Kapitel 9

Berechenbarkeit

Der Begriff der Berechenbarkeit wurde aus mathematischer Sicht erstmals um 1930 von Church, Gödel und Turing untersucht. In erster Linie wollten diese Logiker klären, ob die Gültigkeit bestimmter Formeln algorithmisch entschieden werden kann. Es stellte sich heraus, dass dies bei hinreichend ausdrucksstarken Formelklassen nicht der Fall ist. Zudem gibt es für hinreichend ausdrucksstarke Formelklassen keine volständigen Deduktionssysteme.

Um den Begriff der Berechenbarkeit mathematisch zu definieren, benötigt man ein mathematisches Berechnungsmodell. Wir werden dafür die Programmiersprache IMP benutzen. Zunächst werden wir zeigen, dass IMP die üblichen Datenstrukturen darstellen kann. Insbesondere kann IMP die Menge der Kommandos darstellen. Damit können wir Kommandos schreiben, die Kommandos als Daten verarbeiten (z.B. Interpretierer oder Übersetzer). Wir werden beweisen, dass semantische Eigenschaften von IMP-Programme nicht durch IMP-Programme entschieden werden können.

Der auf IMP basierende Berechenbarkeitsbegriff ist übrigens mit dem auf Turingmaschinen basierenden Berechenbarkeitsbegriff identisch.

Leseempfehlungen

- Neil D. Jones, Computability and Complexity, from a Programming Perspective. MIT Press, 1997.
- Christos H. Papadimitriou, Computational Complexity. Addison Wesley, 1995.
- Christos H. Papadimitriou, Turing (A Novel about Computation). The MIT Press, 2003.

9.1 Die natürlichen Zahlen als universelle Datenstruktur

Wenn wir eine Funktion $X \to Y$ durch ein Programm berechnen wollen, stehen wir zunächst vor der Aufgabe, die Mengen X und Y durch geeignete Datenstruktu-

ren zu realisieren. Es stellt sich heraus, dass alle Datenstrukturen auf die Datenstruktur der natürlichen Zahlen zurückgeführt werden können. Folglich können IMP-Programme mit beliebigen Datenstrukturen rechnen.

Die Universalität der natürlichen Zahlen folgt aus der Tatsache, dass die Primzahlzerlegung einer natürlichen Zahl eindeutig ist und berechnet werden kann. Die ganzen Zahlen können wir beispielsweise mit der injektiven Funktion

$$\#\in\mathbb{Z}\to\mathbb{N}$$
$$\#x = \mathit{if}\ x\geq 0\ \mathit{then}\ 3^x\ \mathit{else}\ 2\cdot 3^{-x}$$

darstellen. Die Menge der Tupel über $\mathbb{N}$ können wir mit der injektiven Funktion

$$\#\in\mathbb{N}^*\to\mathbb{N}$$
$$\#(x_1,\ldots,x_n) = p_0^n\cdot p_1^{x_1}\cdot\ldots\cdot p_n^{x_n}$$

darstellen, wobei p_k die k-te Primzahl bezeichnet. Beispielsweise gilt $p_0 = 2$ und $p_5 = 13$. Auch rekursive Datenstrukturen stellen kein Problem dar. Beispielsweise können wir Bäume über $\mathbb{N}$

$$T \stackrel{\text{def}}{=} \mathbb{N}+\mathbb{N}\times T\times T$$

wie folgt darstellen:

$$\#\in T\to\mathbb{N}$$
$$\#n = 2\cdot 3^n$$
$$\#(n,t_1,t_2) = 3^n\cdot 5^{\#t_1}\cdot 7^{\#t_2}$$

Mit IMP können wir einen unendlichen Speicher realisieren, der für jede der unendlich vielen Adressen in $\mathbb{N}$ eine natürliche Zahl speichern kann, allerdings mit der Einschränkung, dass nur an endlich vielen Adressen ein Wert ungleich 0 stehen darf. Den Zustand dieses Speichers stellen wir durch eine einzige natürliche Zahl dar. Dabei hat die Adresse k in einem Zustand z genau dann den Wert x, wenn z die Primzahl p_k genau x-mal enthält. Beispielsweise gilt für den Zustand $5^7\cdot 11^3$, dass die Adresse 2 den Wert 7, die Adresse 4 den Wert 3, und alle anderen Adressen den Wert 0 haben.

