Beweis einer schwachen Form des ersten Gödelschen Unvollständigkeitssatz - Skript zur Vorlesung 'Einführung in die mathematische Logik'

Merlin Carl

Abstract

Wir skizzieren den Beweis einer schwachen Form des ersten Gödelschen Unvollständigkeitssatzes.

1 Arithmetik

Die Sprache der Arithmetik ist $\mathcal{L}_{\mathrm{PA}} := \{+, \cdot, 0, 1, <\}$, wobei $+$ und $\cdot$ zweistellige Funktionszeichen, $0, 1$ Konstantenzeichen und $<$ ein zweistelliges Relationszeichen ist. Als 'wahre Arithmetik', TA, bezeichnen wir die Menge aller $\mathcal{L}_{\mathrm{PA}}$-Sätze, die (mit der üblichen Interpretation von $+, \cdot, 0, 1, <$) in $\mathbb{N}$ gelten.

Viele Begriffe und Behauptungen der Zahlentheorie erlauben eine unmittelbare Übersetzung in $\mathcal{L}_{\mathrm{PA}}$. Dabei empfiehlt es sich, bereits übersetzte Begriffe als Abkürzungen zu verwenden:

- 'm teilt n' $(m|n)$: $\exists x m \cdot x \equiv n$
- 'm und n sind teilerfremd': $\forall x((x|m \wedge x|n) \rightarrow x \equiv 1)$
- 'c ist der Abstand von a und b' $(c = |a - b|)$: $a + c \equiv b \vee b + c \equiv a$
- 'a und c sind kongruent modulo c' $(a \equiv_c b)$: $\exists d(d = |a - b| \wedge c|d)$
- 'p ist eine Primzahl' $(\mathrm{prim}(p))$: $(\neg p \equiv 1 \wedge \forall x(x|p \rightarrow (x \equiv 1 \vee x \equiv p)))$
- 'Jede natürliche Zahl ist Summe von 4 Quadratzahlen': $\forall x \exists y_1 \exists y_2 \exists y_3 \exists y_4 x \equiv (((y_1 \cdot y_1) + (y_2 \cdot y_2)) + (y_3 \cdot y_3)) + (y_4 \cdot y_4)$

- 'Jede gerade Zahl ist Summe zweier Primzahlen': $\forall x \exists y_1 \exists y_2((\text{prim}(y_1) \wedge \text{prim}(y_2) \wedge 2 \cdot x \equiv y_1 + y_2)$

Bei anderen Begriffen der Zahlentheorie ist die Ausdrückbarkeit in $\mathcal{L}_{\text{PA}}$ weniger offensichtlich: Man versuche sich etwa an 'x ist eine Potenz von 6' oder an 'Es existieren keine positiven natürlichen Zahlen x, y, z, n mit $n > 2$ so, dass $x^n + y^n = z^n$'. Mit etwas technischem Aufwand kann man aber auch solche Begriffe und Behauptungen behandeln: Wir kommen weiter unten dazu.

Hat man sich von der Ausdruckskraft von $\mathcal{L}_{\text{PA}}$ überzeugt, kann man es im wesentlichen als Gegenstand der Zahlentheorie ansehen, TA zu entscheiden. Nun ist TA nach Definition widerspruchsfrei und entscheidet jeden zahlentheoretischen Satz. Für einen zahlentheoretischen Beweisbegriff, also als Axiomatisierung der Zahlentheorie, ist TA als Grundlage aber ungeeignet: Denn um zu entscheiden, ob ein Satz ϕ ein Axiom ist (also zu TA) gehört, muss man genau das tun, was der Beweis doch erst leisten soll: Nämlich feststellen, ob ϕ in $\mathbb{N}$ gilt. Für ein sinnvolles Axiomsystem der Zahlentheorie muss es zumindest ein Verfahren geben, mit dem man feststellen kann, ob ein vorgelegter Satz ein Axiom ist. Wir führen daher folgenden, zunächst noch bloß anschaulichen und formal unbestimmten Hilfsbegriff ein:

Eine Menge X von $\mathcal{L}_{\text{PA}}$-Sätzen heißt 'entscheidbar', falls es ein Verfahren gibt, mit dem von einem beliebigen vorgelegten $\mathcal{L}_{\text{PA}}$-Satz festgestellt werden kann, ob $\phi \in X$.

Wir wollen zeigen: Es existiert keine entscheidbare Teilmenge $X \subseteq \text{TA}$, deren deduktive Hülle TA ist, d.h.: Zu jedem entscheidbaren $X \subseteq \text{TA}$, also jeder entscheidbaren Menge wahrer zahlentheoretischer Sätze existiert ein zahlentheoretischer Satz ϕ, so, dass $\mathbb{N} \models \phi$, aber $\text{TA} \not\vdash \phi$. Das überträgt sich dann auf alle Theorien, die in der Lage sind, zahlentheoretische Aussagen auszudrücken (z.B. Mengenlehre).

