Prof. Dr. H. Brenner **Osnabrück SS 2016**

Einführung in die mathematische Logik

Vorlesung 20

Arithmetische Repräsentierbarkeit

Wir möchten die Wirkungsweise von Registerprogrammen arithmetisch repräsentieren, um so aus der Unentscheidbarkeit des Halteproblems auf die Unentscheidbarkeit der Arithmetik zu schließen. Im Folgenden arbeiten wir mit dem arithmetischen Alphabet $\mathrm{Ar} = \{0, 1, +, \cdot\}$ und der Standardinterpretation in $\mathbb{N}$.

Definition 20.1. Eine Abbildung

$$F\colon \mathbb{N}^r \longrightarrow \mathbb{N}^s$$

heißt *arithmetisch repräsentierbar*, wenn es einen L^{Ar}-Ausdruck ψ in $r+s$ freien Variablen derart gibt, dass für alle $(r+s)$-Tupel $(n_1, \ldots, n_{r+s}) \in \mathbb{N}^{r+s}$ die Äquivalenz $F(n_1, \ldots, n_r) = (n_{r+1}, \ldots, n_{r+s})$ genau dann, wenn $\mathbb{N} \vDash \psi(n_1, \ldots, n_{r+s})$ gilt.

Definition 20.2. Eine Relation $R \subseteq \mathbb{N}^r$ heißt *arithmetisch repräsentierbar*, wenn es einen L^{Ar}-Ausdruck ψ in r freien Variablen derart gibt, dass für alle r-Tupel $(n_1, \ldots, n_r) \in \mathbb{N}^r$ die Äquivalenz $(n_1, \ldots, n_r) \in R$ genau dann, wenn $\mathbb{N} \vDash \psi(n_1, \ldots, n_r)$ gilt.

Die Schreibweise $\psi(n_1, \ldots, n_{r+s})$ bedeutet, dass die $r+s$ nicht namentlich aufgeführten freien Variablen durch die $n_1, \ldots, n_{r+s}$ ersetzt werden, wobei die natürlichen Zahlen n_j als Terme durch die n_j-fache Summe $1+\cdots+1$ (streng genommen mit einer fixierten Klammerung) wiedergegeben werden. Da die repräsentierenden Ausdrücke genau $r+s$ bzw. r freie Variablen besitzen, entsteht durch Substitution der freien Variablen durch die Terme eine Aussage ohne freie Variablen. Diese sind bei Interpretation über den natürlichen Zahlen wahr oder falsch. Aufgrund des Substitutionslemmas ist die Gültigkeit $\mathbb{N} \vDash \psi(n_1, \ldots, n_{r+s})$ äquivalent zur Gültigkeit $\mathbb{N}_{x_1,\ldots,x_{r+s}}^{n_1,\ldots,n_{r+s}} \vDash \psi$. Polynomfunktionen

$$\mathbb{N}^r \longrightarrow \mathbb{N}$$

mit natürlichzahligen Koeffizienten sind unmittelbar arithmetisch präsentierbar.

Wir wollen zeigen, dass Registerprogramme, oder besser gesagt die durch ein Registerprogramm festgelegte Programmabbildung, arithmetisch repräsentierbar sind.

Registerprogramme als Abbildungen

Ein Registerprogramm P, das aus h Programmzeilen besteht und m Register anspricht, möchten wir als eine Abbildung auffassen. Die Wirkungsweise einer jeden Programmzeile hängt dabei nur von den Belegungen der Register zu dem Zeitpunkt ab, an dem diese Zeile aufgerufen wird. Sie ist geschichtsunabhängig, d.h. unabhängig von dem bisherigen Verlauf des Programmes. Man kann daher ein Programm vollständig durch die Abbildung

$$\varphi : \{1, 2, \ldots, h\} \times \mathbb{N}^m \longrightarrow \{1, 2, \ldots, h\} \times \mathbb{N}^m,$$
$$(\ell, n_1, \ldots, n_m) \longmapsto \varphi(\ell, n_1, \ldots, n_m),$$

erfassen. Diese Abbildung nennen wir die *Programmabbildung* $\varphi = \varphi_P$. Dabei steht ℓ für die Programmzeilennummer und n_j steht für den Inhalt des Registers R_j (von denen es ja m Stück gibt). Dem Tupel $(\ell, n_1, \ldots, n_m)$ wird dasjenige Tupel $\varphi(\ell, n_1, \ldots, n_m)$ zugeordnet, das bei Abruf des in der ℓ-ten Programmzeile stehenden Befehls B_ℓ bei der Registerbelegung $(n_1, \ldots, n_m)$ entsteht. Die Abbildung φ besteht dabei aus den $m + 1$ Komponentenfunktionen $\varphi_0, \varphi_1, \ldots, \varphi_m$, wobei φ_0 die Wirkungsweise auf die Programmzeilennummer und die φ_j, $1 \leq j \leq m$, die Wirkungsweise auf das j-te Register beschreibt. Die Wirkung der einzelnen Befehle sieht folgendermaßen aus.

