceil \frac{x+2}{3} \rceil$)

B2

Monday, March 08, 2010
16:01

<u>Satz</u>: Es gibt nichtberechenbare Funktionen. (Bew: es gibt überabz. viele Fkt von $\mathbb{N}$ nach $\mathbb{N}$, aber nur abz. viele Programme) Später werden wir konkrete Fkt kennenlernen.

Behauptung ("These von Church"):
Eine (part.) Fkt $\mathbb{N}^n \to \mathbb{N}$ ist "irgendwie" berechenbar gdw sie Registermaschinen berechenbar ist.

Diese Beh. ist kein math. Satz, weil "irgendwie berechenbar" nicht ordentlich definiert ist.
Die Beh entspricht aber der Erfahrung: jedes "sinnvolle" Computermodell liefert denselben Berechenbarkeitsbegriff.
Insbesondere:

- ein "tatsächlicher" moderner Computer, mit <u>Idealisierung</u>: beliebig viel (=potentiell unend.) (adressierbaren) Speicher, mit "Maschinensprache"
- genauso: C, Basic, LaTeX, Perl, bash, Prolog, Lisp, Java, Python, ...
- bei Registermaschine: können uns (für Fkt von $\mathbb{N}$ nach $\mathbb{N}$) auf 3 Register beschränken (oder: pointer-artige Referenzen zulassen, Modifikation des Programms zulassen etc)
- Turingmaschine,
- μ-rek
- darstellbar in endl. axiomatisierten Theorien

Wegen "beliebigen Speicher" sind das natürlich alles "idealisierte" Modelle. Achtung: Programme sind <u>endlich</u>. Sonst: Jede Fkt f ist trivialerweise berechenbar:
(if $x=0$ print $f(0)$, if $x=1$ print $f(1)$, ----)

B2 (Forts.)

Monday, May 17, 2010
16:52

Einschränkungen / Warnungen:

(.) Tatsächliche Computer natürlich nur endl.
Es ist aber auf diesem Abstraktionsniveau kaum möglich, das in Betracht zu ziehen (auf endl. Computer ist ja nicht mal "lcd" oder "x+1" allgemein berechenbar)

B3

Tuesday, March 16, 2010
19:25

Rek. <u>universelle</u> Begriff der Berechenb. Aber:

(a) Nicht jede "theoretisch berechenbare" Fkt ist auch "praktisch berechenbar": Wenn Berechnung von $f(x)$ 2^{2^x} viele Schritte braucht, dann praktisch sinnlos. $\Rightarrow$ Komplexitätstheorie; Algorithmentheorie

Bsp: Eukl. Alg. berechnet ggT(a,b).

Es gibt aber auch trivialen Algorithmus mit Laufzeit $\sim \min(a,b)$

Eukl. Alg. besser: - Laufzeit ca $\log(\min(a,b))$

- zusätzl. Information (ggT Linearkomb. von a,b etc)

"Gute" Laufzeiten sind in der Praxis $\ln(x)$, $\ln(x)\cdot\ln(\ln(x))$ allenfalls $\ln(x)^2$ [oder, formuliert nicht in der Größe x des Inputs, sondern in der Länge l des Inputs in zB Decimaldarstellung: gute Laufzeiten sind: l, $l\cdot\ln(l)$, l^2, jedenfalls polynomiell in l]

(b) umgekehrt kann ein unberechenbare (oder: praktisch unberechenb.) Fkt $f:\mathbb{N}\to\mathbb{N}$ durchaus praktisch "approximierbar" sein: Es könnte zB ein (schnell) berechenbares g geben s.d. $g(x)=f(x)$ für $x<10^{10^{10}}$, oder so dass $|g(x)-f(x)| \ll x$ für alle x etc

$\Rightarrow$ Numerik, Optimierung, "Heuristik"

(c) natürlich gibt es stärkere Computermodelle: Sei f bel. nicht-berechenbare Fkt. (total)

Ein f-Programm ist ein Programm, das zusätzlich zu $R_e := R_e + 1$ auch $R_e := f(R_e)$ als Grundfunktion verwenden kann.

g heißt f-berechenbar, wenn g durch ein f-Programm berechnet werden kann.