Tatsächlich werden wir etwas stärkeres zeigen, nämlich dass kein Verfahren existiert, das TA entscheidet. Wir werden sehen, dass sich die Nichtexistenz eines $X \subseteq \text{TA}$ wie oben daraus ergibt. Denn gäbe es ein solches X, so könnten wir ein Verfahren zur Entscheidung von TA angeben, das wie folgt funktioniert: Es sei ein $\mathcal{L}_{\text{PA}}$-Satz ϕ gegeben. Es werden systematisch nacheinander alle $\mathcal{L}_{\text{PA}}$-Ableitungen im Sequenzenkalkül (z.B. in lexikalischer Reihenfolge) aufgelistet. Da nach Annahme für jeden $\mathcal{L}_{\text{PA}}$-Satz ψ gilt, dass entweder $X \vdash \psi$ oder $X \vdash \neg\psi$, wird also irgendwann eine Ableitung gefunden, deren Endsequenz entweder $\Gamma\phi$ oder $\Gamma\neg\phi$ (mit $\Gamma \subseteq X$ endlich) ist. Da

X entscheidbar ist, können wir für jede Sequenz Γ überprüfen, ob $\Gamma \subseteq X$. Damit wissen wir dann aber, ob $\phi \in \mathrm{TA}$ oder $\neg\phi \in \mathrm{TA}$ und haben also entschieden, ob $\phi \in \mathrm{TA}$.

(Grobe) Grundidee des Beweises: Die Theorie der natürlichen Zahlen - also die Menge aller in $\mathbb{N}$ gültigen Sätze in der Sprache $\{0, 1, +, \cdot\}$ - ist 'zu kompliziert', um von einem endlichen Verfahren entscheidbar zu sein.

Wenn wir aus dieser ersten Intuition - 'zu kompliziert' - ein Argument machen wollen, müssen wir uns zunächst darüber im Klaren werden, was 'kompliziert' bedeutet: Wir brauchen einen präzisen mathematischen Begriff von von der 'Komplexität' eines Objektes.

2 Chaitin-Komplexität

Die Folge $s_1 := 12345678$ scheint weniger kompliziert zu sein als $s_2 := 15918352$, aber komplizierter als $s_3 := 11111111$. Warum? Ein Unterschied, den man feststellen wird, ist der: Will man jemand die Folge s_1 übermitteln, kann man das kurz tun, indem man sagt 'Zähle von 1 bis 8 in Einerschritten aufwärts'; bei der Folge s_3 reicht 'Schreibe 8 Einsen hintereinander'; bei s_2 scheint es keine wesentliche kürzere Möglichkeit zu geben, als die Folge Glied für Glied zu übermitteln. Wir entnehmen dieser Betrachtung eine erste Annäherung an den Begriff der Komplexität: Die Komplexität einer Symbolfolge s hängt von der Länge der kürzesten Beschreibung ab, die s eindeutig charakterisiert.

Natürlich ist der Begriff 'kürzeste Beschreibung' kaum mathematisch zu präzisieren, solange wir an Beschreibungen denken, die Menschen in natürlicher Sprache an andere Menschen richten: Nicht alle werden eine Beschreibung gleich verstehen, und ob eine Beschreibung überhaupt verstanden wird, wird stark von Vorwissen und evtl. Talent des Empfängers abhängen. Wir müssen also sowohl die 'Beschreibungssprache' als auch den 'Empfänger' 'standardisieren'. Es liegt nahe, als 'Empfänger' einen Computer und als 'Beschreibungssprache' eine passende Programmiersprache zu wählen. Da unsere Betrachtungen zur Vollständigkeit der Arithmetik nicht vom aktuellen Entwicklungsstand der Hardware oder auch den Grenzen physikalisch realisierbarer Rechenanlagen abhängen sollten, nehmen wir einen idealisierten Computer als Basis unserer Betrachtungen, also einen mit unbegrenztem Speicherplatz und unbegrenzter Lebensdauer; zudem erlauben wir beliebig lange endliche Rechnungen. Wir behandeln zunächst allgemein den Begriff des 'Verfahrens' und seine formale Präzisierung, das 'Registermaschinenprogramm'. Beides findet sich in den ersten beiden Abschnitten von Kapitel 10 von Ebbinghaus/Flum/Thomas' 'Einführung in die mathematische Logik'

Das führt uns nun zur folgenden Definition:

Definition 1. Ist P ein Registermaschinenprogramm, so bezeichnen wir mit $|P|$ die 'Länge' von P, d.h. die Anzahl der Zeichen in P. Dabei fassen wir Programmbefehle wie PRINT, LET etc. der Einfachheit halber als einzelne Zeichen auf.

Definition 2. Es sei s eine (endliche oder unendliche) Symbolfolge über einer endlichen Symbolmenge S. Unter der Chaitin-Komplexität von s, geschrieben $C(s)$, verstehen wir das kleinste $k \in \mathbb{N}$ so, dass ein Registerprogramm P existiert mit $|P| = k$ und P berechnet s.

Wenn wir über Mengen von Ausdrücken reden, ist dieser Begriff nur bedingt geeignet. Wir definieren daher weiter:

Definition 3. Es sei M eine Menge von endlichen Symbolfolgen über einer endlichen Symbolmenge S. Unter der Chaitin-Komplexität von M, geschrieben $C_{\text{set}}(M)$, verstehen wir das kleinste $k \in \mathbb{N}$ so, dass ein Registermaschinenprogramm P existiert mit $|P| = k$ und P entscheidet M. (Falls kein solches P existiert, setzen wir $C_{\text{set}}(M) = \infty$.)

Lemma 4. Es sei S ein endliches Alphabet. Zu jedem $k \in \mathbb{N}$ existieren dann nur endlich viele $s \in S^*$ mit $C(s) \leq k$. Folglich gilt für alle bis auf endlich viele $s \in S^*$, dass $C(s) > k$.

