Formale Grundlagen der Informatik 2
(Theoretische Informatik)

(Skriptum WS 2002/2003)

Franz Winkler

Institut für Symbolisches Rechnen
(RISC-Linz)
Johannes Kepler Universität
Linz

1. Algorithmen

1.1. Was ist ein Algorithmus?

Der Begriff des Algorithmus ist in der Mathematik von alters her gebräuchlich. Intuitiv verstehen wir darunter ein Verfahren, welches von einer Anzahl von Grunddaten (Eingabe des Algorithmus) ausgeht und nach einer endlichen Anzahl wohldefinierter, effektiver Schritte eine Anzahl von Resultaten (Ausgabe des Algorithmus) liefert. Als Urahn aller Algorithmen wird vielfach der *Euklid'sche Algorithmus* zur Berechnung des größten gemeinsamen Teilers ganzer Zahlen angesehen. Man findet ihn bereits im 7. Buch der *"Elemente"* von Euklid ($\sim$ 300 v.Chr.), viele Gelehrte vermuten jedoch, daß es sich dabei eigentlich um Euklid's Fassung eines auf Eudoxus ($\sim$ 375 v.Chr.) zurückgehenden Algorithmus handelt.

Euklid'scher Algorithmus (moderne Fassung):
Zu zwei gegebenen nichtnegativen ganzen Zahlen m und n findet der Algorithmus ihren größten gemeinsamen Teiler (ggt).

(1) $[n = 0?]$ Ist $n = 0$, so endet der Algorithmus mit m als Resultat.

(2) $[m \pmod n]$ Setze $l \leftarrow m \pmod n$, $m \leftarrow n$, $n \leftarrow l$ und fahre mit Schritt (1) fort. ■

Die Korrektheit des Algorithmus sieht man unmittelbar aus der Beziehung

$$\mathrm{ggt}(m, n) = \mathrm{ggt}(n, m - q \cdot n),$$

wobei q eine beliebige ganze Zahl ist.

Als Beispiel dafür berechnen wir etwa den größten gemeinsamen Teiler der beiden Zahlen 40902 und 24140:

$$\begin{aligned}
\mathrm{ggt}(40902, 24140) &= \mathrm{ggt}(24140, 16762) = \mathrm{ggt}(16762, 7378) \\
&= \mathrm{ggt}(7378, 2006) = \mathrm{ggt}(2006, 1360) \\
&= \mathrm{ggt}(1360, 646) = \mathrm{ggt}(646, 68) \\
&= \mathrm{ggt}(68, 34) = \mathrm{ggt}(34, 0) \\
&= 34.
\end{aligned}$$

Im Folgenden wollen wir einige Aspekte des *informalen* Begriffs *"Algorithmus"* und *"durch einen Algorithmus berechenbare Funktion"* zusammenstellen. Oft beschränken wir uns auf Algorithmen, welche als Eingabe ein k–Tupel natürlicher Zahlen nehmen (für ein festes k) und als Ausgabe eine natürliche Zahl liefern. Ein Algorithmus berechnet also eine zahlentheoretische Funktion. Praktisch ist das keine Einschränkung, da alle interessanten und vernünftigen Ein– und Ausgabemengen in den natürlichen Zahlen kodiert werden können.

Es ist wichtig zu unterscheiden zwischen dem Begriff des *Algorithmus*, d.h. der Rechenvorschrift, Prozedur, und dem Begriff der *durch einen Algorithmus berechenbaren Funktion*, d.h. der durch eine Rechenvorschrift gegebenen Abbildung. Ein und dieselbe Funktion kann durch viele verschiedene Algorithmen gegeben sein.

Einige Beispiele für Funktionen, für die ein Algorithmus bekannt ist:

a. $\lambda x, y.x + y$.

b. $\lambda x.(x-$te Primzahl). Ein Algorithmus ist zum Beispiel das *Sieb des Eratosthenes*.

c. $\lambda x, y.\mathrm{ggt}(x, y)$. Dazu kann man den *Euklidschen Algorithmus* benutzen.

d. $\lambda x.$(die natürliche Zahl ≤ 9, die als x–te Ziffer in der Dezimaldarstellung von $\pi = 3.14159\ldots$ vorkommt). Etwa durch Quadratur des Kreises mittels der *Simpson Regel*.

Def. 1.1.1: (informal) Wir führen einige Eigenschaften an, welche man üblicherweise mit dem informalen Begriff des *Algorithmus* verbindet. Davon sind die Eigenschaften (1) — (5) praktisch von allen Mathematikern und Computerwissenschaftern als notwendig anerkannt, über (6) — (10) gibt es gelegentlich verschiedene Auffassungen.

