DIDAKTISCHE ÜBERLEGUNGEN ZUR UNLÖSBARKEIT DES HALTEPROBLEMS UND SEINEN BEZUG ZUR RUSSELLSCHEN ANTINOMIE

Christian Maurer

Fachbereich Mathematik und Informatik
Freie Universität Berlin

24.9.96

Zusammenfassung. Die Unlösbarkeit des Halteproblems steht in einem starken formalen Bezug zur Russellschen Antinomie. Sowohl diese Gemeinsamkeit als auch der inhaltliche Unterschied sind auf einem Niveau formulierbar, das die Einsicht in dieses grundlegende Phänomen im Informatik-Unterricht am Gymnasium ermöglicht.

1. Die Unlösbarkeit des Halteproblems

Es gehört zur "Folklore" der Grundlagen der Informatik, daß das "Halteproblem" nicht entscheidbar ist. Ein Beweis dieses "Fundamentalsatzes der Unlösbarkeit" benutzt (siehe z.B. [GS] oder [Re]) folgendes Argument:

Nehmen wir an, es gäbe eine universelle Funktion `haelt_an`, die in der Lage ist, zu entscheiden, ob eine beliebige (durch ihren Quelltext gegebene) Funktion `F` für gegebene Daten `D` (in Form eines Textes) anhält, d.h. zu einem Ende kommt, oder nicht, etwa in Modula-2

```
PROCEDURE haelt_an (F, D: ARRAY OF CHAR): BOOLEAN;
```

(oder funktional, etwa in Miranda,

```
haelt_an:: [char] -> [char] -> bool).
```

1991 *Mathematics Subject Classification.* 03B25, 03E30, 03F03, 68N05.
1991 *CR Classification Scheme.* F.1.0, F.4.0, F.4.1. F.4.3.
Stichworte. Halteproblem, Entscheidbarkeit, Diagonalargument, Metasprache, Objektsprache, Russellsche Antinomie.

Wir betrachten dann die Funktion

```
PROCEDURE seltsam (F: ARRAY OF CHAR): BOOLEAN;
BEGIN
  IF haelt_an (F, F)
    THEN LOOP END;
         RETURN FALSE (* worauf man vergeblich wartet *)
    ELSE RETURN TRUE
  END
END seltsam;
```

(bzw. in Miranda

```
seltsam:: [char] -> bool
seltsam f = seltsam f, haelt_an f f
          = True, otherwise).
```

Diese Funktion hat die Eigenschaft, daß

(1.0) `seltsam (F)`

genau dann gilt (d.h. den Wert `TRUE` liefert), wenn

(1.1) `haelt_an (seltsam, F)`

gilt, also wenn sie ANHÄLT und den Wert `TRUE` liefert, was nach Konstruktion zu

(1.2) `NOT haelt_an (F, F)`

äquivalent ist, also dazu, daß die Funktion `F` NICHT ANHÄLT, falls ihr der eigene Quelltext als Parameter übergeben wird. Die entscheidende Beweisidee ist hierbei, daß der METASPRACHLICH formulierte Sachverhalt 1.0 sich in 1.1 OBJEKTSPRACHLICH formuliert wiederfindet.

Einsetzen von `seltsam` für `F` in 1.1 und 1.2 ergibt nun aber

(1.3) `haelt_an (seltsam, seltsam)`

und gleichzeitig

(1.4) `NOT haelt_an (seltsam, seltsam)`,

also einen Widerspruch:

Die Funktion `seltsam` mit ihrem eigenen Quelltext als Eingabe HÄLT GENAU DANN AN, WENN SIE NICHT ANHÄLT. Da dieser Widerspruch auf der Annahme der Existenz einer Entscheidungsfunktion `haelt_an` basiert, KANN ES EINE DERARTIGE FUNKTION NICHT GEBEN, dh. das “Halteproblem” ist nicht entscheidbar.

Insbesondere zeigt dieses Beispiel, daß es Probleme gibt, die aus prinzipiellen Gründen nicht – auch nicht unter Einsatz beliebig vieler unbegrenzt leistungsfähiger Rechner – lösbar sind.

2. Bezug zur Russellschen Antinomie

Eine Analyse der Grundidee dieser rein informatischen Beweisführung zeigt eine starke formale Analogie zur Russellschen Antinomie[1] (siehe [Ru] oder [H]) in der Mengenlehre, was wie folgt einzusehen ist:

Wir ersetzen wir die (durch ihre Quelltexte repräsentierten) Funktionen und die Daten(-texte), auf denen sie operieren, durch Mengen und das Prädikat `haelt_an` durch das mengentheoretische Symbol $\ni$, d.h.