Bei $B_\ell = i+$ ist

$$\varphi(\ell, n_1, \ldots, n_m) \;=\; (\ell + 1, n_1, \ldots, n_{i-1}, n_i + 1, n_{i+1}, \ldots, n_m).$$

Bei $B_\ell = i-$ ist

$$\varphi(\ell, n_1, \ldots, n_m) \;=\; (\ell + 1, n_1, \ldots, n_{i-1}, n_i - 1, n_{i+1}, \ldots, n_m)$$

bei $n_i \geq 1$ und

$$\varphi(\ell, n_1, \ldots, n_m) \;=\; (\ell + 1, n_1, \ldots, n_{i-1}, n_i, n_{i+1}, \ldots, n_m)$$

bei $n_i = 0$. Bei $B_\ell = C(ij)$ ist

$$\varphi(\ell, n_1, \ldots, n_m) = \begin{cases} (j, n_1, \ldots, n_m), \text{ falls } n_i = 0\,, \\ (\ell + 1, n_1, \ldots, n_m) \text{ sonst}\,. \end{cases}$$

Bei $B_\ell = H$ (also bei $\ell = h$) ist

$$\varphi(h, n_1, \ldots, n_m) \;=\; (h, n_1, \ldots, n_m),$$

die Abbildung wirkt dort also wie die Identität. Der Druckbefehl ist für den Programmablauf nicht relevant und wird hier ignoriert.

In manchen Situationen möchte man eine auf ganz $\mathbb{N} \times \mathbb{N}^m$ definierte Programmabbildung haben. Um dies zu erreichen setzt man für $\ell > h$ einfach

$$\varphi(\ell, n_1, \ldots, n_m) \;=\; (\ell, n_1, \ldots, n_m).$$

Repräsentierbarkeit der Registerbefehle

Ein Registerprogramm kann also in eine Abbildung übersetzt werden, die die Wirkungsweise des Programms widerspiegelt. Die dabei auftretenden Abbildungen sind prinzipiell einfach beschreibbar, auch wenn dafür eine lange Abbildungsdefinition und tief verschachtelte Fallunterscheidungen nötig sind.

Der Ablauf eines Programms P zur Anfangseingabe (Anfangskonfiguration) $e = (1, e_1, \ldots, e_m)$ (die Anfangszeile besitzt die Zeilennummer 1!) wird durch die Hintereinanderschaltung der Programmabbildung $\varphi = \varphi_P$ mit sich selbst beschrieben. Nach dem ersten Programmschritt, bei dem der Befehl in der ersten Programmzeile aufgerufen wird, erhält man die Folgekonfiguration $\varphi(e)$. Die nullte Komponente von $\varphi(e)$ gibt an, mit welcher Programmzeile weitergearbeitet wird. Dies ist aber alles in φ kodiert, so dass das Ergebnis nach dem nächsten Schritt einfach $\varphi(\varphi(e))$ ist. Das Ergebnis nach dem s-ten Rechenschritt (Befehlszeilenwechsel) ist also

$$\varphi(\ldots(\varphi(\varphi(e)))\ldots),$$

wobei s-mal φ angewendet wird. Dafür schreiben wir auch $\varphi^s(e)$. Die aktuelle Zeilennummer ist dabei stets als nullte Komponente von $\varphi^s(e)$ ablesbar, wofür wir $(\varphi^s(e))_0$ schreiben.

Wie wirkt sich nun die Eigenschaft eines Programms, anzuhalten oder nicht, auf diese Iterationen von φ aus? Das Programm hält genau dann an, wenn es bei Eingabe von e ein s gibt mit

$$(\varphi^s(e))_0 = h\,.$$

Wir möchten die Wirkungsweise von Programmen in der Sprache der Arithmetik selbst repräsentieren, um dort das Halteproblem (und seine Unentscheidbarkeit) nachbilden zu können. Dafür müssen wir zunächst die einzelnen Programmschritte arithmetisch erfassen.

