Prof. Dr. H. Brenner **Osnabrück WS 2011/2012**

Einführung in die mathematische Logik

Vorlesung 10

Arithmetische Repräsentierbarkeit

DEFINITION 10.1. Eine Abbildung

$$\varphi : \mathbb{N}^r \longrightarrow \mathbb{N}^s$$

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

DEFINITION 10.2. Eine Relation $R \subseteq \mathbb{N}^r$ heißt *arithmetisch repräsentierbar*, wenn es einen L^{Ar}-Ausdruck ψ in r freien Variablen gibt derart, 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.

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.

Wir wollen zeigen, dass Registerprogramme, oder besser gesagt die durch ein Registerprogramm festgelegte Abbildung, 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

$$\varphi : \{1, 2, \ldots, h\} \times \mathbb{N}^m \longrightarrow \{1, 2, \ldots, h\} \times \mathbb{N}^m,\ (z, r_1, \ldots, r_m) \longmapsto \varphi(z, r_1, \ldots, r_m),$$

auffassen. Diese Abbildung nennen wir die *Programmabbildung*. Dabei steht z für die Programmzeilennummer und r_j steht für den Inhalt des Registers R_j (von denen es ja m Stück gibt). Dem Tupel $(z, r_1, \ldots, r_m)$ wird dasjenige Tupel $\varphi(z, r_1, \ldots, r_m)$ zugeordnet, das bei Abruf des in der z-ten Programmzeile stehenden Befehls B_z bei der Belegung $(r_1, \ldots, r_m)$ entsteht. Die Abbildung φ besteht dabei aus den $k+1$ Komponentenfunktionen $\varphi_0, \varphi_1, \ldots, \varphi_k$, wobei

φ_0 die Wirkungsweise auf die Programmzeilennummer und die φ_j, $1 \leq j \leq k$, die Wirkungsweise auf dem j-ten Register beschreibt. Die Wirkung der einzelnen Befehle sieht folgendermaßen aus.

Bei $B_z = i+$ ist

$$\varphi(z, r_1, \ldots, r_m) = (z+1, r_1, \ldots, r_{i-1}, r_i + 1, r_{i+1}, \ldots, r_m)\,.$$

Bei $B_z = i-$ ist

$$\varphi(z, r_1, \ldots, r_m) = (z+1, r_1, \ldots, r_{i-1}, r_i - 1, r_{i+1}, \ldots, r_m)$$

bei $r_i \geq 1$ und

$$\varphi(z, r_1, \ldots, r_m) = (z+1, r_1, \ldots, r_{i-1}, r_i, r_{i+1}, \ldots, r_m)$$

bei $r_i = 0$. Bei $B_z = C(ij)$ ist

$$\varphi(z, r_1, \ldots, r_m) = \begin{cases} (j, r_1, \ldots, r_m) \text{ falls } r_i = 0 \\ (z+1, r_1, \ldots, r_m) \text{ sonst}\,. \end{cases}$$

Bei $B_z = H$ (also bei $z = h$) ist

$$\varphi(h, r_1, \ldots, r_m) = (h, r_1, \ldots, r_m)\,,$$

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

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 $e = (1, r_1, \ldots, r_k)$ (die Anfangszeile besitzt die Zeilennummer 1!) wird durch die Hintereinanderschaltung der Programmabbildung $\varphi = \varphi_P$ 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 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.

LEMMA 10.3. *Die Programmzeilen eines Registerprogramms bzw. die zugehörige Programmabbildung lassen sich folgendermaßen mit den Variablen $z, r_1, \ldots, r_m, z', r'_1, \ldots, r'_m$ arithmetisch repräsentieren (dabei sei ℓ die aktuelle Programmzeile).*

(1) *$B_\ell = i+$ wird arithmetisiert durch*

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

(2) *$B_\ell = i-$ wird arithmetisiert durch*

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

(3) *$B_\ell = C(i,j)$ wird arithmetisiert durch*

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

(4) *$B_\ell = B_h = H$ wird arithmetisiert durch*

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

Beweis. Die arithmetische Repräsentierbarkeit bedeutet, dass $\varphi_P(z, r_1, \ldots, r_m) = (z', r'_1, \ldots, r'_m)$ genau dann gilt, wenn die entsprechende arithmetische Aussage A_z in $\mathbb{N}$ gilt. Genau so wurden aber die A_z definiert. □

Zu einem gegebenen Programm bestehend aus den Programmzeilen $P_1, \ldots, P_h$ betrachtet man die Konjunktion der soeben eingeführten zugehörigen arithmetischen Repräsentierungen, also

$$A_P = A_1 \wedge A_2 \wedge \ldots \wedge A_h\,.$$

Die Aussage A_P ist somit für eine Variablenbelegung (der Variablen z, z', r_j, r'_j mit Werten in $\mathbb{N}$) genau dann gültig, wenn $\ell = z^{\mathbb{N}} > h$ ist (da dann keine Bedingung der einzelnen konjugierten Aussage erfüllt ist) oder wenn $\ell \leq h$ ist und A_ℓ erfüllt ist, und dies ist nach dem Lemma genau dann der Fall, wenn die (Variablen)-Belegung von z' die Programmzeilennummer ist, die durch den aktuellen (durch die Belegung von z festgelegten) Befehl B_ℓ als nächste Programmzeile aufgerufen wird, und wenn die Belegungen der r'_i die aus diesem Befehl resultierenden Belegungen der r_i sind.

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

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

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 unter der Voraussetzung, dass nur Ziffern zwischen 0 und $p-1$ verwendet werden, eindeutig.

Sei $p \geq 2$ fixiert. Wie berechnet man die Ziffernfolge einer gegebenen Zahl n? Zuerst betrachten wir die Ziffer $a_0(n) = a(n, 0)$. 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 ist, und dass dadurch eine erlaubte rekursive Definition einer Funktion vorliegt.

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

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

Damit ist die Berechnung der $(i+1)$-ten Ziffer auf die Berechnung der i-ten Ziffer 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. Hauptzweck einer solchen Funktion ist es, 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 10.4. 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) $n = b_0 + b_1((i+1) + ap + b_2p^2)$.
(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 10.5. *Zu jeder endlichen Folge $(a_0, \ldots, a_s)$ aus $\mathbb{N}$ gibt es natürliche Zahlen p, n derart, dass $\beta(p,n,i) = a_i$ ist für $i \leq s$.*

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

$$\begin{aligned} n &:= 1 \cdot p^0 + a_0 p^1 + 2p^2 + a_1 p^3 + \ldots + (s+1)p^{2s} + a_s p^{2s+1} \\ &= \sum_{i=0}^{s} a_i p^{2i+1} + \sum_{i=0}^{s} (i+1)p^{2i} \\ &= \sum_{i=0}^{s} (i + 1 + a_i p)p^{2i}. \end{aligned}$$

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

$$b_0 = 1p^0 + a_0 p^1 + 2p^2 + a_1 p^3 + \ldots + kp^{2k-2} + a_{k-1}p^{2k-1}\,,$$

$$b_1 = p^{2k}\,,$$

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

Die erste Eigenschaft ergibt sich aus

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

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