Repräsentierbarkeit in der Theorie der Arithmetik

Christoph Kreitz

Institut für Informatik, Universität Potsdam, 14482 Potsdam

Zusammenfassung *Dieser Artikel gibt einen Überblick über die arithmetische Repräsentierbarkeit berechenbarer Funktionen und ihre Konsequenzen für die Logik. Er ist gedacht als Ergänzung der sehr knappen Abhandlung dieser Thematik in der Vorlesung "Einführung in die Theoretische Informatik II".*

1 Übersicht

Neben der Ausführung von Befehlen auf einem Speicher, der Definition und Anwendung von Funktionen, oder dem Zusammensetzen berechenbarer Elementarfunktionen kann man Berechenbarkeit auch über die Ausführung von Beweisprozessen definieren. Dies ist insbesondere dann sinnvoll, wenn die Berechnung symbolischer Natur ist, also z.B. Programmtransformationen vornimmt, mathematisch-technische Formeln analysiert, oder Schlußfolgerungen aus vorhandenen Daten ziehen soll. Für derartige Anwendungen sind *logische Programmiersprachen* wie z.B. Prolog besser geeignet als andere, da sie Infrastrukturen bereitstellen, die in anderen Programmiersprachen erst vom Anwendungsentwickler selbst geschaffen werden müssen.

Logische Programmiersprachen verwenden Formeln eines formalen logischen Kalküls um das Ein-/Ausgabeverhalten von Funktionen zu beschreiben. Dabei ist es eigentlich irrelevant, welche Variablen der Eingaben und welche die vorgesehene Ausgabe repräsentieren, denn die Formel beschreibt nur den logischen Zusammenhang zwischen den verschiedenen Variablen, liefert also eine logische Spezifikation der zu implementierenden Funktion.

Beispiel 1 (Logische Repräsentation der Subtraktion)
Die Subtraktionsfunktion $sub : \mathbb{N}^2 \to \mathbb{N}$ *ist definiert durch* $sub(x_1, x_2) = x_1 \dot{-} x_2$. *Wir wollen schrittweise eine logische Spezifikation dieser Funktion herleiten. Hierzu nutzen wir aus, daß die Subtraktion normalerweise als Umkehrung der Addition angesehen wird. Für das Ergebnis* y *der Subtraktion gilt also* $x_1 - x_2 = y \Leftrightarrow x_1 = x_2 + y$.

Dieser Zusammenhang gilt jedoch nur auf den ganzen Zahlen, bei denen negative Zahlen als Funktionsergebnisse zugelassen sind. Auf den natürlichen Zahlen gilt jedoch $x_1 - x_2 = 0$ *wenn* $x_1 < x_2$ *ist. Es ist offensichtlich, daß in diesem Fall die Gleichung* $x_1 = x_2 + y$ *nicht mehr korrekt ist, denn diese würde* $x_1 = x_2$ *verlangen, obwohl* x_2 *größer als* x_1 *ist. Damit folgt insgesamt* $x_1 \dot{-} x_2 = y \Leftrightarrow x_1 = x_2 + y \vee x_1 \leq x_2 \wedge y = 0$.

Die Formel $x_1 = x_2 + y \vee x_1 \leq x_2 \wedge y = 0$ *ist also eine logische Spezifikation des Verhaltens der Funktion* sub. *Diese Formel enthält drei Variablen* x_1, x_2 *und* y *und könnte zur Vereinfachung mit dem Namen* $SUB(x_1, x_2, y)$ *abgekürzt werden.*

Um logische Formeln für Berechenbarkeitszwecke nutzbar zu machen, muß man sie zunächst in einer logischen Programmiersprache, also der Syntax eines festgelegten logischen Kalküls, formulieren. Spätestens zur Ausführungszeit muß man dann festlegen, welche der Variable als Ausgabe zu betrachten ist, da ansonsten ja keinerlei Berechnungen stattfinden können. Da wir logische Programmierung zur *Darstellung berechenbarer Funktionen* verwenden wollen, muß die Ausgabe eindeutig sein, d.h. für jedes Eingabetupel darf es maximal einen Ausgabewert geben, der hierzu paßt, und genau dies muß mit den Mitteln des logischen Kalküls auf schematische Weise überprüft werden können.[1] Zur Vereinfachung legen wir fest, daß die Ausgabevariable in der Formel als letzte genannt wird und alle anderen Variablen Eingaben darstellen. Bei $SUB(x_1, x_2, y)$ sind also x_1, x_2 die Eingaben und y die Ausgabe.

Um die Eindeutigkeit der Ausgabe logisch überprüfen zu können, muß man neben der formalen Sprache auch *Axiome* und *Beweisregeln* festlegen. Die Axiome geben an, welche Formeln als Grundwahrheiten angesehen werden, wie z.B. $\forall x\ x = x$, $P \Rightarrow P$ oder $\forall x\ 0 \neq s(x)$. Regeln wiederum besagen, wie aus wahren Formeln neue Wahrheiten gefolgert, oder *abgeleitet* werden können. So besagt z.B. der sogenannte *Modus Ponens*, daß eine Formel B wahr sein muß, wenn $A \Rightarrow B$ gilt und die Formel A wahr ist. Eine Formel, die auf diese Art aus den Axiomen folgt, wird als logisch *gültig* bezeichnet.

Üblicherweise trennt man die reine Prädikatenlogik von den zusätzlichen Axiomen ab und bezeichnet letztere dann als logische *Theorie*. Auf die Art kann man die allgemeingültigen logischen Formeln von denen unterscheiden, die nur unter bestimmten Annahmen, den Axiomen einer Theorie, gelten. So gehören Axiome wie $\forall x\ 0 \neq s(x)$ nicht zur Prädikatenlogik, sondern beschreiben Eigenschaften der Symbole 0 und s, die in der Prädikatenlogik keinerlei Bedeutung besitzen. Für die Darstellung berechenbarer Funktionen werden wir sogenannte *numerische Theorien* benötigen, also Theorien, die jede natürliche Zahl durch einen Term der formalen Sprache beschreiben können und Axiome über die Eigenschaften dieser Terme bereitstellen.

Wir werden im folgenden kurz die wichtigsten Begriffe und Erkenntnisse der Logik zusammenfassen und eine einfache numerische Theorie vorstellen, die neben Termen für die natürlichen Zahlen auch Axiome für Addition und Multiplikation, im Gegensatz zu den Peano Axiomen aber keine Induktionsaxiome bereitstellt. Sie liefert daher keine exakte Beschreibung der bekannten natürlichen Zahlen, ist dafür aber schematisch leicht zu verarbeiten und hinreichend, um alle berechenbaren Funktionen über den natürlichen Zahlen zu repräsentieren. Wir werden den Begriff der arithmetischen Repräsentierbarkeit präzise definieren, an vielen Beispielen ausführlich illustrieren und anschließend beweisen, daß dieses Konzept äquivalent zu dem der rekursiven Funktionen ist. Zum Abschluß werden wir zeigen, daß die Turing-Mächtigkeit arithmetisch repräsentierbarer Funktionen die Unlösbarkeit einer Reihe logischer Probleme zur Folge hat und Beweise für die Gödelschen Unvollständigkeitssätze geben. Dieses Thema führt aber weit über eine Einführung in die Theoretische Informatik hinaus und ist nur als Zusatznformation für Interessierte gedacht.

[1] In Programmiersprachen wie Prolog werden die Ausgabevariablen der Formeln erst zur Laufzeit festgelegt, indem bei einer Anfrage gewisse Werte als Konstante und andere als Variablen gekennzeichnet werden. Das Laufzeitsystem bestimmt dann der Reihe nach alle möglichen Werte für die Variablen, welche die Formeln des Logikprogramms korrekt machen. In diesem Sinne ist auch eine Eindeutigkeit der Ausgabewerte nicht erforderlich, denn man betrachtet alle Formeln als Darstellung *mengenwertiger* Funktionen.

2 Logik und numerische Theorien

In der Mathematik wird Logik oft als Hilfsmittel zur Präzisierung und Kurzdarstellung textlicher Aussagen eingesetzt. Logische Konnektive, Quantoren und Rechenregeln wie die *De Morganschen Gesetze* sind daher vielen bekannt. Der wirkliche Zweck der Logik geht jedoch weit über die Verwendung als Hilfsmittel zur Abkürzung hinaus. Formale Logik hat zum Ziel, mathematische ("logische") Schlußfolgerungen, die üblicherweise ein tiefes Verständnis der Bedeutung einer Aussage verlangen, durch Regeln für eine symbolische Manipulation von Formeln zu ersetzen. Formale Logik umgeht die Mehrdeutigkeiten der natürlichen Sprache und soll durch eine schematische Lösung mathematischer Probleme jeden Streit über die Gültigkeit einer Aussage aus dem Weg räumen. Die Vision war, im Zweifelsfall die korrekte Lösung einfach auszurechnen. Auch wenn sich diese Vision, wie wir in Abschnitt 4 zeigen werden, nicht im vollen Umfang realisieren läßt, wird formale Logik heute dazu verwendet, aufwendige Beweise zu überprüfen und Hard- und Software zu verifizieren – eine Aufgabe, die für Menschen wegen ihrer Komplexität kaum durchführbar ist.

Kernbestandteile einer formalen Logik sind eine *formale Sprache*, bestehend aus *Syntax* und *Semantik*, und ein *Ableitungssystem*, bestehend aus *Axiomen* und *Inferenzregeln*. Man spricht oft auch von einem logischen *Kalkül*, um den Aspekt des Ableitungssystems deutlicher hervorzuheben. Dabei kann es für durchaus eine Reihe verschiedener Ableitungssysteme für dieselbe Logik (genauer, dieselbe logische Sprache) geben. So gibt es z.B. für die *Prädikatenlogik* sogenannte *Frege-Hilbert Kalküle*, in denen es viele Axiome aber nur eine Inferenzregel (den Modus Ponens) gibt, Kalküle des *natürlichen Schließens*, die je zwei Regeln für jedes Konnektiv aber keine Axiome haben, *Sequenzen-* und *Tableaukalküle* sowie *Resolutions-* und *Konnektionskalküle*, die auf eine effiziente Verarbeitung durch Computer getrimmt sind. Auch bei der Syntax einer Logik gibt es durchaus Abweichungen, je nachdem ob die Sprache maschinenlesbar sein muß oder logische Konnektive mit den vertrauten Symbolen kennzeichnet.

Wir konzentrieren uns im folgenden auf die Prädikatenlogik. Deren Syntax kennt *logische Symbole* wie $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, $\forall$, $\exists$ und das Gleichheitssymbol $=$, deren Bedeutung festgelegt ist, sowie Variablen, Funktions- und Prädikatszeichen, die keine feste Interpretation besitzen. Nullstellige Funktionszeichen werden als Konstante aufgefaßt, nullstellige Prädikatszeichen als Aussagen. Aus Variablen und Funktionszeichen (und Klammern) kann man nun *Terme* bilden, die widerum mithilfe von Prädikatssymbolen und dem Gleichheitssymbol zu *Formeln* zusammengesetzt werden können. Formeln, in denen alle Variablen durch einen Quantor gebunden sind, werden als *Sätze* bezeichnet.

Definition 2 (Syntax der Prädikatenlogik).
Es sei $\mathcal{V}$ eine Menge von Variablensymbolen, $\mathcal{F}$ eine Menge von Funktionssymbolen und $\mathcal{P}$ eine Menge von Prädikatssymbolen.

Ein Term ist entweder eine Variable $x \in \mathcal{V}$, ein nullstelliges Funktionssymbol $k \in \mathcal{F}$, oder eine Funktionsanwendung $f(t_1, \ldots, t_n)$, wobei $t_1, .., t_n$ Terme sind und $f \in \mathcal{F}$.
Eine Formel ist entweder ein nullstelliges Prädikatssymbol $P \in \mathcal{P}$, eine Prädikatsanwendung $P(t_1, \ldots, t_n)$ oder $t_1 = t_2$, wobei $t_1, .., t_n$ Terme sind und $P \in \mathcal{P}$, oder eine zusammengesetzte Formel der Gestalt $\neg A$, $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $\forall x\, A$, $\exists x\, A$, (A), wobei A, B Formeln sind und $x \in \mathcal{V}$.

Um die verschiedenen Alphabete nicht immer wieder neu kennzeichnen zu müssen, bezeichnen wir Variablen üblicherweise mit Symbolen wie $x,y,z,a,b,c,\ldots$, Funktionssymbole mit $f,g,h,a,b,c,\ldots$, und Prädikatssymbole mit $P,Q,R,\ A,\ B,\ C\ \ldots$. Gelegentlich verwenden wir auch Spezialsymbole wie $+, -, *, \ldots, 0, 1, 2, \ldots$ und $<, >, \geq, \ldots$ sowie Infix-Notation, um Funktionen und Prädikate zu kennzeichnen. Meist besagt der Kontext eindeutig, um welche Art von Symbol es sich handelt.

Manchmal läßt sich der syntaktische Aufbau einer Formel nicht eindeutig aus ihrer textlichen Darstellung rekonstruieren. Die Formel $\exists y\ gerade(y) \wedge y \geq 2 \Rightarrow y = 2 \wedge y > 20$ kann man zum Beispiel auf mindestens drei verschiedene Arten lesen. Um Klarheit zu schaffen, könnte man die Formel um Klammern ergänzen, aber Formeln mit vielen Klammern sind ebenfalls schwer zu lesen. Aus diesem Grunde vereinbart man Konventionen, welche Konnektive stärker binden als andere. Leider gibt es in der Literatur sehr unterschiedliche Konventionen. Für diesen Artikel legen wir fest, daß $\neg$ stärker bindet als $\wedge$, dann folgt $\vee$, dann $\Rightarrow$, dann $\exists$ und schließlich $\forall$. Implikation wird als *rechtsassoziativ* angesehen, d.h. $A \Rightarrow B \Rightarrow C$ steht für $A \Rightarrow (B \Rightarrow C)$. Im Zweifelsfall werden wir Klammern ergänzen, um Mißverständnisse zu vermeiden.

Variablen kommen *frei* in Formeln vor, solange sie nicht durch einen Quantor *gebunden* werden. Bei der mechanischen Verarbeitung von Formeln durch Inferenzregeln kann es vorkommen, daß eine Variable mehrfach durch Quantoren gebunden wird. In diesem Fall wird die Variable durch den "nächsten" Quantor gebunden. So ist in $\forall x P(x) \wedge (\forall x Q(x) \wedge R(x))$ das erste x in $P(x)$ durch den äußeren Quantor gebunden und die anderen durch den inneren. Um Klarheit zu schaffen, sollte man Variablen konsistent umbenennen: $\forall x P(x) \wedge (\forall y Q(y) \wedge R(y))$ ist erheblich verständlicher.

2.1 Semantik der Prädikatenlogik

Die *Semantik* ordnet syntaktisch korrekten Formeln eine Bedeutung zu. Sie gibt uns die Handhabe, darzustellen, daß wir den Satz $\forall x\ x > 2 \Rightarrow x^2 > 4$ für wahr halten, den Satz $\forall x\ \ x > 2 \Rightarrow x^2 > 5$ aber nicht. Die Semantik der Prädikatenlogik wird üblicherweise dadurch beschrieben, daß man allen nichtlogischen Symbolen eine Bedeutung zuweist. Dabei werden Variablen als Objekte eines Universums *interpretiert*, Funktionssymbole als Funktionen auf diesem Universum, und Prädikatsymbole als Relationen über diesem Universum. Der Einfachheit halber fassen wir Relationen als Funktionen mit Bildbereich {wahr,falsch} auf. Die Bedeutung eines Terms $f(t_1, .., t_n)$ kann man bestimmen, indem man die Interpretation von f auf die Interpretation der Terme $t_1, .., t_n$ anwendet. Auf diese Art wird jeder Term durch ein Objekt interpretiert. Die Bedeutung einer Formel $P(t_1, .., t_n)$ bestimmt man analog und erhält einen Wahrheitswert als Ergebnis.

Man beachte, daß bis auf die genannten Randbedingungen keinerlei Einschränkungen an die Interpretation der nichtlogischen Symbole existieren. Anders als bei der Verwendung von Logik als Hilfsmittel zur Abkürzung mathematischer Aussagen haben Symbole keine feste Bedeutung[2] , auch wenn dies beim Aufschreiben vielleicht

[2] Es ist natürlich möglich, für viele Standardsymbole eine feste Bedeutung in den logischen Kalkül mit aufzunehmen. Dies macht den Kalkül jedoch erheblich komplexer und automatische Unterstützung für eine Beweisführung schwierig. Außerdem müsste der Kalkül ständig erweitert werden, wenn neue Konzepte hinzukommen.

beabsichtigt war. In der Prädikatenlogik ist nur die Bedeutung der logischen Symbole $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, $\forall$, $\exists$ und $=$ festgelegt. Die Bedeutung aller anderen Symbole ist dagegen frei wählbar. Diese Eigenschaft ist essentiell für eine Verarbeitung logischer Formeln mit von computergestützten Beweissystemen, da Computer Formeln syntaktisch verarbeiten müssen, ohne die Bedeutung von Symbolen zu "verstehen".

Beispiel 3 (Interpretation nichtlogischer Symbole)

In der Mathematik bezeichnet $4{+}5$ normalerweise die Zahl neun. Hier wird $4{+}5$ als Text verstanden, der eine mathematische Idee, nämlich die Addition der Zahlen vier *und* fünf, *schriftlich formuliert. In diesem Sinn ist der Text nur ein Hilfsmittel zur Kommunikation einer Idee und deshalb wird der Text oft mit dem mathematischen Konzept identifiziert, das sich dahinter verbirgt.*

In der formalen Logik ist dies anders. Hier ist $4{+}5$ ein Term der formalen Sprache, in der weder das Symbol $+$ noch die Symbole 4 und 5 zu den reservierten Symbolen mit eine vordefinierten Bedeutung gehören. Man könnte diese Symbole also genauso gut auch mit der Subtraktionsfunktion und den Zahlen siebzehn und acht interpretieren.

Im Gegensatz zu den nichtlogischen Symbolen haben die Symbole $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, $\forall$, $\exists$ und $=$ eine feste Bedeutung in der Prädikatenlogik. Die Formel $t_1{=}t_2$ wird mit wahr interpretiert, wenn die Interpretationen der Terme t_1 und t_2 dasselbe Objekt bezeichnen, und der Wahrheitsgehalt zusammengesetzter Formeln folgt den bekannten logischen Regeln. Die folgende Definition präzisiert das Konzept der Interpretationen.