Definition 20.3. Den Programmzeilen $B_1, \ldots, B_h$ eines Registerprogramms mit m Registern werden die folgenden arithmetischen Ausdrücke $A_1, \ldots, A_h$ in den freien Variablen $z, r_1, \ldots, r_m, z', r'_1, \ldots, r'_m$ zugeordnet.

(1) Bei $B_\ell = i+$ setzt man

$$\begin{aligned} A_\ell := (z = \ell) \to (z' = z + 1) \wedge (r'_1 = r_1) \wedge \ldots \\ \wedge (r'_{i-1} = r_{i-1}) \wedge (r'_i = r_i + 1) \wedge (r'_{i+1} = r_{i+1}) \wedge \ldots \wedge (r'_m = r_m). \end{aligned}$$

(2) Bei $B_\ell = i-$ setzt man

$$\begin{aligned} A_\ell := (z = \ell) \to (z' = z + 1) \wedge (r'_1 = r_1) \wedge \ldots \\ \wedge (r'_{i-1} = r_{i-1}) \wedge ((r_i = 0) \to (r'_i = r_i)) \wedge (\neg(r_i = 0) \to (r'_i + 1 = r_i)) \\ \wedge (r'_{i+1} = r_{i+1}) \wedge \ldots \wedge (r'_m = r_m). \end{aligned}$$

(3) Bei $B_\ell = C(i,j)$ setzt man

$$A_\ell := (z = \ell) \to ((r_i = 0) \to (z' = j)) \wedge (\neg(r_i = 0) \to (z' = z+1)) \\ \wedge (r'_1 = r_1) \wedge \ldots \wedge (r'_m = r_m).$$

(4) Bei $B_\ell = B_h = H$ setzt man

$$A_h := (z = h) \to (z' = z) \wedge (r'_1 = r_1) \wedge \ldots \wedge (r'_m = r_m)\,.$$

Hierbei werden die natürlichen Zahlen ℓ und j in den arithmetischen Ausdrücken durch die entsprechenden Summen $1 + \cdots + 1$ repräsentiert, die Abfrage am i-ten Register schlägt sich in der Position im Ausdruck, wo etwas passiert, nieder.

Wie bei der Programmabbildung ist es sinnvoll, für alle Programmzeilennummern (also auch für $\ell > h$) einen arithmetischen Ausdruck zu haben. Dazu setzen wir

$$A_{h+1} := (z \geq h+1) \to (z' = z) \wedge (r'_1 = r_1) \wedge \ldots \wedge (r'_m = r_m)$$

(wobei $z \geq h+1$ eine Abkürzung ist). Zu einem gegebenen Programm bestehend aus den Programmzeilen $B_1, \ldots, B_h$ betrachtet man die Konjunktion der soeben eingeführten zugehörigen arithmetischen Repräsentierungen, also $A_P = A_1 \wedge A_2 \wedge \ldots \wedge A_h \wedge A_{h+1}$. Dieser Ausdruck repräsentiert die Programmabbildung.

LEMMA 20.4. *Es sei P ein Registerprogramm mit den Programmzeilen B_1, $\ldots, B_h$ und m Registern mit den zugehörigen arithmetischen Ausdrücken $A_1, \ldots, A_h$ in den freien Variablen $z, r_1, \ldots, r_m, z', r'_1, \ldots, r'_m$. Es sei $A_P = A_1 \wedge \ldots \wedge A_h \wedge A_{h+1}$. Dann ist A_P eine arithmetische Repräsentierung der Programmabbildung φ_P.*

Beweis. Die Variablen z, z', r_j, r'_j seien durch die natürlichen Zahlen ℓ, ℓ', n_j, n'_j belegt. Wir müssen zeigen, dass die Gleichheit

$$\varphi_P(\ell, n_1, \ldots, n_m) = (\ell', n'_1, \ldots, n'_m)$$

genau dann gilt, wenn

$$\mathbb{N} \vDash A_P(\ell, \ell', n_1, \ldots, n_m, n'_1, \ldots, n'_m)$$

gilt. Der Ausdruck A_P gilt genau dann, wenn sämtliche Ausdrücke A_1, A_2, $\ldots, A_h, A_{h+1}$ gelten. Es sei ℓ fixiert. Dann gelten sämtliche A_k, $k \neq \ell$, automatisch, da für diese Ausdrücke der Vordersatz nicht gilt. Die Gültigkeit von A_P bei dieser Belegung bedeutet also, dass der Nachsatz in A_ℓ gelten muss. Sowohl $\varphi(\ell, -)$ als auch der Nachsatz von A_ℓ drücken die Wirkungsweise des Befehls B_ℓ aus, daher gilt die Abbildungsgleichheit genau dann, wenn A_ℓ wahr ist. □

Die β-Funktion

Das Halteproblem führte zu der Existenzaussage, dass es eine Iteration der Programmabbildung gibt, für die die 0-te Komponente gleich der Haltezeilennummer ist. Die arithmetische Repräsentierung dieser Existenzaussage bedarf einiger Vorbereitungen.

