3. Stunde

Tuesday, March 16, 2010
19:25

Rek. universelle 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. ⇒ 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 eine 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
⇒ 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.
⇒ partielle Ordnung der "Turing-Grade":
$g \leq_T f \;:\Leftrightarrow\; g$ ist f-berechenbar

3. Stunde (Fortstzung)

Tuesday, March 16, 2010
19:25

Es gibt unberechenbare Funktionen.

Bew: Es gibt überabz. viele Funktionen $\mathbb{N} \to \mathbb{N}$,
aber nur abz. viele Programme

Es gibt also nur abz. viele Funktionen:

ber. Fkt ↓ / Input →	0	1	2	...
f_0	2	17	3	...
f_1	5	1	0	...
f_2	0	1	0	...
⋮	⋮	⋮	⋮	

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

Q Ist g berechenbar?
Wenn ja, dann $g \equiv f_a$
für irgendein a

(Die f_n sind ja alle berechenb. Fkt.).
Dann ist aber $f_a(a) = g(a) = f_a(a) + 1$.
Widerspruch?

(a) Wir können f_n so wählen, dass g ber.:
- Kodiere Programme als natürliche Zahlen
- Zeige: Folgende Fkt $U: \mathbb{N}^2 \to \mathbb{N}$ ist berechenb.
  $U(e,n) :=$ output von Programm Nummer e auf Input n
  ("universelles Programm" ≅ Interpreter)

Dann ist $g(n) := U(n,n) + 1$

(b) Trotzdem bekommen wir keinen 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:
$f_3(3) = \text{undef} = g(3) = f_3(3)+1 = \text{undef}+1 = \text{undef}$.

(c) Das impliziert die Unentscheidbarkeit des Halteproblems, z.B.:

$$h(x) = \begin{cases} 1, \text{ wenn } \varphi(x,x) \text{ def.} \\ 0 \text{ sonst} \end{cases}$$

! h ist nicht berechenbar !

Sonst wäre $g_1(x) := \begin{cases} \varphi(x,x)+1, \text{ wenn } h(x)=1 \\ 0 \text{ sonst} \end{cases}$

ebenfalls rek.,

das liefert nun aber wirklich einen Widerspruch.

(d) Unentscheidbarkeit des Halteproblems hat wichtige Anwendungen:

(1) [ohne Bew.] 10. Hilbertsches Problem, die allg. Diophant. Gleichung:

Es gibt keinen Algorithmus, der entscheidet ob ein $f \in \mathbb{Z}[X_1 \dots X_7]$ eine ganzz. NS hat.

(2) [ohne Bew.] Wortproblem in Gruppen:

Es gibt eine Gruppe G die durch endlich viele Erzeugenden $a_1 \dots a_n$ und endl. viele Rel. $R_1 \dots R_N$ def. ist, so dass folgende Frage algorithmisch unentscheidbar ist:

Ist ein Wort (zB: $a_1^5 a_2^{-1} a_1^3 a_7 a_4^{-1}$) gleich 1?