Auch Kommandos können durch natürliche Zahlen dargestellt werden. Im Folgenden gehen wir davon aus, dass Darstellungen für Tupel über $\mathbb{Z}$ und Kommandos gegeben sind. Die Darstellung eines Tupels $t\in\mathbb{Z}^*$ bezeichnen wir mit $\#t$, und die Darstellung eines Kommandos $c\in \mathit{Com}$ mit $\#c$. Das Symbol wird also überladen verwendet.

Es ist jetzt klar, dass IMP aus theoretischer Sicht ein äußerst leistungsfähiges Berechnungsmodell ist. Anders als konkrete Computer arbeitet das Berechnungsmodell IMP mit einem beliebig großen Speicher.

Die Universalität der natürlichen Zahlen wurde zuerst von Kurt Gödel erkannt. Turings Berechnungsmodell verwendet statt natürlicher Zahlen Zeichenreihen als universelle Datenstruktur. Auch heutige Computer verwenden Zeichenreihen als universelle Datenstruktur, wobei sie sich auf nur zwei Zeichen beschränken.

9.2 Berechenbare Funktionen

Sei $\mathbb{Z}_\perp \stackrel{\text{def}}{=} \mathbb{Z} \cup \{\perp\}$, wobei $\perp$ ein einmal gewähltes Objekt mit $\perp \notin \mathbb{Z}$ ist. Unter einer berechenbaren Funktion wollen wir eine Funktion $\mathbb{Z} \to \mathbb{Z}_\perp$ verstehen, die sich mit einem IMP-Kommando berechnen lässt. Dabei bedeutet das Ergebnis $\perp$, dass das berechnende Kommando divergiert. Die Funktion, die zu einem Kommando die durch das Kommando berechnete Funktion liefert, definieren wir mithilfe einer einmal gewählten Lokation X_0, über die wir die Eingabe und Ausgabe des Kommandos kommunizieren:

$$\begin{aligned}
&\mathcal{F} \in \mathit{Com} \to \mathbb{Z} \to \mathbb{Z}_\perp \\
&\mathcal{F}cx \;=\; \text{let } s = Cc(\sigma_0[X_0 := x]) \text{ in if } s = \perp \text{ then } \perp \text{ else } sX_0 \\
&\qquad\quad \text{wobei } \sigma_0 = \lambda X \in \mathit{Loc}.\ 0
\end{aligned}$$

Damit ergibt sich die **Menge der berechenbaren Funktionen** wie folgt:

$$BF \stackrel{\text{def}}{=} \{\, \mathcal{F}c \mid c \in \mathit{Com} \,\}$$

Ein Kommando c heißt **reinlich**, wenn für alle $x \in \mathbb{Z}$ gilt:

$$\mathcal{F}cx \neq \perp \Longrightarrow \exists y \in \mathbb{Z}\colon\ Cc(\sigma_0[X_0 := x]) = \sigma_0[X_0 := y]$$

Proposition 9.2.1 Jede berechenbare Funktion kann durch ein reinliches Kommando berechnet werden.

Beweis Zu jedem Kommando erhält man ein reinliches Kommando, indem man dem Kommando Zuweisungen anfügt, die die vom Kommando benutzten und von X_0 verschiedenen Lokationen auf 0 setzen. ∎

Proposition 9.2.2 Seien f und g berechenbare Funktion. Dann ist die „Komposition“ $f \circ_\perp g = \lambda x \in \mathbb{Z}.\ \mathit{if}\ fx = \perp\ \mathit{then} \perp \mathit{else}\ g(fx)$ berechenbar.

Beweis Sei c_f ein reinliches Kommando, das f berechnet, und c_g ein Kommando, das g berechnet. Dann berechnet $c_f; c_g$ die Komposition $f \circ_\perp g$. ∎

Aufgabe 9.1 Zeigen Sie, dass die Funktionen $\lambda x \in \mathbb{Z}.\ x$ und $\lambda x \in \mathbb{Z}.\ \perp$ berechenbar sind. ∎

9.3 Universelle Kommandos

Da wir Kommandos als Zahlen darstellen können, können wir in IMP einen Interpreter für IMP schreiben. Die Idee eines solchen Interpreters geht auf Turing zurück. Das den Interpretierer realisierende Kommando bezeichnet man als **universelles Kommando**.

