Prof. Dr. H. Brenner **Osnabrück SS 2021**

Einführung in die mathematische Logik

Vorlesung 18

Wir kehren nun zur Ausgangsfrage dieses Kurses zurück, ob es eine Maschine geben kann, die mathematische Probleme (etwa aus der Zahlentheorie) löst. In den vorhergehenden Vorlesungen haben wir eine formale Sprache entwickelt, in der man solche nichttrivialen Probleme präzise formulieren kann. Ferner haben wir gesehen, wie ein formaler Beweis (eine Ableitung im Prädikatenkalkül) in dieser Sprache aussieht, und dass es nach dem Vollständigkeitssatz für jeden mathematisch beweisbaren Ausdruck der Sprache auch einen formalen Beweis gibt.

In dem vorgestellten Ableitungskalkül der Prädikatenlogik sind die Starttautologien und die Ableitungsregeln übersichtlich strukturiert. Zwar nehmen die Starttautologien häufig Bezug auf beliebige Ausdrücke (und Variablen) der Sprache, doch da die Ausdrücke prinzipiell auflistbar sind, gilt dies auch für die Starttautologien. Daher kann man sich auch gut einen Algorithmus vorstellen, der nach und nach alle formalen Beweise und somit auch alle formal-beweisbaren Ausdrücke ausgibt. Ein andersgelagertes Problem ist die Fragestellung, ob es ein Entscheidungsverfahren für die Prädikatenlogik gibt, ob es also ein algorithmisches Verfahren gibt, dass zu einem gegebenen Ausdruck überprüfen kann, ob es dafür einen formalen Beweis gibt oder nicht.

Wenn wir bisher von Algorithmen gesprochen haben, so haben wir dabei immer an intuitiv durchführbare Algorithmen gedacht, ohne ein konkretes Durchführungsmodell vor Augen zu haben. In dieser Vorlesung stellen wir die Arbeitsweise einer konkreten algorithmischen Maschine vor, der Registermaschine, die wir von nun an als mechanische Realisierung unserer intuitiven Vorstellung von Algorithmen auffassen wollen.

Registermaschinen

Es gibt verschiedene Möglichkeiten, eine deterministisch arbeitende Maschine zu modellieren. Wir arbeiten hier mit Registermaschinen, da diese einem wirklichen Computer ziemlich nahe kommen und daher etwas vertrauter sind als Turingmaschinen oder rekursive Funktionen (wobei letztere vom mathematischen Standpunkt her eleganter sind).

Statue von Alan Turing (1912-1954).

DEFINITION 18.1. Unter einer *Registermaschine* versteht man eine endliche Folge von Registern $R_1, R_2, \ldots, R_m$ (oder Speichern), deren Inhalt jeweils eine natürliche Zahl ist, die durch eine endliche (eventuell leere) Folge von Strichen repräsentiert wird.

Ein *Programm für eine Registermaschine* ist eine endliche durchnummerierte Folge von Befehlen $B_1, B_2, \ldots, B_h$, wobei es für die einzelnen Befehle B_j die folgenden Möglichkeiten gibt.

(1) $i+$ (erhöhe den Inhalt des Registers R_i um 1, d.h. um einen Strich).
(2) $i-$ (reduziere den Inhalt des Registers R_i um 1, d.h. ziehe einen Strich ab; wenn der Inhalt leer ist, so lasse ihn leer).
(3) $C(ij)$ (wenn das i-te Register leer ist, so gehe zum Befehl B_j, andernfalls zum nächsten Befehl).
(4) Drucke (drucke den Inhalt des ersten Registers).
(5) Halte an.

Dabei muss $i \leq m$ für alle in einer Programmzeile adressierten Register und $j \leq h$ für alle adressierten Befehlszeilen gelten. Die letzte Befehlszeile B_h ist ein Haltebefehl und sonst gibt es keinen Haltebefehl.

Die beiden ersten Befehle nennt man *Inkrementierung* bzw. *Dekrementierung*. Der dritte Befehl ist der *Abfragebefehl* oder die (bedingte) *Sprunganweisung*. Es folgen *Druckbefehl* (*Ausgabebefehl*) und *Haltebefehl*.