Proof. Es gibt nur endlich viele Zeichenfolgen der Länge k über einem endlichen Alphabet, folglich auch nur endlich viele Programme P mit $|P| = n$ für jedes $n \in \mathbb{N}$ und folglich auch nur endlich viele Programme P mit $|P| \leq k$ für jedes $k \in \mathbb{N}$. Da jedes Programm höchstens eine Symbolfolge ausgeben kann, gibt es also auch nur endlich viele Symbolfolgen in S^* mit Komplexität $\leq k$. □

Wir skizzieren nun ein Argument, warum die Menge der in $\mathbb{N}$ wahren S_{PA}-Sätze nicht entscheidbar ist. Wir werden dann sehen, wie daraus die Nichtexistenz einer wahren und vollständigen Erweiterung von PA folgt. Wir nehmen dazu zunächst an, dass sich '$C(s) = k$' in S_{PA} ausdrücken läßt. Wir werden im nächsten Abschnitt sehen, dass das tatsächlich der Fall ist.

Theorem 5. Es sei $S := \{0, 1\}$, S^* die Menge der endlichen Folgen von Elementen von S. Dann ist die Menge $\mathcal{C}$ der Paare (s, k) mit $s \in S^*$, $k \in \mathbb{N}$ und $C(s) = k$ ist nicht entscheidbar.

Proof. Angenommen, P wäre ein Programm, das $\mathcal{C}$ entscheidet; es sei $|P| = k$. Wir benutzen P, um ein weiteres Programm Q zu schreiben, das folgendes tut: Q zählt der Reihe nach alle Paare $(s, m) \in S^* \times \mathbb{N}$ auf und wendet auf jedes davon P an, bis ein (s, k) gefunden wird mit $(s, m) \in \mathcal{C}$ und $m > 10^{10+k}$; sobald es gefunden ist, wird s ausgegeben. Nach Lemma 4 existiert ein s mit $C(s) > 10^{10+k}$, daher wird so ein Paar schließlich gefunden werden.

Es ist nicht schwer, zu sehen, dass dieses Verfahren so implementiert werden kann, dass $|Q| < 10^{10+k}$: tatsächlich sind nur sehr überschaubar viele zustätzliche Zeilen Code erforderlich: Für ein Programm, das alle Paare $(s, k) \in S^* \times \mathbb{N}$ aufzählt, braucht man eine gewisse, fixe Anzahl A von Zeichen. Um zu gegebenem k die Zahl 10^{10+k} zu berechnen, braucht es ein weiteres Programm mit einer gewissen, fixen Anzahl B von Zeichen, gleiches gilt für den Vergleich zweier Zahlen mit Zeichenzahl C. All diese Algorithmen lassen sich leicht explizit hinschreiben und benötigen zusammen (erheblich) weniger als 10^9 Zeichen. Die Ausführung überlassen wir dem skeptischen und interessierten Leser als Übung.

Da wir Zahlen als Dezimalzahlen verarbeiten, genügt es, um die Zahl k in das Register R_i zu schreiben, die Anweisungsfolge LET $R_i = R_i + z_l$ an den Anfang des Programms zu setzen, wobei z_i die i-te Ziffer in der Dezimaldarstellung von k sei. OBdA können wir annehmen, dass k zu Beginn in R_0 geschrieben wird. Damit erhält man $\log(z)$ zusätzliche Zeilen, deren Zeichenzahl durch eine Konstante $D < 100$ nach oben begrenzt ist.

Insgesamt ist damit die Länge $|Q|$ von Q höchstens $2(A + B + C + D\log(k)) < 2(10^9 + \log(k)D) < 2 \cdot 10^9 + 200\log(k) < 10^{10}10^k = 10^{10+k}$.

Dann ist aber Q ein Programm mit $|Q| < 10^{10+k}$, das eine Zeichenfolge s mit Komplexität $> 10^{10+k}$ ausgibt, im Widerspruch zur Definition der Chaitin-Komplexität. □

Von hier aus sieht man die Unentscheidbarkeit der wahren Arithmetik wie folgt: Angenommen, Aussagen der Form $C(s) = k$ sind in S_{PA} tatsächlich formulierbar. Dann würde ein Entscheidungsverfahren für die wahre Arithmetik insbesondere alle Aussagen der Form $C(s) = k$ entscheiden und also die Menge der Paare (s, k), für die $C(s) = k$ wahr ist. Das widerspricht aber Theorem 5.

Daraus erhalten wir nun leicht das gewünschte Resultat:

Corollary 6. Es existiert kein entscheidbares $X \subseteq \mathrm{TA}$ so, dass $X \vdash \phi$ gdw. $\phi \in \mathrm{TA}$ für alle $\mathcal{L}_{\mathrm{PA}}$-Sätze ϕ.

Proof. (Skizze) Angenommen, X wäre eine solche Menge, P ein $\mathcal{L}_{\mathrm{PA}}$-Programm, das X entscheidet. Wir können P dann ebenfalls benutzen, um von endlichen Folgen von $\mathcal{L}_{\mathrm{PA}}$-Ausdrücken $(\phi_1, ..., \phi_n)$ festzustellen, ob ihre sämtlichen Elemente in X liegen, indem wir P sukzessive auf jedes Folgenglied anwenden. Wir wollen nun zeigen, dass wir aus diesem P ein Entscheidungsverfahren für TA gewinnen können, im Widerspruch zu Theorem 5 (bzw. der darauf folgenden Bemerkung): Unter Verwendung von P können wir ein Verfahren konstruieren, das für einen gegebenen $\mathcal{L}_{\mathrm{PA}}$-Satz ϕ alle endlichen Folgen von endlichen Folgen (sic!) von $\mathcal{L}_{\mathrm{PA}}$-Ausdrücken aufzählt und jede neue solche Folge daraufhin überprüft, ob sie (1) eine Ableitung im Sequenzenkalkül ist und (2) ihr letztes Element von einer der Formen $\Gamma\phi$ oder $\Gamma\neg\phi$ mit $\Gamma \subseteq X$ ist; im ersten Fall wird $\square$ ausgegeben, im zweiten 1, anschließend wird gehalten. Nun gilt entweder $\mathbb{N} \models \phi$ oder $\mathbb{N} \models \neg\phi$, also entweder $\phi \in \mathrm{TA}$ oder $\neg\phi \in \mathrm{TA}$. Nach Annahme über X existiert also eine endliche Folge $\Gamma \subseteq X$ so, dass $\vdash \Gamma\phi$ oder $\vdash \Gamma\neg\phi$. Da unser Verfahren sämtliche endlichen Ableitungen durchsucht, wird schließlich eine solche Ableitung gefunden werden, die mit $\Gamma\phi$ oder $\Gamma\neg\phi$, $\Gamma \subseteq X$, endet und es wird $\square$ ausgegeben, falls $\phi \in \mathrm{TA}$ und 1 andernfalls. Folglich haben wir ein Entscheidungsverfahren für TA angegeben. $\square$