Satz 9.3.1 (Universelles Kommando) Es gibt ein Kommando c_u wie folgt:

$$\forall c \in \mathit{Com}\ \forall x \in \mathbb{Z}\colon\ \mathcal{F}c_u(\#(\#c, x)) = \mathcal{F}cx$$

Man kann noch einen Schritt weitergehen und einen gesteuerten Interpreter betrachten. Dieser bekommt zusätzlich zu c und x eine als *Reichweite* bezeichnete Zahl y als Eingabe, die vorgibt, wieviele Schleifendurchläufe bei der Ausführung von c erlaubt sind. Wenn die Ausführung von c für x weniger als y Schleifendurchläufe erfordert, liefert der gesteuerte Interpreter das Ergebnis, ansonsten terminiert er mit einer Ausnahme. Die Ausnahme realisieren wir durch das Ergebnis 0 und reguläre Ergebnisse stellen wir durch Zahlen verschieden von 0 dar.

Satz 9.3.2 (Gesteuertes universelles Kommando) Es gibt ein Kommando c_v, sodass für alle $c \in \mathit{Com}$ und $x \in \mathbb{Z}$ gilt:

1. $\mathcal{F}cx = \bot \iff \forall y \in \mathbb{Z}\colon\ \mathcal{F}c_v(\#(\#c, x, y)) = 0$
2. $\mathcal{F}cx \neq \bot \iff \exists y \in \mathbb{Z}\colon\ \mathcal{F}c_v(\#(\#c, x, y)) = \mathit{if}\ \mathcal{F}cx \geq 0\ \mathit{then}\ 1 + \mathcal{F}cx\ \mathit{else}\ \mathcal{F}cx$

Aufgabe 9.2 Warum existiert eine endliche Menge $L \subseteq \mathit{Loc}$, sodass jede berechenbare Funktion durch ein Kommando berechnet werden kann, das nur Lokationen aus L verwendet? ■

9.4 Prüfbare und entscheidbare Mengen

Unter einer **Darstellung einer Menge** X verstehen wir eine injektive Funktion $\# \in X \to \mathbb{Z}$. Unter einem **Problem** verstehen wir ein Tupel $(X, \#, M)$ aus einer Menge X, einer Darstellung $\#$ von X und einer Menge $M \subseteq X$. Ein Problem $(X, \#, M)$ heißt

- **prüfbar**,[1] wenn $\exists f \in \mathit{BF}\ \forall x \in X\colon\ x \in M \iff f(\#x) \neq \bot$.
- **entscheidbar**, wenn $\exists f \in \mathit{BF}\ \forall x \in X\colon\ f(\#x) \in \{0,1\} \wedge (x \in M \iff f(\#x) = 1)$.
- **unentscheidbar**, wenn es nicht entscheidbar ist.

[1] Prüfbarkeit wird oft auch als Semientscheidbarkeit bezeichnet. In der Literatur findet man auch den Begriff *recursively enumerable (r.e.)*, der eine zur Prüfbarkeit äquivalente Eigenschaft bezeichnet.

Ein Kommando c heißt **Prüfer** für ein Problem $(X, \#, M)$, wenn für alle $x \in X$ gilt: $x \in M \iff \mathcal{F}c(\#x) \neq \bot$. Ein Problem ist genau dann prüfbar, wenn es einen Prüfer für das Problem gibt. Der Begriff des **Entscheiders** ist analog definiert.

Im Folgenden ergibt sich die Grundmenge X und die Darstellung $\#$ eines Problems $(X, \#, M)$ meist aus dem Kontext. Wir beschreiben dann das Problem $(X, \#, M)$ einfachheitshalber nur durch seine Menge M und sagen, dass die Menge M prüfbar oder entscheidbar ist. Für die Grundmenge $\mathbb{Z}$ werden wir stets die Identitätsfunktion $\lambda x \in \mathbb{Z}.\ x$ als Darstellung verwenden. Darstellungen werden wir generell mit dem Symbol $\#$ bezeichnen, das wir überladen verwenden werden.

Satz 9.4.1 $M \subseteq X$ ist entscheidbar genau dann, wenn M und $X - M$ prüfbar sind.

Beweis Sei $\# \in X \to \mathbb{Z}$ eine Darstellung der Menge X. Wir betrachten die Probleme $(X, \#, M)$ und $(X, \#, X - M)$.

Aus einem Entscheider für X können wir leicht Prüfer für M und $X - M$ konstruieren (Übungsaufgabe!).