$\Rightarrow$ partielle Ordnung der "Turing-Grade":

$g \leq_T f \;:\Leftrightarrow\; g$ ist f-berechenbar

A2

Wednesday, March 10, 2010
13:49

= Ziegler S 61–62
Unterschied: wir verwenden
partielle Fkt

1 2 μ-Rekursion

Grundfunktionen:

(·) $S : \mathbb{N} \to \mathbb{N} \quad x \mapsto x+1$

(·) $I_i^n : \mathbb{N}^n \to \mathbb{N} \quad (x_1 \ldots x_n) \mapsto x_i \qquad (i \leq n)$

(·) $C_0^0 : \mathbb{N}^0 \to \mathbb{N} \quad 0$

oder, wenn das zu unheimlich ist:

$C_0^1 : \mathbb{N} \to \mathbb{N} \quad x \mapsto 0$

(alle Grundfkt sind total)

Einsetzung: Gegeben $f : \mathbb{N}^n \to \mathbb{N}$ und

$g_i : \mathbb{N}^m \to \mathbb{N} \qquad (i = 1, \ldots, n)$

Dann liefert Einsetzung $f(g_1(x_1 \ldots x_m), \ldots, g_n(x_1 \ldots x_m))$

(Bem: wenn f, g_i total, dann auch $f(g_1 \ldots g_n)$)

Primitive Rekursion
Geg $g : \mathbb{N}^n \to \mathbb{N}$ und $h : \mathbb{N}^{n+2} \to \mathbb{N}$

die prim Rek $f : \mathbb{N}^{n+1} \to \mathbb{N}$ ist def durch:

$f(x_1 \ldots x_n, 0) = g(x_1 \ldots x_n)$

$f(x_1 \ldots x_n, y+1) = h(x_1 \ldots x_n, y, f(x_1 \ldots x_n, y))$

(wenn g, h total, dann f total)

A2 (Forts.)

Monday, May 17, 2010
17:22

<u>μ-Rek.</u> Geg $g : \mathbb{N}^{n+1} \to \mathbb{N}$ Die μ Rekursion μg ist die Fkt $f : \mathbb{N}^n \to \mathbb{N}$ def durch

$f(x_1 \dots x_n) = \min(y : g(x_1, \dots x_n, y) = 0)$

wenn g total ist! Allgemein:

$f(x_1 .. x_n) = \min(y : \forall y' < y \; g(x_1 .. x_n, y') \text{ def } \& \; g(x_1, \dots, x_n, y) = 0)$

(g tot. $\not\Rightarrow$ f tot.)

<u>Def</u> Die Klasse der prim. rek. Funktionen [bzw μ-rek. Fkt] ist die kleinste Klasse die die Grundfkt. enthält und unter Einsetzung, prim. Rek [bzw: <u>zusätzlich</u> μ-Rek.] abgeschlossen ist.

Äquiv.: $f : \mathbb{N}^n \to \mathbb{N}$ (partielle) ist prim. rek [bzw μ-rek.], wenn f aus den Grundfkt durch Einsetzung, prim. Rek. [bzw: zusätzlich μ-Rek.] gebildet werden kann.

Es gilt: jede prim. rek. Fkt. ist total.

Bsp: (·) $x+3$ ist prim rek:

$x + 3 = S(S(S(x)))$, d.h. entsteht durch Einsetzung der Grundfkt. S

(·) Die konstante Fkt 3 ist prim rek

$3 = (x+3) \circ C_0^0$

(·) $f = y \dot{-} 1$ ist prim rek:

$f(0) = 0 = g() \to g = C_0^0$

$f(y+1) = y = h(y, f(y)) \to h = I_1^2$

$\Rightarrow \underset{1\text{-st.}}{f}$ ist prim. Rek von $\underset{0\text{-st.}}{C_0^0}$ und $\underset{2\text{-st.}}{I_1^2}$

(·) $f(x,y) = x + y$ ist prim rek

$f(x, 0) = x = g(x) \to g = I_1^1$

$f(x, y+1) = f(x,y) + 1 = h(x, y, f(x,y)) \to$

$h = S \circ I_3^3$

B4

Tuesday, March 16, 2010
19:57

<u>Nachtrag</u>: Wir kümmern uns allg. um <u>part</u> Fkt.
Insbes: für f, g $\mathbb{N}^n \to \mathbb{N}$ part, $f = g$ heißt:
$dom(f) = dom(g)$ (d.h., $f(\bar{x})$ def gdw $g(\bar{x})$ def) <u>und</u>
$\bar{x} \in dom(f) \to f(\bar{x}) = g(\bar{x})$

Wenn M eine Registermaschine (d.h. ein Programm) ist, dann ist $f_M^n : \mathbb{N}^n \to \mathbb{N}$ (die von M berechnete n-dim. Fkt.) natürlich i.A partiell.
Genauso für Def von rekursiver Funktion
(Grundfkt + Abschluss unter Einsetzung, prim. Rek. und μ-Rek.):

Insbesondere: Wenn $f_1 \ldots f_n$ $\mathbb{N}^m \to \mathbb{N}$ und $g : \mathbb{N}^n \to \mathbb{N}$ part, dann ist
$g(f_1(x_1 .. x_m), .., f_n(x_1 ... x_m))$ per definition undefiniert gdw $f_1(x_1 .. x_m)$ undef. oder ... oder $f_n(x_1 ... x_m)$ undef. oder $g(f_1(x_1 .. x_m), .., f_n(x_1 .. x_m))$ undefiniert.