Definition 4 (Semantik der Prädikatenlogik).
Eine Interpretation $\mathcal{I}$ besteht aus einem Universum $\mathcal{U}$ und einer Interpretationsfunktion ι, die Variablen in Objekte aus $\mathcal{U}$, n-stellige Funktionssymbole in Funktionen aus $\mathcal{U}^n{\to}\mathcal{U}$ und n-stellige Prädikatssymbole in Funktionen aus $\mathcal{U}^n{\to}\{\mathsf{wahr},\mathsf{falsch}\}$ abbildet und auf Termen und Formeln wie folgt homomorph fortgesetzt wird.

$$
\begin{aligned}
\iota(f(t_1,\ldots,t_n)) &= \iota(f)(\,\iota(t_1),\ldots,\iota(t_n)\,)\\
\iota(P(t_1,\ldots,t_n)) &= \iota(P)(\,\iota(t_1),\ldots,\iota(t_n)\,)\\
\iota((A)) &= \iota(A)\\
\iota(t_1{=}t_2) &= \begin{cases}\mathsf{wahr} & \textit{falls } \iota(t_1)=\iota(t_2)\\ \mathsf{falsch} & \textit{sonst}\end{cases}\\
\iota(\neg A) &= \begin{cases}\mathsf{wahr} & \textit{falls } \iota(A)=\mathsf{falsch}\\ \mathsf{falsch} & \textit{sonst}\end{cases}\\
\iota(A\wedge B) &= \begin{cases}\mathsf{wahr} & \textit{falls } \iota(A)=\mathsf{wahr}\ \textit{und}\ \iota(B)=\mathsf{wahr}\\ \mathsf{falsch} & \textit{sonst}\end{cases}\\
\iota(A\vee B) &= \begin{cases}\mathsf{wahr} & \textit{falls } \iota(A)=\mathsf{wahr}\ \textit{oder}\ \iota(B)=\mathsf{wahr}\\ \mathsf{falsch} & \textit{sonst}\end{cases}\\
\iota(A\Rightarrow B) &= \begin{cases}\mathsf{wahr} & \textit{falls aus } \iota(A)=\mathsf{wahr}\ \textit{immer}\ \iota(B)=\mathsf{wahr}\ \textit{folgt}\\ \mathsf{falsch} & \textit{sonst}\end{cases}\\
\iota(\forall x\ A) &= \begin{cases}\mathsf{wahr} & \textit{falls } \iota^u_x(A)=\mathsf{wahr}\ \textit{für alle}\ u\in\mathcal{U}\\ \mathsf{falsch} & \textit{sonst}\end{cases}\\
\iota(\exists x\ A) &= \begin{cases}\mathsf{wahr} & \textit{falls } \iota^u_x(A)=\mathsf{wahr}\ \textit{für ein}\ u\in\mathcal{U}\\ \mathsf{falsch} & \textit{sonst}\end{cases}
\end{aligned}
$$

Dabei ist die Modifikation ι^u_x von ι definiert durch $\iota^u_x(y)=\begin{cases}u & \textit{falls } y=x\\ \iota(y) & \textit{sonst}\end{cases}$

Da die Variablen, Funktions- und Prädikatssymbole einer Formel sehr frei interpretiert werden dürfen, gibt es für viele Formeln Interpretationen, die sie wahr werden lassen, und andere Interpretationen, die das Ergebnis falsch liefern. Erstere werden oft auch als *Modelle* der Formel bezeichnet, da sie Möglichkeiten aufzeigen, warum die Formel gelten könnte. Für die Logik sind *(allgemein)gültige* Formeln von besonderem Interesse, also Formeln, die unter jeder Interpretation wahr werden. Diese Formeln sind *aus logischen Gründen wahr* und ein Beweis muß nicht mehr auf die konkrete Interpretation eingehen. Damit sind sie gut geeignet für schematische Beweisverfahren, die bei der Beweisführung ausschließlich syntaktisch vorgehen und kein tieferes Verständnis der betrachteten Formeln benötigen.

Definition 5 (Modelle, Erfüllbarkeit und Gültigkeit).
Es sei A eine beliebige prädikatenlogische Formel

- *Eine Interpretation $\mathcal{M}=(\iota,\mathcal{U})$ ist ein Modell von A, wenn $\iota(A)=\mathsf{wahr}$ ist.*
  Kurzschreibweise: $\mathcal{M}\models A$
- *A ist gültig, wenn jede Interpretation ein Modell für A ist.*
  A ist erfüllbar, wenn es ein Modell für A gibt.
  A ist widerlegbar, wenn es ein Modell für $\neg A$ gibt.
  A ist widersprüchlich, wenn es kein Modell für A gibt.
- *A folgt logisch aus Formelmenge $\mathcal{E}=\{E_1,..,E_n\}$, wenn jede Interpretation $\mathcal{I}$, die ein Modell aller Formeln $E_i\in\mathcal{E}$ ist, auch ein Modell von A ist.*
  Kurzschreibweise: $\mathcal{E}\models A$

In der Prädikatenlogik läßt sich der logische Folgerungsbegriff durch das Implikationssymbol simulieren, denn aufgrund des sogenannten *Deduktionstheorems* ist die Schlußfolgerung $\{E_1,..,E_n\}\models A$ genau dann semantisch korrekt, wenn $(E_1\wedge..\wedge E_n)\Rightarrow A$ eine gültige Formel ist. Damit ist es möglich, das semantische Konzept der logischen Schlüsse auf syntaktische Art auszudrücken und mit schematisch-logischen Beweismethoden zu analysieren.

2.2 Inferenzsysteme

Durch die Definition der Semantik ist die Bedeutung von Formeln eindeutig festgelegt. Rein hypothetisch wäre es möglich, diese Bedeutung dadurch zu bestimmen, daß man die Interpretation präzisiert und dann den Wert einer Formel ausrechnet. Für eine gegebene Interpretation kann diese Auswertung gelegentlich sogar automatisch durchgeführt werden. Die Analyse der Gültigkeit oder Erfüllbarkeit einer Formel müsste jedoch wieder von Hand geschehen, da man Aussagen über alle möglichen Interpretationen zu zeigen hätte. Wesentlich sinnvoller ist es daher, Verfahren zu entwickeln, welche die Gültigkeit von Formeln durch syntaktische Manipulationen analysieren können, ohne dabei ihren Wert ermitteln zu müssen. Diese Verfahren werden *Inferenzsysteme*, *Ableitungssysteme* oder oft einfach auch *Kalküle* genannt.

Ein logischer Kalkül besteht aus einer Menge von Regeln zur Manipulation von Formeln und Termen. Dabei wird eine Regel der Art "*aus A_1 und ... und A_n folgt C*" häufig in schematischer Form aufgeschrieben als

$$\frac{A_1,\ldots,A_n}{C}.$$

Wir nennen $A_1, \ldots, A_n$ die *Prämissen* dieser Regel und C die *Konklusion*. Eine typische Regel ist z.B. der *Modus Ponens* $\frac{A, A \Rightarrow B}{B}$. Sie besagt, daß aus der Gültigkeit der Formeln A und $A \Rightarrow B$ die Gültigkeit von B folgt. Eine Regel ohne Prämissen wird als *Axiom*, da die Konklusionsformel ohne jede Voraussetzung Gültigkeit hat, also eine Grundannahme darstellt. Gelegentlich werden Axiome und Regeln mit Prämissen separat betrachtet, aber das ist nicht wirklich erforderlich.

Genaugenommen bezeichnet die Schreibweise $\frac{A_1,\ldots,A_n}{C}$ ein Regel*schema*, da die Prämissen und die Konklusion Platzhalter für Formeln und Terme enthalten. Im Modus Ponens dürfen für A und B beliebige Formeln eingesetzt werden. Wichtig ist nur, daß dieselbe Formel, die für A eingesetzt wird, auch auf der linken Seite der Implikation erscheint und die für B eingesetzte Formel auf der rechten Seite. Ein Regelschema wird auf eine Menge $\mathcal{T}$ von Formeln angewandt, indem die Platzhalter in den Prämissen so durch Formeln erseetzt werden, daß sich Formeln aus $\mathcal{T}$ ergeben. Die entsprechend instantiierte Konklusion ist dann die abgeleitete neue Formel. Eine Formel A, die sich durch endlich viele Regelanwendungen aus $\mathcal{T}$ und den zwischendurch abgeleiteten Formeln ergibt, heißt *ableitbar aus* $\mathcal{T}$. Wir schreiben hierfür kurz $\mathcal{T} \vdash A$. Eine Formel A, die ohne Annahmen ableitbar ist, für die also $\emptyset \vdash A$ gilt, wird als *Theorem* bezeichnet.

Wie bereits erwähnt, gibt es für die Prädikatenlogik eine Vielfalt von Kalkülen, die sich durch Art und Umfang der Regeln unterscheiden. *Frege-Hilbert Kalküle* bestehen aus vielen Axiomen und einer einzigen "echten" Inferenzregel, dem Modus Ponens. Dies läßt formale Beweise ähnlich erscheinen wie in rigorosen mathematischen Abhandlungen, aber wenn man genau arbeitet, muß eine Fülle von Axiomen kombiniert werden, um ein einfaches Ergebnis zu erzielen. Die Kalküle des *natürlichen Schließens*, *Sequenzen-* und *Tableaukalküle* lassen sich sehr schematisch anwenden, orientieren sich aber immer noch an einem menschlichen Anwender. *Resolutions-* und *Konnektionskalküle* sind auf eine effiziente Verarbeitung durch Computer getrimmt und am besten geeignet für automatische Beweisprozeduren. Die Beweise sind für den Menschen aber kaum noch zu lesen. Allen Kalkülen ist gemein, daß sie *korrekt* und *vollständig* sind, d.h. alle Theoreme sind logisch gültige Formeln und umgekehrt. Damit ist es möglich, dem Ergebnis eines schematischen Beweises zu vertrauen (Korrektheit) und für alle gültigen Formeln mithilfe des Kalküls einen schematischen Beweis zu finden (Vollständigkeit). Es gibt jedoch gravierende Unterschiede in der Effizienz einer Beweissuche. Für vertiefende Details verweisen wir interessierte Leser auf die Lehrveranstaltung *Inferenzmethoden* und die dort bereitgestellten Materialien.

Die Möglichkeit, Beweise effektiv zu finden, ist essentiell für die Definition eines Berechenbarkeitsbegriffs auf der Basis logischer Gültigkeit. Da die obengenannten Kalküle korrekt und vollständig sind, wissen wir daß logische Gültigkeit $\models$ identisch ist mit Ableitbarkeit $\vdash$. Daher liefern diese Kalküle einen Berechnungsmechanismus für die logisch repräsentierbaren Funktionen, die wir in Definition 8 auf Seite 12 definieren werden. Da formale Ableitungen in einem logischen Kalkül wegen der Fülle notwendiger Details üblichweise jedoch sehr lang sind, werden wir in Korrektheitsbeweisen meist weniger formale Argumente führen und die Gültigkeit einer Formel mit rigorosen mathematischen Methoden und bekannten logischen Gesetzen nachweisen. Wegen der Vollständigkeit der Kalküle folgt hieraus dann die Existenz einer formale Ableitung dieser Formel im Kalkül.

2.3 Numerische Theorien und die Theorie $\mathcal{Q}$

Die Semantik der Prädikatenlogik erlaubt es, die Bedeutung der nichtlogischen Symbole Symbole frei zu interpretieren. Dies ist essentiell für die Verwendung logischer Beweiskalküle für die Beweisführung, da man ansonsten für jedes neu verwendete Symbol eine feste Bedeutung in einen Beweiskalkül hineincodieren und jedes Mal die Korrektheit und Vollständigkeit aufs Neue nachweisen müsste.

Die einzige Möglichkeit, die Bedeutung nichtlogischer Symbole einzuschränken, ist daher die Angabe einer Menge $\mathcal{T}$ von Formeln, welche die Eigenschaften dieser Symbole spezifizieren. Diese Formeln müssen dann als Voraussetzung jeder zu beweisenden Formel A hinzugefügt werden, so daß man im Endeffekt zeigen würde, daß A logisch aus $\mathcal{T}$ folgt bzw. daß $\mathcal{T} \Rightarrow A$ eine gültige Formel ist. Genau besehen heißt dies, daß die Formel A für jede Interpretation der genannten nichtlogischen Symbole gilt, welche die Formeln aus $\mathcal{T}$ erfüllen.

In der Mathematik wird diese Vorgehensweise häufig bei einem *axiomatischen Aufbau* einer Theorie verwendet. Anstatt die Objekte, Funktionen und Relationen der Theorie explizit zu konstruieren und hieraus weitere Eigenschaften zu folgern, beschreibt man die Konzepte der Theorie axiomatisch durch Spezifikation von Eigenschaften, welche sie erfüllen müssen. Alle weiteren Erkenntnisse werden dann nur aus diesen Axiomen hergeleitet, was oft einfacher ist, als explizite Konstruktionen zu verwenden. So wird z.B. die Mengentheorie meist auf den Axiomen von Zermelo & Fraenkel aufgebaut und die Theorie der natürlichen Zahlen auf den Peano Axiomen. Auch die ganzen, rationalen und reellen Zahlen werden in der Analysis meist axiomatisch beschrieben. Um sicherzustellen, daß die Axiome nicht unsinnig sind, wird die Theorie gelegentlich durch eine Menge von Konstruktionen unterstützt, welche die Axiome erfüllen.

Aus diesem Grund wird in der Logik eine *erfüllbare Menge $\mathcal{T}$ von Formeln*, aus der Schlußfolgerungen gezogen werden sollen, als *Theorie* bezeichnet. Eine Formel heißt dann *gültig in $\mathcal{T}$* (oder *$\mathcal{T}$-Theorem*, wenn sie logisch aus $\mathcal{T}$ folgt, d.h. wenn $\mathcal{T} \models A$ gilt. Wir schreiben hierfür oft auch $\models_{\mathcal{T}} A$. Manche Lehrbücher bezeichnen auch die Menge aller $\mathcal{T}$-Theoreme als die eigentliche Theorie und betrachten die Ausgangsformeln als *Menge der Axiome der Theorie*.

Man muß jedoch bedenken, daß nicht alle Theorien die Bedeutung ihrer Symbole so stark einschränken, daß es nur noch eine einzige Interpretation gibt, die alle Axiome erfüllt. Neben der *Standardinterpretation* gibt es oft weitere Interpretationen, die zwar nicht erwünscht, prinzipiell aber möglich sind. Wir werden dies in Abschnitt 2.4 am Beispiel der arithmetischen Theorie $\mathcal{Q}$ illustrieren.

Für die Darstellung berechenbarer Funktionen benötigt man logische Theorien, die es ermöglichen, Aussagen über Zahlen und Funktionen auf Zahlen zu formulieren. Sie müssen also zumindest Terme für die natürlichen Zahlen bereitstellen. Wie in vielen anderen Berechnungsmechanismen ist die einfachste Zahlendarstellung eine *unäre* Repräsentation: man stellt ein Symbol $\overline{0}$ als Repräsentation der Null bereit und ein Symbol s für die Nachfolgerfunktion. Da die Bedeutung dieser Symbole in der Logik nicht festgelegt ist, gibt man außerdem Axiome an, welche Randbedingungen für mögliche Interpretationen festlegen. Eine konkrete Zahl n wird damit durch den Term

$$\underbrace{\mathsf{s}(\ldots(\mathsf{s}(\overline{0}))\ldots)}_{n-mal}$$

dargestellt, den wir kurz mit $\overline{n}$ bezeichnen. So wird z.B. die Zahl 7 durch den Term $\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\overline{0}))))))$, also $\overline{7}$ dargestellt. Man beachte, daß $\overline{7}$ nur eine Abkürzung für diesen prädikatenlogischen Term und nicht etwa selbst eine Zahl ist.[3]

Theorien, die mindestens die Symbole $\overline{0}$ und s bereitstellen, nennt man *Numerische Theorien*. Unter den numerischen Theorien ist die *Peano Arithmetik* die bekannteste. Sie stellt zusätzlich ein Additionssymbol $+$ und ein Multiplikationssymbol $*$ bereit und spezifiziert die vier Symbole durch die bekannten Peano Axiome. Dazu gehört insbesondere auch das *Induktionsaxiom* $P(0) \wedge (\forall x\, P(x) \Rightarrow P(\mathsf{s}(x))) \Rightarrow \forall x\, P(x)$. Dieses Axiom ist aber in Wirklichkeit ein Axiomenschema, da für P jede beliebige Formel mit einer freien Variablen eingesetzt werden darf. Streicht man die Multiplikation und ihre Axiome, so entsteht die sogenannte *Presburger Arithmetik*, die unerwartet ausdrucksstark und entscheidbar ist und sich daher für automatisierte Beweisführung gut geeignet. Für die Repräsentation berechenbarer Funktionen reicht Presburger Arithmetik jedoch nicht aus, da die Theorie berechenbarer Funktionen nicht entscheidbar ist.

Behält man jedoch die Multiplikation und streicht stattdessen das Induktionsschema, so erhält man eine einfache numerische Theorie, mit der man alle berechenbaren Funktionen beschreiben kann. Sie ist bekannt als *Robinson Arithmetik* oder noch mehr unter dem Namen Theorie $\mathcal{Q}$ und benötigt nur sieben Axiome.

Definition 6 (Robinson's Theorie $\mathcal{Q}$).
Die Theorie $\mathcal{Q}$ ist eine numerische Theorie mit zwei ausgezeichneten zweistelligen Funktionssymbolen $+$ und $*$ in Infixnotation und den folgenden Axiomen

$$\begin{array}{lll} Q_1\colon \forall x,y\ \ \mathsf{s}(x){=}\mathsf{s}(y) \Rightarrow x{=}y & & Q_4\colon \forall x\ \ x{+}\overline{0} = x \\ Q_2\colon \forall x\ \ \mathsf{s}(x) \neq \overline{0} & & Q_5\colon \forall x,y\ \ x{+}\mathsf{s}(y) = \mathsf{s}(x{+}y) \\ Q_3\colon \forall x\ \ x{\neq}\overline{0} \Rightarrow \exists y\ x{=}\mathsf{s}(y) & & Q_6\colon \forall x\ \ x{*}\overline{0} = \overline{0} \\ & & Q_7\colon \forall x,y\ \ x{*}\mathsf{s}(y) = (x{*}y){+}x \end{array}$$

Im Gegensatz zur Peano Arithmetik verwendet die Theorie $\mathcal{Q}$ also nur ein *endliches Axiomensystem*, da neben den genannten sieben Axiomen nur noch die Axiome der logischen Konnektive und die Axiome der Gleichheit, also *Reflexivität*, *Transitivität*, *Symmetrie* und *Substitutivität* benötigt werden. Das Substitutionsaxiom ist, genau besehen, ebenfalls ein Axiomenschema, da es besagt, daß in jeder Formel und in jedem Term die linke Seite einer Gleichheit durch die rechte ersetzt werden kann. In der Theorie $\mathcal{Q}$ gibt es jedoch keine ausgezeichneten Prädikatssymbole und nur drei ausgezeichnete Funktionszeichen. Damit kann das Substitutionsschema explizit durch fünf Axiome beschrieben werden.

$$\begin{array}{lll} \mathit{subst}_{\mathsf{s}}\colon \forall x,y\ \ x{=}y \Rightarrow \mathsf{s}(x){=}\mathsf{s}(y) & & \\ \mathit{subst}_{+l}\colon \forall x,y,z\ x{=}y \Rightarrow x{+}z{=}y{+}z & & \mathit{subst}_{+r}\colon \forall x,y,z\ x{=}y \Rightarrow z{+}x{=}z{+}y \\ \mathit{subst}_{*l}\colon \forall x,y,z\ x{=}y \Rightarrow x{*}z{=}y{*}z & & \mathit{subst}_{*r}\colon \forall x,y,z\ x{=}y \Rightarrow z{*}x{=}z{*}y \end{array}$$

Dies bedeutet, daß die Theorie $\mathcal{Q}$ *endlich axiomatisierbar* ist und somit leicht mithilfe automatischer Beweisverfahren verarbeitet werden kann obwohl sie, wie sich herausstellen wird, unentscheidbar ist.

[3] Anstelle des des Funktionssymbols s verwenden viele frühe Arbeiten einen nachgestellten Strich und stellen die Zahl n durch den Term $\underbrace{\overline{0}'\ldots'}_{n-mal}$ dar. Diese Notation ist kompakter, entspricht aber nicht der Syntax der Prädikatenlogik.

2.4 Modelle der Theorie $\mathcal{Q}$

Es ist leicht zu sehen, daß die Axiome von $\mathcal{Q}$ erfüllbar sind, da die Standardinterpretation der natürlichen Zahlen mit Null, Nachfolgerfunktion, Addition und Multiplikation die Axiome erfüllt. Andererseits können wegen der Abwesenheit des Induktionsschemas viele bekannte Eigenschaften der Nachfolgerfunktion, Addition und Multiplikation, wie z.B. Kommutativität und Assoziativität, in der Theorie $\mathcal{Q}$ nicht für die Funktionssymbole s, $+$ und $*$ bewiesen werden. Der Grund hierfür ist, daß es Nichtstandardinterpretationen der Symbole von $\mathcal{Q}$ gibt, welche die Axiome $Q_1 \ldots Q_7$ erfüllen, diese Eigenschaften aber nicht besitzen. Wir wollen dies an einem einfachen Beispiel illustrieren.