Wenn wir Prüfer für M und $X - M$ haben, können wir mithilfe des gesteuerten universellen Kommandos einen Entscheider für M konstruieren. Der Entscheider lässt beide Prüfer für schrittweise erhöhte Reichweite laufen, bis einer der beiden Prüfer Erfolgt meldet. ■

Aufgabe 9.3 Sei eine Darstellung für X gegeben und seien M, N entscheidbare Teilmengen von X. Beweisen Sie:

a) $M \cap N$ ist entscheidbar.

b) $M \cup N$ ist entscheidbar.

c) $M - N$ ist entscheidbar. ■

Aufgabe 9.4 Sei eine Darstellung für X gegeben und seien M, N prüfbare Teilmengen von X. Beweisen Sie:

a) $M \cap N$ ist prüfbar.

b) $M \cup N$ ist prüfbar. ■

Aufgabe 9.5 Sei $M \subseteq E \subseteq X$ und sei E entscheidbar. Beweisen Sie: $X - M$ ist genau dann prüfbar, wenn $(X - M) \cap E$ prüfbar ist. ■

9.5 Reduktionen

Seien $(X, \#, M)$ und $(Y, \S, N)$ Probleme. Eine **Reduktion** $(X, \#, M) \to (Y, \S, N)$ ist eine Funktion $f \in X \to Y$, für die eine berechenbare Funktion $g \in \mathbb{Z} \to \mathbb{Z}_\bot$ existiert, sodass für alle $x \in X$ gilt: $x \in M \iff fx \in N$ und $\S(fx) = g(\#x)$.

Proposition 9.5.1 Seien $(X,\#,M)$ und $(Y,\S,N)$ Probleme. Eine Funktion $f \in X \to Y$ ist genau dann eine Reduktion $(X,\#,M) \to (Y,\S,N)$, wenn sie eine Reduktion $(X,\#,X-M) \to (Y,\S,Y-N)$ ist.

Die folgende Proposition besagt, dass Berechenbarkeitseigenschaften von Problemen mithilfe von Reduktionen bewiesen werden können. Wenn man beispielsweise zeigen will, dass ein Problem P nicht prüfbar ist, genügt es, eine Reduktion $P' \to P$ für ein bereits als nicht prüfbar erkanntes Problem P' anzugeben.

Proposition 9.5.2 Sei f eine Reduktion $P \to Q$. Dann:

1. Q prüfbar $\Longrightarrow$ P prüfbar.
2. Q entscheidbar $\Longrightarrow$ P entscheidbar.
3. P nicht prüfbar $\Longrightarrow$ Q nicht prüfbar.
4. P unentscheidbar $\Longrightarrow$ Q unentscheidbar.

Beweis Sei f eine Reduktion $(X,\#,M) \to (Y,\S,N)$.

Wir beginnen mit Behauptung (1). Sei c_N ein Prüfer für N. Weiter sei c_f ein reinliches Kommando mit $\forall x \in X\colon\ \S(fx) = \mathcal{F}c_f(\#x)$. Dann gilt für alle $x \in X$:

$$x \in M \iff fx \in N \in \mathcal{F}c_N(\S(fx)) \neq \bot \iff \mathcal{F}(c_f;c_N)(\#x) \neq \bot$$

Also ist $c_f;c_N$ ein Prüfer für M.

Behauptung (2) ergibt sich aus Behauptung (1) mithilfe von Proposition 9.5.1 und Satz 9.4.1. Behauptungen (3) und (4) folgen mit Kontraposition aus (1) und (2). ■

9.6 Unentscheidbarkeit von Programmeigenschaften

Es stellt sich heraus, dass semantische Programmeigenschaften grundsätzlich unentscheidbar sind. Diese für den Uneingeweihten völlig überraschende Tatsache wurde 1953 von Rice gezeigt. Etwa 20 Jahre früher konnte Turing bereits zeigen, dass das sogenannte Halteproblem unentscheidbar ist (Divergiert eine Turingmaschine für eine bestimmte Eingabe?). Diese Ergebnisse sind für die Logik und die Informatik von fundamentaler Bedeutung. Sie können relativ einfach bewiesen werden.

Proposition 9.6.1 (Halteproblem) $\{\, c \in \mathit{Com} \mid \mathcal{F}c(\#c) = \bot \,\}$ ist nicht prüfbar.