`haelt_an (F, D)` (die Funktion `F` hält für die Daten `D` an)

durch

$D \in F$ (die Menge F enthält die Menge D als Element).

Die Funktion `seltsam` entspricht dann einer Menge S, die Aussage 1.1 der Aussage

$$F \in S \tag{2.1}$$

und 1.2 der Aussage

$$F \notin F, \tag{2.2}$$

womit sich $S = \{F \mid F \notin F\}$ ("Menge" aller derjenigen Mengen, die sich nicht selbst als Element enthalten) ergibt. Einsetzen von S für F in 2.1 und 2.2 liefert nun aber die Russellsche Antinomie (entsprechend 1.3 und 1.4):

$$S \in S \tag{2.3}$$

genau dann, wenn

$$S \notin S. \tag{2.4}$$

Allerdings besteht ein Unterschied zwischen der oben angeführten informatischen Argumentation und dieser Situation in der Mathematik:

In der Mengenlehre gibt es das Prädikat $\in$ sehr wohl, folglich muß die Bildung der "Supermenge" S ("Menge aller Mengen ... ") mangels Konsistenz abgelehnt werden.

Aufgelöst wird die Antinomie durch die Unterscheidung zwischen "Mengen" und "Klassen" ("Mengen" einer Meta-Ebene) im Rahmen einer axiomatischen Grundlegung einer Mengenlehre nach Zermelo-Fraenkel oder Neumann-Bernays-Gödel, in der die "Klasse aller Mengen" zulässig ist.

Dagegen ist die informatische Konstruktion von `seltsam` — relativ zur Existenz der Entscheidungsfunktion `haelt_an` — syntaktisch wie semantisch einwandfrei, woraus die Nichtexistenz eines solchen Prädikates folgt.

[1] oder, allgemeiner, zum Beweis, daß es nie eine surjektive Abbildung von einer Menge A auf ihre Potenzmenge $\mathcal{P}A$ geben kann (also auch keine injektive Abbildung von $\mathcal{P}A$ nach A): wenn $s\colon A \to \mathcal{P}A$ surjektiv wäre, hätte $\{a \in A \mid a \notin s(a)\} \in \mathcal{P}A$ ein Urbild $x \in A$ unter s, folglich wäre $a \in s(x)$ für alle $a \in A$ äquivalent zu $a \notin s(a)$, was für $a = x$ auf einen Widerspruch führt.

3. Didaktische Überlegungen

Die Erkenntnis, daß es grundsätzlich unlösbare Probleme gibt, ist so fundamental, daß sie sich – gerade aufgrund ihrer Formulierbarkeit mit elementaren Mitteln – als Gegenstand des Informatik-Unterrichts an der gymnasialen Oberstufe anbietet.

Voraussetzung für die Behandlung des Problems ist – abgesehen von Basiskenntnissen über eine Programmiersprache – lediglich ein sauberer Funktionsbegriff, wie er wohl im Laufe eines jeden ersten Unterrichtsjahres in Informatik bereitgestellt wird.

Die logische Spielerei mit dem Diagonal-Argument kann anhand der Bildung der Klasse aller Mengen, die sich nicht selbst als Element enthalten, oder einer geeigneten Antinomie vorbereitet werden; die Einsicht in die Legitimität der Konstruktion von `seltsam` ist eine einfache Übung zur Syntax von Funktionen in der verwendeten Programmiersprache.

Die Übertragung des vorgestellten Problemkreises auf Antinomien, wie sie schon in der Antike bekannt waren, liegt auf der Hand. Sie gibt Stoff für Aufgaben im Informatik-Unterricht, die der Lernzielsicherung (oder dem Einstieg in das Thema) dienen können.

Als Beispiel sei die klassische Antinomie des Barbiers angeführt: Wir betrachten die Funktion, die angibt, ob ein Herr `F` einen Herren `H` rasiert:

```
PROCEDURE rasiert (F, H: Herren): BOOLEAN.
```

und die naheliegende Funktion

(3.1)
```
PROCEDURE rasiert_sich_nicht_selbst (H: Herren): BOOLEAN;
BEGIN
  RETURN NOT rasiert (H, H)
END rasiert_sich_nicht_selbst.
```

Die Annahme, wir könnten diese Funktion durch einen `Barbier` vom Typ `Herren` repräsentieren, der alle diejenigen Herren rasiert, die sich nicht selbst rasieren:

(3.2)
```
rasiert_sich_nicht_selbst (H) = rasiert (Barbier, H)
```

führt durch Einsetzen von `Barbier` für `H` in 3.1 und 3.2 zu dem Paradoxon, daß sich der `Barbier`

(3.3) genau DANN selbst rasiert,

(3.4) wenn er sich NICHT selbst rasiert.