Beispiel 7 (Nichtstandardinterpretation von $\mathcal{Q}$)

Die Formel $\forall x\ \mathsf{s}(x){\neq}x$ ist in der Theorie $\mathcal{Q}$ nicht gültig.

Um diese Behauptung zu beweisen, müssen wir ein Modell von $\mathcal{Q}$ konstruieren, in dem die Formel $\forall x\ \mathsf{s}(x){\neq}x$ nicht wahr ist. Es ist offensichtlich, daß dieses Modell in irgendeiner Form von der Standardinterpretation der natürlichen Zahlen abweichen muß, da die Formel ein bekanntes Gesetz der Arithmetik ist. Eine kurze Überlegung zeigt auch, daß $\mathsf{s}(\overline{n}){\neq}\overline{n}$ für jedes Numeral $\overline{n}$ gilt. Dies läßt sich leicht per Induktion beweisen:[4] *Für $n{=}0$ entspricht die Aussage dem Axiom Q_2 und wenn $\mathsf{s}(\overline{n}){\neq}\overline{n}$ für ein $n \in \mathbb{N}$ gilt, dann kann $\mathsf{s}(\mathsf{s}(\overline{n})){=}\mathsf{s}(\overline{n})$ nicht gelten, da hieraus $\mathsf{s}(\overline{n}){=}\overline{n}$ mit Axiom Q_1 folgen würde. Wir brauchen also für das Universum unseres Modells ein zusätzliches Element, das sich nicht durch ein Numeral $\overline{n}$ darstellen läßt. Es ist nicht nötig eine Intuition für dieses Element anzugeben, aber man könnte sich vorstellen, daß es jenseits aller natürlichen Zahlen liegt. Daher bezeichnen wir es mit ∞. Auf diesem Element müssen wir die Interpretation von* s *so wählen, daß die Formel $\mathsf{s}(\infty){=}\infty$ den Wert* wahr *erhält.*

Auf den natürlichen Zahlen dagegen müssen wir alle Symbole von $\mathcal{Q}$ auf die vertraute Art interpretieren, um sicherzustellen, daß die sieben Axiome erfüllt sind. Wir interpretieren also $\overline{0}$ mit der Zahl Null, s *mit der Nachfolgerfunktion, $+$ mit der Addition und $*$ mit der Multiplikation. Tabellarisch dargestellt erhalten wir also folgende Interpretationsfunktion ι auf dem Universum $\mathcal{U} = \mathbb{N} \cup \{\infty\}$.*

$\iota(\mathsf{s})$	$j \in \mathbb{N}\ \infty$
	$j{+}1\ \ \infty$

$\iota(+)$	$j \in \mathbb{N}\ \infty$	
$i \in \mathbb{N}$	$i+j$	∞
∞	∞	∞

$\iota(*)$	0	$j \in \mathbb{N}^+$	∞
0	0	0	0
$i \in \mathbb{N}^+$	0	$i*j$	∞
∞	0	∞	∞

In dieser Nichtstandardinterpretation der Symbole von $\mathcal{Q}$ ist das Gesetz $\forall x\ \mathsf{s}(x){\neq}x$ verletzt. Dies kann man an der Tabelle unmittelbar erkennen. Zu zeigen bleibt, daß ι die Axiome von $\mathcal{Q}$ erfüllt. Da sich ι auf den natürlichen Zahlen genauso verhält wie

[4] Man beachte, daß wir für den Nachweis von Eigenschaften der Theorie $\mathcal{Q}$ Induktionsbeweise verwenden dürfen, da wir hier die Theorie von außen ansehen. Dagegen darf das Induktionsschema nicht in Ableitungen von Theoremen innerhalb der Theorie $\mathcal{Q}$ eingesetzt werden, weil es nicht zu den sieben Axiomen gehört. Das mag anfangs etwas verwirrend erscheinen, aber man muß trennen zwischen den Beweisen, die *mit* den Methoden einer Theorie geführt werden, und sogenannten *metasprachlichen* Beweisen die *über* eine Theorie geführt werden.

die Standardinterpretation, sind die Axiome dort erfüllt und wir müssen nur noch das Verhalten von ι untersuchen, wenn die Interpretationen der Symbole s, $+$ *und* $*$ *auf da Zusatzelement* ∞ *angewandt werden. Der Einfachheit halber schreiben wir im folgenden kurz* s, $+$ *und* $*$ *anstelle von* $\iota(\mathsf{s})$, $\iota(+)$ *und* $\iota(*)$.

Q_1: *Es gelte* $s(\infty){=}s(y)$. *Wegen* $s(\infty){=}\infty$ *folgt hieraus* $\infty{=}s(y)$ *und damit* $\infty{=}y$, *da nur die Anwendung von* s *auf* ∞ *den Wert* ∞ *ergibt.*
Aus dem gleichen Grund folgt aus $s(y){=}s(\infty)$ *auch* $y{=}(\infty)$.

Q_2: $s(\infty){\neq}0$ *folgt direkt aus der Tabelle.*

Q_3: *Wegen* $\infty{=}s(\infty)$ *und* $\infty{\neq}0$ *gilt* $\infty{\neq}0 \;\Rightarrow\; \exists y\, s(y){=}\infty$.

Q_4: $\infty{+}0{=}\infty$ *folgt direkt aus der Tabelle*

Q_5: *Es gilt* $\infty{+}s(y){=}\infty{=}\; s(\infty) \;{=}s(\infty{+}y)$
und $x{+}s(\infty){=}\; x{+}\infty{=}\infty{=}\; s(\infty) \;{=}s(x{+}\infty)$

Q_6: $\infty{*}0{=}0$ *folgt direkt aus der Tabelle*

Q_7: *Es gilt* $\infty{*}s(y){=}\infty{=}\infty{+}(\infty{*}y)$
und $0{+}s(\infty){=}\; 0 \;{=}\; 0{+}0 \;{=}(0{*}\infty){+}0$
und $x{*}s(\infty){=}\; x{*}\infty{=}\infty{=}\infty{+}x \;{=}(x{*}\infty){+}x$ *für alle anderen* $x{\in}\mathcal{U}$.

$(\iota,\mathcal{U})$ *ist also ein Modell von* $\mathcal{Q}$, *welches die Formel* $\forall x\; \mathsf{s}(x){\neq}x$ *nicht erfüllt, und damit ist diese Formel in* $\mathcal{Q}$ *nicht gültig.*

Mit einem ähnlichen Ansatz und mehreren Zusatzelementen kann man leicht Nichtstandardmodelle von $\mathcal{Q}$ konstruieren, welche die Kommutativität und Assoziativität von $+$ und $*$ verletzen und in denen auch Formeln wie $\forall x\, \overline{0}{+}x{=}x$ oder $\forall x\, \overline{0}*x{=}\overline{0}$ nicht mehr wahr sind. Die Theorie $\mathcal{Q}$ ist also wesentlich zu schwach, um die bekannten Gesetze der Arithmetik zu beweisen, da man hierfür wirklich das Induktionsschema benötigt. Nichtsdestotrotz ist sie ausdrucksstark genug, um alle berechenbaren Funktionen darstellen zu können. Dies wollen wir im folgenden Abschnitt zeigen.

3 Arithmetische Repräsentierbarkeit

Um zeigen zu können, daß eine so einfache Theorie wie die Robinson Arithmetik ausreicht, um alle berechenbaren Funktionen darstellen zu können, müssen wir zunächst einmal präzise definieren, was es bedeutet, daß eine logische Formel eine Funktion beschreibt. Die formale Sprache der Prädikatenlogik ermöglicht zwar die Verwendung von Funktionszeichen, gibt uns aber keinen direkten Mechanismus an die Hand, Funktionen zu definieren oder gar zu programmieren. Numerische Theorien wie die Peano Arithmetik oder die Robinson Arithmetik verwenden Axiome, um die Bedeutung einiger Symbole wie s, $+$ oder $*$ einzuschränken. Dies aber liefert keine geschlosssene Darstellung einer Funktion, denn man kann auf die Art nur schwer ausdrücken, daß das Symbol $-$ das eindeutig definierte Inverse der Additionsfunktion beschreibt.

In der Mathematik werden Funktionen oft als Spezialfälle von Relationen, also als Mengen von Paaren $\{(x_0,y_0),(x_1,y_1),\ldots\}$, angesehen. Diese Relationen müssen *rechtseindeutig* sein, dürfen also für jedes x maximal ein Paar (x,y) enthalten. Damit entsprechen sie der Menge der Ein-/Ausgabepaare einer (möglicherweise partiellen) Funktion. So beschreibt z.B. die Relation $\{(0,0),(1,0),(2,1),(3,2),\ldots\}$ die Vorgängerfunktion p auf den natürlichen Zahlen.

Relationen lassen sich in der Logik durch Formeln beschreiben, welche angeben, wann ein Ausgabewert y zu einem Eingabewert x paßt. Diese Formeln spezifizieren also das Ein-/Ausgabeverhalten einer Funktion. So läßt sich zum Beispiel die Relation $\{(0,0),(1,0),(2,1),(3,2),\ldots\}$ durch die Formel $x{=}0 \wedge y{=}0 \;\vee\; y{+}1{=}x$ beschreiben, die damit also eine Spezifikation der Vorgängerfunktion p darstellt.

Die Spezifikation des Ein-/Ausgabeverhaltens einer Funktion f durch eine Formel R_f reicht jedoch nicht aus, um *berechenbare* Funktionen zu spezifizieren, da R_f für sich alleine genommen keinen Berechnungsmechanismus enthält. Eine Formel kann nur dann eine berechenbare Funktionen spezifizieren, wenn wir in der Lage sind, die Zugehörigkeit eines Ausgabewertes zu einem Eingabewert mechanisch zu überprüfen und die Eindeutigkeit dieses Ausgabewertes nachzuweisen. Wenn also $f(x){=}y$ für ein konkretes Paar (x,y) gilt, dann müssen wir die Gültigkeit der Formel $R_f(\bar{x},\bar{y})$ formal beweisen können, wobei $\bar{x}$ und $\bar{y}$ Termdarstellungen der Werte x und y sind. Ist dagegen $f(x)$ verschieden von y, so müssen wir auch das beweisen können, also zeigen, daß $\neg R_f(\bar{x},\bar{y})$ gilt. Diese Überlegungen führen zu folgender Definition.

Definition 8 (Repräsentierbarkeit arithmetischer Funktionen).
Es sei $\mathcal{T}$ *eine numerische Theorie.*
Eine Funktion $f{:}\mathbb{N}^k{\to}\mathbb{N}$ *heißt repräsentierbar in* $\mathcal{T}$, *wenn es in der formalen Sprache von* $\mathcal{T}$ *eine* $k{+}1$*-stelliges Prädikat* R_f *gibt, so daß für alle* $i_1,..,i_k,k{\in}\mathbb{N}$ *gilt*

$$f(i_1,..,i_k){=}j \quad \textit{impliziert} \quad \models_{\mathcal{T}} \quad R_f(\overline{i_1},..,\overline{i_k},\overline{j})$$
$$\textit{und} \quad f(i_1,..,i_k){\neq}j \quad \textit{impliziert} \quad \models_{\mathcal{T}} \neg R_f(\overline{i_1},..,\overline{i_k},\overline{j})$$

$f{:}\mathbb{N}^k{\to}\mathbb{N}$ *heißt arithmetisch repräsentierbar, wenn* f *repräsentierbar in* $\mathcal{Q}$ *ist.*

Die beiden Anforderungen an R_f besagen, daß die letzte Variable eindeutig die Funktionswerte von f beschreibt. Ist f an der Stelle i definiert, dann ist $R_f(\overline{i},\overline{f(i)})$ in $\mathcal{T}$ ableitbar und $\neg R_f(\overline{i},\overline{j})$ für alle $j{\neq}f(i)$. Ist f an der Stelle i nicht definiert, dann ist $\neg R_f(\overline{i},\overline{j})$ für alle j ableitbar. Man beachte, daß in der Formel anstelle der Zahlen die entsprechenden Numerale verwendet werden.

Um also zu zeigen, daß die Vorgängerfunktion p arithmetisch repräsentierbar ist, müssten wir für die Formel $R_{\mathsf{p}}(x,y) \;\equiv\; x{=}0 \wedge y{=}0 \;\vee\; y{+}1{=}x$, beweisen, daß korrekte Ein-/Ausgabepaare zu gültige Instanzen von R_{p} führen, während unpassende Ein-/Ausgabepaare zu gültigen Instanzen der Negation von R_{p} führen. Zu zeigen ist also

- Für alle i,j mit $\mathsf{p}(i){=}j$ ist $R_{\mathsf{p}}(\overline{i},\overline{j})$ gültig in $\mathcal{Q}$,
  d.h. es gilt $\models_{\mathcal{Q}} R_{\mathsf{p}}(\overline{0},\overline{0})$, $\models_{\mathcal{Q}} R_{\mathsf{p}}(\overline{1},\overline{0})$, $\models_{\mathcal{Q}} R_{\mathsf{p}}(\overline{2},\overline{1})$, $\models_{\mathcal{Q}} R_{\mathsf{p}}(\overline{3},\overline{2})$, …
- Für alle i,j mit $\mathsf{p}(i){\neq}j$ ist $\neg R_{\mathsf{p}}(\overline{i},\overline{j})$ gültig in $\mathcal{Q}$,
  d.h. es gilt $\models_{\mathcal{Q}} \neg R_{\mathsf{p}}(\overline{0},\overline{1})$, $\models_{\mathcal{Q}} \neg R_{\mathsf{p}}(\overline{0},\overline{2})$, …
  und $\models_{\mathcal{Q}} \neg R_{\mathsf{p}}(\overline{1},\overline{1})$, $\models_{\mathcal{Q}} \neg R_{\mathsf{p}}(\overline{1},\overline{2})$, …
  und $\models_{\mathcal{Q}} \neg R_{\mathsf{p}}(\overline{2},\overline{0})$, $\models_{\mathcal{Q}} \neg R_{\mathsf{p}}(\overline{2},\overline{2})$, …

Man muß also für jedes konkrete Paar (i,j) natürlicher Zahlen nur mithilfe der Axiome der Theorie $\mathcal{Q}$ beweisen können, daß die Ausgabe j tatsächlich der Wert von $\mathsf{p}(i)$ ist bzw. daß sie es nicht ist. Natürlich führt man diesen Beweis nicht einzeln für jeden möglichen Wert von i und j, sondern führt Induktionsbeweise auf der sogenannten *Meta-Ebene* der Theorie $\mathcal{Q}$, also indem wir die Theorie von außen betrachten. So

zeigen wir, daß $R_{\mathsf{p}}(\overline{0},\overline{0})$ in $\mathcal{Q}$ gültig ist und daß aus der Gültigkeit von $R_{\mathsf{p}}(\overline{i},\overline{j})$ für ein i,j mit $\mathsf{p}(i){=}j$ auch folgt, daß $R_{\mathsf{p}}(\overline{i{+}1},\overline{k})$ für das eindeutige k mit $\mathsf{p}(i{+}1){=}k$ in $\mathcal{Q}$ gültig ist.[5] Daraus folgt dann, $\models_{\mathcal{Q}} R_{\mathsf{p}}(\overline{i},\overline{j})$ für alle i,j mit $\mathsf{p}(i){=}j$. Einen ähnlichen (Meta-)Beweis muß man für den Fall führen, daß $\mathsf{p}(i){\neq}j$ ist. Im Beispiel 10 werden wir diese Beweise im Detail vorführen.

In der Definition der Repräsentierbarkeit erscheinen die beiden Anforderungen an R_f auf den ersten Blick unnötig kompliziert und man ist geneigt, sie durch

$$f(i_1,..,i_k){=}j \quad \text{gilt genau dann, wenn} \quad \models_T \quad R_f(\overline{i_1},..,\overline{i_k},\overline{j})$$

zu ersetzen. Dies reicht jedoch nicht aus, da hierdurch die zweite Bedinungzu

$$f(i_1,..,i_k){\neq}j \quad \text{impliziert} \quad \not\models_T \quad R_f(\overline{i_1},..,\overline{i_k},\overline{j}),$$

abgeschwächt wird, also dazu daß $R_f(\overline{i_1},..,\overline{i_k},\overline{j})$ in der Theorie $\mathcal{T}$ nicht gültig ist, wenn $f(i_1,..,i_k){=}j$ nicht gilt. Wenn aber eine Formel in $\mathcal{T}$ nicht gültig ist, dann heißt dies nur, daß wir es nicht möglich ist, aus den Axiomen von $\mathcal{T}$ einen Beweis für sie zu konstruieren. Daraus folgt aber nicht, daß wir dann auch aus den Axiomen der Theorie $\mathcal{T}$ einen Beweis für die Negation der Formel konstruieren können, denn dies würde verlangen, daß die Theorie $\mathcal{T}$ entscheidbar ist.

3.1 Beispiele arithmetisch repräsentierbarer Funktionen

Um eine Funktion f als arithmetisch repräsentierbar nachzuweisen, müssen wir eine Formel R_f konstruieren, die das Ein-/Ausgabeverhalten von f beschreibt, und anschließend die Anforderungen an R_f aus Definition 8 nachweisen. Zur Konstruktion der Formel R_f muß man im Prinzip nur $R_f(x_1,..,x_k,y) \equiv f(x_1,..,x_k){=}y$ definieren und dann die rechte Seite schrittweise in eine Formel umwandeln, in der nur s, $+$, $*$, Numerale $\overline{i}$, Abkürzungen für bekannte Formeln und bereits bekannte Repräsentationen anderer Funktionen vorkommen. Dies ist nicht anders als in jeder anderen Programmiersprache, in der man Programme schrittweise aus den Basiselementen der Sprache zusammensetzt. Wir wollen dies an einigen Beispielen illustrieren

Beispiel 9 (Repräsentierbarkeit der Addition)
Wir wollen zeigen, daß die Addition auf den natürlichen Zahlen arithmetisch repräsentierbar ist. Da die Addition zu den vordefinierten Symbolen der Theorie $\mathcal{Q}$ gehört, sollte die Repräsentation einfach sein. Wir können uns also darauf konzentrieren zu beweisen, daß das Symbol $+$ tatsächlich die Additionsfunktion beschreibt.[6]

Wir müssen also zunächst eine dreistelliges Prädikat R_+ angeben, welche die Addition repräsentieren soll. Wir definieren also $R_+(x_1,x_2,y) \equiv y = x_1{+}x_2$. Da alle Symbole der rechten Seite in $\mathcal{Q}$ vordefiniert sind, müssen wir die rechte Seite nicht weiter umwandeln und haben somit R_+ als eine Formel der Theorie $\mathcal{Q}$ beschrieben.

[5] Wir wissen natürlich, daß k in diesem Fall identisch mit i ist. Die gewählte Formulierung entspricht mehr der allgemeinen Vorgehensweise.

[6] Wegen Beispiel 7 wissen wir, daß die Addition auf $\mathbb{N}$ nicht die einzig mögliche Interpretation des Symbols $+$ ist. Die Nichtstandardmodelle von $\mathcal{Q}$ verhalten sich jedoch nur *außerhalb* der natürlichen Zahlen anders als gewohnt. Da jede natürliche Zahl in $\mathcal{Q}$ durch ein Numeral repräsentiert wird, umfassen alle Modelle von $\mathcal{Q}$ die Menge der natürlichen Zahlen und verhalten sich hierauf wie gewohnt. Dies muß jedoch für jedes Symbol einzeln bewiesen werden.