Ein Durchlauf eines Programmes für eine Registermaschine arbeitet die Befehle des Programmes der Reihe nach ab und zwar unter den jeweiligen zum Bearbeitungszeitpunkt vorgefundenen Registerbelegungen. Wenn die aktuelle Programmzeile ein bedingter Sprungbefehl $C(ij)$ ist, so wird, falls die Bedingung zu diesem Zeitpunkt erfüllt ist (also falls das Register R_i leer ist), zur Programmzeile B_j gewechselt. Wenn die Endzeile B_h, also der Haltebefehl erreicht wird, so ist die Bearbeitung beendet.

Die Belegung (oder der Inhalt) des Registers R_i, die sich im Laufe des Programmdurchlaufs mehrfach ändern kann, werden wir häufig mit r_i bezeichnen. Dies ist stets eine natürliche Zahl. Wenn das Register R_i leer ist, so ist sein Inhalt $r_i = 0$.

Die Möglichkeiten einer Registermaschine scheinen auf den ersten Blick recht bescheiden zu sein. Man sieht aber recht schnell, dass man aus diesen wenigen Befehlen Programmabschnitte zusammensetzen kann, die zunehmend komplexere Befehle ausführen. Komplexe Befehle, von denen schon gezeigt wurde, dass sie sich mit Hilfe der Grundbefehle realisieren lassen, werden ohne weiteren Kommentar weiterverwendet.

Man sagt, dass ein Programm *korrekt* ist, wenn es das tut, was es tun soll. Wenn beispielsweise gesagt wird, dass ein Programm zwei Zahlen addiert, so wird die Korrektheit dadurch bewiesen, dass man eben durch Analyse des Programmcodes nachweist, dass bei beliebiger Belegung der beiden Register, deren Inhalte addiert werden sollen, das Programm schließlich anhält und in ein weiteres Register wirklich die Summe der beiden Zahlen gespeichert ist. Ein Korrektheitsnachweis ist häufig eine mühevolle Kleinarbeit mit aufwändigen Fallunterscheidungen, in den natürlich auch mathematische Überlegungen eingehen, wie z.B. bei der Addition die Eigenschaft, dass $s+t = s+(t-1)+1$ ist, was einen induktiven Korrektheitsbeweis ermöglicht. Wir werden diese Korrektheitsüberlegungen häufig kurz halten.

Programmbeispiele

Wir beschreiben einige Programme bzw. Programmabschnitte für Registermaschinen. Wenn man Programme aus schon entwickelten Programmabschnitten zusammensetzt, so ändern sich natürlich die absoluten Befehlsnummern im Programm, was wir aber ignorieren werden.

Beispiel 18.2. Einen unbedingten Sprung (ein „Go to-Befehl“) zu einer bestimmten Programmzeile, der also nicht von einer Abfrage abhängt, kann man dadurch realisieren, dass man ein neues Register R_k hinzunimmt, das von keiner anderen Programmzeile adressiert wird und dessen Inhalt auf 0 gesetzt wird. Dann bewirkt der Befehl $C(kj)$, dass zur j-ten Programmzeile gewechselt wird, da der Inhalt des Registers R_k im gesamten Programmverlauf gleich 0 bleibt.

Beispiel 18.3. Ein Programm soll sämtliche natürlichen Zahlen der Reihe nach ausdrucken. Dazu brauchen wir eine Registermaschine mit zwei Registern R_1 und R_2, die zum Start beide leer sind. Das zweite Register bleibt unverändert und wird nur für den unbedingten Sprungbefehl verwendet. Die Haltezeile wird nie erreicht.

(1) Drucke
(2) 1+

(3) Gehe zu 1
(4) Halte an

BEISPIEL 18.4. Das Register R_i soll geleert werden. Dies geschieht durch das folgende Programm.

(1) $C(i, 4)$
(2) $i-$
(3) Gehe zu 1
(4) Halte an

BEMERKUNG 18.5. Wir erlauben, dass bei einer Registermaschine die Anfangsbelegung der Register von außen festgelegt wird. Man könnte aber auch festlegen, dass die Anfangsbelegung stets die Nullbelegung ist, ohne die Berechnungsmöglichkeiten der Registermaschine einzuschränken. Dann kann man die eigentlich gewünschte Anfangsbelegung dadurch erreichen, dass man dem Programm ein „Belegungsprogramm“ voranstellt, das den einzelnen Registern R_i durch die s_i Befehle $i+, \ldots, i+$ die gewünschte Belegung s_i zuweist.