Im Paradigma der Russellschen Antinomie wird das Problem einfach dadurch gelöst, daß verboten wird, diesen `Barbier` zu betrachten, weil seine Einführung Bestandteil der metasprachlichen Ebene ist.

Im Paradigma des Halteproblems ist die Situation etwas schwieriger:

Der dortige Beweis basiert auf der Äquivalenz von 1.0 und 1.1, also der erwähnten Grundidee, metasprachliche Bestandteile des Beweises in der Objektsprache zu formulieren, was ermöglicht, daß AUF DIE FUNKTION `seltsam`, DIE SICH AUF DIE DEFINITION VON `haelt_an` STÜTZT, WIEDERUM `haelt_an` ANGEWENDET WERDEN KANN.

Auf das Beispiel des Barbiers läßt sich dieses Prinzip jedoch nicht ohne weiteres übertragen, weil der Typ `Herren` nicht mit dem DER FUNKTIONEN AUF `Herren` übereinstimmt. Durch eine Modifikation der bisherigen Betrachtungen läßt sich aber auch in diesem Paradigma das Halteproblem wiederfinden:

Wir unterstellen, es gäbe eine Funktion `rasiert_und_wird_fertig`, die entscheidet, ob ein Herr `A` einen Herrn `H` rasiert UND MIT DIESER RASUR JEMALS FERTIG WIRD, und betrachten, darauf aufbauend, die Funktion

```
PROCEDURE Barbier_rasiert (H: Herren): BOOLEAN;
BEGIN
  IF rasiert_und_wird_fertig (H, H)
    THEN LOOP (* endloser Streit zwischen Barbier und H *) END;
         RETURN FALSE (* was deshalb nicht passiert *)
    ELSE RETURN rasiert (Barbier, H)
  END
END Barbier_rasiert;
```

für die wir nun den zu 1.0 bis 1.2 vollständig analogen Sachverhalt haben:

`Barbier_rasiert (H)` gilt, d.h. liefert genau dann den Wert `TRUE`, wenn der

(4.0) `Barbier` den Herren `H` rasiert und damit irgendwann FERTIG WIRD,

also wenn `rasiert (Barbier, H)` und

(4.1) `rasiert_und_wird_fertig (Barbier, H)`.

Das ist aber nach Konstruktion von `Barbier_rasiert` äquivalent zu

(4.2) `NOT rasiert_und_wird_fertig (H, H)`.

Einsetzen von `Barbier` für `H` in 4.1 und 4.2 liefert das gleiche Desaster wie beim Halteproblem, folglich KANN DIE FUNKTION `rasiert_und_wird_fertig` NICHT EXISTIEREN.

An der vorgestellten Beispielklasse läßt sich vortrefflich demonstrieren, daß – nicht nur in der Mathematik, sondern auch in der Informatik – äußerst sorgfältig zwischen Sprache und Metasprache unterschieden werden muß, was die Logiker erst in unserem Jahrhundert in die Lage versetzt hat, jahrtausendealte Paradoxien aufzulösen.

UND WIE WAR DAS NUN MIT DEM KRETER [E], DER BEHAUPTET HABEN SOLL, ALLE KRETER SEIEN LÜGNER, UND ALLE AUSSAGEN VON KRETERN SEIEN BESTIMMT LÜGEN?

Literatur

[E] Epimenides aus Kreta, *in Athen verbreitetes Paradoxon* (etwa 400 v. Chr).

[GS] L. Goldschlager, A. Lister, *Informatik – Eine moderne Einführung*, Kapitel 3.1.3, Carl-Hanser-Verlag, München Wien, 1986, pp. 82–85.

[Re] P. Rechenberg, *Was ist Informatik?*, Kapitel 8.2, Carl-Hanser-Verlag, München Wien, 1991, pp. 159–163.

[H] B. Russell, *Letter to Frege (1902)*, From Frege to Gödel (J. v. Heijenoort, ed.), A Source Book in Mathematical Logic, 1879–1931, Harvard University Press, Cambridge (Mass.), 1967, pp. 124–125.

[Ru] B. Russell, *The Principles of Mathematics*, vol. 1, Cambridge University Press, Cambridge, 1903, pp. 101–107.

Institut für Informatik, Freie Universität Berlin, Takustr. 9, 14195 Berlin

E-mail address: maurer@inf.fu-berlin.de