Wir müssen nun zeigen, daß R_+ eine Repräsentation der Additionsfunktion gemäß Definition 8 ist. Hierzu zeigen wir, daß $\models_{\mathcal{Q}} R_+(\overline{i},\overline{j},\overline{k})$ für alle $i,j,k{\in}\mathbb{N}$ mit $i{+}j{=}k$ gilt und daß $\models_{\mathcal{Q}} \neg R_+(\overline{i},\overline{j},\overline{k})$ für alle $i,j,k{\in}\mathbb{N}$ mit $i{+}j{\neq}k$ gilt, und verwenden dafür jeweils Induktion auf der Meta-Ebene von $\mathcal{Q}$.

1. *Es sei i beliebig, aber fest. Wir zeigen durch Induktion über j, daß $\models_{\mathcal{Q}} R_+(\overline{i},\overline{j},\overline{k})$ (bzw. $\models_{\mathcal{Q}} \overline{k} = \overline{i}{+}\overline{j}$) für alle $j,k{\in}\mathbb{N}$ mit $i{+}j{=}k$ gilt.*
   Für $j{=}0$ folgt $i{=}k$, also sind die Numerale $\overline{i}$ und $\overline{k}$ identisch und die Formel $R_+(\overline{i},\overline{j},\overline{k})$ ist identisch mit $\overline{i} = \overline{i}{+}\overline{0}$. Diese Formel läßt sich in $\mathcal{Q}$ aus dem Axiom Q_4 und dem Symmetrieaxiom der Gleichheit ableiten. Damit folgt $\models_{\mathcal{Q}} R_+(\overline{i},\overline{j},\overline{k})$.
   Für ein festes j gelte $\models_{\mathcal{Q}} R_+(\overline{i},\overline{j},\overline{k})$ bzw. $\models_{\mathcal{Q}} \overline{k} = \overline{i}{+}\overline{j}$ für alle $k{\in}\mathbb{N}$ mit $i{+}j{=}k$.
   Es gelte $i{+}(j{+}1){=}k$. Dann ist $k{>}0$, $\overline{k}$ identisch mit $\mathsf{s}(\overline{k{-}1})$ und $\overline{j{+}1}$ mit $\mathsf{s}(\overline{j})$.
   Wegen $i{+}j{=}k{-}1$ folgt mit der Induktionsannahme $\models_{\mathcal{Q}} \overline{k{-}1} = \overline{i}{+}\overline{j}$. Mit dem Axiom $\mathsf{subst}_{\mathsf{s}}$ folgt hieraus $\models_{\mathcal{Q}} \mathsf{s}(\overline{k{-}1}) = \mathsf{s}(\overline{i}{+}\overline{j})$ und mit Axiom Q_5, Symmetrie und Transitivität folgt $\models_{\mathcal{Q}} \mathsf{s}(\overline{k{-}1}) = \overline{i}{+}\mathsf{s}(\overline{j})$. Da die abgeleitete Formel aber syntaktisch identisch ist mit $\overline{k} = \overline{i}{+}\overline{j{+}1}$, folgt insgesamt $\models_{\mathcal{Q}} R_+(\overline{i},\overline{j{+}1},\overline{k})$.
2. *Wir zeigen durch Induktion über $i{+}j$, daß $\models_{\mathcal{Q}} \neg R_+(\overline{i},\overline{j},\overline{k})$ (bzw. $\models_{\mathcal{Q}} \overline{k} \neq \overline{i}{+}\overline{j}$) für alle $i,j,k{\in}\mathbb{N}$ mit $i{+}j{\neq}k$ gilt.*
   Für $i{+}j{=}0$ ist $i{=}j{=}0$, also $\overline{i}{=}\overline{0}$ und $\overline{j}{=}\overline{0}$. Es sei $k{\in}\mathbb{N}$ mit $i{+}j{\neq}k$. Dann ist $k{>}0$, $\overline{k}$ identisch mit $\mathsf{s}(\overline{n})$ für ein n und $\neg R_+(\overline{i},\overline{j},\overline{k})$ identisch mit $\mathsf{s}(\overline{n}) \neq \overline{0}{+}\overline{0}$. Diese Formel folgt in $\mathcal{Q}$ aus den Axiomen Q_4 und Q_2, also gilt $\models_{\mathcal{Q}} \neg R_+(\overline{i},\overline{j},\overline{k})$.
   Für ein $n{\in}\mathbb{N}$ und alle i,j mit $i{+}j{=}n$ gelte $\models_{\mathcal{Q}} \neg R_+(\overline{i},\overline{j},\overline{k})$ (bzw. $\models_{\mathcal{Q}} \overline{k} \neq \overline{i}{+}\overline{j}$) für alle $k{\in}\mathbb{N}$ mit $i{+}j{\neq}k$.
   Es sei $i{+}j = n{+}1$ und $i{+}j{\neq}k$. Dann ist $\overline{n{+}1}$ identisch mit $\mathsf{s}(\overline{n})$ und wegen Teil 1 gilt $\models_{\mathcal{Q}} R_+(\overline{i},\overline{j},\mathsf{s}(\overline{n}))$ bzw. $\models_{\mathcal{Q}} \mathsf{s}(\overline{n}) = \overline{i}{+}\overline{j}$.
   Falls $k{=}0$ ist, dann folgt aus Axiom Q_2 und Symmetrie $\models_{\mathcal{Q}} \overline{k} \neq \mathsf{s}(\overline{n})$ und mit der obigen Erkenntnis und Transitivität auch $\models_{\mathcal{Q}} \overline{k} \neq \overline{i}{+}\overline{j}$ bzw. $\neg R_+(\overline{i},\overline{j},\overline{k})$.
   Andernfalls ist $\overline{k}$ identisch mit $\mathsf{s}(\overline{k{-}1})$ und $n{\neq}k{-}1$, also nach Induktionsannahme $\models_{\mathcal{Q}} \overline{k{-}1} \neq \overline{i'}{+}\overline{j'}$ für alle $i',j'{\in}\mathbb{N}$ mit $i'{+}j'{=}n$.
   Falls $j{=}0$ ist, dann ist $(i{-}1){+}j{=}n$ und es folgt $\models_{\mathcal{Q}} \overline{k{-}1} \neq \overline{i{-}1}{+}\overline{j}$. Mit Q_4 und Transitivät folgt $\models_{\mathcal{Q}} \overline{k{-}1} \neq \overline{i{-}1}$ und mit $\mathsf{subst}_{\mathsf{s}}$ ergibt sich $\models_{\mathcal{Q}} \mathsf{s}(\overline{k{-}1}) \neq \mathsf{s}(\overline{i{-}1})$ bzw. $\models_{\mathcal{Q}} \overline{k} \neq \overline{i}$, woraus mit Q_4 und Symmetrie wiederum $\models_{\mathcal{Q}} \overline{k} \neq \overline{i}{+}\overline{j}$ folgt.
   Andernfalls ist $\overline{j}$ identisch mit $\mathsf{s}(\overline{j{-}1})$ und es folgt $\models_{\mathcal{Q}} \overline{k{-}1} \neq \overline{i}{+}\overline{j{-}1}$ wegen $i{+}(j{-}1){=}n$. Mit $\mathsf{subst}_{\mathsf{s}}$ ergibt sich $\models_{\mathcal{Q}} \mathsf{s}(\overline{k{-}1}) \neq \mathsf{s}(\overline{i}{+}\overline{j{-}1})$ und mit Q_5 und Transitivät wiederum $\models_{\mathcal{Q}} \mathsf{s}(\overline{k{-}1}) \neq \overline{i}{+}\mathsf{s}(\overline{j{-}1})$ bzw. $\models_{\mathcal{Q}} \overline{k} \neq \overline{i}{+}\overline{j}$.
   Damit folgt $\models_{\mathcal{Q}} \neg R_+(\overline{i},\overline{j},\overline{k})$für alle $i,j,k{\in}\mathbb{N}$ mit $i{+}j = n{+}1$ und $i{+}j{\neq}k$.

Wie das obige Beispiel zeigt, sind Korrektheitsbeweise auch für einfache Repräsentationsformeln wie R_+ ausgesprochen kompliziert. Die liegt daran, daß wir nachweisen müssen, daß Instanzen dieser Formel in jedem möglichen Fall *innerhalb der Theorie $\mathcal{Q}$ ableitbar* sind und manche vertraute Gesetze wie die Kommutativität der Addition in $\mathcal{Q}$ nicht gelten. Deswegen fallen manche Beweise etwas aufwendiger aus, wenn man ganz genau argumentieren möchte. Dies ist aber bei Korrektheitsbeweisen für Turingprogramme, λ-Terme oder μ-rekursive nicht anders und schon gar nicht bei Korrektheitsbeweisen für C- oderJava-Programme, da man hier genau genommen sogar die Eigenschaften des Compiler mit in den Beweis einbeziehen müsste.

In den meisten Korrektheitsbeweisen geht man daher nicht so detailliert vor wie in Beispiel 9 und verzichtet insbesondere auf die Angabe von Zwischenschritten, die mit den Gleichheitsaxiomen – also Transitivität, Symmetrie und Substitutivität – zu tun haben. Stattdessen werden oft einfache Gleichungsketten aufgestellt und als Begründung (wenn überhaupt) nur noch die relevanten Axiome von $\mathcal{Q}$ benannt. Wir wollen dies an einem weiteren Beispiel illustrieren.

Beispiel 10 (Repräsentierbarkeit der Vorgängerfunktion)

Die Vorgängerfunktion p *ist das Inverse der Nachfolgerfunktionen auf positiven natürlichen Zahlen, d.h. es gilt* $\mathsf{p}(\mathsf{s}(x))=x$ *für alle* x*. Für* $x=0$ *ist der Wert der Vorgängerfunktion ebenfalls* 0*, da negative Zahlen als Resultate nicht möglich sind. Damit ist* $y=\mathsf{p}(x)$ *genau dann, wenn* $y+1=x$ *ist, oder* x *und* y *den Wert* 0 *annehmen. Diese Analyse führt zu folgender Repräsentationsformel für die Vorgängerfunktion* p*.*

$$R_{\mathsf{p}}(x,y) \;\equiv\; x=\overline{0} \wedge y=\overline{0} \;\vee\; \mathsf{s}(y)=x$$

Zu zeigen bleibt, daß R_{p} *tatsächlich eine Repräsentation der Vorgängerfunktion ist. Hierzu zeigen wir, daß* $\models_{\mathcal{Q}} R_{\mathsf{p}}(\overline{i},\overline{k})$ *für alle* $i,k\in\mathbb{N}$ *mit* $\mathsf{p}(i)=k$ *gilt und daß auch* $\models_{\mathcal{Q}} \neg R_{\mathsf{p}}(\overline{i},\overline{k})$ *für alle* $i,k\in\mathbb{N}$ *mit* $\mathsf{p}(i)\neq k$ *gilt, und unterscheiden* $i=0$ *und* $i>0$*.*

1. *Es sei* $i=0$ *und* $\mathsf{p}(i)=k$*. Dann ist* $\overline{i}=\overline{k}=\overline{0}$ *und* $R_{\mathsf{p}}(\overline{i},\overline{k})$ *ist* $\overline{0}=\overline{0}\wedge\overline{0}=\overline{0} \;\vee\; \mathsf{s}(\overline{0})=\overline{0}$*. Diese Formel ist gültig in* $\mathcal{Q}$ *aufgrund der Reflexivität der Gleichheit.*

   Es sei $\mathsf{p}(i)\neq k$*. Dann ist* $\overline{k}=\mathsf{s}(\overline{j})$ *für ein* j *und die Negation von* $R_{\mathsf{p}}(\overline{i},\overline{k})$ *ist die Formel* $\neg(\overline{0}=\overline{0}\wedge\mathsf{s}(\overline{j})=\overline{0} \;\vee\; \mathsf{s}(\mathsf{s}(\overline{j}))=\overline{0})$*. Wandelt man diese Formel mit den üblichen Gesetzen der Prädikatenlogik um, so ergibt sich* $(\overline{0}\neq\overline{0}\vee\mathsf{s}(\overline{j})\neq\overline{0})\wedge\mathsf{s}(\mathsf{s}(\overline{j}))\neq\overline{0}$*. Beide Teilformeln dieser Formel folgen in* $\mathcal{Q}$ *aus dem Axiom* Q_2 *und damit ist* $\neg R_{\mathsf{p}}(\overline{i},\overline{k})$ *gültig in* $\mathcal{Q}$*.*

2. *Es sei* $i>0$ *und* $\mathsf{p}(i)=k$*. Dann ist* $i=k+1$*,* $\overline{i}=\mathsf{s}(\overline{k})$*, und* $R_{\mathsf{p}}(\overline{i},\overline{k})$ *ist die Formel* $\mathsf{s}(\overline{k})=\overline{0}\wedge\overline{k}=\overline{0} \;\vee\; \mathsf{s}(\overline{k})=\mathsf{s}(\overline{k})$*, die in* $\mathcal{Q}$ *aus der Reflexivität der Gleichheit folgt.*

   Es sei $\mathsf{p}(i)\neq k$*. Wegen* $i>0$ *ist* $\overline{i}=\mathsf{s}(\overline{j})$ *für ein* j *und es ist entweder* $k>j$ *oder* $k<j$*. Im ersten ersten Fall hat* $\overline{k}$ *die Gestalt* $\mathsf{s}^n(\overline{j})$ *für ein* $n>0$ *und* $\neg R_{\mathsf{p}}(\overline{i},\overline{k})$ *ist die Formel* $(\mathsf{s}(\overline{j})\neq\overline{0}\vee\mathsf{s}^n(\overline{j})\neq\overline{0})\wedge\mathsf{s}(\mathsf{s}^n(\overline{j}))\neq\mathsf{s}(\overline{j})$*. Die linke Teilformel folgt aus Axiom* Q_2*. Wendet man das Axiom* Q_1 *i-mal auf die rechte Teilformel an, so ergibt sich* $\mathsf{s}^n(\overline{0})\neq\overline{0}$*, was wiederum aus* Q_2 *folgt.*
   Ansonsten hat $\overline{j}$ *die Gestalt* $\mathsf{s}^m(\overline{k})$ *für ein* $m>0$ *und* $\neg R_{\mathsf{p}}(\overline{i},\overline{k})$ *ist die Formel* $(\mathsf{s}(\mathsf{s}^m(\overline{k}))\neq\overline{0}\vee\overline{k}\neq\overline{0})\wedge\mathsf{s}(\overline{k})\neq\mathsf{s}(\mathsf{s}^m(\overline{k}))$*, die in* $\mathcal{Q}$ *durch* $k+1$*-faches Anwenden von* Q_1 *und Anwendung von* Q_2 *ableitbar ist.*
   Damit ist $\neg R_{\mathsf{p}}(\overline{i},\overline{k})$ *gültig in* $\mathcal{Q}$*.*

In den bisherigen Beispielen haben wir gezeigt, daß die Addition durch ein Prädikat R_+ repräsentiert wird, das durch $R_+(x_1,x_2,y) \equiv y=x_1+x_2$ definiert ist, und daß die Vorgängerfunktion durch ein Prädikat R_{p} repräsentiert wird, das definiert ist durch $R_{\mathsf{p}}(x,y) \equiv x=\overline{0}\wedge y=\overline{0}\vee\mathsf{s}(y)=x$. Diese Ausdrucksweise wirkt etwas umständlich. Wir werden daher gelegentlich sagen, daß eine Funktion durch die Formel repräsentiert wird, die ihre Eigenschaften beschreibt, wenn wir den Namen des repräsentierenden Prädikates nicht benötigen. So könnten wir z.B. sagen, daß die Vorgängerfunktion durch die Formel $x=\overline{0}\wedge y=\overline{0}\vee\mathsf{s}(y)=x$ repräsentiert wird, wobei aus dem Kontext hervorgeht, welche Variablen die Eingaben und welche die Ausgaben kennzeichnen.

Wir wollen nun für einige wichtige arithmetische Funktionen zeigen, daß sie in der Theorie $\mathcal{Q}$ repräsentierbar sind. Dabei werden wir auch einige Hilfsprädikate als Abkürzungen für komplexere Formeln einführen und hierfür, wo möglich, vertraute (Infix-)Notationen einführen. Wir werden die Korrekheit der Repräsentationen intuitiv begründen, aber nicht so detailliert beweisen wir in den obigen Beispielen.

Beispiel 11 (Wichtige repräsentierbare Funktionen und Hilfsprädikate)

– *Die Nachfolgerfunktion* s *ist vordefiniert in* $\mathcal{Q}$ *und kann daher durch das Prädikat* R_{s} *mit* $R_{\mathsf{s}}(x,y) \equiv y=\mathsf{s}(x)$ *repräsentiert werden.*

– *Die Multiplikation* $*$ *ist ebenfalls vordefiniert und wird repräsentiert durch das Prädikat* R_* *mit* $R_*(x_1,x_2,y) \equiv y=x_1{*}x_2$*.*

– *Die Subtraktion auf natürlichen Zahlen ist das Inverse der Addition, die aber keine negativen Werte annehmen kann. Wie in Beispiel 1 illustriert, ist also*
$x_1 \dot{-} x_2 = y \;\Leftrightarrow\; x_1=x_2+y \;\vee\; x_1\leq x_2 \wedge y=0$*.*
Um diese Formel auf eine Form zu bringen, die der Sprache von $\mathcal{Q}$ *entspricht, müssen wir noch das Prädikat* $\leq$ *durch* s*,* $+$*,* $*$*, Numerale und prädikatenlogische Ausdrücke beschreiben. Es gilt* $x_1\leq x_2$*, wenn man* x_2 *als Summe von* x_1 *und einer natürliche Zahl* n *beschreiben kann. Wir definieren daher folgende Hilfsprädikate*
$x_1\leq x_2 \;\equiv\; \exists z\; x_1+z=x_2$ *und*
$x_1<x_2 \;\equiv\; \mathsf{s}(x_1)\leq x_2$
Damit wird die Subtraktion repräsentiert durch
$R_-(x_1,x_2,y) \equiv x_1=x_2+y \;\vee\; x_1\leq x_2 \wedge y=\overline{0}$

– *Das Hilfsprädikat* $\leq$ *beschreibt in der Theorie* $\mathcal{Q}$*, daß eine Zahl kleiner oder gleich einer anderen Zahl ist. Dies ist jedoch keine Repräsentation des entsprechenden Größenvergleichs* als berechenbare Funktion*, da diese bei Eingabe zweier Werte* x_1 *und* x_2 *ein Ergebnis liefern muß, welches besagt, ob* $x_1\leq x_2$ *gilt oder nicht.*
Die Größenvergleichsfunktion $t_\leq$ *ist definiert durch* $t_\leq(x_1,x_2) = \begin{cases} 1 & \text{falls } x_1\leq x_2 \\ 0 & \text{sonst} \end{cases}$
und wird in $\mathcal{Q}$ *dargestellt durch*
$R_\leq(x_1,x_2,y) \equiv x_1\leq x_2\wedge y=\overline{1} \;\vee\; x_2<x_1\wedge y=\overline{0}$