Eine natürliche Zahl n lässt sich bekanntlich im Zehnersystem als

$$n = a_0 1 + a_1 10 + a_2 10^2 + \cdots + a_k 10^k$$

schreiben, wobei die a_i zwischen 0 und 9 liegen. Umgekehrt definiert eine endliche Ziffernfolge $(a_0, a_1, \ldots, a_k)$ (bzw. in alltäglicher Schreibweise $a_k a_{k-1} \ldots a_1 a_0$) eine natürliche Zahl. Anstatt der Basis 10 kann man jede natürliche Zahl $p \geq 2$ als Basis nehmen (für viele Zwecke ist auch die Basis 1 erlaubt, eine Zahl n wird dann einfach durch das n-fache Hintereinanderschreiben der 1 repräsentiert). Man spricht dann von der p-adischen Entwicklung (oder Darstellung) der Zahl. Die p-adische Entwicklung einer natürlichen Zahl ist eindeutig.

Sei $p \geq 2$ fixiert. Wie berechnet man die Ziffernfolge einer gegebenen Zahl n? Zuerst betrachten wir die Ziffer (die Einerziffer) $a_0(n) = a(n, 0)$. Es gilt die rekursive Beziehung

$$a(n,0) = \begin{cases} n, \text{falls } n < p\,, \\ a(n-p,0) \text{ sonst}\,. \end{cases}$$

Dies beruht einfach darauf, dass bei $n \geq p$ das Abziehen von p die Ziffer zu p^0 nicht ändert. Man beachte, dass sowohl die Abfrage, die die Fallunterscheidung in dieser Definition konstituiert, als auch die Subtraktion im Fall 2 mit einer Registermaschine durchführbar sind, und dass dadurch eine R-berechenbare Funktion vorliegt.

Auch die Definition der anderen Ziffern geschieht rekursiv. Wenn man von n die (schon berechnete) Ziffer zu p^0 abzieht, so erhält man eine durch p teilbare Zahl. Zwischen der Ziffernentwicklung von n und von $m = \frac{n-a(n,0)}{p}$ besteht ein direkter Zusammenhang, die Ziffer a_{i+1} von n ist einfach die Ziffer a_i von m. Daher ist für $i \geq 0$

$$a(n,i+1) = \begin{cases} 0, \text{falls } n < p^{i+1}\,, \\ a(\frac{n-a(n,0)}{p}, i) \text{ sonst}\,. \end{cases}$$

Damit ist die Berechnung der $(i+1)$-ten Ziffer auf die Berechnung der i-ten Ziffer einer kleineren Zahl rekursiv zurückgeführt. Die Bedingung in der Abfrage und die Subtraktion und die Division in der Definition sind durch eine Registermaschine durchführbar. Diese Funktionsvorschrift berechnet nicht nur die „benötigten“ Ziffern, sondern auch alle höheren, wobei natürlich für alle unbenötigten 0 herauskommt.

Wir führen nun die β-Funktion ein. Der Hauptzweck dieser Funktion soll sein, endliche Folgen von natürlichen Zahlen unterschiedlicher Länge durch drei Zahlen zu kodieren. Die Grundidee ist, dies über die p-adische Entwicklung zu tun, wobei die drei Eingabezahlen einen Zahlwert, eine Basis und eine Ziffernstelle repräsentieren, und die Ausgabe die Ziffernfolge ist. Zugleich soll diese Funktion arithmetisch repräsentierbar sein, so dass die folgende Funktion etwas komplizierter aussieht. Wir folgen weitgehend dem Zugang von Ebbinghaus, Flum, Thomas.

