GTI

Hannes Diener

ENC B-0123,
diener@math.uni-siegen.de

2. Juli

In diesem Kapitel wollen wir sehen, daß alle unserer drei Berechenbarkeitsmodelle zum gleichen Funktionsbegriff führen. D.h. für eine partielle Funktion $f : \mathbb{N}^k \rightharpoonup \mathbb{N}$ sind äquivalent:

- f ist Turing-berechenbar
- f ist `WHILE`-berechenbar
- f ist μ-rekursiv.

Genauer werden wir in drei Schritten zeigen, daß

- jede μ-rekursive Funktion `WHILE`-berechenbar ist,
- jede `WHILE`-berechenbare Funktion Turing-berechenbar ist,
- und jede Turing-berechenbare Funktion μ-rekursiv ist.

Satz

Jede μ-rekursive Funktion ist `WHILE`-berechenbar.

Beweis.

Es reicht zu zeigen, daß die Basisfunktionen `WHILE`-berechenbar sind, und die Menge der `WHILE`-berechenbaren Funktionen unter Substitution, primitiver Rekursion und Minimalisierung abgeschlossen sind. Die ersten beiden Aussagen sind einfach (Übung), primitive Rekursion hatten wir schon im Kapitel über `WHILE`-Berechenbarkeit behandelt, es bleibt also der Abschluss unter Minimalisierung zu zeigen. □

Beweis.

...

Sei hierzu P ein WHILE-Programm, welches eine Funktion $f : \mathbb{N}^{k+1} \rightharpoonup \mathbb{N}$ berechnet. Dann wird μf durch folgendes WHILE-Programm berechnet.[1]

$x_\ell := 0;$

$x_j := P(x_1, \ldots, x_k, x_\ell);$

WHILE x_j DO

    $x_\ell := x_\ell + 1$

    $x_j := P(x_1, \ldots, x_k, x_\ell);$

END

$x_1 := x_\ell$

[1] Natürlich sind x_ℓ und x_j frische Variablen (also ist $x_\ell := 0$ überflüssig). Man beachte, daß der Unterprogramm-aufruf syntaktischer Zucker ist und die Variablen nicht verändert.

Satz

Jede `WHILE`-berechenbare Funktion ist Turing-berechenbar.

Beweis.

Vorlesung. □

Satz

Jede Turing-berechenbare Funktion ist μ-rekursiv.

Beweis.

Wir können annehmen, daß die TM M, mit dem Bandalphabet $\{0, 1, B\}$ arbeitet und eine totale Übergangsfunktion hat (d.h. nie verwirft). Ausserdem können wir annehmen, daß $Q = \{q_0, \ldots, q_n\}$ und q_1 der Startzustand und q_0 der einzige Endzustand. Eine Konfiguration $\kappa = (\langle a_1, \ldots, a_m \rangle, q_i, \langle b_1, \ldots, b_n \rangle)$ können wir nun leicht als natürlich Zahl kodieren über

$$\underline{\kappa} = \left\langle \langle a_1, \ldots, a_m \rangle^* , i, \langle b_1, \ldots, b_n \rangle^* \right\rangle^3 ,$$

wobei B durch 2 interpretiert wird. Mit etwas Arbeit kann man zeigen, daß es eine primitiv-rekursive Funktion f gibt, so daß $f(\underline{\kappa}) = \underline{\kappa'}$ wenn $\kappa \vdash_M \kappa'$. □

Beweis.

...

Ist jetzt m die kleinste Zahl, so daß

Man sieht leicht, daß die Funktion $g : \mathbb{N} \to \mathbb{N}$, die jedem n die Kodierung der Startkonfiguration $\left\langle \langle\rangle^* , 1, \langle b_1, \ldots, b_m\rangle^* \right\rangle^3$ mit $\mathrm{bin}(n) = b_1 \ldots b_m$ zuordnet, primitiv-rekursiv ist. Um die Turingmaschine zu simulieren wenden wir also f iterativ so lange auf $g(n)$ an, bis wir ein Tripel erhalten, welches an der 2 Koordinate eine 0 stehen hat. Das heißt wenn die Rechnung akzeptiert liefert uns $\mu(f^k \circ g)^3_2(n)$ die genaue Anzahl der Schritte. Nennen wir diese *partielle* Funktion $\mathrm{st} : \mathbb{N} \rightharpoonup \mathbb{N}$. Offensichtlich ist st μ-rekursiv. □

Beweis.

. . .

Das Ergebnis (wenn die Turingmaschine hält) ist kodiert also

$$(f^{\mathrm{st}(n)}(n))_3^3 \ .$$

Auch hier sieht man leicht, daß es eine primitiv-rekursive Funktion gibt, die aus einem kodierten Ergebnis das korrekte Ergebnis extrahiert. Insgesamt folgern wir, daß jede Turing-berechenbare Funktion μ-rekursiv ist. □

Auch alle anderen Modelle der Berechenbarkeit, welche bis heute vorgeschlagen wurden, führen zur gleichen Funktionsklasse. Von nun an werden wir nur von *berechenbaren Funktionen* reden, ohne ein Modell zu spezifizieren (es sei denn es ist nötig für Details).
Diese Einsicht ist bekannt unter dem Namen

> *Church'sche These, Church-Turing These oder auch Church-Markov-Turing These.*

Man beachte, daß es nur eine These, aber kein beweisbares Theorem ist.[2]

[2] Wie auch sollte man diese Aussage beweisen? Da der Begriff der intuitiv berechenbaren Funktion nicht formalisiert ist; dies ist ja genau was man machen will.

Analysieren wir den letzten Beweis genauer, so sehen wir, daß sich jede berechenbare Funktion mit "Anwendung einer einzigen `WHILE` Schleife" darstellen lässt. Genauer

Satz (Kleenes Normalformtheorem)

Ist $f : \mathbb{N}^k \to \mathbb{N}$ *berechenbar, so gibt es primitiv rekursive Funktionen* h, π*, so daß gilt*

$$f = \pi \circ (\mu h)$$

Ohne Beweis (dessen Idee aber zumindest klar sein müsste)

Satz

Für eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ sind äquivalent:

- *f ist LOOP-berechenbar*
- *f ist primitiv-rekursiv.*