Bsp: $g : \mathbb{N} \to \mathbb{N}$ konstante 0-Fkt, $f : \mathbb{N} \to \mathbb{N}$ part.
dann ist $g \circ f$ i.A. <u>nicht</u> total (und nicht die konst. 0-Fkt)

Analog bei der Konstr. von f mittels prim. rek. bzw μ-Rek. aus part. Fkt: f ist nur dann def, wenn alle zur Konstr. von f verwendeten Funktionswerte definiert sind.

B4 (Forts.)

Monday, May 17, 2010
16:57

Bem: Um diese notationellen Probleme zu vermeiden, definiert Ziegler rekursive Fkt nur für totale Funktionen. Dadurch kann aber zB nicht die (partielle, rekursive) universelle Funktion definiert werden.

<u>Notation</u>: (•) Zur Erinnerung:
Eine (part.) Fkt $f: \mathbb{N}^n \to \mathbb{N}$ ist berechenbar, wenn es eine Registermaschine M gibt sd: $f = f_M^n$ (d.h. M berechnet f)
Insbes: $\bar{x} \in dom(f)$ gdw (M hält auf input $\bar{x}$)

(•) statt "Registermaschine" sagen wir (entscheidbar) auch: Maschine, Computer, Programm, Computerprogramm

(•) Eine part. Fkt $f: \mathbb{N}^n \to \mathbb{N}$ ist rek, wenn sie aus Grundfkt durch Einsetzung, prim. Rek und μ-Rek. entsteht.

(•) statt "rek.", sagen wir auch: partiell rekursiv, μ-rekursiv.

(•) $A \subseteq \mathbb{N}$ ist rek., wenn F_A (natürlich total und) rek. ist. Statt rek. sagen wir auch: entscheidbar.

2A (Forts.) = Ziegler 63–65

1.3. Jede rek. Fkt ist berechenbar

Bew mit Ind: Wir müssen zeigen:

(1) Grundfkt. sind berechenbar

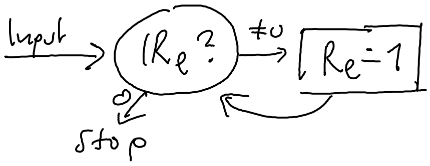
(2) Wenn $g, f_1 \dots f_n$ berechenb., dann auch $g(f_1, \dots, f_n)$

(3) Wenn g, h berechenb., dann auch $f = \text{Prim-rek}(g,h)$

(4) Wenn g berechenb., dann auch $f = \mu g$

Bew: Vorbereitung:

(*) Hilfsprogramm $\mathcal{L}^l$ (löscht Register l)



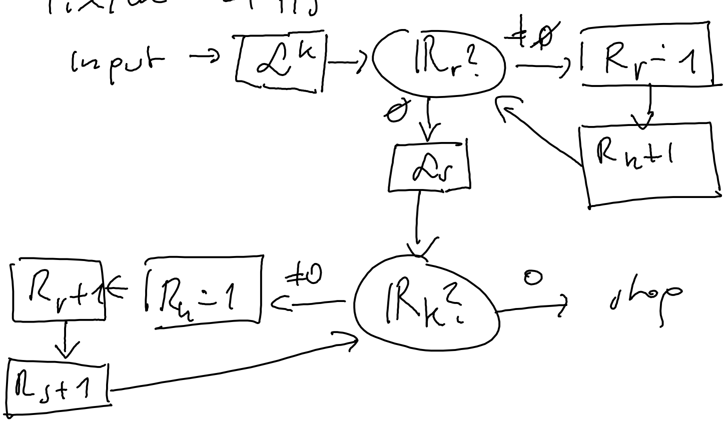
D.h. formal: $\mathcal{L}^l = ((l,3,1),(l,-1),(0,0,0),0)$

(*) Hilfsprogramm $K^{r,s}$:

Kopiere Register r nach Register s

D.h. bei Input $\begin{pmatrix} \mathbb{R}_0, & \mathbb{R}_1 & \dots & \mathbb{R}_r & \dots & \mathbb{R}_s \\ x_0, & x_1, & & x_r & & x_s \end{pmatrix}$

Fixiere $h \neq r, s$



Grundfunktionen:

S: Input $\to \boxed{K^{1,0}} \to \boxed{R_0 + 1} \to$ stop

I^n_l Input $\to \boxed{K^{l,0}} \to$ stop

C^0_0 Input $\to$ stop

= Ziegler 63–65

Monday, March 15, 2010
09:18

Hintereinanderausführen von Programmen
(haben wir bereits verwendet):

Formel: $M_1 = (b^1_0, \dots, b^1_n)$ $M_2 = (b^2_0, \dots, b^2_m)$

Dann ist $M_1 M_2$ ("zuerst M_1 dann M_2) def
als $(c_0, \dots, c_{n+m+1})$, wobei:

für $0 \le i \le n$ $c_i = b^1_i$, ausser:

(·) $b^1_i = 0$ (STOP) dann $c_i = (0, n+1, n+1)$ (GOTO $n+1$)

für $0 \le i \le m$, $c_{n+1+i} = b^2_i$, ausser:

(·) $b^2_i = (l, a, b)$, dann $c_i = (l, n+1+a, n+1+b)$

Informell: $\longrightarrow \boxed{M_1} \longrightarrow \boxed{M_2}$

Schritt im Bew: Jede rek. Fkt. ist berechenb.
Haben bereits gezeigt: (·) Grundfkt. berechenb.
(·) Folgende Hilfspr: L^r (lösche r-tes Reg.)
$K^{r,s}_h$ (kopiere Reg. r auf s, verw. Hilfsreg. h)

Ang. M berechnet n-stellige Fkt. und verwendet
Register $R_0 \dots R_R$. Dann $M^* \equiv L^0 L^{n+1} \dots L^{R-1} M$

Bsp: Wenn $h(\cdot,\cdot)$ und $f_1(\cdot), f_2(\cdot)$ berechenb.,
dann auch $x \mapsto h(f_1(x), f_2(x))$ [andere Dim. analog]

Ang H, F_1, F_2 berechnen h, f_1 und f_2, und
verwenden alle nur $R_0 \dots R_{R-1}$

Dann verwende

$$\underbrace{K^{1,R}_{R+2}}_{\text{rette } x \text{ nach } R} \; \underbrace{F_1 K^{0,R+1}_{R+2}}_{\text{ber. } f_1(x) \text{, Kop nach } R+1} \; \underbrace{K^{R,1}_{R+2} F_2^* K^{0,2}}_{\text{hole } x\text{, berechne } f_2(x)\text{, speichere in } R_2} \; \underbrace{K^{R+1,1} H^*}_{\text{kopiere } f_1(x) \text{ nach } R_1\text{, ber. } h}$$

3A (Fortsetzung)

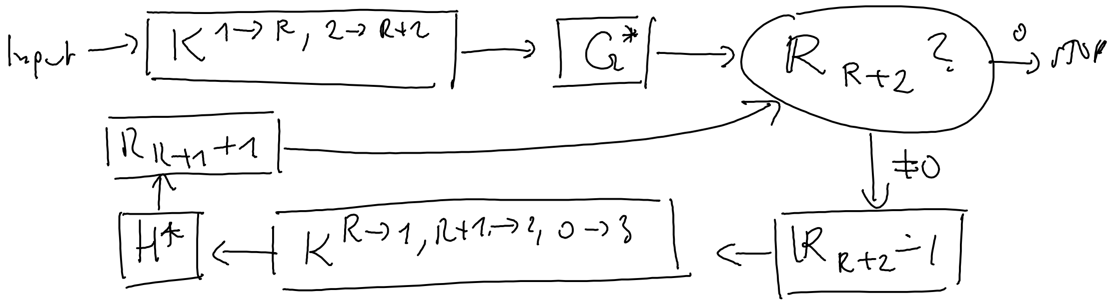
Monday, March 15, 2010
10:08

<u>Prim. rek.:</u>

Wenn $g(x)$ und $h(x,y,z)$ berechenbar, dann auch $f(x,y)$ def durch: $f(x,0) = g(x)$ und $f(x,y+1) = h(x,y,f(x,y))$

Bew: Ang G, H berechnen g, h, und verw. nur Register $R_0 \dots R_{k-1}$.

Verwende folgende Maschine (wobei die Kopiermaschinen das Hilfsreg R_{k+3} verw.)



μ-Rek.: analog (siehe Übungen)

4A

Monday, March 15, 2010
10:38

1.9. Jede berechenb. Fkt ist rekursiv.

$\boxed{\text{Ziegler Skriptum S 66 – 68, exklusive 12.6}}$

Bemerkungen:

(·) Eine Relation P auf $\mathbb{N}^n$ (auch Prädikat genannt) ist schlicht eine Eigenschaft von n-Tupeln.

Bsp für $n=1$ $\quad P(n) \equiv$ "n ist Primzahl"

$n=2$ $\quad P(n,m) \equiv$ "n teilt m" etc

Auf offensichtliche Weise ist P äquivalent zu einer Teilmenge von $\mathbb{N}^n$: $P' = \{\bar{x} \in \mathbb{N}^n : P(\bar{x})\}$

(In den obigen Bsp, P' ist Menge der Primzahlen bzw die Menge $\{(n,m) : n|m\}$

(·) Üblicherweise wird die charakteristische Funktion anders def. als im Ziegler Skriptum, nämlich als $1 - K_R$, aber das macht natürlich keinen Unterschied

(·) ⅟.

4A (Forts.)