Beweis Sei p eine Variable des Typs $X \to X \to \mathbb{B}$. Dann gilt die Gleichung $\exists x \forall y.\ pxy \Leftrightarrow \neg(pyy) = 0$. Also gibt es kein Kommando c_0, sodass für alle Kommandos c gilt: $\mathcal{F}c_0(\#c) \neq \bot \iff \mathcal{F}c(\#c) = \bot$. Also existiert kein Prüfer für $\{\, c \in \mathit{Com} \mid \mathcal{F}c(\#c) = \bot \,\}$. ■

Die Unentscheidbarkeit des Halteproblems „terminiert ein Kommando auf seiner Darstellung?“ folgt also unmittelbar aus der Tautologie[2] $\neg\exists x\forall y.\ pxy \Leftrightarrow \neg(pyy)$. Dabei spielt das zugrundeliegende Berechnungsmodell keine Rolle, bis auf die Annahme, dass Programme als Daten darstellbar sind, die von Programmen bearbeitet werden können.

Mithilfe von Reduktionen können jetzt weitere Mengen als nicht prüfbar bewiesen werden. Hier ist ein Beispiel:

Proposition 9.6.2 $\{ c \in Com \mid \mathcal{F}c0 = \bot \}$ ist nicht prüfbar.

Beweis Es genügt, eine Reduktion

$$\{ c \in Com \mid \mathcal{F}c(\#c) = \bot \} \to \{ c \in Com \mid \mathcal{F}c0 = \bot \}$$

anzugeben. Für alle Kommandos c gilt:

$$\mathcal{F}c(\#c) = \bot \iff \mathcal{F}(X_0 := \#c; c)0 = \bot$$

Also ist $\lambda c.\ X_0 := \#c; c$ eine Reduktion wie gefordert. ∎

Aufgabe 9.6 Beweisen Sie, dass die folgenden Mengen nicht prüfbar sind:

a) $\{ c \in Com \mid \exists x \in \mathbb{Z}\colon \mathcal{F}cx = \bot \}$

b) $\{ c \in Com \mid \forall x \in \mathbb{Z}\colon \mathcal{F}cx = \bot \}$

c) $\{ c \in Com \mid \mathcal{F}ca = \bot \}$ wobei $a \in \mathbb{Z}$

d) $\{ (c,x) \in Com \times \mathbb{Z} \mid \mathcal{F}cx = \bot \}$

e) $\{ (c,x) \in Com \times \mathbb{Z} \mid \mathcal{F}cx = \bot \wedge x \leq 0 \}$ ∎

Wir betrachten jetzt die Menge aller Kommandos, die für alle Eingaben terminieren:

$$T = \{ c \in Com \mid \forall x \in \mathbb{Z}\colon\ \mathcal{F}cx \neq \bot \}$$

Satz 9.6.3 Weder T und $Com - T$ ist prüfbar.

Beweis Es genügt zu zeigen, dass das Halteproblem sowohl auf T und $Com - T$ reduziert werden kann.

Wir beginnen mit $Com - T = \{ c \in Com \mid \exists x \in \mathbb{Z}\colon \mathcal{F}cx = \bot \}$. Für alle $c \in Com$ gilt:

$$\mathcal{F}c(\#c) = \bot \iff \exists x \in \mathbb{Z}\colon\ \mathcal{F}(X_0 := \#c; c)x = \bot$$

Also reduziert $\lambda c.\ X_0 := \#c; c$ das Halteproblem auf $Com - T$.

[2] Wir benutzen den Begriff Tautologie hier mit etwas anderer Bedeutung als im Kapitel über Aussagenlogik. Hier verstehen wir unter einer Tautolgie eine Aussage, die allein kraft ihrer logischen Form wahr ist.

Wir betrachten jetzt T. Sei c_v ein gesteuertes universelles Kommando und sei loop ein Kommando, das immer divergiert. Dann gilt für alle $c \in Com$:

$$\mathcal{F}c(\#c) = \bot \iff$$
$$\forall x \in \mathbb{Z}\colon\ \mathcal{F}(X_0 := \#(\#c, \#c, X_0);\ c_v;\ \text{if } X_0 = 0 \text{ then skip else loop})x \neq \bot$$

Damit haben wir eine Reduktion des Halteproblems auf T. Die Idee für diese Reduktion kam von David Steurer, 10. Juli 2004. ■

Auch der Beweis des Satzes von Rice beruht auf einer Reduktion, die mithilfe eines universellen Kommandos konstruiert ist.

Satz 9.6.4 (Rice) Sei F eine Menge von berechenbaren Funktionen $\mathbb{Z} \to \mathbb{Z}_\bot$, die die Funktion $\lambda x \in \mathbb{Z}.\ \bot$ enthält und mindestens eine berechenbare Funktion nicht enthält. Dann ist $\{ c \in Com \mid \mathcal{F}c \in F \}$ nicht prüfbar.