Man könnte auch erstmal ein „Entleerungsprogramm“ vorschalten, das alle Register leert und daran anschließend die Belegung durchführt, doch muss man für den Entleerungsvorgang, der nach Beispiel 18.4 einen unbedingten Sprungbefehl verwendet, zumindest ein leeres Register zur Verfügung haben.

Wenn der Registerinhalt r_i um eine natürliche Zahl k erhöht werden soll, also k-fach direkt hintereinander inkrementiert werden soll, so schreiben wir dafür auch $i + \cdots +$ mit k Pluszeichen.

BEISPIEL 18.6. Es soll mit einer Registermaschine festgestellt werden, ob der Inhalt r_i des Registers R_i größer oder gleich dem Inhalt r_j des Registers R_j ist. Dazu reserviert man das leere Register R_k (i, j, k seien paarweise verschieden) und baut einen Programmabschnitt der folgenden Art.

(1) $C(j, 6)$
(2) $j-$
(3) $C(i, 7)$
(4) $i-$
(5) Gehe zu 1
(6) $k+$
(7) Halte an

Wenn dieser Programmabschnitt abgelaufen ist, so steht im Register R_k der Wert $r_k = 1$ oder $r_k = 0$, je nachdem, ob $r_i \geq r_j$ ist oder nicht, und zwar unabhängig davon, ob man damit die Eingangsdaten oder Zwischendaten, wenn das Programm den ersten Befehl abarbeitet, meint. Die Korrektheit dieses Programms beruht darauf, dass $r \geq s$ genau dann gilt, wenn $r - 1 \geq s - 1$ ist. Dies ermöglicht einen Induktionsbeweis.

Beispiel 18.7. Wir wollen überprüfen, ob die Inhalte von zwei Registern R_i und R_j übereinstimmen. Dazu kann man das Programm aus Beispiel 18.6 einfach abändern zu

(1) $C(j, 6)$
(2) $j-$
(3) $C(i, 9)$
(4) $i-$
(5) Gehe zu 1
(6) $C(i, 8)$
(7) Gehe zu 9
(8) $k+$
(9) Halte an

Bei Gleichheit erhält man $r_k = 1$, bei Ungleichheit $r_k = 0$.

In den obigen beiden Beispielen wurde die Antwort im Register R_k (in der Form 0 oder 1 abgespeichert). Der Druckbefehl nimmt aber immer Bezug auf R_1. Daher ist es nötig, Registerinhalte in andere Register zu verschieben.

Beispiel 18.8. Wir wollen den Registerinhalt r_i des Registers R_i in das Register R_j übertragen (unabhängig von dessen Inhalt). Dies leistet das folgende Programm.

(1) Leere R_j
(2) $C(i, 6)$
(3) $i-$
(4) $j+$
(5) Gehe zu 2
(6) Halte an

Bei diesem Programm wird im Laufe der Durchführung das Ausgangsregister der Übertragung leer gemacht. Dies ist nicht immer erwünscht, häufig möchte man den Inhalt eines Registers kopieren und sich den Inhalt zugleich merken.

Beispiel 18.9. Wir wollen den Registerinhalt r_i des Registers R_i in das Register R_j übertragen (unabhängig von dessen Inhalt), ohne R_i zu leeren. Dazu brauchen wir ein drittes Register R_k und das folgende Programm.

(1) Leere R_j
(2) Leere R_k
(3) $C(i, 8)$
(4) $i-$
(5) $j+$
(6) $k+$
(7) Gehe zu 3
(8) Übertrage den Inhalt von R_k nach R_i
(9) Halte an

Hier wird zwar im Laufe des Programms der Inhalt von R_i verändert, zum Schluss wird der ursprüngliche Inhalt aber wieder hergestellt.

Beispiel 18.10. Die zwei Registerinhalte r_i (von R_i) und r_j (von R_j) sollen addiert werden, wobei die Summe zum Schluss in R_k stehen soll (es seien i, j, k paarweise verschieden). Dies leistet das folgende Programm.