Thursday, April 29, 2010
14:12

(*) Lem 12.3 braucht anderen Bew, wenn f, g nicht total sind:

Lem: Wenn f, g, P rek, dann auch $h: x \mapsto \begin{cases} f(x) & \text{wenn } P(x) \\ g(x) & \text{sonst.} \end{cases}$

(insbes: $h(x)$ def gdw:
($P(x)$ und $f(x)$ def.) oder ($\neg P(x)$ und $g(x)$ def.)

Bew vom Ziegler-Skr:
$h = f \cdot K_P + g(1 - K_P)$. Das funktioniert aber nur für f, g total! (sonst: $\mathrm{dom}(h) = \mathrm{dom}(f) \cap \mathrm{dom}(g)$)

Beweis für allg. f, g: $f \cong M_f$, $g \cong M_g$, $P \cong M_P$
schreibe Programm M:
- Kopiere Input x in Hilfsregister
- Entscheide ob $P(x)$ (terminiert immer)
- wenn ja:
  Kopiere Input aus Hilfsregister zurück,
  Lösche alle anderen Reg
  starte M_f
- wenn nein: analog für M_g

Bem: Nach unserer Def: $\mathrm{dom}(f \cdot g) = \mathrm{dom}(f) \cap \mathrm{dom}(g)$
Bsp: "$0 \cdot g$" (x) def gdw $g(x)$ def
Bem: f rek, $\mathrm{dom}(g) \subseteq \mathrm{dom}(f)$, $g(x) = f(x)$ für $x \in \mathrm{dom}$
$\not\Rightarrow$ g rek;
(A r.e. $:\Leftrightarrow$ $A = \mathrm{dom}(g)$ für g rek.)

5A

Sunday, April 25, 2010
16:18

(Ziegler 68 - Cam 12.7 Mil. Beu)

A6

Monday, May 17, 2010
17:07

Ziegler Skriptum S70 – 71.

B6

Sunday, April 25, 2010
16:18

Das universelle Programm

<u>Def</u>: (1) Sei $T(m,x,y)$ das Kleene Präd T_1 aus dem Ziegler Skriptum. T ist prim. rek.

(2) Das universelle Programm $U : \mathbb{N}^2 \to \mathbb{N}$ (eine partielle rek. Fkt) ist def durch:
$U(m,x) = (\mu y\, T(m,x,y))_0$
Das heißt: Sei $\mathcal{M}$ eine Maschine. Dann $f^1_{\mathcal{M}}(x)$ (der Output von $\mathcal{M}$ auf Input x) $= U(\ulcorner \mathcal{M} \urcorner, x)$ (wobei $\ulcorner \mathcal{M} \urcorner$ der Code (= die Gödel-Nummer von $\mathcal{M}$ ist)

1.5 Folgerungen des universellen Programms

(a) berechenbare $\Longleftrightarrow$ rekursiv
(da ja $f^1_{\mathcal{M}}(x) = U(\ulcorner \mathcal{M} \urcorner, x)$, $U : \mathbb{N}^2 \to \mathbb{N}$ rek
daher $U(\ulcorner \mathcal{M} \urcorner, \cdot) : \mathbb{N} \to \mathbb{N}$ rek;
(rek $\to$ berechenbar wissen wir bereits)

(b) Jede (part) rek Fkt f hat Darstellung
$f(x) = (\mu y\; g(x,y))_0$, wobei g prim rek.
und $(\cdot)_0$ die "erste Komp. Fkt" $\langle x_0, \dots, x_n \rangle \mapsto x_0$ ebenfalls prim rek, zusätzl. mit $(\text{undef.})_0 = \text{undef.}$
D.h.: Abgesehen von $(\cdot)_0$ reicht es, prim. Rekursion und Einsetzung auf totale Fkt anzuwenden, und μ-rek. ein einziges Mal ebenfalls auf tot. Fkt (aber das Ergebnis ist natürlich i.A. <u>nicht</u> total)

B6 (Forts.)

Monday, May 17, 2010
17:11

(c) Unlösbarkeit des Halteproblems!

Erinnerung / Motivation

Es gibt nur abz. viele ber. Fkt. Es gibt wegen des univ. Prog U sogar eine uniforme Aufzählung $f_n = U(n, \cdot)$

	0	1	2	...
f_0	2	17	3	...
f_1	5	1	0	...
f_2	0	1	0	...
⋮	⋮	⋮	⋮	

Input →; rek. Fkt ↓

Diagonalfkt:

$g(n) := f_n(n) + 1$

g ist berechenbar!

Daher: $g \equiv f_a$ für irgendein a

Dann ist aber $f_a(a) = g(a) = f_a(a) + 1$.

Das ist aber kein Widerspruch;

Programme terminieren im Allgemeinen nicht!

Die berechenb. Fkt. sind also i.A. partiell.

Ang, $f_3(3) =$ undef. Dann könnte $g \equiv f_3$ sein:

$f_3(3) = \text{undef} = g(3) = f_3(3)+1 = \text{undef} + 1 = \text{undef}$.