Tatsächlich existiert zu jeder entscheidbaren Theorie $T \subseteq \mathrm{TA}$ eine natürliche Zahl K so, dass T keinen Satz der Form $C(s) > K$ beweisen kann. Insbesondere kann jedes solche T nur endlich viele Sätze dieser Form beweisen, obwohl unendlich viele davon wahr sind!

Es muss also nun darum gehen, zu sehen, dass diese Aussagen tatsächlich in der Sprache der Arithmetik ausdrückbar sind.

3 Codierung

Unser Ziel ist es, zu zeigen, dass wir in der Sprache der Arithmetik über den Ablauf von Programmen reden können, also Aussagen wie 'Das Programm P wird halten' oder 'Das Programm P wird den Wert w ausgeben' als zahlentheoretische Aussagen formulieren können.

Wir benötigen einige Werkzeuge, die aus den Grundvorlesungen bekannt sein sollten.

Erinnerung 1: (Chinesischer Restsatz) Es seien $a_1, ..., a_n \in \mathbb{N}$ paarweise teilerfremd, ferner $r_1, ..., r_n \in \mathbb{Z}$ beliebig. Dann existiert $R \in \mathbb{N}$ so, dass $R \equiv r_i \bmod a_i$ für alle $1 \leq i \leq n$.

Erinnerung 2: (Cantorsche Paarfunktion) Die Funktion $p : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$,

gegeben durch $p(m,n) = \frac{(m+n)(m+n+1)}{2} + n$ ist eine Bijektion zwischen $\mathbb{N} \times \mathbb{N}$ und $\mathbb{N}$, genannt 'Cantorsche Paarfunktion'.

Definition 7. Für $2 \leq n \in \mathbb{N}$ definieren wir rekursiv bijektive Funktionen $p_n : \mathbb{N}^n \to \mathbb{N}$: $p_2(a,b) = p(a,b)$, $p_{n+1}(z_1, ..., z_{n+1}) = p(z_1, p_n(z_2, ..., z_{n+1}))$.

Mit Induktion sieht man leicht, dass die p_i alle bijektiv sind.

Das Auslesen des i-ten Elements aus einem n-Tupel-Code läßt sich in der Sprache der Zahlentheorie leicht beschreiben:

Proposition 8. Es sei $a = p_n(a_1, ..., a_n)$. Für alle $1 \leq i \leq n$ gilt dann $a_i = x$ gdw. $\exists x_1, ..., x_{n-1} a = p_n(x_1, ..., x_{i-1}, x, x_i, ..., x_{n-1})$.

Proof. □

Wir schreiben nun kurz $\mathrm{Tpl}_{n,i}(a,x)$, falls x das i-te Element des durch a codierten n-Tupels ist.

Unser Ziel ist es, in der Sprache der Zahlentheorie nicht nur über natürliche Zahlen, sondern über endliche Folgen natürlicher Zahlen beliebiger Länge reden, d.h. insbesondere quantifizieren zu können. Dazu müssen wir endliche Folgen in geeigneter Weise codieren. Die iterierten Paarfunktionen, die wir oben definiert haben, haben den Nachteil, dass das Auslesen des i-ten Elements aus einer n-elementigen Folge zum einen sehr umständlich ist, vor allem aber die Auslesefunktion nicht in offensichtlicher Weise uniform in i und n definierbar ist. Den Schlüssel zur Lösung liefert die Verbindung des chinesischen Restsatzes mit dem folgenden Lemma:

Lemma 9. Zu beliebigen natürlichen Zahlen m, n existiert eine natürliche Zahl $N > m$ so, dass $N+1, 2N+1, ..., nN+1$ paarweise teilerfremd sind

Proof. Wähle $k \in \mathbb{N}$ groß genug, dass $k \cdot n! > m$ und setze $N = k \cdot n!$. Dann ist sicherlich $N > m$. Um zu sehen, dass $N+1, 2N+1, ..., nN+1$ paarweise teilerfremd sind, betrachte beliebige $1 \leq i < j \leq n$; wir wollen zeigen, dass $\mathrm{ggT}(iN+1, jN+1) = 1$. Das sieht man wie folgt: Es ist zunächst $\mathrm{ggT}(iN+1, jN+1) = \mathrm{ggT}(iN+1, (j-i)N)$. Da $\mathrm{ggT}(iN+1, N) = 1$, ist weiter $\mathrm{ggT}(iN+1, (j-i)N) = \mathrm{ggT}(iN+1, j-i)$. Wegen $1 \leq i < j \leq n$ ist aber $1 \leq j - i < n$, also $(j-i) | n!$, also $(j-i) | k \cdot n! = N$. Wieder wegen $\mathrm{ggT}(iN+1, N) = 1$ ist also $\mathrm{ggT}(iN+1, (j-i)) = 1$, d.h. $\mathrm{ggT}(iN+1, jN+1) = 1$. □