– *Die Division auf natürlichen Zahlen ist das Inverse der Multiplikation, die aber ganzzahlige Werte annehmen muß. Die ganzzahlige Division rundet daher das rationale Divisionsergebnis auf die nächste ganze Zahl ab. Division durch* 0 *ist nicht erlaubt. Damit läßt sich die ganzzahlige Division wie folgt präzisieren*
$div(x_1,x_2) = \begin{cases} \lfloor x_1/x_2 \rfloor & \text{falls } x_2\neq 0 \\ \bot & \text{sonst} \end{cases}$
Es gilt also $y=div(x_1,x_2)$ *genau dann, wenn* $x_2\neq 0$ *ist und* x_1 *einen Wert zwischen* $x_2{*}y$ *und* $x_2{*}y+(x_2-1)$ *annimmt. Damit läßt sich die* div *darstellen durch*
$R_{div}(x_1,x_2,y) \equiv x_2{*}y\leq x_1 \;\wedge\; x_1<\mathsf{s}(x_2){*}y \;\wedge\; x_2\neq\overline{0}$
Eine alternative Herangehensweise ist die Feststellung, daß sich die Zahl x_1 *die Summe von* $x_2{*}y$ *und Divisionsrest ist, wobei letzterer eine Zahl zwischen* 0 *und* x_2-1 *ist. Dies liefert gleichzeitig eine Repräsentation der ganzzahlige Division und des Divisionsrestes – also der "modulo"-Funktion.*
$R_{div'}(x_1,x_2,y) \equiv x_2\neq\overline{0} \wedge \exists z\; (z<x_2 \wedge x_1=x_2{*}y+z)$
$R_{mod}(x_1,x_2,z) \equiv x_2\neq\overline{0} \wedge z<x_2 \wedge \exists y\; x_1=x_2{*}y+z)$

– *Ein Primzahltest prüft, ob eine gegebene Zahl x eine Primzahl ist. Dies ist der Fall, wenn x mindestens 2 ist und durch keine Zahl, außer sich selbst und natürlich der 1, teilbar ist. Dabei ist eine Zahl x_1 Teiler von x_2, im Zeichen $x_1|x_2$, wenn x_2 ein Vielfaches von x_1 ist. Beide Begriffe lassen sich direkt in die Sprache der Theorie $\mathcal{Q}$ übertragen, was zur Definition von zwei weiteren Hilfsprädikaten führt:*

$$x_1 \mid x_2 \quad \equiv \exists y\ x_2{=}x_1{*}y \quad \textit{und}$$
$$Prime(x) \ \equiv \overline{2}{\leq}x \ \wedge\ \forall y\,(\overline{1}{<}y \wedge y{<}x) \Rightarrow \neg(y \mid x)$$

Die Umwandlung dieser Hilfsprädikate in Repräsentationen eines Teilbarkeits- bzw. Primzahltest ist analog zur Konstruktion von $R_{\leq}$ und ergibt

$$R_{divides}(x_1,x_2,y) \equiv x_1 \mid x_2 \wedge y = \overline{1} \ \vee\ \neg(x_1 \mid x_2) \wedge y = \overline{0}$$
$$R_{prime}(x,y) \quad \equiv Prime(x) \wedge y = \overline{1} \ \vee\ \neg Prime(x) \wedge y = \overline{0}$$

– *Das kleinste gemeinsame Vielfache zweier Zahlen x_1 und x_2 ist – wie der Name schon sagt – die kleinste Zahl y, die von x_1 und x_2 geteilt wird. Das bedeutet, daß x_1 und x_2 Teiler von y sind und daß jede andere Zahl z, die von x_1 und x_2 geteilt wird, mindestens so groß ist wie y. Diese Eigenschaft läßt sich direkt in Logik übertragen und liefert*

$$R_{kgV}(x_1,x_2,y) \equiv x_1 \mid y \ \wedge\ x_2 \mid y \ \wedge\ \forall z\ (x_1 \mid z \wedge x_2 \mid z \Rightarrow y{\leq}z)$$

Analog hat der größte gemeinsame Teiler zweier Zahlen ebenfalls eine sehr direkte Räpresentation in $\mathcal{Q}$

$$R_{ggT}(x_1,x_2,y) \equiv y \mid x_1 \ \wedge\ y \mid x_2 \ \wedge\ \forall z\ (z \mid x_1 \wedge z \mid x_2 \Rightarrow z{\leq}y)$$

Die Repräsentationen von Primzahl- und Teilbarkeitstests, *kgV* und *ggT* zeigen, daß die arithmetische Repräsentierbarkeit eine besonders geeignete Darstellungsfunktion für Funktionen ist, die üblicher durch ihre Eigenschaften spezifiziert werden und nicht durch ihr operationales Verhalten. In diesen Fällen ist das "Logikprogramm" eine direkte Übertragung der Spezifikation in eine logische Formel. Man muß dabei nur sicherstellen, daß die Spezifikation eindeutig ist oder durch die Hinzunahme weiterer Formeln eindeutig gemacht wird. Die Konstruktion konkreter Ausgaben wird dann durch das Laufzeitsystem der logischen Programmiersprache durchgeführt – der Programmierer muß sich hierum nicht kümmern. In imperativen und funktionalen Programmiersprachen dagegen fällt dem Programmierer die Aufgabe zu, selbst ein Verfahren zu beschreiben, das die gesuchte Lösung konstruiert.

Wir könnten die obige Liste der Beispiele arithmetisch repräsentierbarer Funktionen noch beliebig fortsetzen, da im Endeffekt jede bekannte berechenbare Funktion arithmetisch repräsentierbar ist und damit insbesondere alle Beispielfunktionen, die wir im Zusammenhang mit Turingmaschinen, μ-rekursiven Funktionen und λ-Termen betrachtet haben. Das dies tatsächlich der Fall ist, wollen wir im nächsten Abschnitt beweisen.

3.2 Arithmetische Repräsentierbarkeit ist Turing-mächtig

Es ist leicht einzusehen, daß alle arithmetisch repräsentierbaren Funktionen auch berechenbar sind. Da die Robinson Arithmetik endlich axiomatisierbar ist, kann man einen Algorithmus konstruieren, der schrittweise alle Beweise generiert und somit alle gültigen Formeln der Theorie $\mathcal{Q}$ aufzählt. Bei Eingabe eines Zahlentupels $(x_1,..,x_n)$ erhöht man simultan die Werte für mögliche Ausgabewerte und für die Anzahl der erlaubten

Beweissuchschritte, bis ein Beweis für eine Formel der Art $R_f(\overline{x_1},..,\overline{x_n},\overline{y})$ gefunden ist. Da R_f rechtseindeutig ist, gibt es nur ein mögliches y und dies ist dann der gesuchte Funktionswert.[7] Damit haben wir folgendes Theorem bewiesen.

Theorem 12 *Jede arithmetisch repräsentierbaren Funktion ist berechenbar.*

Um zu beweisen, daß jede berechenbare Funktion auch arithmetisch repräsentierbar ist, müssen wir eines der bekannten Berechenbarkeitsmodelle durch das Konzept der Repräsentierbarkeit ausdrücken. Hierzu bieten sich besonders die μ-rekursiven Funktionen an, die eine mathematische Sicht auf berechenbare Funktionen beschreibt und sich schon relativ nahe an formaler Logik befinden. Der Vollständigkeit wiederholen wir hier ihre Definition.

Definition 13 (μ-rekursive Funktionen).
Die Klasse $\mathcal{R}$ der μ-rekursiven Funktionen ist induktiv wie folgt definiert.

1. *Die Nachfolgerfunktion s:$\mathbb{N}\to\mathbb{N}$ mit $s(x) = x{+}1$ für alle $x \in \mathbb{N}$ ist μ-rekursiv.*
2. *Die Projektionsfunktionen pr^n_k:$\mathbb{N}^n\to\mathbb{N}$ mit der Eigenschaft $pr^n_k(x_1,..,x_n) = x_k$ für alle $x_1,..,x_n \in \mathbb{N}$ sind μ-rekursiv für alle $n \in \mathbb{N}$ und $k \in \{1..n\}$.*
3. *Die Konstantenfunktionen c^n_k:$\mathbb{N}^n\to\mathbb{N}$ mit der Eigenschaft $c^n_k(x_1,..,x_n) = k$ für alle $x_1,..,x_n \in \mathbb{N}$ sind μ-rekursiv für alle $n,k \in \mathbb{N}$.*
4. *Die Komposition $f{\circ}(g_1,..,g_n)$:$\mathbb{N}^k\to\mathbb{N}$ der Funktionen f:$\mathbb{N}^n\to\mathbb{N}$, $g_1,..g_n$:$\mathbb{N}^k\to\mathbb{N}$ ist μ-rekursiv für alle $n,k \in \mathbb{N}$, wenn $f, g_1...g_n$ μ-rekursive Funktionen sind.*
   Dabei ist $f{\circ}(g_1,..,g_n)$ die eindeutig bestimmte Funktion h mit der Eigenschaft
   $$h(\hat{x}) = f(\,g_1(\hat{x}),..,g_n(\hat{x})\,)$$[8]
5. *Die primitive Rekursion $Pr[f,g]$:$\mathbb{N}^k\to\mathbb{N}$ zweier Funktionen f:$\mathbb{N}^{k-1}\to\mathbb{N}$ und g:$\mathbb{N}^{k+1}\to\mathbb{N}$ ist μ-rekursiv für alle $k \in \mathbb{N}$, wenn f und g μ-rekursiv sind.*
   Dabei ist $Pr[f,g]$ die eindeutig bestimmte Funktion h für die gilt
   $$h(\hat{x},0) = f(\hat{x}) \ \textit{ und }\ h(\hat{x},y{+}1) = g(\hat{x},y,\,h(\hat{x},y)).$$
6. *Die Minimierung μf:$\mathbb{N}^k\to\mathbb{N}$ einer Funktion f:$\mathbb{N}^{k+1}\to\mathbb{N}$ ist μ-rekursiv für alle $k \in \mathbb{N}$, wenn f μ-rekursiv ist.*
   Dabei ist μf die eindeutig bestimmte Funktion h, für die gilt
   $$h(\hat{x}) = \begin{cases} min\{y \mid f(\hat{x},y){=}0\} & \textit{falls dies existiert und } f(\hat{x},i) \textit{ für alle } i{<}y \textit{ definiert ist} \\ \bot & \textit{sonst} \end{cases}$$

Für den Beweis der arithmetischen Repräsentierbarkeit der rekursiven Funktionen müssen wir also die Grundfunktionen repräsentieren und zeigen, daß repräsentierbare Funktionen durch Komposition, primitive Rekursion und Minimierung wieder zu repräsentierbaren Funktionen zusammengesetzt werden.

[7] Diese Methode ist zwar nicht sehr effizient, zeigt aber, daß es prinzipiell möglich ist, jede repräsentierbare Funktionen zu berechnen. Die Forschung im Bereich *Automatisches Theorembeweisen* und *Logikprogrammierung* hat in den vergangen Jahrzehnten eine Vielfalt von Techniken entwickelt, mit denen logisch repräsentierte Funktionen sehr effizient berechnet werden können, und Logikprogrammiersprachen wie Prolog sind in vielen Anwendungen nicht weniger effizient als Sprachen wie C++ oder Java und gelegentlich sogar deutlich effizienter.

[8] $\hat{x}$ ist abkürzend für ein Tupel $(x_1,...,x_m)$

- Die *Nachfolgerfunktion* wird repräsentiert durch die Formel $R_{\mathsf{s}}(x,y) \equiv y = \mathsf{s}(x)$
- Die *Projektionsfunktion* pr_k^n wird repräsentiert durch die Formel
$$R_{pr_k^n}(x_1,..,x_n,y) \equiv y = x_k$$
- Die *Konstantenfunktion* c_k^n wird repräsentiert durch die Formel
$$R_{c_k^n}(x_1,..,x_n,y) \equiv y = \overline{k}$$
- Die *Komposition* $f\circ(g_1,..,g_n)$ kann durch folgende Formel repräsentiert werden
$R_{f\circ(g_1,..,g_n)}(x_1,..,x_n,y) \equiv$
$\exists z_1,..z_k\ R_{g_1}(x_1,..x_n,z_1)\wedge\ldots\wedge R_{g_k}(x_1,..x_n,z_k)\wedge R_f(z_1,..z_k,y)$,
wobei $R_{g_1},\ldots,R_{g_k}$, and R_f die Funktionen $g_1,..,g_k$ und f repräsentieren.
- Die *Minimierung* μf kann durch folgende Formel repräsentiert werden
$$R_{\mu f}(x_1,..,x_n,y) \equiv \forall z{\leq}y\ (R_f(x_1,..x_d,z,0)\Leftrightarrow z=y)$$

Problematisch ist nur die Repräsentation der primitiven Rekursion. Während Komposition und Minimierung sich in $\mathcal{Q}$ mithilfe von Quantoren beschreiben lassen, gibt es in der Sprache von $\mathcal{Q}$ kein Konstrukt zur Beschreibung von Rekursion. In der Tat ist es sehr mühsam, eine direkte Repräsentation der primitiven Rekursion zu entwickeln. Allerdings hat sich herausgestellt, daß man das Verhalten der primitiven Rekursion mithilfe von Polynom-Codierungen und den oben genannten Konstrukten simulieren kann, also in einem Berechnungsmodell beschreiben kann, das den μ-rekursiven Funktionen sehr ähnlich ist, aber anstelle der primitiven Rekursion drei weitere Grundfunktionen, nämlich die Addition, die Multiplikation und einen Test auf Gleichheit, enthält. Man nennt dieses Modell *min-rekursive Funktionen* um die Ähnlichkeit zu den μ-rekursiven Funktionen hervorzuheben.

Wir werden im folgenden zunächst die min-rekursiven Funktionen genau definieren und beweisen daß alle μ-rekursiven Funktionen auch min-rekursiv sind und daß alle min-rekursiven Funktionen arithmetisch repräsentierbar sind. Hieraus folgt dann die Turing-Mächtigkeit der arithmetisch repräsentierbaren Funktionen.

Definition 14 (min-rekursive Funktionen).
Die Klasse $\mathcal{R}_{min}$ der min-rekursiven Funktionen ist induktiv wie folgt definiert.

1. *Die Nachfolgerfunktion s:$\mathbb{N}\to\mathbb{N}$ ist min-rekursiv.*
2. *Die Addition $+$:$\mathbb{N}^2\to\mathbb{N}$ ist min-rekursiv.*
3. *Die Multiplikation $*$:$\mathbb{N}^2\to\mathbb{N}$ ist min-rekursiv.*
4. *Der Test auf Gleichheit $t_=$:$\mathbb{N}^2\to\mathbb{N}$ mit der Eigenschaft* $t_=(x,y) = \begin{cases} 1 \text{ falls } x{=}y \\ 0 \text{ sonst} \end{cases}$
für alle $x,y\in\mathbb{N}$ ist min-rekursiv.
5. *Die Projektionsfunktionen pr_k^n:$\mathbb{N}^n\to\mathbb{N}$ mit der Eigenschaft $pr_k^n(x_1,..,x_n) = x_k$ für alle $x_1,..,x_n\in\mathbb{N}$ sind min-rekursiv für alle $n\in\mathbb{N}$ und $k\in\{1..n\}$.*
6. *Die Konstantenfunktionen c_k^n:$\mathbb{N}^n\to\mathbb{N}$ mit der Eigenschaft $c_k^n(x_1,..,x_n) = k$ für alle $x_1,..,x_n\in\mathbb{N}$ sind min-rekursiv für alle $n,k\in\mathbb{N}$.*
7. *Die Komposition $f\circ(g_1,..,g_n)$:$\mathbb{N}^k\to\mathbb{N}$ der Funktionen f:$\mathbb{N}^n\to\mathbb{N}$, $g_1,..g_n$:$\mathbb{N}^k\to\mathbb{N}$ ist min-rekursiv für alle $n,k\in\mathbb{N}$, wenn $f,g_1...g_n$ min-rekursive Funktionen sind.*
8. *Die Minimierung μf:$\mathbb{N}^k\to\mathbb{N}$ einer Funktion f:$\mathbb{N}^{k+1}\to\mathbb{N}$ ist min-rekursiv für alle $k\in\mathbb{N}$, wenn f min-rekursiv ist.*

Im Prinzip ist es möglich, in der Definition der min-rekursiven Funktionen auf die Nachfolgerfunktion und die Konstantenfunktionen zu verzichten. Die Konstante c_1^n kann durch die Komposition $t_=\circ(pr_n^n, pr_n^n)$ ausgedrückt werden, die Nachfolgerfunktion s mithilfe der Addition $+\circ(pr_1^1, c_1^1)$, die Konstante c_2^n als $s\circ c_1^n$, die höheren Konstanten entsprechend mit weiteren Nachfolgerperationen, und die Nullfunktion c_0^n als $t_=\circ(c_1^n, c_2^n)$. Da all diese Funktionen jedoch gebraucht werden und leicht in der Theorie $\mathcal{Q}$ repräsentiert werden können, sind sie in unserer Definition explizit enthalten.

Es ist leicht zu sehen, daß jede min-rekursive Funktionen auch μ-rekursiv ist, da Addition, Multiplikation und Gleichheitstest μ-rekursiv sind und alle anderen min-rekursiven Konstrukte auch in der Definition μ-rekursiver Funktionen genannt werden.

Theorem 15 *$\mathcal{R}_{min}{\subseteq}\mathcal{R}$: Jede min-rekursive Funktion ist auch rekursiv.*

Für den Beweis der Gegenrichtung müssen wir vor allem zeigen, daß die primitive Rekursion min-rekursiver Funktionen wieder eine min-rekursive Funktion liefert. Hierzu müssen wir zeigen, wie wir für eine Funktion $h = Pr[f,g]$ den Wert von $h(\hat{x},y)$ berechnen können, ohne dabei Rekursion zu verwenden.

Eine Möglichkeit ist, die gesamte Berechnungsfolge $h(\hat{x},0),\ h(\hat{x},1),\ \ldots h(\hat{x},y)$ zu bestimmen und dann das letzte Element dieser Folge herauszugreifen. Aber auch das ist ohne Verwendung von Rekursion nicht trivial. Wir wissen zwar, daß $h(\hat{x},0){=}f(\hat{x})$ ist, $h(\hat{x},1){=}g(\hat{x},0,\ h(\hat{x},0))$, $h(\hat{x},2){=}g(\hat{x},1,\ h(\hat{x},2))$, usw., und daß bei der Bestimmung jedes einzelnen $h(\hat{x},i)$ aus dem Vorgänger keine primitive Rekursion erforderlich ist. Allerdings wäre die Konstruktion der gesamten Folge mit genau y Elementen dann doch wieder ein rekursiver Prozess.

Deswegen müssen wir die Berechnungsfolge auf eine andere Weise generieren, die nur Komposition, Minimierung und einfache Arithmetik auf der Basis von Addition und Multiplikation benötigt. Anstatt also die Folge $h(\hat{x},0),\ h(\hat{x},1),\ \ldots h(\hat{x},y)$ zu *konstruieren*, zählen wir einfach alle möglichen Zahlenfolgen $z_0,z_1,\ldots,z_y$ auf und *suchen* mittels Minimierung die erste solche Zahlenfolge, welche die Bedingungen $z_0{=}f(\hat{x})$, $z_1{=}g(\hat{x},0,z_0))$, $z_2{=}g(\hat{x},1,z_1))$, usw. erfüllt.

Auch dies ist immer noch nicht ganz einfach, da wir Zahlenfolgen beliebiger Länge aufzählen müssen. Diese müssen für die Minimierung jedoch als eine einzige Zahl codiert werden, auf deren Komponenten wir mit einer min-rekursiven Funktion zugreifen können. Hierzu bietet sich eigentlich die Standardtupelfunktion $\langle\_,\_\rangle$ an, die wir von den μ-rekursiven Funktionen kennen. Sie war definiert als
$$\langle x,y\rangle = (x{+}y)(x{+}y{+}1){\div}2 + y$$
und ist damit auch min-rekursiv und bijektiv. Auch ihre Umkehrfunktionen π_i lassen sich min-rekursiv beschreiben, denn es ist
$$\pi_1(z) = \mu_i[\exists j{\leq}z\ \langle i,j\rangle{=}z] \quad \text{und} \quad \pi_2(z) = \mu_j[\exists i{\leq}z\ \langle i,j\rangle{=}z]$$
Für die Codierung und Decodierung von Zahlenfolgen beliebiger Länge würden wir jedoch die Iteration der Standardtupelfunktion benötigen, was wieder eine Rekursion erfordert. Deswegen müssen wir Zahlenfolgen durch Tupel fester Größe codieren.