(1) Leere R_k
(2) Übertrage r_i nach R_k
(3) $C(j, 7)$
(4) $j-$
(5) $k+$
(6) Gehe zu 3
(7) Halte an

Mit der Addition und der Kopie von Inhalten kann man auch den Inhalt eines Registers zu einem anderen Register dazuaddieren. Dies kann man natürlich auch einfach direkt realisieren.

Beispiel 18.11. Die beiden Registerinhalte r_i (von R_i) und r_j (von R_j) sollen multipliziert werden, wobei das Produkt zum Schluss in R_k stehen soll (es seien i, j, k paarweise verschieden). Dies leistet das folgende Programm mit dem Hilfsregister R_ℓ.

(1) Leere R_k
(2) Übertrage den Inhalt von R_i nach R_ℓ ohne R_i zu leeren
(3) $C(j, 7)$
(4) Addiere den Inhalt von R_ℓ zu R_k hinzu
(5) $j-$
(6) Gehe zu 2
(7) Halte an

Die Korrektheit dieses Programms beruht auf $r \cdot s = (r-1)s + s$; für das Produkt rs muss man r-mal s mit sich selbst addieren.

Beispiel 18.12. Es soll überprüft werden, ob der Registerinhalt r_t (von R_t) den Registerinhalt r_j (von R_j) teilt. Falls ja soll das Programm 1 ausgeben, andernfalls 0. Dies leistet das folgende Programm mit den Hilfsregistern R_k und R_ℓ (für Teilprogramme braucht man noch weitere Hilfsregister). Das Ausgaberegister R_1 soll zu Beginn leer sein.

(1) Leere R_ℓ
(2) Berechne $r_t \cdot r_\ell$ und schreibe das Ergebnis in R_k (ohne r_t, r_ℓ zu verändern)
(3) Bei $r_k > r_j$ gehe zu 8
(4) Bei $r_k = r_j$ gehe zu 7
(5) $\ell+$
(6) Gehe zu 2

(7) $1+$
(8) Drucke
(9) Halte an

Beispiel 18.13. Es soll überprüft werden, ob der Registerinhalt r_j (von R_j) eine Primzahl ist. Falls ja soll das Programm 1 ausgeben, andernfalls 0. Dies leistet das folgende Programm mit dem Hilfsregister R_t (für Teilprogramme braucht man noch weitere Hilfsregister). Das Ausgaberegister R_1 soll zu Beginn leer sein.

(1) Leere R_t
(2) $t+$
(3) $t+$
(4) Wenn $r_t = r_j$, so gehe zu 8
(5) Wenn $r_t \geq r_j$, so gehe zu 9[1]
(6) Wenn r_j von r_t geteilt wird, so gehe zu 9
(7) Gehe zu 3
(8) $1+$
(9) Drucke
(10) Halte an

Beispiel 18.14. Es sollen die geraden Zahlen ≥ 4 daraufhin überprüft werden, ob sie die Eigenschaft in der Goldbachvermutung erfüllen, also ob sie die Summe von zwei Primzahlen sind. Das Programm soll die Ausgabe 0 machen, falls ein Gegenbeispiel gefunden wurde. Dies leistet das folgende Programm mit den Registern R_n, R_k und R_i, die alle zu Beginn auf 0 gesetzt seien. Auch das Ausgaberegister R_1 soll zu Beginn leer sein. Wir testen ab 6, um uns auf ungerade Primzahlen als Summanden beschränken zu können.

(1) $n++++$
(2) $n++$
(3) Leere R_k
(4) $k+$
(5) $k++$
(6) Wenn $r_k \geq r_n$, so gehe zu 12
(7) Wenn r_k eine Primzahl ist, so gehe zu 9
(8) Gehe zu 5
(9) Berechne $r_n - r_k$, schreibe das Ergebnis in R_i
(10) Wenn r_i eine Primzahl ist, so gehe zu 2
(11) Gehe zu 5
(12) Drucke
(13) Halte an

[1]Die Programmzeile (5) ist nur für $r_j = 0, 1$ von Bedeutung.

Abbildungsverzeichnis