Wir können nun eine beliebige endliche Folge natürlicher Zahlen $(a_1, ..., a_n)$ codieren, indem wir N so wählen, dass $N+1, ..., nN+1$ paarweise teilerfremd sind und N größer ist als das Maximum der a_i; nach dem chinesischen Restsatz existiert dann ein A so, dass $A \equiv a_i \bmod iN+1$ und nach der Wahl

von N ist damit a_i eindeutig bestimmt als der kleinste positive Vertreter der Restklasse von A modulo $iN+1$, bzw. der einzige, der echt zwischen 0 und $iN+1$ liegt. Die Folge $(a_1, ..., a_n)$ ist damit also eindeutig bestimmt durch die Angabe von A, N und der Länge n und kann durch die natürliche Zahl $p_3(A, N, n)$ codiert werden. Wir nennen $p_3(A, N, n)$ einen Folgencode für $(a_1, ..., a_n)$.

Wir wollen nun sehen, dass elementare Operationen auf Folgen wie das Auslesen eines Elementes sich arithmetisch ausdrücken lassen.

Lemma 10. Es sei $c = p_3(A, N, n)$ ein Folgencode für $(a_1, ..., a_n)$. Dann ist a_i charakterisiert als das einzige x, für das folgender Ausdruck wahr wird, für den wir ab jetzt abkürzend $\mathrm{Aus}(c, i, x)$ schreiben:

$\exists A, N, n[c = p_3(A, N, n) \wedge A \equiv x \bmod iN + 1 \wedge x < N]$

Proof. □

Wir benötigen noch einige elementare Relationen zwischen Folgen: Zum einen die Relation $\mathrm{append}(s, j, s')$, die genau dann erfüllt ist, wenn die durch s' codierte Folge dadurch aus der durch s codierten Folge entsteht, dass man j anhängt, zum anderen die Relation $\mathrm{replace}(s, i, j, s')$, die genau dann erfüllt ist, wenn die durch s' codierte Folge dadurch aus der durch s codierten Folge ensteht, dass man das i-te Element von s durch j ersetzt, schließlich $\mathrm{last}(a, b)$, um auszudrücken, dass b das letzte Element der durch a codierten Folge ist.

$\mathrm{append}(s, j, s') := \exists A_1, N_1, n_1, A_2, N_2(s = p_3(A_1, N_1, n_1) \wedge s' = p_3(A_2, N_2, n_1 + 1) \wedge \forall i \exists x(i \leq n \rightarrow \mathrm{Aus}(s_1, i, x) \wedge \mathrm{Aus}(s_2, i, x)) \wedge \mathrm{Aus}(s', n_1 + 1, j))$

$\mathrm{replace}(s, i, j, s') := \exists A_1, N_1, n, A_2, N_2(s = p_3(A_1, N_1, n) \wedge s' = p_3(A_2, N_2, n) \wedge \mathrm{Aus}(s', i, j) \wedge \forall k \exists x((k \leq n \wedge k \neq i) \rightarrow (\mathrm{Aus}(s, k, x) \wedge \mathrm{Aus}(s', k, x))))$

$\mathrm{last}(s, j) := \exists A, N, n(s = p_3(A, N, n) \wedge \mathrm{Aus}(s, n, j))$

Wir können nun Registermaschinenprogramme als endliche Folgen natürlicher Zahlen codieren. Dazu definieren wir zunächst eine Codierung der einzelnen Befehle. Es sei $S = \{a_1, ..., a_n\}$ das zugrundeliegende Alphabet. OBdA sei bereits $a_i = i$ (andernfalls codieren wir a_i vermöge einer geeigneten Bijektion durch i).

Definition 11. Wir definieren die Funktion c von Registermaschinenkommandos auf natürliche Zahlen wie folgt:

- $c(\mathrm{LET}\ R_i = R_i + a_j) = p(1, p(i, j))$

- $c(\text{LET } R_i = R_i - a_j) = p(2, p(i, j))$
- $c(\text{IF } R_i = \square \text{ THEN } Z' \text{ ELSE } Z_1 \text{ OR ... OR } Z_n) = p(3, p(i, z))$, wobei z ein Code für $Z_1, ..., Z_n))$
- $c(\text{PRINT}) = p(4, 0)$
- $c(\text{STOP}) = p(5, 0)$

Es ist leicht zu sehen, dass c injektiv ist. Ist K das Kommando in Zeile Z, so codieren wir nun

Definition 12. Es sei nun P das Programm:
1 K_1
2 K_2
...
l K_l
Dann codieren wir P durch einen Code c für die Zahlenfolge $(p(1\ K_1), ..., p(\text{l}\ K_l))$.

Angenommen, P enthält als maximalen Registerindex m. Dann ist ein **Berechnungszustand** von P gegeben durch ein Tupel $(Z, r_0, ..., r_m)$, das zunächst die aktive Programmzeile Z und dann die Inhalte der Register $R_0, ..., R_m$ enthält. Wir codieren einen solchen Berechnungszustand durch $p(Z, c)$, wobei c ein Code für $r_0, ..., r_m)$ ist.