Eine beliebte, wenn auch nicht sehr effiziente, Methode der Mathematik zur Codierung einer Liste von Zahlen $z_0,z_1,\ldots,z_y$ ist die Verwendung von *eindeutig decodierbaren Polynomen*. Um dieses Polynom zu erzeugen, suchen wir zunächst die kleinste Primzahl p, die größer ist als y und alle z_i. Anschließend verwenden wir $z_0,z_1,\ldots,z_y$

als Koeffizienten einer p-adischen Darstellung einer Zahl $z \equiv z_0 + z_1{*}p + \ldots + z_y{*}p^y$. Um sicherzustellen, daß wir auf die einzelnen Koeffizienten zugreifen können, koppeln wir z mit der Primzahl p und erhalten insgesamt die Zahl $\hat{z} \equiv \langle z, p\rangle$. Will man nun eine Komponente z_i aus $\hat{z}$ extrahieren, so muß man zunächst $z{=}\pi_1(\hat{z})$ und $p{=}\pi_2(\hat{z})$ bestimmen, dividiert anschließend z durch p^{i+1}, was eine Zahl z' liefert, welche die Folge $z_i, z_{i+1}, \ldots, z_y$ repräsentiert. Wir bestimmen nun den Rest der Division von z' und p und erhalten somit den Wert von z_i.

Diese Konstruktion benötigt jedoch immer noch eine primitive Rekursion für die Berechnung von p^{i+1}. Dies kann man jedoch umgehen, indem man in der Zahl z neben den eigentlichen Elementen der Folge $z_0, z_1, \ldots, z_y$ jeweils noch den Index des Elementes und die Zahl $p\dot{-}1$ als Trennsymbol mitführt, insgesamt also die Folge $(p\dot{-}1), 0, z_0,\ (p\dot{-}1), 1, z_1, \ldots,\ (p\dot{-}1), y, z_y$ als p-adische Zahl repräsentiert und dann einfach nur nach dem Vorkommen der Teilfolge $(p\dot{-}1), i, z_i$ sucht.

Es ist wichtig, daß wir die Zahl $\hat{z}$ nicht aus einer Zahlenfolge $z_0, z_1, \ldots, z_y$ konstruieren müssen, was wiederum Rekursion erfordern würde, sondern nur eine geeignete Repräsentation benötigen, die einen *Zugriff auf die Elemente der Folge mit min-rekursiven Funktionen erlaubt*. Denn bei der Suche nach Berechnungsfolgen für eine primitiv-rekursive Berechnung werden Zahlen der Reihe nach generiert, und dann in "Komponenten" zerlegt, deren Eigenschaften überprüft werden können. Es reicht also zu zeigen, daß zu jeder Zahlenfolge $z_0, z_1, \ldots, z_y$ eine Zahl $\hat{z}$ existiert, aus der sich die einzelnen z_i mit min-rekursiven Mitteln extrahieren lassen.

Wir geben hierzu eine Reihe min-rekursiver Konstruktionen an, mit denen wir insgesamt eine Zahlenfolgenrepräsentation in ihre Komponenten zerlegen können.[9]

- Die beschränkte Minimierung min-rekursiver Funktionen und Relationen ist min-rekursiv, denn es gilt $\mu_{i<n}[P(i)] \equiv \mu_i[P(i) \vee i{=}n]$. Dies liefert ein wirksames Mittel zur min-rekursiven Implementierung rekursiver Funktionen.
- Ein ebenso wichtiges Hilfsmittel sind beschränkte Quantoren. Wir definieren
  $\forall i{<}n\ P(i) \equiv \mu_{i<n}[\neg P(i){=}n$ und $\exists i{<}n\ P(i) \equiv \mu_{i<n}[P(i){\neq}n$.
  $\forall i{\leq}n\ P(i)$ und $\exists i{\leq}n\ P(i)$ sind analog definiert.
- Wir benötigen einen Größenvergleich: $x{<}y \equiv \exists i{<}y\ i{=}x$ und analog $x{\leq}y$.
- Teilbarkeit und Primzahltest sind ähnlich zu Beispiel 11 definiert:
  $x \mid y \equiv \exists i{\leq}n\ x{*}i{=}y$ und $Prime(x) \equiv 2{\leq}x \wedge \forall y\,(1{<}y \wedge y{<}x) \Rightarrow \neg(y \mid x)$
- Bei der Berechung von β müssen wir testen, ob eine Zahl n Potenz einer Primzahl p ist. Dies ist der Fall, wenn jeder echte Teiler von n durch p teilbar ist.
  $n \in p^* \equiv 1{\leq}n \wedge Prime(p) \wedge \forall i{\leq}n\ (i \mid n \Rightarrow (m{=}1 \vee p \mid m)$
- Wir benötigen auch die erste Primzahlpotenz von p, die größer als n ist.
  $n{\uparrow}p^* \equiv \mu_i[n{<}i \wedge i \in p^* \wedge Prime(p)]$
- Wir simulieren die Konkatenation zweier Zahlenfolgen auf p-adischen Polynomen.
  $z_1 \circ_p z_2 \equiv z_1{*}(z_2{\uparrow}p^*) + z_2$

[9] Der Einfachheit halber verwenden wir oft Relationssymbole anstelle der zugehörigen Testfunktionen, schreiben also z.B. $\mu_i[P(i)]$ statt $\mu_i[t_P(i){=}1]$ oder $x{=}y$ statt $t_=(x,y)$ und verwenden logischen Konnektive $\wedge$, $\vee$, $\neg$, $\Rightarrow$ anstelle der sonst erforderlichen Multiplikation, Addition, etc. von Testfunktionen.

- Da $0 \circ_p z_2{=}z_2$ und $z_1 \circ_p 0{=}z_1$ ist, können wir Teilfolgen wie folgt beschreiben
  $z_1\ part_p\ z_2 \equiv \exists u{\leq}z_2\ \exists v{\leq}z_2\ \ z_2 = (u \circ_p z_1) \circ_p v$
- Der Zugriff auf die i-te Komponente einer Folge z wird wie folgt berechnet
  $z\ @_p\ i \equiv \mu_{j\leq z}[\ ((p\dot{-}1) \circ_p i) \circ_p j\ \ part_p\ \ z\ ]$
  Die Minimierung zählt also die möglichen Zahlen durch, die als Komponente an der Stelle i stehen können, bis das richtige Element identifiziert ist.

Lemma 16 *Es gibt eine min-rekursive Funktion $\beta{:}\mathbb{N}^2{\to}\mathbb{N}$ so daß für jede Zahlenfolge $z_0, z_1, \ldots, z_y$ eine Zahl $\hat{z}$ existiert mit der Eigenschaft $\beta(\hat{z}, i) = z_i$ für alle $i{\leq}y$.*

Beweis: Wir definieren $\beta(\hat{z}, i) \equiv \pi_1(\hat{z})\ @_{\pi_2(\hat{z})}\ i$.
Damit ist β min-rekursiv. Um zu beweisen, daß β die gewünschten Eigenschaften besitzt, sei $z_0, z_1, \ldots, z_y$ eine endliche Folge natürlicher Zahlen und p eine Primzahl, die größer ist als y und alle z_i, und $p' = p\dot{-}1$.
Wir wählen $z = p' \circ_p 0 \circ_p z_0 \circ_p p' \circ_p 1 \circ_p z_1 \ldots \circ_p p' \circ_p y \circ_p z_y$ und $\hat{z} \equiv \langle z, p\rangle$
Dann ist $\beta(\hat{z}, i) = z\ @_p\ i = z_i$ für alle $i{\leq}y$. □

Um die Rolle von β als Komponentenzugriffsfunktion besser zu kennzeichnen, verwenden wir die Notation $\hat{z}@i$ anstelle von $\beta(\hat{z}, i)$. Damit können wir nun beweisen, daß min-rekursive Funktionen abgeschlossen unter primitiver Rekursion sind.

Lemma 17 *Die primitive Rekursion $Pr[f, g]{:}\mathbb{N}^k{\to}\mathbb{N}$ zweier Funktionen $f{:}\mathbb{N}^{k-1}{\to}\mathbb{N}$ und $g{:}\mathbb{N}^{k+1}{\to}\mathbb{N}$ ist min-rekursiv für alle $k \in \mathbb{N}$, wenn f und g min-rekursiv sind.*

Beweis: Es sei $h = Pr[f, g]$ und $(\bar{x}, y) \in \mathbb{N}^k$. Dann gibt es nach Lemma 16 eine Zahl $\hat{z}$ mit der Eigenschaft $\hat{z}@i = h(\hat{x}, i)$ für alle $i{\leq}y$. Insbesondere liefert $\hat{z}@y = h(\hat{x}, y)$ das gewünschte Funktionsergebnis.
Für diese Zahl gilt also $\hat{z}@0{=}f(\hat{x})$ und $\forall i{<}y\ \hat{z}@(i{+}1){=}g(\hat{x}, 0, \hat{z}@i)$. Da f, g, $t_=$ und beschränkte Quantifizierung min-rekursiv ist, kann diese Eigenschaft mit min-rekursiven Mitteln geprüft werden, und somit die Zahl $\hat{z}$ durch Minimierung bestimmt werden. Insgesamt erhalten wir
$h(\hat{x}, y) = \mu_{\hat{z}}[\hat{z}@0{=}f(\hat{x}) \wedge \forall i{<}y\ \hat{z}@(i{+}1){=}g(\hat{x}, 0, \hat{z}@i)] @ y$
und damit ist h min-rekursiv. □

Nach diesen Vorarbeiten sind wir nun in der Lage, zu beweisen, daß min-rekursive Funktionen genauso ausdrucksstark sind wie rekursive Funktionen.

Theorem 18 *$\mathcal{R}{\subseteq}\mathcal{R}_{min}$: Jede rekursive Funktion ist auch min-rekursiv.*

Beweis: Wir beweisen die Behauptung durch strukturelle Induktion über den Aufbau der μ-rekursiven Funktionen.

- Per Definitionen sind alle μ-rekursiven Grundfunktionen auch min-rekursiv.
- Es sei gezeigt, daß alle μ-rekursiven Funktionen, die durch n Schachtelungen von Komposition, primitiver Rekursion und Minimierung entstanden sind, auch min-rekursiv sind.
- Ist h μ-rekursiv und durch $n+1$ Schachtelungen von Komposition, primitiver Rekursion und Minimierung entstanden, dann ist $h{=}f{\circ}(g_1 \ldots g_n)$, $h{=}Pr[f, g]$ oder $h{=}\mu\, f$, wobei f, g und die g_i rekursive Funktionen der maximalen Schachtelungstiefe n sind, und somit nach Induktionsannahme auch min-rekursiv sind.
  Dann ist h nach Definition bzw. nach Lemma 17 ebenfalls min-rekursiv. □

Um zu zeigen, daß die Menge der arithmetisch repräsentierbare Funktionen Turingmächtig ist, müssen wir nun noch beweisen, daß jede min-rekursive Funktion arithmetisch repräsentierbar ist. Hierzu verwenden wir die Erkenntnisse, die wir unmittelbar vor der Definition der min-rekursiven Funktionen auf Seite 19 gesammelt haben.

Theorem 19 *Alle min-rekursiven Funktionen sind arithmetisch repräsentierbar.*

Beweis: Wieder beweisen wir die Behauptung durch strukturelle Induktion.

– Alle min-rekursiven Grundfunktionen sind arithmetisch repräsentierbar
  · Die *Nachfolgerfunktion* wird repräsentiert durch die Formel $R_{\mathsf{s}}(x,y) \equiv y = \mathsf{s}(x)$
  · Die *Addition* wird repräsentiert durch $R_{+}(x_1,x_2,y) \equiv y = x_1{+}x_2$
  · Die *Multiplikation* wird repräsentiert durch $R_{*}(x_1,x_2,y) \equiv y = x_1{*}x_2$
  · *Projektionsfunktion* pr^n_k wird repräsentiert durch $R_{pr^n_k}(x_1,..,x_n,y) \equiv y = x_k$
  · *Konstantenfunktion* c^n_k wird repräsentiert durch $R_{c^n_k}(x_1,..,x_n,y) \equiv y = \overline{k}$
– Es sei gezeigt, daß alle min-rekursiven Funktionen, die durch n Schachtelungen von Komposition und Minimierung entstanden sind, arithmetisch repräsentierbar sind.
– Ist h μ-rekursiv und durch $n{+}1$ Schachtelungen von Kompositionund Minimierung entstanden, dann ist $h{=}f{\circ}(g_1...g_k)$ oder $h{=}\mu\, f$, wobei f und die g_i min-rekursive Funktionen der maximalen Schachtelungstiefe n sind, und somit nach Induktionsannahme auch arithmetisch repräsentierbar sind. Es seien R_f und $R_{g_1},\ldots,R_{g_k}$ die entsprechenden Repräsentationen der Funktionen $f, g_1, .., g_k$.

  Dann wird die Komposition $h{=}f{\circ}(g_1...g_k)$ durch folgende Formel repräsentiert, welche das Weiterleiten der Ergebnisse der g_i an f codiert.

$$R_h(x_1,..,x_m,y) \equiv \exists z_1,..z_k\ R_{g_1}(x_1,..x_m,z_1) \wedge \ldots \wedge R_{g_k}(x_1,..x_m,z_k) \wedge R_f(z_1,..z_k,y)$$

  Die Minimierung $h{=}\mu\, f$ wird durch folgende Formel repräsentiert:

$$R_h(x_1,..,x_m,y) \equiv \forall z{\leq}y\ \exists t\ R_f(x_1,..x_m,z,t) \wedge (t = 0 \Leftrightarrow z = y)$$

  Diese Formel codiert die Tatsache, daß $f(x_1,..x_m,z)$ für alle $z{\leq}y$ definiert ist und erstmalig eine Nullstelle an der Stelle y hat.

  Damit ist h in beiden möglichen Fällen arithmetisch repräsentierbar. □

Aus Theorem 18 und Theorem 19 folgt insgesamt

Theorem 20 *Jede rekursive Funktion ist repräsentierbar in $\mathcal{Q}$.*

Theorem 20 ist nicht nur eine Aussage über die Ausdruckskraft logischer Programme sondern spielt auch eine wichtige Rolle für den Zusammenhang zwischen mathematischer Logik und der Theorie der Berechenbarkeit. Es ist die Grundlage für eine Beschreibung logischer Methoden zur Konstruktion von Programmen aus logischen Spezifikationen und damit auch die Grundlage aller Logik-Programmiersprachen. Darüber hinaus ist es das Fundament für den Beweis einer Reihe von Unmöglichkeitsaussagen für die Logik und Arithmetik, wie z.B. die Unentscheidbarkeit der Prädikatenlogik, die Unvollständigkeit aller formalen Systeme für die Arithmetik, die Nicht-Axiomatisierbarkeit der Arithmetik und die Undefinierbarkeit des Wahrheitsbegriffs in der Arithmetik. Dies werden wir im folgenden Abschnitt zeigen.

4 Unlösbare Probleme der Logik

Die Tatsache, daß die Robinson Arithmetik ausdruckstark genug ist, um alle berechenbaren Funktionen repräsentieren zu können, bedeutet, daß viele unlösbare Probleme der Theorie der Berechenbarkeit sich auf die Prädikatenlogik übertragen lassen. In der Berechenbarkeitstheorie, die wir in einem separaten Artikel ausführlich besprechen, lassen sich Probleme formulieren, die nachweislich unentscheidbar oder sogar nicht einmal rekursiv aufzählbar sind, und damit von keinem Algorithmus gelöst werden können. Da sich diese Probleme innerhalb der Theorie $\mathcal{Q}$ beschreiben lassen, bedeutet dies, daß es für die Prädikatenlogik ebenfalls eine Reihe von Problemstellungen gibt, die sich algorithmisch nicht lösen lassen. Hierzu gehören insbesondere

Der Unentscheidbarkeitssatz von Church:
"*Die Prädikatenlogik erster Stufe ist unentscheidbar*", d.h. wahre und falsche Sätze lassen sich nicht durch ein Entscheidungsverfahren trennen.

Der Undefinierbarkeitssatz von Tarski:
"*Der Begriff der Wahrheit kann innerhalb der Arithmetik nicht definiert werden*"

Die Gödelschen Unvollständigkeitssätze:
"*Die Theorie der Arithmetik ist nicht vollständig axiomatisierbar*", d.h. es gibt keine konsistentes aufzählbares Axiomensystem, mit dem alle gültigen arithmetischen Aussagen formal bewiesen werden können.

"*Keine hinreichend ausdruckstarke formale Theorie kann ihre eigene Konsistenz beweisen*", d.h. wenn eine Theorie mindestens so mächtig ist wie $\mathcal{Q}$, dann kann man innerhalb dieser Theorie nicht beweisen, daß sie keine Widersprüche enthält.

Diese Erkenntnisse bedeuten, daß es nicht möglich ist, Mathematik mit rein formalen Mitteln zu betreiben, da man für jede formale Theorie eine mächtigere mathematische Theorie benötigt, um zu beweisen, daß die formale Theorie keine falschen Ergebnisse liefern kann. Die Konsistenz dieser mächtigeren Theorie muß aber wieder mit externen Mitteln gezeigt werden, so daß man im Endeffekt kein zuverlässiges formales Fundament für die Mathematik angeben kann. Selbst die Konsistenz der Mengentheorie, die von Vielen als Fundament der Mathematik angesehen wird, kann nicht gezeigt werden, ohne eine noch fundamentalere Mathematik zugrunde zu legen.

Aufgrund der Vorarbeiten der vorhergehenden Abschnitte sind wir nun in der Lage, diese Aussagen direkt innerhalb der Logik – also ohne Querverweise auf die Theorie der Berechenbarkeit zu beweisen.

4.1 Gödelnumerierungen und Diagonalisierung

Der Schlüssel zu all den obengenannten Resultaten ist ein fundamentale Zusammenhang, der erstmalig in den 1930er von dem Mathematiker Kurt Gödel erkannt wurde: es ist möglich, alle berechenbaren Funktionen mit einer einheitlichen Methode – der nun nach ihm benannten Gödelnumerierung – effektiv zu numerieren und es ist ebenso möglich, berechenbare Funktionen mithilfe von *Diagonalisierung* durch eine fiktiven

Tabelle aller berechenbaren Funktionen zu definieren. [10] Kombiniert man nun diese Methode mit der Annahme, daß Prädikatenlogik entscheidbar oder Arithmetik axiomatisierbar wäre, dann entsteht eine neue berechenbare Funktion, die sich von allen anderen berechenbaren Funktionen unterscheidet, was offensichtlich ein Widerspruch ist. Heraus folgt dann, daß gewisse Probleme in der Logik unlösbar sein müssen. Wir werden beide Methoden im folgenden kurz beschreiben.

Das Konzept der Gödelnumerierung beruht auf der Beobachtung, daß jeder Formalismus für berechenbare Funktionen im Endeffekt aus einer Menge von formalen Ausdrücken besteht, die jeweils das Verhalten einer berechenbaren Funktion beschreiben. Formale Ausdrücke sind im Endeffekt nur Text, der gewissen Vorschriften einer formalen Sprache genügt. Texte wiederum lassen sich der Reihe nach aufzählen. Man legt dazu für die verwendeten Einzelsymbole eine Reihenfolge fest und zählt dann erst die einelementigen Texte in der Reihenfolge der Symbole auf, dann die zweielementigen, dreielementigen usw. Für jeden der so erzeugten Texte prüft man nun, ob er einen formalen Ausdruck beschreibt. Wenn dies der Foll ist, dann ist die Nummer des Textes in der Aufzählungsreihenfolge die Gödelnummer des formalen Ausdrucks bzw. der zugehörigen berechenbaren Funktion. Diese Erkenntnis führt zu folgender Definition.