Beweis Es genügt, eine Reduktion

$$\{ c \in Com \mid \mathcal{F}c(\#c) = \bot \} \to \{ c \in Com \mid \mathcal{F}c \in \mathcal{F} \}$$

anzugeben. Sei $f \in BF - F$. Dann gilt für alle $c \in Com$:

$$\mathcal{F}c(\#c) = \bot \iff (\mathit{if}\ \mathcal{F}c(\#c) = \bot\ \mathit{then}\ \lambda x \in \mathbb{Z}.\ \bot\ \mathit{else}\ f) \in F$$

Wir konstruieren jetzt zu c ein Kommando, das die Funktion auf der rechten Seite der Äquivalenz berechnet. Sei c_f ein Kommando mit $\mathcal{F}c_f = f$. Sei c_u ein reinliches universelles Kommando. Sei X_s eine Lokation, die nicht in c_u vorkommt. Dann gilt für alle $c \in Com$:

$$\mathcal{F}c(\#c) = \bot \iff \mathcal{F}(X_s := X_0;\ X_0 := \#(\#c, \#c);\ c_u;\ X_0 := X_s;\ c_f) \in F$$

Die Richtung „$\Longrightarrow$“ ist klar. Die andere Richtung zeigen wir durch Kontraposition. Sei also $\mathcal{F}c(\#c) \neq \bot$. Dann berechnet $X_s := X_0;\ X_0 := \#(\#c, \#c);\ c_u;\ X_0 := X_s$ die Identitätsfunktion. Also berechnet das Kommando auf der rechten Seite der Äquivalenz die Funktion $f \notin F$.

Folglich ist $\lambda c.\ X_s := X_0;\ X_0 := \#(\#c, \#c);\ c_u;\ X_0 := X_s;\ c_0$ eine Reduktion wie gefordert. ■

Unter einer semantischen Eigenschaft eines Kommandos c wollen wir eine Eigenschaft der duch c berechneten Funktion verstehen. Formal können wir eine semantische Eigenschaft von Kommandos durch die Menge aller berechenbaren Funktionen darstellen, die die Eigenschaft erfüllen. Aus dem Satz von Rice folgt, dass jede nichttriviale semantische Eigenschaft unetscheidbar ist:

Korollar 9.6.5 Sei F eine nichtleere Menge von berechenbaren Funktionen $\mathbb{Z} \to \mathbb{Z}_\perp$, die mindestens eine berechenbare Funktion nicht enthält. Dann ist $\{ c \in Com \mid \mathcal{F}c \in F \}$ nicht entscheidbar.

Beweis Entweder F oder $BF - F$ enthält $\lambda x \in \mathbb{Z}.\ \perp$. Also folgt mit dem Satz von Rice, dass entweder $\{ c \mid \mathcal{F}c \in F \}$ oder $\{ c \mid \mathcal{F}c \notin F \}$ nicht prüfbar ist. Also ist $\{ c \mid \mathcal{F}c \in F \}$ unentscheidbar. ∎

Die folgende Proposition ist ein Beispiel für die Anwendung des Satzes von Rice. Sie stellt fest, dass sematische Programmäquivalenz nicht prüfbar ist.

Proposition 9.6.6 $\{ (c, c') \in Com^2 \mid \mathcal{F}c = \mathcal{F}c' \}$ ist nicht prüfbar.

Beweis Die Funktion $\lambda c.\ (c,$ `while true do skip`$)$ ist eine Reduktion

$$\{ c \in Com \mid \mathcal{F}c \in \{\lambda x \in \mathbb{Z}.\ \perp\} \} \to \{ (c, c') \in Com^2 \mid \mathcal{F}c = \mathcal{F}c' \}$$

Mit dem Satz von Rice folgt, dass die linke Menge nicht prüfbar ist. Also ist auch die rechte Menge nicht prüfbar. ∎

Aufgabe 9.7 Beweisen Sie:

a) Für alle $a, b \in \mathbb{Z}$: $\{ c \in Com \mid \mathcal{F}ca = b \}$ nicht entscheidbar.

b) $\{ c \in Com \mid \{ x \in \mathbb{Z} \mid \mathcal{F}cx \neq \perp \}$ endlich $\}$ nicht prüfbar. ∎