Eine Berechnung ist einfach eine Folge von Berechnungszuständen. Wir wollen nun sagen, was es heißt, 'Die Berechnung von P' zu sein. Dazu definieren wir zunächst, was der 'richtige nächste Schritt gemäß P nach dem Zustand $(Z, r_0, ..., r_m)$' ist.

Wir beginnen mit Codierungen der einzelnen Schritte; es sei im Folgenden l der Code einer Programmzeile in einem Programm P, das n Register benutze; ferner z und z' Berechnungszustände. Wir konstruieren für jeden Grundbefehl β einen S_{PA}-Ausdruck $\phi_\beta(l, z, z')$, der genau dann in $\mathbb{N}$ erfüllt ist, wenn l einen Befehl der Form β enthält und bei Ausführung von Zeile l im von z codierten Berechnungszustand sich als nächstes der von z' codierte Berechnungszustand ergibt.

(1) Nachfolgerbefehl: LET $R_i = R_j + a_j$:
$\phi_+(l, z, z') := \exists i, j, k[l = p(k, p(1, p(i, j))) \wedge$
$\exists r_1, r_2, u_1, u_2(z = p(k, r_1) \wedge z' = p(k, r_2) \wedge \text{Aus}(r_1, i, u_1) \wedge \text{Aus}(r_2, i, u_2) \wedge$
$\text{append}(u_1, j, u_2) \wedge$
$\forall 0 \leq i_1 \leq m \exists x(i_1 \neq i \rightarrow (\text{Aus}(r_1, i_1, x) \wedge \text{Aus}(r_2, i_1, x))))]$

(2) Dekrementierungsbefehl: LET $R_i = R_j - a_j$:
$\phi_-(l, z, z') := \exists i, j, k[l = p(k, p(1, p(i, j))) \wedge$
$\exists r_1, r_2, u_1, u_2(z = p(k, r_1) \wedge z' = p(k, r_2) \wedge \mathrm{Aus}(r_1, i, u_1) \wedge \mathrm{Aus}(r_2, i, u_2) \wedge$
$(\mathrm{append}(u_2, j, u_1) \vee \exists y_1, y_2(u_1 = p_3(y_1, y_2, 0))) \wedge$
$\forall 0 \leq i_1 \leq m \exists x(i_1 \neq i \rightarrow (\mathrm{Aus}(r_1, i_1, x) \wedge \mathrm{Aus}(r_2, i_1, x))))]$

(3) Bedingter Sprung: IF $R_i = \square$ THEN Z' ELSE Z_1 OR ... OR Z_n
$\phi_{\mathrm{jump}}(l, z, z') := \exists r, k, k', y, t, k, i, z[l = p(k, p(3, p(i, z))) \wedge z = p(k, r) \wedge z' =$
$p(k', r) \wedge \mathrm{Aus}(r, i, y) \wedge$
$[(\mathrm{last}(y, t) \wedge \mathrm{Aus}(z, t, k')) \vee (\exists y_1, y_2(y = p_3(y_1, y_2, 0) \wedge \mathrm{Aus}(z, 0, k')))]]$

(4) Druckbefehl: PRINT
$\phi_{\mathrm{print}}(l, z, z') := \exists j, x, z_1, z_1'[l = p(j, p(4, 0)) \wedge z = p(j, z_1) \wedge z' = p(j + 1, z')]$

(5) Stopbefehl: STOP
$\phi_{\mathrm{stop}}(l, z, z') := \exists j[l = p(j, p(5, 0)) \wedge z = z']$

Definition 13. Es sei $c(P)$ der Code eines Programms, ferner $b = p(Z, r)$ Code eines P-Berechnungszustandes. Wir arithmetisieren nun die Relation 'Berechnungszustand d folgt gemäß P auf Berechnungszustand b' durch

$$\mathrm{NextStep}(b, d, c(P)) := \exists l, Z, r[b = p(Z, r) \wedge \mathrm{Aus}(c(P), Z, l) \wedge (\phi_+(l, b, d) \vee \phi_-(l, b, d) \vee \phi_{\mathrm{jump}}(l, b, d) \vee \phi_{\mathrm{print}}(l, b, d) \vee \phi_{\mathrm{stop}}(l, b, d))]$$

Wir müssen noch den Startzustand eines Programms P, das k Register verwendet, arithmetisieren: Alle Register enthalten die leere Folge, die aktive Programmzeile ist die erste (ein Code für die leere Folge ist nach unserer Definition jede Zahl der Form $p_3(a, b, 0)$ mit $a, b \in \mathbb{N}$):

Definition 14. $\mathrm{Start}(k, z) := \exists r[z = p(1, r) \wedge \forall 1 \leq i \leq k \exists x, y \mathrm{Aus}(r, i, p_3(x, y, 0))]$.

Nun codiert eine natürliche Zahl p den Ablauf von des Programmes P, das k Register verwendet, bis zum s-ten Schritt genau dann, wenn p eine Folge der Länge s von Berechnungszuständen codiert, so dass zwischen jedem Zustand und seinem Nachfolger (falls er einen hat) die NextStep-Relation gilt und so, dass am Anfang der Startzustand vorliegt:

Definition 15. Wir setzen $\mathrm{COMP}(p, s, c(P), k) := \exists z(\mathrm{Aus}(p, 0, z) \wedge \mathrm{Start}(k, z)) \wedge$
$(\forall j < s \exists z_1, z_2 \mathrm{Aus}(p, j, z_1) \wedge \mathrm{Aus}(p, j + 1, z_2) \wedge \mathrm{NextStep}(z_1, z_2, c(P)))$.