(1) Ein Algorithmus besteht aus einer endlichen Anzahl von Anweisungen oder Instruktionen.

(2) Es gibt einen *Rechner*, Mensch oder Maschine, der diese Instruktionen ausführen kann.

(3) Es besteht die Möglichkeit, Rechenschritte abzuspeichern und aus dem Speicher wieder abzurufen.

(4) Sei P eine Menge von Instruktionen wie in (1) und R ein Rechner wie in (2). Dann besteht die Abarbeitung von P durch R in einem diskreten, schrittweisen

Rechenvorgang (keine analogen Methoden).

(5) Die Abarbeitung von P durch R ist deterministisch, ohne Bezugnahme auf Zufallsmethoden, z.B. Würfeln.

(6) Die Länge der Eingabe ist nach oben hin nicht beschränkt. (Die Theorie der Algorithmen beschäftigt sich mit der Frage der prinzipiellen Berechenbarkeit, ohne Bezugnahme auf praktische Einschränkungen.)

(7) Die Anzahl der Instruktionen ist nach oben hin nicht beschränkt. (Siehe (6).)

(8) Der verfügbare Speicherplatz ist nach oben hin nicht beschränkt. (Siehe (6).)

(9) Der Rechner R kann nur eine endliche Anzahl von Typen von Instruktionen verarbeiten.

(10) Eine a priori Schranke für die Anzahl der benötigten Rechenschritte muß nicht vorhanden sein. ■

Dieses intuitive Konzept eines Algorithmus bezieht seine Legitimität aus der vielfach geteilten Überzeugung, daß es möglich ist zu erkennen, ob ein Verfahren effektiv ist oder nicht. Das mag noch angehen, wenn es sich darum handelt von einem vorgelegten Verfahren zu entscheiden, ob es ein Algorithmus ist oder nicht. Die Intuition ist aber von geringem Nutzen, wenn es darum geht von einem gewissen Problem nachzuweisen, daß es keinen wie immer gearteten Algorithmus zu dessen Lösung gibt. So gibt es z.B. keinen Algorithmus, der für zwei beliebig vorgegebene Pascal–Programme entscheiden könnte, ob sie das gleiche Input–Output Verhalten haben! Probleme dieser Art treten sehr häufig in der theoretischen Informatik auf, und haben in den letzten Jahrzehnten besondere Aufmerksamkeit erlangt. Eine wissenschaftlich einwandfreie Definition des Begriffs "Algorithmus" ist also in höchstem Maße wünschenswert.

Ein wesentlicher Anstoß zur exakten Fassung des Begriffs "Algorithmus" kam von David Hilberts Programm, das das Ziel verfolgte, einen Algorithmus zu finden, der es gestatten würde, von jeder logischen Formel über den ganzen Zahlen, etwa

$$\neg\Big(\exists n, x, y, z\Big)\Big(\quad x^n + y^n = z^n,\ x \cdot y \cdot z \neq 0,\ n > 2\Big)$$

(Fermats Letzter Satz [1]) zu entscheiden, ob sie gültig ist oder nicht. Daß dieses

[1] Die Gültigkeit von Fermats Letztem Satz wurde in jüngster Zeit von Andrew Wiles bewiesen. Zuvor war diese Frage mehr als 300 Jahre lang ein offenes Problem.

Vorhaben zum Scheitern verurteilt war, wurde 1931 von Kurt Gödel [2] in seinem Unvollständigkeitssatz nachgewiesen:

In jeder Axiomatisierung der Theorie der ganzen Zahlen gibt es Formeln, deren Gültigkeit sich im System weder beweisen noch widerlegen läßt.

Um 1930 arbeiteten viele Mathematiker an diesem Hilbertschen Projekt, unter ihnen A.M. Turing, E.L. Post, A. Church. Verschiedene Modelle für das intuitive Konzept des Algorithmus wurden entwickelt: Turing–Maschinen von A.M. Turing [3], ein ähnliches Modell von E.L. Post, der λ–Kalkül von A. Church. Es stellte sich jedoch heraus, daß alle diese Modelle dieselbe Berechnungspotenz besitzen.

Im folgenden wollen wir verschiedene Annäherungen an den Begriff der Berechenbarkeit etwas näher betrachten.

[2] K. Gödel: *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I*, *Monatshefte f. Math. u. Physik*, 38, 171–198 (1931).

[3] A.M. Turing: *On computable numbers with an application to the Entscheidungsproblem*, *Proc. London Math. Soc.*, 2/42, 230–265 (1936). Korrektur, ibid., 43, 544–546.