Definition 21 (Gödelnumerierung).
Eine Gödelnumerierung ist eine Abbildung $\nu{:}\mathcal{X}\to\mathbb{N}$ von einer Menge $\mathcal{X}$ von Ausdrücken auf die natürlichen Zahlen, welche die folgenden Bedinungen erfüllt.

1. *ν ist injektiv, d.h. verschiedene Ausdrücke erhalten verschiedene Gödelnummern.*
2. *ν ist berechenbar, d.h. die Gödelnummer eines Ausdrucks kann effektiv bestimmt werden.*
3. *Es ist effektiv entscheidbar, ob eine gegebene Zahl eine Gödelnummer darstellt.*

Es ist leicht zu sehen, daß die Sprachen der Robinson Arithmetik und der Peano Arithmetik Gödelnumerierungen besitzen. Codiert man die Variablen-, Funktions- und Prädikatssymbole durch den ASCII-Text, den man beim Hinschreiben der entsprechenden Namen verwendet, dann verwendet die Prädikatenlogik nur endlich viele verschiedene Symbole. Wir weisen nun jedem dieser Symbol eine Nummer zu, z.B. erhält (eine 0,) eine 1, $\forall$ eine 2, $\exists$ eine 3, usw. Damit entspricht jede Formel einer eindeutigen Zahlenfolge, welche die Folge der Symbol im zugehörigen Text repräsentiert.

Es gibt nun vielfältige Arten, Zahlenfolgen durch Zahlen zu codieren. Eine möglichkeit ist die oben beschriebene lexikographische Ordnung, eine weitere die im letzten Abschnitt verwendeten eindeutig decodierbaren Polynome, und eine dritte die Standardtupelfunktion $\langle \_,\_ \rangle$ (siehe Seite 20), die berechenbar und bijektiv ist und darüber hinaus berechenbare Umkehrfunktionen π_1 und π_2 besitzt. Die Standardtupelfunktion läßt sich auf komplexere Zahlentupel und Zahlenfolgen beliebiger Länge fortsetzen, indem man definiert:

$$\begin{aligned}\langle x\rangle^{1} &= x\\ \langle x_1,..,x_{k+1}\rangle^{k+1} &= \langle\langle x_1,..,x_k\rangle^{k}, x_{k+1}\rangle\\ \langle x_1..x_k\rangle^{*} &= \langle k, \langle x_1,..,x_k\rangle^{k}\rangle\end{aligned}$$

[10] Diese Idee wird sehr ausführlich in dem Artikel zur elementaren Berechenbarkeitstheorie diskutiert und an Beispielen illustriert. In diesem Abschnitt konzentrieren wir uns auf die ursprüngliche logische Formulierung dieser Methode.

Da diese Funktionen bijektiv und berechenbar sind und berechenbare Umkehrfunktionen besitzen, kann man mithilfe der Numerierung der Einzelsymbole jede Formel in eine Zahl umwandeln und jede Zahl in einen Text umwandeln, den man mithilfe der Syntaxregeln der Prädikatenlogik darauf überprüfen kann, ob er eine Formel darstellt oder nicht. Damit sind alle Bedingungen an eine Gödelnumerierung erfüllt.

Diagonalisierung ist eine Methode, um aus bestimmten Annahmen über unendliche Objekte wie Mengen oder Funktionen neue Objekte zu konstruieren, die sich widersprüchlich verhalten, und somit die Annahme zu widerlegen. Die Idee geht zurück auf Cantor's Beweis der Überabzählbarkeit der Menge aller totalen Funktionen auf $\mathbb{N}$. Nimmt man an, daß die Menge $\mathbb{N}\to\mathbb{N}$ abzählbar ist und bezeichnet die i-te Funktion mit f_i, dann kann man eine Funktion $f{:}\mathbb{N}\to\mathbb{N}$ definieren durch $f(i) = f_i(i)+1$. Diese Funktion kann nicht zu den aufgezählten Funktionen gehören, denn sonst wäre f ein f_j und es würde $f_j(j) = f(j) = f_j(j)+1$ gelten. Somit kann $\mathbb{N}\to\mathbb{N}$ nicht abzählbar sein.

Gödels wichtigste Erkenntnis war, daß dieses Argument auf berechenbare Funktionen übertragen werden kann, obwohl die Menge der berechenbaren Funktionen abzählbar ist. Verwendet man eine Gödelnumerierung der berechenbaren Funktionen, dann läßt sich zeigen, daß *Diagonalisierung über berechenbare Funktionen wieder eine berechenbare Funktion liefert*. Damit ist Diagonalisierung ein geeignetes Mittel, um die "*berechenbare* Abzählbarkeit", also die *Aufzählbarkeit* bestimmter Mengen zu widerlegen und daraus die Unentscheidbarkeit einiger mathematischer Probleme zu folgern.

Auch wenn die Idee der Berechenbarkeit in der Mathematik der 1920er Jahre bereits eine große Bedeutung hatte, gab es noch keine ausformulierte Berechenbarkeitstheorie. Daher enthält die ursprüngliche Gödelsche Diagonalkonstruktion natürlich keinerlei explizite Bezüge auf das Konzept der berechenbaren Funktionen. Sie basiert stattdessen ausschließlich auf formaler Logik und codiert das Schlüsselargument der Diagonalisierung als formale *Selbstreferenz*, also der Möglichkeit auf die eigene Beschreibung zuzugreifen. In Cantors Beweis bestand die Selbstreferenz darin, daß die Definition der Diagonalfunktion die Nummern der abgezählten Funktionen verwendet, und damit im Endeffekt auch an irgendeiner Stelle auf ihre eigene Nummer zugreifen muß. In Gödels logischer Formulierung kann Selbstreferenz mithilfe der Codierung einer Gödelnummer als Numeral, also als einem Bestandteil logischer Formeln, erreicht werden.

Wir bezeichnen im folgenden die Termdarstellung der Gödelnummer einer Formel X mit $\lceil X\rceil$. Es ist nicht relevant, wie diese Termdarstellung genau aussieht. Es reicht zu wissen, daß sie effektiv aus der Formel X berechnet werden kann. So könnte man für die Robinson-Arithmetik $\lceil X\rceil \equiv \overline{\nu_{\mathcal{Q}}(X)}$ definieren , wobei $\nu_{\mathcal{Q}}$ eine Gödelnumerierung der Terme von $\mathcal{Q}$ ist. Die Selbstreferenz besteht nun darin, das Numeral $\lceil X\rceil$ mit der Formel X in Bezug zu setzen. Wir nennen dies die *Diagonalisierung der Formel X*.

Definition 22 (Diagonalisierung von Formeln).
Es sei X eine Formel, in der die Variable x frei vorkommen kann.
Die Diagonalisierung von X ist die Formel $\exists x\ x{=}\lceil X\rceil \wedge X$.

Es ist leicht einzusehen, daß die Diagonalisierung einer Formel X effektiv berechnet werden kann. Hierzu bestimmt man zunächst die Gödelnummer $n{=}\nu(X)$ und setzt (den Text von) X mit dem Numeral $\overline{n}$, dem Gleichheitssymbol, logischen Konnektiven und dem Variablennamen x zusammen und generiert die Formel $\exists x\ x{=}\overline{n} \wedge X$.

Lemma 23 (Diagonalisierung ist berechenbar)
Es gibt eine berechenbare Funktion diag, so daß für alle $n \in \mathbb{N}$ gilt

$$\overline{n} = \lceil X \rceil \text{ impliziert } \overline{diag(n)} = \lceil \exists x\ x{=}\lceil X \rceil \wedge X \rceil,$$

d.h. wenn n die Gödelnummer einer Formel X ist, dann ist $diag(n)$ die Gödelnummer der Diagonalisierung von X.

Aus heutiger Sicht kann man die fundamentale Bedeutung dieser Aussage kaum noch erfassen, da wir aufgrund der Church'schen These informal beschriebene Algorithmen in Beweisen verwenden können. Im ursprünglichen Beweis von Gödel mußte die Aussage von Lemma 23 und der Beweis jedoch ausschließlich in Logik formuliert und sehr detailliert gestaltet werden. Hierzu benötigt man eine konkrete logische Theorie, welche in der Lage ist, die Funktion *diag* zu repräsentieren (also z.B. $\mathcal{Q}$ oder Peano Arithmetik), und eine konkrete Gödelnumerierung wie z.B. $\nu_{\mathcal{Q}}$. Die Aussage von Lemma 23 kann dann z.B. wie folgt formuliert werden

Es gibt ein Prädikat R_{diag}, so daß für alle $n, k \in \mathbb{N}$ gilt

$\models_{\mathcal{Q}} R_{diag}(\overline{n}, \overline{k})$, falls $n{=}\nu_{\mathcal{Q}}(X)$ für eine Formel X ist und $k{=}\nu_{\mathcal{Q}}(\exists x\ x{=}\lceil X \rceil \wedge X)$
$\models_{\mathcal{Q}} \neg R_{diag}(\overline{n}, \overline{k})$, sonst

Für den Beweis testet man bei Eingabe einer Zahl n zunächst, ob diese eine Formel X repräsentiert und generiert die Zahlencodierung der Liste der Symbole in X. Diese ergänzt man um die entsprechenden Codierungen von '$\exists$', 'x', ' ', 'x' '$=$' '$\overline{n}$' '$\wedge$' und berechnet die Codierung des Gesamtresultats. Diese informale Berechnung wird schließlich als Formel beschrieben, welche die Berechnung arithmetisch repräsentiert. Man kann sich gut vorstellen, wie aufwendig dies ist, wenn man jeden Schritt des Algorithmus präzise und ausschließlich in Logik beschreiben muß. Der Beweis lieferte jedoch eine wichtige Erkenntnis, welche der Ausgangspunkt für die Gödelschen Unvollständigkeitssätze und eine Reihe anderer Unmöglichkeitsaussagen war. In heutiger Denkweise kann man diese kurz wie folgt formulieren.

Korollar 24 *Die Funktion diag ist repräsentierbar in $\mathcal{Q}$.*

Mit der Repräsentation der Diagonalisierungsfunktion als Formel wird ein Weg eröffnet, für jede beliebige Formel dieser Theorie eine Art Fixpunkt zu bestimmen.

Lemma 25 (Diagonalisierungslemma)
Es sei $\mathcal{T}$ eine numerische Theorie, in der diag repräsentierbar ist, und B ein einstelliges Prädikat. Dann gibt es eine Formel G mit der Eigenschaft $\models_{\mathcal{T}} G \Leftrightarrow B(\lceil G \rceil)$.

Beweis: Es sei R_{diag} eine Repräsentation der Funktion *diag* in der Theorie $\mathcal{T}$ und F die Formel $\exists y\ R_{diag}(x,y) \wedge B(y)$ und n die Gödelnummer von F. Aufgrund der Definition von R_{diag} wissen wir, daß $\models_{\mathcal{T}} R_{diag}(\overline{n}, \overline{k})$ gilt, wenn k die Gödelnummer der Diagonalisierung von F ist.
Als Formel G wählen wir die Diagonalisierung von F, also $G \equiv \exists x\ x{=}\lceil F \rceil \wedge F$. Dann ist $G = \exists x\ x{=}\overline{n} \wedge (\exists y\ R_{diag}(x,y) \wedge B(y))$, was wiederum logisch äquivalent zur Formel $\exists y\ R_{diag}(\overline{n}, y) \wedge B(y)$ ist.
Um $\models_{\mathcal{T}} G \Leftrightarrow B(\lceil G \rceil)$ zu zeigen, sei g die Gödelnummer von G. Dann gilt $\models_{\mathcal{T}} R_{diag}(\overline{n}, \overline{g})$ (d.h. $R_{diag}(\overline{n}, \overline{g})$ ist "wahr") und wegen der Rechtseindeutigkeit von R_{diag} ist g die einzige Zahl mit dieser Eigenschaft. Damit gilt
$\models_{\mathcal{T}} G \Leftrightarrow \exists y\ R_{diag}(\overline{n}, y) \wedge B(y) \Leftrightarrow R_{diag}(\overline{n}, \overline{g}) \wedge B(\overline{g}) \Leftrightarrow B(\overline{g}) = B(\lceil G \rceil)$. □

Damit enthält jede numerische Theorie, welche die Funktion *diag* repräsentieren kann, eine Repräsentation eines generischen Fixpunktoperators ähnlich zu den Fixpunktkombinatoren des λ-Kalküls. Fixpunktoperatoren wiederum machen es möglich, widersprüchliche Formeln zu konstruieren, die äquivalent zu ihrer eigenen Negation sind. Dies wollen wir im folgenden Abschnitt zeigen.

4.2 Unvollständigkeiten in mathematischen Theorien

Mit dem Diagonalisierungslemma sind wir nun in der Lage zu zeigen, daß jede formale mathematische Theorie in irgendeinem Sinne beschränkt ist. Entweder ist sie nicht sehr ausdrucksstark, d.h. manche gültigen mathematischen Aussagen können in ihr nicht bewiesen werden, oder sie ist inkonsistent, also widersprüchlich in sich selbst. Um dies zu beweisen, präzisieren wir zunächst den nötigen mathematischen Begriffsapparat.

Definition 26.
Es seien $\mathcal{T}$ und $\mathcal{T}'$ Theorien.

1. *$\mathcal{T}$ heißt konsistent, wenn es kein $\mathcal{T}$-Theorem gibt, dessen Negation ebenfalls ein $\mathcal{T}$-Theorem ist.*
2. *$\mathcal{T}$ heißt vollständig, wenn für jede $\mathcal{T}$-Formel X entweder X oder $\neg X$ ein $\mathcal{T}$-Theorem ist.*
3. *Eine Menge S von $\mathcal{T}$-Formeln heißt entscheidbar, wenn die Menge aller Gödelnummern von S entscheidbar ist, also wenn die charakteristische Funktion von $\{\nu_{\mathcal{T}}(X) \mid X \in S\}$ berechenbar ist.*
   Die Theorie $\mathcal{T}$ ist entscheidbar, wenn die Menge der $\mathcal{T}$-Theorem entscheidbar ist.
4. *$\mathcal{T}$ heißt axiomatisierbar, wenn es eine entscheidbare Teilmenge von $\mathcal{T}$ gibt, deren logische Folgerungen genau die $\mathcal{T}$-Theoreme sind.*
   $\mathcal{T}$ heißt endlich axiomatisierbar, wenn $\mathcal{T}$ mit einer endlichen Menge von Axiomen axiomatisierbar ist.
5. *Eine Menge $S{\subseteq}\mathbb{N}^k$ heißt definierbar in $\mathcal{T}$, wenn es in der formalen Sprache von $\mathcal{T}$ eine $k{+}1$-stelliges Prädikat R_S gibt, so daß für alle $i_1, .., i_k \in \mathbb{N}$ gilt*

$$(i_1, .., i_k) \in S \text{ impliziert } \models_{\mathcal{T}} R_S(\overline{i_1}, .., \overline{i_k})$$
$$\text{und } (i_1, .., i_k) \notin S \text{ impliziert } \models_{\mathcal{T}} \neg R_S(\overline{i_1}, .., \overline{i_k})$$

6. *$\mathcal{T}'$ ist eine Erweiterung von $\mathcal{T}$. wenn jedes $\mathcal{T}$-Theorem auch ein Theorem in $\mathcal{T}'$ ist.*
7. *Die Theorie der Arithmetik ist die Theorie mit vordefinierten Symbolen $\bar{0}, \mathsf{s}, +, *$, deren Theoreme genau die Formeln sind, die unter der Standardinterpretation der natürlichen Zahlen wahr sind.*

Definierbarkeit in $\mathcal{T}$ ist eng verwandt mit Repräsentierbarkeit. Wenn nämlich die charakteristische Funktion χ_S von S durch as Prädikat R_χ repräsentiert wird, dann gilt $\models_{\mathcal{T}} R_\chi(\overline{n}, \overline{1})$ für alle $n \in S$ und $\models_{\mathcal{T}} \neg R_\chi(\overline{n}, \overline{1})$ für alle $n \notin S$. Dies bedeutet, daß das Prädikat R_S mit $R_S(X) \equiv R_\chi(x\overline{1})$ die Menge S in $\mathcal{T}$ definiert.

Korollar 27 *Eine Menge $S{\subseteq}\mathbb{N}^k$ ist genau dann definierbar in $\mathcal{T}$, wenn ihre charakteristische Funktion χ_S in $\mathcal{T}$ repräsentierbar ist.*

Da alle charakteristischen Funktionen entscheidbarer Mengen auch berechenbar sind, bedeutet dies, daß *alle entscheidbaren Mengen in $\mathcal{Q}$ definierbar* sind und ebenso in jeder anderen Theorie, welche die berechenbaren Funktionen repräsentieren kann.

Repräsentierbarkeit und Definierbarkeit bleibt erhalten unter Theorie-Erweiterungen. Dabei bleibt sogar das repräsentierende Prädikat bestehen, d.h. wenn f in $\mathcal{T}$ durch ein Prädikat R_f repräsentiert wird und $\mathcal{T}'$ eine Erweiterung von $\mathcal{T}$ ist, dann ist R_f auch eine Repräsentation von f in $\mathcal{T}'$. *Arithmetik ist eine Erweiterung der Peano-Arithmetik*, da die Axiome der Peano Arithmetik offensichtlich in der tandardinterpretation der natürlichen Zahlen wahr sind. *Peano-Arithmetik ist eine Erweiterung der Robinson Arithmetik* , da alle Axiome von $\mathcal{Q}$ Theoreme der Peano Arithmetik sind. Damit überträgt sich Repräsentierbarkeit und Definierbarkeit von der Theorie $\mathcal{Q}$ auch auf die Peano Arithmetik und die Arithmetik.

Es gibt jedoch bestimmte Arten von Funktionen und Mengen, die sich nicht mehr repräsentieren bzw. definieren lassen, sobald eine Theorie zu ausdrucksstark ist. Eine Theorie, die alle berechenbaren Funktionen repräsentieren kann, ist nicht fähig, die Menge ihrer eigenen Theoreme durch eine Formel zu definieren.

Lemma 28 *Es sei $\mathcal{T}$ eine konsistente Theorie, in der diag repräsentierbar ist. Dann ist die Menge der Gödelnummern von $\mathcal{T}$-Theoremen nicht in $\mathcal{T}$ definierbar.*

Beweis: Wir nehmen an, daß die Menge der Gödelnummern von $\mathcal{T}$-Theoremen durch das Prädikat $R_{\mathcal{T}}$ definiert werden kann. Dann gilt $\models_T R_{\mathcal{T}}(\lceil X \rceil)$ für eine beliebige $\mathcal{T}$-Formel X, wenn X ein $\mathcal{T}$-Theorem ist und ansonsten $\models_T \neg R_{\mathcal{T}}(\lceil X \rceil)$. Nach dem Diagonalisierungslemma gibt es außerdem eine Formel G mit der Eigenschaft $\models_T G \Leftrightarrow \neg R_{\mathcal{T}}(\lceil G \rceil)$.
Dann ist G ein $\mathcal{T}$-Theorem, da ansonsten $\models_T \neg R_{\mathcal{T}}(\lceil G \rceil)$ gelten muß, woraus wiederum $\models_T G$ folgt. Da G ein $\mathcal{T}$-Theorem ist, muß auch $\models_T R_{\mathcal{T}}(\lceil G \rceil)$ gelten, woraus $\models_T \neg G$ folgt. Damit sind G und $\neg G$ gleichzeitig $\mathcal{T}$-Theorem, was der Konsistenz von $\mathcal{T}$ widerspricht. □

Der Beweis von Lemma 28 hat eine große Ähnlichkeit zum Cantorschen Diagonalbeweis, den wir auf Seite 26 skizziert hatten. Die Annahme, daß $R_{\mathcal{T}}$ die Menge der $\mathcal{T}$-Theoreme definieren kann, entspricht der Annahme, daß man jeder Funktion auf den natürlichen Zahlen eine Nummer zuweisen kann. Die Anwendung des Diagonalierungslemmas auf die Negation von $R_{\mathcal{T}}$ entspricht der Idee, eine neue Funktion f diagonal durch alle Funktionen f_i zu konstruieren und dabei jeweils an der Stelle i eine Abweichung zu erzeugen, also $f(i) \equiv f_i(i)+1$ zu definieren. Der Widerspruch ergibt sich nun, wenn man das durch Diagonalisierung erzeugte Objekt analysiert. In Cantors Beweis folgt aus der Tatsache, daß f ein f_j sein muß, der Widerspruch $f(j)=f(j)+1$, während sich in obigem Beweis aus der Tatsache, daß $\neg R_{\mathcal{T}}$ einen Fixpunkt haben muß, eine logische Inkonsistenz ergibt. Damit kann der Beweis von Lemma 28 als logische Darstellung des ursprünglichen Cantorschen Arguments verstanden werden.