(Ende Motivation)

Damit sehen wir aber auch, dass das Halteproblem $H = \{x : U(x,x) \text{ def}\}$ nicht rek. sein kann: Sonst wäre

$$g(x) := \begin{cases} U(x,x)+1 & \text{wenn } U(x,x) \text{ def} \\ 0 & \text{sonst} \end{cases}$$

rek., d.h. $g(x) = U(\ulcorner M \urcorner, x)$ für ein M,

d.h. $g(\ulcorner M \urcorner) = U(\ulcorner M \urcorner, \ulcorner M \urcorner) = \begin{cases} U(\ulcorner M \urcorner, \ulcorner M \urcorner)+1 & \text{wenn def} \\ 0 & \text{sonst} \end{cases}$

Widerspruch

B7

Sunday, April 25, 2010
16:41

1.6. Rekursive und r.e. Teilmengen von $\mathbb{N}$

Wiederholung: $A \subseteq \mathbb{N}^n$ rek. $:\Leftrightarrow$ $K_A : \mathbb{N}^n \to \mathbb{N}$ rek
(charakteristische Fkt) $\Leftrightarrow$ Es gibt Computerprogr.
das auf jeden Input x hält und ausgibt
(entscheidet) ob $x \in A$ oder $x \notin A$.
Haben gesehen: Wenn $A, B \subseteq \mathbb{N}$ rek und
$f: \mathbb{N} \to \mathbb{N}$ rek, total, dann ist
(·) $A \cap B$, $A \cup B$, $\mathbb{N} \setminus A$, $A \setminus B$, $A \Delta B$ rek
(·) $\{x : f(x) \in A\}$ rek
(☺) (·) $\{ x : \exists z < f(x) \; z \in A \}$ rek etc
(·) $\{x : x$ Gerade$\}$, $\{x : x$ Primzahl$\}$, ... rekursiv

Im Unterschied dazu:
Rekursiv aufzählbar (r.e., recursively enumerable)
Intuitiv: Es gibt ein Computerprogramm, das auf Input x "Ja" ausgibt wenn $x \in A$. Aber wenn $x \notin A$ dann kann M entweder "Nein" ausgeben <u>oder gar nicht terminieren</u>.

Formal: $A \subseteq \mathbb{N}^n$ r.e. $:\Leftrightarrow$ $\exists$ $B \subseteq \mathbb{N}^n \times \mathbb{N}$ rek sd
$A = \{x : \exists y \; (x, y) \in B\}$

H ist r.e., weil $x \in H \Leftrightarrow U(x, x)$ def $\Leftrightarrow \exists g \; T(x, x, g)$
und T ist rek. (sogar: prim. rek.)
Bem: Identifiziere $T \subseteq \mathbb{N}^3$ mit Präd. $T(\cdot, \cdot, \cdot)$

B7 (Forts.)

Thursday, April 29, 2010
14:11

Lem: TFAE (the following are equivalent):

(1) A r.e. (d.h.: $\exists B \subseteq \mathbb{N}\times\mathbb{N}$ rek, $A = \{x : \exists y\, (x,y) \in B\}$

(2) $\exists g: \mathbb{N} \to \mathbb{N}$ part. rek $A = dom(g) = \{x \in \mathbb{N} : g(x) \text{ def.}\}$

(3) $\exists f: \mathbb{N} \to \mathbb{N}$ part. rek. $A = f''\mathbb{N} = \{f(x) : x \in dom(f)\}$

(4) $A = \emptyset$ oder $\exists f: \mathbb{N} \to \mathbb{N}$ total rek $A = f''\mathbb{N}$

(5) A endl oder $\exists f: \mathbb{N} \to \mathbb{N}$ total, inj., rek $A = f''\mathbb{N}$

Bew: $(5) \to (4)$ und $(4) \to (3)$ klar

$(3) \to (2)$ $A = f''\mathbb{N} = \{y : \exists \langle x, z\rangle : T(m, x, z) \;\&\; y = (z)_0\}$

(dabei ist m der Code einer Maschine die f berechnet).

$g(y) := \mu k : T(m, (k)_0, (k)_1) \;\&\; y = ((k)_1)_0$

$g(y)$ def gdw $f(y)$ def gdw $y \in A$

$(2) \to (1)$ $A = dom(g) = \{x : \exists y \;\; T(m, x, y)\}$ ist

Proj. einer rek. (sogar: primrek) TM von $\mathbb{N}\times\mathbb{N}$

B7 (Forts.)

Monday, May 17, 2010
17:27

$(1) \to (5)$ Sei A unendl und die Projektion von B.

$P_0(s,y) :\iff \forall m < lg(s) : y \neq (s)_m$ ist prim. rek.