DEFINITION 20.5. Unter der *β-Funktion* versteht man die Abbildung

$$\mathbb{N}^3 \longrightarrow \mathbb{N},\ (p,n,i) \longmapsto \beta(p,n,i),$$

die folgendermaßen festgelegt ist. $\beta(p,n,i)$ ist die kleinste Zahl $a \in \mathbb{N}$, die die Bedingung erfüllt, dass es natürliche Zahlen b_0, b_1, b_2 gibt, die die folgenden Eigenschaften erfüllen:

(1) $n = b_0 + b_1\left((i+1) + ap + b_2p^2\right)$.
(2) $a < p$.
(3) $b_0 < b_1$.
(4) b_1 ist eine Quadratzahl.
(5) Alle Teiler $d \neq 1$ von b_1 sind ein Vielfaches von p.

Wenn kein solches a existiert, so ist $\beta(p,n,i) = 0$.

Zunächst ist klar, dass diese Funktion arithmetisch repräsentierbar ist. Wenn p eine Primzahl ist, so bedeutet Teil (5), dass b_1 eine Primzahlpotenz ist, und Teil (4), dass der Exponent geradzahlig ist. Das folgende Lemma sichert die gewünschte Eigenschaft der β-Funktion, nämlich die Eigenschaft, endliche Folgen zu repräsentieren.

LEMMA 20.6. *Zu jeder endlichen Folge $(a_0, \ldots, a_s)$ aus $\mathbb{N}$ gibt es natürliche Zahlen p, n derart, dass $\beta(p,n,i) = a_i$ für $i \leq s$ ist.*

Beweis. Es sei die endliche Folge $(a_0, a_1, \ldots, a_s)$ vorgegeben. Wir wählen eine Primzahl p, die größer als alle a_i und größer als $s+1$ ist. Es sei

$$\begin{aligned} n &:= 1 \cdot p^0 + a_0p^1 + 2p^2 + a_1p^3 + \cdots + (s+1)p^{2s} + a_sp^{2s+1} \\ &= \sum_{i=0}^{s} a_ip^{2i+1} + \sum_{i=0}^{s}(i+1)p^{2i} \\ &= \sum_{i=0}^{s}(i+1+a_ip)p^{2i}. \end{aligned}$$

Die vorgegebene Folge ist also die Folge der Ziffern der ungeraden Stellen in der p-adischen Ziffernentwicklung von n. Wir behaupten $\beta(p,n,k) = a_k$ für $k \leq s$. Zunächst erfüllt a_k die in der Definition der β-Funktion formulierten Eigenschaften, und zwar mit

$$b_0 = 1p^0 + a_0p^1 + 2p^2 + a_1p^3 + \cdots + kp^{2k-2} + a_{k-1}p^{2k-1},$$

$$b_1 = p^{2k},$$
$$b_2 = (k+2) + a_{k+1}p + (k+3)p^2 + \cdots + (s+1)p^{2(s-k)-2} + a_s p^{2(s-k)-1}.$$

Die erste Eigenschaft ergibt sich aus

$$\begin{aligned}
n &= \sum_{i=0}^{s} a_i p^{2i+1} + \sum_{i=0}^{s} (i+1)p^{2i} \\
&= \sum_{i=0}^{k-1} a_i p^{2i+1} + \sum_{i=0}^{k-1} (i+1)p^{2i} + \sum_{i=k}^{s} a_i p^{2i+1} + \sum_{i=k}^{s} (i+1)p^{2i} \\
&= b_0 + p^{2k}\left(\sum_{i=0}^{s-k} a_{k+i} p^{2i+1} + \sum_{i=0}^{s-k} (k+i+1)p^{2i}\right) \\
&= b_0 + p^{2k}\left(k+1+a_k p + \sum_{i=1}^{s-k} a_{k+i} p^{2i+1} + \sum_{i=1}^{s-k} (k+i+1)p^{2i}\right) \\
&= b_0 + b_1\left(k+1+a_k p + b_2 p^2\right),
\end{aligned}$$

die anderen sind klar. Wenn umgekehrt ein a die Bedingungen erfüllt (mit c_0, c_1, c_2), wobei $c_1 = p^{2\ell}$ ist, so ist

$$\begin{aligned}
n &= b_0 + (k+1)p^{2k} + a_k p^{2k+1} + b_2 p^{2k+2} \\
&= c_0 + (k+1)p^{2\ell} + a p^{2\ell+1} + c_2 p^{2\ell+2}.
\end{aligned}$$

Da die p-adische Entwicklung von n eindeutig ist, folgen daraus und aus den weiteren Bedingungen die Gleichheiten $\ell = k$ und $a = a_k$. □