9.7 Church-Turing-These

Wir haben die Begriffe Entscheidbarkeit und Prüfbarkeit mithilfe des Berechnugsmodells IMP definiert. Es stellt sich die Frage, ob man dieselben entscheidbaren und prüfbaren Mengen bekommt, wenn man ein anderes Berechnungsmodell wie zum Beispiel Turingmaschinen zugrunde legt. Es stellt sich heraus, dass man genau dieselben Mengen bekommt. Diese Tatsache ist zunächst völlig überraschend. In den dreißiger Jaren des zwanzigsten Jahrhunderts haben Logiker wie Church, Gödel und Turing viele verschiedene Berechnungsmodelle untersucht. Sie konnten zeigen, dass sich jeweils dieselben prüfbaren und entscheidbaren Mengen ergeben. Der damit verbundene Berechenbarkeitsbegriff wird als **Turing-Berechenbarkeit** bezeichnet. Die Annahme, dass die damit erhaltenen prüfbaren und entscheidbaren Mengen den intuiven Begriff von algorithmischer Berechenbarkeit korrekt formalisieren, bezeichnet man als **Church-Turing-These**.[3]

[3] Unter plato.stanford.edu/entries/church-turing finden Sie historische und philosophische Betrachtungen zur Church-Turing-These.

9.8 Unentscheidbare arithmetische Probleme

Auf der Mathematikertagung in Paris formulierte David Hilbert im Jahr 1900 das folgende Problem, das heute als „Hilberts 10. Problem“ bekannt ist:

> Gibt es einen Algorithmus, der entscheidet, ob eine polynomielle Gleichung mit ganzahligen Koeffizienten (z.B. $x^2y + 3yz - y^2 - 17 = 0$) eine ganzzahlige Lösung hat?

Als Hilbert das Problem formulierte, ging die mathematische Welt davon aus, dass ein solcher Algorithmus existiert. Kurt Gödel, ein Schüler Hilberts, konnte 1930 zeigen, dass zumindest für komplexere arithmetische Probleme kein Entscheidungsalgorithmus existiert. Das führte zu einer radikalen Änderung der mathematischen Weltsicht. Weitere 40 Jahre vergingen, bis der russische Mathematiker Yuri Matiyasevich 1970 zeigen konnte, dass auch für Hilberts 10. Problem kein Entscheidungalgorithmus existiert.

Wir wollen die Sätze von Matiyasevich und Gödel präzise formulieren. Dazu definieren wir arithmetische Ausdrücke und arithmetische Formeln wie folgt:

$x \in Var = \mathbb{N}$	Variablen
$k \in \mathbb{Z}$	Konstanten
$e \in AE = x \mid k \mid e + e \mid e \cdot e$	Arithmetische Ausdrücke
$A \in AF = e = e \mid \neg A \mid A \wedge A \mid \exists x A$	Arithmetische Formeln

Die Denotationsfunktionen für arithmetische Ausdrücke $AE \to (Var \to \mathbb{Z}) \to \mathbb{Z}$ und arithmetische Formeln $AF \to (Var \to \mathbb{Z}) \to \mathbb{Z}$ seien wie üblich definiert.

Satz 9.8.1 (Matiyasevich, 1970) Die Menge $\{ e \in AE \mid \forall s \in Var \to \mathbb{Z}\colon \mathcal{D}es \neq 0 \}$ der arithmetischen Ausdrücke, die für keine Variablenbelegung den Wert 0 annehmen, ist nicht prüfbar.

Der Beweis dieses Satzes ist jenseits unserer Möglichkeiten. Ausgehend vom Satz von Matiyasevich ist es einfach, den Gödelschen Unvollständigkeitssatz zu beweisen.

Eine arithmetische Formel A heißt **gültig**, wenn $\forall s \in Var \to \mathbb{Z}\colon\ \mathcal{D}As = 1$.

Korollar 9.8.2 (Gödelscher Unvollständigkeitssatz, 1930) Die Menge der gültigen arithmetischen Formeln ist nicht prüfbar.