$h_0 : s \mapsto \mu z : \quad z = \langle x, y\rangle, \ (x,y) \in B, \ P_0(s,y)$

ist also rek, und (weil A unendl) total.

$h_1(\langle a_0, a_1, .., a_n\rangle) := \langle a_0, a_1, .., a_n, h_0(\langle a_0, .. a_n\rangle)\rangle$

ist tot. / rek. (weil die "append" Fkt rek ist),

$f(0) := (h_1(0))_0 \in A$

$f(1) := (h_1(h_1(0)))_1 = (\langle f(0), h_0(\langle f(0)\rangle)\rangle)_1 =$

$h_0\langle f(0)\rangle$ ist in A und ungleich $f(0)$.

Allgemein:

$f(n) = (h_1^{n+1}(0))_n$ ist in A, ungl. $f(0), .., f(n-1)$

$\Rightarrow f$ ist die gesuchte Fkt:

total, inj und $f''\mathbb{N} \subseteq A$ nach Konstruktion

$f''\mathbb{N} = A$: Sei $x \in A$. Dann gibt es y sd $(x,y) \in B$

Sei y sd $(x,y) \in B$ und $\langle x, y\rangle$ <u>min.</u> mit dieser Eig. Es gibt nur endl viele $z < \langle x,y\rangle$, h_1 nimmt also irgendwann den Wert $\langle x,y\rangle$ an, damit ist x im Bild von f.

5B6B

Sunday, April 25, 2010
20:46

Berechnung für andere Bereiche als $\mathbb{N}$

Haben gesehen: Wir können Folgen natürlicher Zahlen $(x_0, \dots, x_{n-1})$ als nat. Zahl $\langle x_0, \dots, x_{n-1}\rangle$ kodieren.

Auf diese und ähnliche Weise können wir viele andere Bereiche kodieren und Berechenbarkeit, Entscheidbarkeit und r.e. für diese Bereiche kodieren.

Insbesondere:

$\mathbb{Z}$: kodiere $z = (-1)^a b$ als $\ulcorner z \urcorner = \langle a, b\rangle$ (um Fkt eindeutig zu machen: für $a \in \{0,1\}$ und $b > 0$ oder $a = b = 0$)

Sei $f: \mathbb{Z} \to \mathbb{Z}$. Def $\tilde{f}: \mathbb{N} \to \mathbb{N}$ sd.
$\tilde{f}(\ulcorner z \urcorner) = \ulcorner f(z) \urcorner$ (und zB $\tilde{f}(x) = 0$, falls x kein Code)

Def: f rekursiv gdw $\tilde{f}$ rekursiv.

Genauso: $A \subseteq \mathbb{Z}$ rekursiv $:\Leftrightarrow \{\ulcorner z \urcorner : z \in A\} \subseteq \mathbb{N}$ rek.
(und analog für r.e. etc).

Bem: (·) $A \subseteq \mathbb{Z}$ rek. $\Leftrightarrow K_A : \mathbb{Z} \to \mathbb{Z}$ rek.

(·) Hätten Registermaschinen so def. können dass in jedem Register Element von $\mathbb{Z}$ steht, das hätte zu selben Begriff von rek. geführt.

(·) Analog für Fkt: $\mathbb{Z}^n \to \mathbb{Z}^m$ etc

Bsp: natürlich sind Addition, Multiplikation, Division mit Rest, etc etc alles rek. Fkt

$\mathbb{Q}$: Kodiere $q = (-1)^a \frac{b}{c}$ als $\ulcorner q \urcorner = (a, b, c)$
für $(a = b = 0, c = 1)$ oder $a \in \{0,1\}, b, c > 0$, $\mathrm{ggT}(b,c) = 1$
Rest analog wie für $\mathbb{Z}$

5B6B (Forts.)

Sunday, April 25, 2010
21:05

<u>$\mathbb{N}^n$</u>: Wir haben bereits def. was es heißt dass $f: \mathbb{N}^n \to \mathbb{N}$ rekursiv ist (bzw dass $A \subseteq \mathbb{N}^n$ rek. ist). Äquivalent hätten wir verwenden können:
Für $f: \mathbb{N}^n \to \mathbb{N}$ def $\tilde{f}: \mathbb{N} \to \mathbb{N}$ durch

$$\tilde{f}(z) = \begin{cases} f(x_1 .. x_n), & \text{wenn } z = \langle x_1 .. x_n \rangle \\ 0 & \text{sonst.} \end{cases}$$

Dann gilt: f rek. gdw $\tilde{f}$ rek.
Wir hätten uns also bei der Def von rek. gleich auf Fkt von $\mathbb{N}$ nach $\mathbb{N}$ beschränken können.