Eine unmittelbare Folge von Lemma 28 ist, daß die wahren arithmetischen Aussagen nicht durch eine arithmetische Formel repräsentiert werden können, da die Funktion *diag* in der Arithmetik repräsentierbar ist. Diese Erkenntnis wird of als *Tarski's Satz von der Undefinierbarkeit der arithmetischen Wahrheit* bezeichnet.

Theorem 29 (Undefinierbarkeit der arithmetischen Wahrheit (Tarski))
Die Menge der Gödelnummern wahrer arithmetischer Aussagen ist in der Arithmetik nicht definierbar.

Undefinierbarkeit ist eng verwandt mit Unentscheidbarkeit. Wenn eine Theorie $\mathcal{T}$ entscheidbar ist, dann bedeutet dies, daß es ein Verfahren gibt, für jede $\mathcal{T}$-Formel X zu entscheiden, ob sie ein $\mathcal{T}$-Theorem ist oder nicht. Hierzu muß man nur die Gödelnummer von X bestimmen und dann die charakteristische Funktion $\chi_{\mathcal{T}}$ für die Menge $\{\nu_{\mathcal{T}}(X) \mid X \in \mathcal{T}\}$ der Gödelnummern von $\mathcal{T}$-Theoremen aufrufen. Allerdings kann diese Funktion nur dann berechenbar sein, wenn $\mathcal{T}$ nicht sehr mächtig ist. Denn wenn $\mathcal{T}$ die berechenbaren Funktionen repräsentieren kann, dann würde die Berechenbarkeit von $\chi_{\mathcal{T}}$ nach Korollar 27 zur Folge haben, daß die Menge der Gödelnummern von $\mathcal{T}$-Theoremen in $\mathcal{T}$ definierbar ist, was aber nach Lemma 28 für konsistente Theorien nicht der Fall sein kann. Diese Erkenntnis wird in folgendem Satz zusammengefaßt.

Theorem 30 (Unentscheidbarkeit mächtiger Theorien)
Jede konsistente Theorie, welche die berechenbaren Funktionen repräsentieren kann, ist unentscheidbar.

Dies bedeutet, daß man für eine hinreichend mächtige Theorie nicht effektiv entscheiden kann, ob eine Formel ein Theorem ist oder nicht. Da bereits die Robinson Arithmetik mächtig genug ist, alle berechenbaren Funktionen zu repräsentieren, gilt die Aussage von Theorem 30 für die Theorie $\mathcal{Q}$ und jede konsistente Erweiterung davon, wie z.B. die Peano Arithmetik oder die Arithmetik.

Korollar 31 *$\mathcal{Q}$ und jede konsistente Erweiterung von $\mathcal{Q}$ ist unentscheidbar.*

Korollar 32 *Die Arithmetik ist unentscheidbar*

Da die Theorie $\mathcal{Q}$ ein endliches Axiomensystem besitzt, sind wir nun sogar in der Lage, die Unentscheidbarkeit der Prädikatenlogik zu beweisen. Wir müssen hierzu nur zeigen, daß die Entscheidbarkeit der Prädikatenlogik auch die Entscheidbarkeit der Robinson Arithmetik zur Folge hätte.

Theorem 33 (Unentscheidbarkeit der Prädikatenlogik (Church))
Die Gültigkeit von Formeln der Prädikatenlogik erster Stufe ist unentscheidbar.

Beweis: Es sei $Ax_{\mathcal{Q}}$ die Konjunktion aller Axiome der Theorie $\mathcal{Q}$. Dann ist eine Formel X genau dann ein $\mathcal{Q}$-Theorem, wenn die Formel $Ax_{\mathcal{Q}} \Rightarrow X$ in der Prädikatenlogik gültig ist.
Wenn prädikatenlogische Gültigkeit entscheidbar wäre, dann könnte man Gültigkeit in $\mathcal{Q}$ dadurch entscheiden, indem man zunächst eine Formel X in $Ax_{\mathcal{Q}} \Rightarrow X$ transformiert und hierauf das Entscheidungsverfahren für die Prädikatenlogik anwendet. Nach Korollar 31 ist $\mathcal{Q}$ jedoch unentscheidbar und damit ist auch die Prädikatenlogik unentscheidbar. □

Die Unentscheidbarkeit der Prädikatenlogik hat zur Folge, daß es keine Verfahren geben kann, mit denen man die Gültigkeit prädikatenlogischer Formeln automatisch überprüfen kann. Daher können computergestützte Beweisverfahren bestenfalls nach Beweisen *suchen* und somit bestätigen, daß eine Formel tatsächlich ein Theorem ist. Sie können jedoch die Gültigkeit einer Formel nicht widerlegen, da die Suche nach einer Widerlegung nicht immer eine Antwort findet.

Nun könnte man versuchen, eine Widerlegung dadurch zu finden, indem man die Formel zunächst negiert und dann hierfür einen Beweis sucht. Dies setzt jedoch voraus, daß die Prädikatenlogik *vollständig* ist, also daß die Negation einer Formel gültig ist, wenn die Formel selbst kein Theorem ist. Für vollständige Theorien führt die eben beschriebene Methode tatsächlich zu einer Entscheidung.

Lemma 34 *Jede axiomatisierbare vollständige Theorie ist entscheidbar.*

Beweis: Es sei $\mathcal{T}$ eine axiomatisierbare vollständige Theorie und Ax die entscheidbare Menge der Axiome von $\mathcal{T}$.

- Wenn $\mathcal{T}$ konsistent ist, dann kann man aufgrund der Entscheidbarkeit von Ax testen, ob eine gegebene Folge von $\mathcal{T}$-Formeln eine gültige logische Ableitung in $\mathcal{T}$ darstellt. Damit ist es möglich alle gültigen Ableitungen von $\mathcal{T}$-Formeln und somit auch alle $\mathcal{T}$-Theoreme aufzuzählen.
  Da $\mathcal{T}$ vollständig ist, wird diese Aufzählung für eine beliebige $\mathcal{T}$-Formel X entweder X oder $\neg X$ nach endlich vielen Schritten aufzählen und liefert damit eine Entscheidung, ob X gültig ist oder nicht.
- Wenn $\mathcal{T}$ inkonsistent ist, dann gibt es eine Formel X, für die sowohl X und $\neg X$ Theoreme in $\mathcal{T}$ und damit logische Folgerungen der Axiome sind. Damit ist auch $X \wedge \neg X$ ein $\mathcal{T}$-Theorem und hieraus folg, daß jede $\mathcal{T}$-Formel auch ein $\mathcal{T}$-Theorem ist. Damit ist $\mathcal{T}$ entscheidbar. □

Eine unmittelbare Konsequenz von Lemma 34 und der Tatsache, daß eine Theorie, welche alle berechenbaren Funktionen repräsentieren kann, unentscheidbar sein muß (Theorem 30), ist ausdrucksstarke Theorien nicht gleichzeitig konsistent, vollständig und axiomatisierbar sein können.

Theorem 35 (Erster Gödelscher Unvollständigkeitssatz)
Es gibt keine konsistente, vollständige und axiomatisierbare Theorie, welche alle berechenbaren Funktionen repräsentieren kann.

Da die Robinson Arithmetik konsistente, axiomatisierbar und hinreichend mächtig ist, kann sie somit nicht vollständig sein. Das gleiche gilt auch für jede axiomatisierbare und konsistente Erweiterung der Robinson Arithmetik. Da weder Inkonstenz noch unentscheidbare Axiomensysteme wünschenswerte Eigenschaften formaler Theorien sind, muß man bei ausdrucksstarken formalen Theorien notwendigerweise auf die Vollständigkeit verzichten. In diesen Theorien wird es also immer Sätze geben, die man weder beweisen noch widerlegen kann.

Korollar 36 *Keine konsistente axiomatisierbare Erweiterung von $\mathcal{Q}$ ist vollständig.*

Da die Arithmetik per Definition konsistent und vollständig ist, kann es für sie somit kein entscheidbares Axiomensystem geben. Dies bedeutet, daß jeder Versuch, ein computergestütztes Beweissystem für die Arithmetik zu entwickeln, von vorneherein zum Scheitern verurteilt ist. Man kann bestenfalls erwarten, ein hinreichend großes Fragment der Arithmetik zu axiomatisieren und somit einen Großteil der praktisch relevanten Fragestellungen mithilfe von automatischen Beweisverfahren zu lösen, muß sich aber im Klaren sein, daß es immer Theoreme geben wird, die mit diesem Verfahren nicht behandelt werden können.

Korollar 37 *Die Arithmetik ist nicht axiomatisierbar.*

Wegen der Unentscheidbarkeit der Prädikatenlogik ist übrigens auch die *Theorie der Prädikatenlogik unvollständig*, da sie offentischtlich endlich axiomatisierbar und konsistent ist. Dies ist aber nicht verwunderlich, da es für die Prädikatenlogik eine unendliche Vielfalt von Interpretationen gibt, die eine Formel erfüllen oder falsifizieren können. Damit gibt es natürlich auch viele logische Formeln, die in einigen, aber nicht in allen Interpretationen wahr werden. Diese Formeln sind somit nicht gültig und haben auch keine gültige Negation.

Gödels erster Unvollständigkeitssatz hat zur Folge, daß in hinreichend mächtigen Theorien wie z.B. der Arithmetik viele wahre Formeln nicht formal bewiesen werden können. Zu jedem Axiomensystem lassen sich arithmetisch gültige Formeln konstruieren, die mit diesem System nicht beweisbar sind. *Wahrheit und Beweisbarkeit sind also keineswegs dasselbe*.

Diese Erkenntnis führte zu der Frage, ob man denn wenigstens formal beschreiben könnte, welche Formeln in einer Theorie beweisbar sind, d.h. ob man Beweisbarkeit in einer Theorie durch ein Prädikat innerhalb der Theorie selbst repräsentieren könnte. Es hat sich jedoch herausgestellt, daß auch *Beweisbarkeit nicht vollständig definierbar* ist, da es nicht möglich ist, ein Beweisprädikat so zu formalisieren, daß die Widerspruchsfreiheit der Theorie durch dieses Prädikat beweisbar wird.

Definition 38 (Beweisprädikat).
Sei $\mathcal{T}$ eine Theorie. Ein einstelliges Prädikat $Prov$ heißt Beweisprädikat für $\mathcal{T}$, wenn für alle $\mathcal{T}$-Formeln X und Y die folgenden Bedingungen erfüllt sind

1. $\models_{\mathcal{T}} X$ *impliziert* $\models_{\mathcal{T}} Prov(\lceil X \rceil)$
2. $\models_{\mathcal{T}} Prov(\lceil X \Rightarrow Y \rceil) \Rightarrow (Prov(\lceil X \rceil) \Rightarrow Prov(\lceil Y \rceil))$
3. $\models_{\mathcal{T}} Prov(\lceil X \rceil) \Rightarrow Prov(\lceil Prov(\lceil X \rceil) \rceil)$

Die erste Bedingung besagt, daß *jedes $\mathcal{T}$-Theorem auch beweisbar* sein muß, also das Beweisbarkeitsprädikat hierfür gelten muß. Die zweite Bedingung ist ein *Modus Ponens für Beweisbarkeit* und die dritte verlangt, daß *Beweisbarkeit beweisbar sein* muß. Man beachte dabei, daß die zweite und dritte Bedingung stärkere Formulierungen sind als die erste, da sie verlangen, daß die Implikation *innerhalb* der Theorie $\mathcal{T}$ gelten muß, also ein Theorem der Theorie $\mathcal{T}$ ist, während die erste den Zusammenhang zwischen Gültigkeit und Beweisbarkeit nur auf der Meta-Ebene der Theorie herstellt. Eine Bedingung der Art $\models_{\mathcal{T}} Prov(\lceil X \rceil) \Rightarrow X$ kann nicht sinnvoll für alle $\mathcal{T}$-Formeln gefordert werden, da dies die Inkonsistenz der Theorie $\mathcal{T}$ zur Folge hätte. Das folgende Theorem zeigt nämlich, daß die Gültigkeit von $Prov(\lceil X \rceil) \Rightarrow X$ impliziert, daß X selbst ein $\mathcal{T}$-Theorem ist.

Theorem 39 (Satz von Löb)
Sei $Prov$ ein Beweisprädikat für eine Theorie $\mathcal{T}$, die alle berechenbaren Funktionen repräsentieren kann, und X eine beliebige $\mathcal{T}$-Formel.

Wenn $\models_{\mathcal{T}} Prov(\lceil X \rceil) \Rightarrow X$ gilt, dann ist X ein $\mathcal{T}$-Theorem.

Beweis: Es gelte $\models_{\mathcal{T}} Prov(\lceil X \rceil) \Rightarrow X$.

Aufgrund des Diagonalisierungslemmas (Lemma 25) besitzt das Prädikat B mit $B(x) \equiv Prov(x) \Rightarrow X$ eine Fixpunkt, d.h. es gibt eine Formel G mit der Eigenschaft $\models_{\mathcal{T}} G \Leftrightarrow (Prov(\lceil G \rceil) \Rightarrow X)$.

Da $Prov$ ein Beweisprädikat ist, folgt hieraus mit Bedingung (1)
$\models_{\mathcal{T}} Prov(\lceil G \Rightarrow (Prov(\lceil G \rceil) \Rightarrow X) \rceil)$ und mit (2)
$\models_{\mathcal{T}} Prov(\lceil G \rceil) \Rightarrow Prov(\lceil Prov(\lceil G \rceil) \Rightarrow X \rceil)$. und wieder mit (2)
$\models_{\mathcal{T}} Prov(\lceil G \rceil) \Rightarrow Prov(\lceil Prov(\lceil G \rceil) \rceil) \Rightarrow Prov(\lceil X \rceil)$.

Da $\models_{\mathcal{T}} Prov(\lceil G \rceil) \Rightarrow Prov(Prov(\lceil G \rceil))$ nach (2) gilt, können wir den mittleren Implikanten $Prov(Prov(\lceil G \rceil))$ fallen lassen und erhalten

$\models_{\mathcal{T}} Prov(\lceil G \rceil) \Rightarrow Prov(\lceil X \rceil)$, woraus mit der Annahme folgt

$\models_{\mathcal{T}} Prov(\lceil G \rceil) \Rightarrow X$, was nach dem Diagonalisierungslemma äquivalent ist zu
$\models_{\mathcal{T}} G$, was mit Bedingung (1) wiederum zu
$\models_{\mathcal{T}} Prov(\lceil G \rceil)$ führt. Mit $\models_{\mathcal{T}} Prov(\lceil G \rceil) \Rightarrow X$ folgt hieraus schließlich
$\models_{\mathcal{T}} X$ und damit ist X ein $\mathcal{T}$-Theorem. □

Eine unmittelbare Folgerung des Satzes von Löb ist, daß Beweisprädikate für konsistente Theorien nicht vollständig sein können, da es in einer hinreichend mächtigen Theorie nicht möglich sein ist, die eigene Konsistenz innerhalb der Theorie zu beweisen.

Theorem 40 (Zweiter Gödelscher Unvollständigkeitssatz)
Es sei $Prov$ ein Beweisprädikat für eine konsistente Theorie $\mathcal{T}$, welche alle berechenbaren Funktionen repräsentieren kann. Dann ist $\neg Prov(\lceil \overline{0}{=}\overline{1} \rceil)$ in $\mathcal{T}$ nicht ableitbar.

Beweis: Wir nehmen an, es gelte $\models_{\mathcal{T}} \neg Prov(\lceil \overline{0}{=}\overline{1} \rceil)$. Da per Definition die Negation einer Formel X äquivalent ist dazu, daß X den Widerspruch $\overline{0}{=}\overline{1}$ impliziert, folgt $\models_{\mathcal{T}} Prov(\lceil \overline{0}{=}\overline{1} \rceil) \Rightarrow \overline{0}{=}\overline{1}$. Mit dem Satz von Löb folgt hieraus $\models_{\mathcal{T}} \overline{0}{=}\overline{1}$, was wegen der Konsistenz von $\mathcal{T}$ nicht der Fall sein kann. □

Neben der *Undefinierbarkeit arithmetischer Wahrheit*, der *Unentscheidbarkeit der Logik*, der *Unvollständigket von Systemen der Arithmetik* haben wir somit die *Undefinierbarkeit von Beweisbarkeit* als viertes unlösbares Problem der Logik nachgewiesen.

Der Nachweis dieser Unlösbarkeiten versetzte dem Versuch, die Mathematik vollständig zu begründen und zu formalisieren, einen schweren Schlag. Es ist unmöglich, die Mathematik auf ein festes Fundament zu setzen, dessen Widerspruchsfreiheit zweifelsfrei bewiesen werden kann. Man muß daher die Korrektheit von gewissen formalen Systemen als gegeben annehmen und kann diese bestenfalls intuitiv begründen, denn beweisen läßt sie sich nicht. Damit bleibt die Frage nach einem einheitlichen Fundament der Mathematik leider ungeklärt, da es keineswegs unumstritten ist, welche mathematischen Fundamentalgesetze als unumstößliche Wahrheiten anzusehen sind und welche nicht. Dies gilt insbesondere für das Auswahlaxiom der Mengentheorie oder das hierzu äquivalente logische Gesetz vom ausgeschlossenen Dritten. Denn die Frage, ob eine beliebige Aussage wahr ist, wenn sie nicht falsch ist (formal $\models X \vee \neg X$), enthält implizit die Vision einer gewissen Vollständigkeit der Mengentheorie (“*entweder ist X ein Theorem oder $\neg X$*”). Aufgrund ihrer Ausdruckskraft kann die Mengentheorie aber nicht vollständig sein, wenn sie konsistent ist.

Die Gödelschen Unvollständigkeitssätze zeigen also, daß es nicht sinnvoll ist, die Mathematik als rein formales Gebilde ohne Bezug zur realen Welt anzusehen, Formale Theorien sollten immer nur als Hilfsmittel zur Beschreibung und schematischen Analyse eines Fragmentes der Mathematik betrachtet werden, die nicht für sich selbst stehen sondern ihre Begründung in der Erkenntnis der mathematischen Realität finden müssen.

Literatur

1. George S. Boolos, John P. Burgess, and Richard C. Jeffrey. Computability and Logic. 4th edition, Cambridge University Press 2002.
2. G. Cantor Über eine elementare Frage der Mannigfaltigkeitslehre. *Deutsche Mathematiker-Vereinigung* 1:75–78, 1890.
3. Kurt Gödel. Zur intuitionistischen Arithmetik und Zahlentheorie. *Ergebnisse eines mathematischen Kolloquiums (Wien)*, 4:34–38, 1932.
4. R. M. Robinson. An Essentially Undecidable Axiom System. *Proceedings of the International Congress of Mathematics*, 729-730, 1950.
5. Alfred Tarski, A. Mostowski, and R. M. Robinson. Undecidable theories. North Holland, 1953.

Darüber hinaus gibt es im Web jede Menge guter Manuskripte, die mit dem Suchstichwort *axiomatisierbare Theorie* zu finden sind.