Beweis Wieder genügt es, eine Reduktion der Matiyasevich-Menge anzugeben:

$$\{ e \in AE \mid \forall s \in Var \to \mathbb{Z}\colon\ \mathcal{D}es \neq 0 \} \ \to\ \{ A \in AF \mid A \text{ gültig} \}$$

Das ist einfach: $\lambda e.\ \neg(e = 0)$ ist eine solche Reduktion. ∎

Aufgabe 9.8 Zeigen Sie, dass die Menge der ungültigen arithmetischen Fomeln nicht prüfbar ist. ■

Der Satz von Gödel hat die wichtige Konsequenz, dass es für die Menge der gültigen arithmetischen Formeln kein vollständiges Deduktionssystem geben kann. Das liegt daran, dass Deduktionssysteme per Definition immer so konstruiert sein müsssen, dass man die ableitbaren Formeln algorithmisch aufzählen kann. Das bedeutet aber gerade, dass die ableitbaren Formeln immer eine prüfbare Menge darstellen.

Mithilfe einer Technik, die als Quantorenelimination bezeichnet wird, konnte Mojzesz Presburger 1929 zeigen, dass die Menge der gültigen arithmetischen Formeln entscheidbar ist, wenn man nur Formeln ohne Multiplikation betrachtet.

Satz 9.8.3 (Presburger, 1929) Die Menge der gültigen arithmetischen Formeln ohne Multiplikation ist entscheidbar.

9.9 Abzählbare Mengen

Dieser Abschnitt gehört eigentlich ins Kapitel Mengenlehre

Der Begriff der abzählbaren Menge geht auf Cantor zurück. Cantor zeigte, dass es nicht abzählbare Mengen gibt (Cantorscher Diagonalschluss). Der Begriff der Abzählbarkeit interessiert uns, da syntaktisch beschreibare Mengen immer abzählbar sind.

Eine Menge M heißt **abzählbar**, wenn es eine injektive Funktion $M \to \mathbb{N}$ gibt. Jede endliche Menge ist abzählbar. Auch $\mathbb{N}$ und $\mathbb{Z}$ sind abzählbar.

Informell gesprochen ist eine Menge abzählbar, wenn sie höchstens soviele Elemente enthält wie die Menge der natürlichen Zahlen.

Eine Menge heißt **überabzählbar**, wenn sie nicht abzählbar ist.

Proposition 9.9.1 Seien X und Y zwei nichtleere Mengen. Dann existiert eine injektive Funktion $X \to Y$ genau dann, wenn eine surjektive Funktion $Y \to X$ existiert.

Proposition 9.9.2 Eine nichtleere Menge M ist genau dann abzählbar, wenn es eine surjektive Funktion $\mathbb{N} \to M$ gibt.

Proposition 9.9.3 Der Schnitt, die Vereinigung und die Differenz von zwei abzählbaren Mengen sind abzählbar.

Beweis Wir zeigen nur, dass die Vereinigung von zwei abzählbaren Mengen abzählbar ist. Die anderen Behauptungen folgen mit ähnlichen Argumenten.

Seien $f \in M \to \mathbb{N}$ und $f' \in M' \to \mathbb{N}$ injektive Funktionen. Dann ist

$h \in M \cup M' \to \mathbb{N}$
$h(x) = \text{if } x \in M \text{ then } 2 \cdot f(x) \text{ else } 2 \cdot f'(x) + 1$

eine injektive Funktion. ■

Jede darstellbare Menge ist abzählbar. Folglich ist auch die Menge *Com* der Kommandos abzählbar. Also ist auch die Menge der berechenbaren Funktionen abzählbar.

Proposition 9.9.4 Die Menge der berechenbaren Funktionen ist abzählbar.

Proposition 9.9.5 Die Menge $\mathbb{Z} \to \mathbb{Z}_\perp$ ist überabzählbar.

Beweis Durch Widerspruch. Sei $\alpha \in \mathbb{N} \to (\mathbb{Z} \to \mathbb{Z}_\perp)$ surjektiv. Wir definieren

$f \in \mathbb{Z} \to \mathbb{Z}_\perp$
$fx = \text{if } X < 0 \text{ then } 0 \text{ else } \alpha x x + 1$

Offensichtlich gilt

$\forall n \in \mathbb{N}\colon\ \alpha n n \neq fn$

Das ist ein Widerspruch zu der Annahme, dass α surjektiv ist. ■

Die gerade verwendete Beweistechnik wird als **Cantorscher Diagonalschluss** bezeichnet. Als Cantor diesen Beweis erstmals vorlegte, sorgte er in der mathematischen Welt für einige Verblüffung.

Wir wissen jetzt, dass es überabzählbar viele Funktionen $\mathbb{Z} \to \mathbb{Z}_\perp$ gibt, die nicht berechenbar sind (Übung: Warum?).