Ein Programm P mit k Registern hält, wenn ein $s \in \mathbb{N}$ so existiert, dass der letzte Zustand der Berechnung bis s ein STOP-Zustand ist:

Definition 16. $\mathrm{HALT}(c(P), k) := \exists p \exists n[\mathrm{COMP}(p, n, c(P), k) \wedge \exists L, y(\mathrm{Aus}(p, n, p(L, y)) \wedge \mathrm{Aus}(c(P), L, p(L, p(5, 0))))]$ (L ist die im Berechnungsschritt n aktive Programmzeile; hier steht also: Im n-ten Schritt der P-Berechnung p ist die aktive Programmzeile L, und in der steht STOP)

Wir beschreiben nun die Beziehung zwischen einem Programmcode und dem von ihm berechneten (d.h. ausgegebenen) Ergebnis r; PRINT gibt den Inhalt des ersten Registers aus; da die Berechnung nur einmal drucken darf, nehmen wir OBdA an, dass sie nach dem Drucken stoppt, die Registerinhalte sich nach dem PRINT-Befehl bis zum STOP also nicht mehr ändern und die Ausgabe also mit dem Registerinhalt in R_0 im Stop-Zustand identisch ist.
$\mathrm{OUTPUT}(c(P), k, r) := \exists p, n, r'[\mathrm{COMP}(p, n, c(P), k) \wedge \exists L, y(\mathrm{Aus}(p, n, p(L, y)) \wedge \mathrm{Aus}(c(P), L, p(L, p(5, 0))) \wedge \mathrm{Aus}(y, 0, r) \wedge]$
$\exists a_0, b_0, k, a_1, b_1(r = p_3(a_0, b_0, k) \wedge r' = p_3(a_0, b_0, k) \wedge \forall 0 \leq i \leq k \exists x(\mathrm{Aus}(r, i, x) \wedge \mathrm{Aus}(r', i, x))))]$

Wir brauchen noch eine Arithmetisierung für die 'Länge eines Programms'; hierzu benötigen wir ein Prädikat $\mathrm{len}(c(P), k)$, das 'die Länge des durch $c(P)$ codierten Programms ist k' ausdrückt. Der Leser, der den Ausführen bis hierher gefolgt ist, sollte keine Zweifel haben, dass dieses Prädikat sich in $\mathcal{L}_{\mathrm{PA}}$ formulieren läßt; wir überlassen es als Übungsaufgabe zur Arithmetisierung, die Formulierung auszuführen.

Wir können nun für ein endliches Alphabet $\mathcal{A}$ die Aussage 's hat Chaitin-Komplexität k' arithmetisch formulieren (wobei wir s als Code einer endlichen $\mathcal{A}$-Folge interpretieren und OBdA annehmen, dass $\mathcal{A} = \{0, 1, ..., r\}$, $r \in \mathbb{N}$):

Definition 17. $C(s, k) := \exists P, l(\mathrm{len}(P, k) \wedge \mathrm{OUTPUT}(P, l, s)) \wedge \forall Q, l'(\mathrm{len}(Q) < k \rightarrow \neg \mathrm{OUTPUT}(Q, l', s))$

4 Konkrete Unvollständigkeit

'TA entscheiden' ist nur eine bedingt adäquate Beschreibung der Intentionen eines Zahlentheoretikers. Zu TA gehören etwa Sätze aus $10^{10^{10}}$ Zeichen, die mit 10^{10} Quantorenwechseln beginnen und faktisch von keinem Menschen gelesen oder verstanden werden können. Sicherlich kann es kaum als sinnvolle zahlentheoretische Frage angesehen werden, ob ein solcher Satz zu TA gehört. Falls das Unvollständigkeitsphänomen etwas mit 'echter' Mathematik zu tun haben und nicht nur ein Artefakt der logischen Modellierung sein soll,

stellt sich die Frage nach 'konkreter Unvollständigkeit': Gibt es zahlentheoretisch/kombinatorisch/... 'interessante' Sätze, die weder beweisbar noch widerlegbar sind?

Um diese Frage sinnvoll stellen zu können, muss man natürlich angeben, welches Axiomsystem man zugrunde legt: Nimmt man ϕ als Axiom, ist ϕ natürlich auch beweisbar. Es gibt aber eine Reihe 'kanonischer' Axiomsysteme, die eine Reihe grundlegender Einsichten über ihren Gegenstandsbereich (z.B. natürliche Zahlen oder Mengen) ausdrücken und in denen sich erfahrungsgemäss alle oder jedenfalls die meisten Beweise eines Gebietes formalisieren lassen. Für die Zahlentheorie leistet das z.B. die Peano-Arithmetik (PA):

PA besteht aus den Axiomen der sogenannten Robinson-Arithmetik Q:

1. $\forall x(x+1 \neq 0)$
2. $\forall x \forall y(x+1 = y+1 \rightarrow x = y)$
3. $\forall x(x \neq 0 \rightarrow \exists y(x = y+1))$
4. $\forall x(x+0 = 0)$
5. $\forall x \forall y(x+(y+1) = (x+y)+1)$
6. $\forall x(x \cdot 0 = 0)$
7. $\forall x \forall y(x \cdot (y+1) = (x \cdot y)+x)$
8. $\forall x \forall y(x \leq y \rightarrow \exists z(x+z = y))$

Und zusätzlich den (unendlich vielen) Induktionsaxiomen $\{\mathrm{Ind}_\phi : \phi \in \mathcal{L}_{\mathrm{PA}}\}$, wobei Ind_ϕ den Ausdruck

$$(\phi[\tfrac{0}{x}] \wedge \forall x(\phi \rightarrow \phi[\tfrac{x+1}{x}])) \rightarrow \forall x \phi$$

bezeichnet.