<u>Polynome</u> zB aus $\mathbb{Z}[X_1 \dots X_{17}]$
Ein Monom π hat die Form $(-1)^a \cdot b \cdot X_1^{c_1} \dots X_{17}^{c_{17}}$
für $a \in \{0,1\}$, $b, c_1, .., c_{17} \in \mathbb{N}$
Der Code $\ulcorner \pi \urcorner$ von π sei $\langle a, b, c_1, .., c_{17} \rangle$.
Der Code $\ulcorner p \urcorner$ eines Polynoms $p \in \mathbb{Z}[X_1 .. X_{17}]$ der Form $\sum_{i<n} \pi_i$ sei $\langle \ulcorner \pi_0 \urcorner, \dots, \ulcorner \pi_{n-1} \urcorner \rangle$.
(Bem: man kann noch eine Ordnung einführen, um jedem Polynom einen eindeutigen, von der Ordnung unabhängigen, Code zu geben)
Ein Fkt $f: \mathbb{Z}[X_1, .., X_{17}] \to \mathbb{Z}[X_1 \dots X_{17}]$ ist rek, wenn $\tilde{f}: \ulcorner p \urcorner \mapsto \ulcorner f(p) \urcorner$ (od 0, falls kein Code) rek. ist.
Genauso: $A \subseteq \mathbb{Z}[X_1 \dots X_{17}]$ entscheidbar (= rekursiv), wenn $\{\ulcorner p \urcorner : p \in A\} \subseteq \mathbb{N}$ entscheidbar.

5B6B(Forts.)

Sunday, April 25, 2010
21:10

10. Hilbertsches Problem / Satz von Matijasevitch:

Satz: $N = \{ \ulcorner p \urcorner \in \mathbb{Z}[X_1 .. X_{17}] : p$ hat zumindest eine ganzzahlige Nullstelle$\}$ ist nicht entscheidbar (aber natürlich r.e.).

(Ohne Beweis. Beweisidee: Konstruiere auf effiziente (= rekursive) Art zu jeder Maschine M ein Polynom p sd: M hält auf Input $\ulcorner M \urcorner$ gdw p hat ganzz. NS.
(D.h. finde $f: \mathbb{N} \to \mathbb{N}$ rek. sdass für alle x gilt $f(x) \in N$ gdw $x \in H$. Wäre N rek., dann auch H, Widerspruch.
Diese Konstruktion braucht etwas (elementare) Zahlentheorie.)

Ähnliche Ergebnisse gibt es in anderen Gebieten, zB Wortproblem in Gruppen:
Sei G erzeugt durch $a_1 \dots a_n$. Ein Wort w hat die Form $b_1^{e_1} \cdot b_2^{e_2} \dots b_m^{e_m}$, wobei $b_i \in \{a_1 \dots a_n\}$ und $e_i \in \mathbb{Z}$ für alle $i \leq m$.

Satz (ohne Bew.)

Es gibt Gruppe G die def ist durch endl. viele Erzeuger $a_1 \dots a_n$ und endl. viele Relationen $w_1 = 1$, $w_2 = 1$, $\dots$ $w_N = 1$ so dass das Wortproblem: (Ist $w = 1$?) unentscheidbar ist.
(Bem: Auch hier kann man eine Übersetzung finden von Computer M zu Wort w^M sd. M hält gdw $w^M = 1$)
Bem: Wortproblem ist natürl. r.e. (wieso?)

5B6B (Forts.)

Sunday, April 25, 2010
21:28

Zeichenketten

Man kann auch (was dem Begriff des math. Algorithmus vielleicht besser entspricht) ganz allgemein Zeichenketten (zu einem recht umfassenden mathematischen Alphabet) kodieren: Dann wäre das Polynom "Menge von Buchstaben
" $-17 \cdot X^{\wedge}12 + 3 \cdot X^{\wedge}5$ " eine Folge
$(-, 1, 7, \cdot, X, {}^\wedge, 1, 2, +, 3, \cdot, X, {}^\wedge, 5)$
jeder Buchstabe bekommt eine eindeutige Zahl, zB
$(11, 1, 7, 12, 20, 13, 1, 2, 14, 3, 12, 20, 13, 5)$
und diese Folge von Zahlen kodiert man mit der üblichen $\langle\rangle$ Fkt als einzige Zahl.
(Auf diese Weise kann man zu jedem abzählb. Alphabet Σ den Zeichenketten (= endliche Folge von Buchstaben) Σ^* codes in $\mathbb{N}$ zuordnen)
Auch so kann rek. zB für $\mathbb{Z}[X_1 .. X_{17}]$ def, natürlich mit demselben Ergebnis.
Im Ziegler-Skr. sind Registermaschinen so def., daß sie mit Zeichenketten operieren.
Es gilt: $f: \Sigma^* \to \Sigma^*$ ist Ziegler-Skript-berechenbar $\iff$ $\tilde{f}: \mathbb{N} \to \mathbb{N}$ $\ulcorner \sigma \urcorner \mapsto \ulcorner f(\sigma) \urcorner$ ist berechenbar