Die Erfahrung zeigt, dass sich zahlentheoretische Beweise, die in der Mathematik faktisch geführt werden, üblicherweise in PA formalisieren lassen. Wir können die Frage also z.B. so präzisieren: 'Gibt es zahlentheoretisch interessante $\mathcal{L}_{\mathrm{PA}}$-Sätze, die in PA weder beweisbar noch widerlegbar sind?' Nun ist die Frage, ob ein Satz 'interessant' ist, natürlich potenziell Geschmackssache. Es gibt aber tatsächlich einige Kandidaten für solche Aussagen, die wir hier kurz erläutern:

(1) Der Satz von Paris-Harrington: Es seien $1 \leq k, m, n, C$ natürliche Zahlen. Wir färben die n-Tupel, die aus verschiedenen Elementen von $\{0, 1, ..., k\}$

bestehen„ mit m Farben. Eine Teilmenge $A \subseteq \{0, 1, ..., k\}$ heißt 'einfarbig', falls alle n-Tupel aus verschiedenen Elementen von A die gleiche Farbe haben. Der Satz von Paris-Harrington besagt, dass es zu gegebenen m, n, C stets ein k gibt, so dass für jede solche Färbung eine einfarbige Menge $A \subseteq \{0, 1, ..., k\}$ existiert, die mindestens C Elemente hat und außerdem $|A| > \min(A)$ erfüllt. Der Satz von Paris-Harrington ist nicht in PA beweisbar.

(2) Der Satz von Goodstein: Um eine Zahl n 'vollständig in Basis b' darzutellen, stellen wir sie zunächst zur Basis b dar; falls dabei Exponenten $\geq b$ auftreten, stellen wir auch diese Exponenten in Basis b dar; falls dabei wiederum Exponenten $\geq b$ auftreten, stellen wir auch sie zur Basis b dar usw., bis die Darstellung keine Koeffizienten oder Exponenten $\geq b$ mehr vorkommen. So ist die Darstellung von 12 vollständig in Basis 2 z.B. so zu erhalten: $12 = 2^3 + 2^2 = 2^{2^1+2^0} + 2^{2^1}$. Eine Zahl m wird 'zur Basis $b > 1$ aufgeblasen', indem man m zunächst vollständig zur Basis b darstellt und anschließend in dieser Darstellung jedes b durch $b + 1$ ersetzt. So wird 12 zur Basis 2 z.B. zu $3^{3^1+3^0} + 3^{3^1} = 3^4 + 3^3 = 81 + 27 = 108$ aufgeblasen. Für die sich durch Aufblasen zur Basis b aus m ergebende Zahl schreiben wir $\text{Aufb}(m, b)$. Wir definieren nun, beginnend mit einer beliebigen natürlichen Zahl m und einer natürlichen Zahl b eine Folge $(a_i^{m,b} : i \in \mathbb{N})$ wie folgt: $a_0^{m,b} = m$, $a_{i+1}^{m,b} = \text{Aufb}(a_i^{m,b}, b + i) - 1$. Man beachte, dass das Aufblasen einer Zahl die Zahl im Allgemeinen erheblich wachsen läßt. Trotzdem sagt der Satz von Goodstein, dass **jede** solche Folge schließlich bei 0 endet: Für alle m, b existiert ein $n \in \mathbb{N}$ so, dass $a_n^{m,b} = 0$. Auch von diesem Satz ist bewiesen, dass er in PA nicht beweisbar ist.

(3) Der 'Tod der Hydra': Es sei T ein endlicher Baum; einen Knoten von T nennen wir die 'Wurzel' von T und bezeichnen ihn mit r. Wir stellen uns T als Hydra vor, die Blätter von T als Köpfe. im Kampf mit der Hydra sucht sich Herkules in jedem Schritt einen Kopf k der Hydra aus und schlägt ihn ab (d.h. entfernt ihn aus T). Falls dieser Kopf direkt an r saß, geht es weiter mit dem nächsten Schritt. Andernfalls sei b der Elternknoten von k (d.h. der Knoten von T, mit dem k verbunden ist) und c der (eindeutig bestimmte) mit k verbundene Knoten auf dem kürzesten Pfad in T von r zu k und T_b der größte Teilbaum von T, der b, aber nicht c enthält. Die wütende Hydra sucht sich dann eine natürliche Zahl n aus und läßt an c sofort n Kopien von B_b nachwachsen, ehe der Kampf weitergeht. Nach einem Satz von Kirby und Paris wird Herkules den Kampf am Ende immer gewinnen, egal, welche Köpfe er abschlägt oder wie viele Köpfe die Hydra in jedem Schritt nachwachen läßt; Kirby und Paris haben ferner bewiesen, dass dieser Satz in PA nicht beweisbar ist.

Solche Beispiele zeigen, dass durchaus verständliche und interessante Aussagen von den üblichen Axiomsystemen unabhängig sein können. Es mag durchaus sein, dass die eine oder andere noch offene Frage der Zahlentheorie (Goldbachsche Vermutung, Primzahlzwillingsvermutung, Riemannsche Vermutung, Collatzsche Vermutung,...) dazu gehört. Die Aussagen (1)-(3) sind aber (bei geeigneter Umformulierung) alle beweisbar im deutlich stärkeren Axiomsystem der Mengenlehre, ZFC. Ob es nun auch von ZFC unabhängige 'interessante' zahlentheoretische Aussagen gibt, ist eine derzeit diskutierte Frage. Einige Kandidaten für solche Aussagen liefert die 'Boolean Relation Theory' von Harvey Friedman.