Da Machine

Thomas Pajor

2004

1 Beschreibung

1.1 Das Problem

Gegeben sei die Menge der natürlichen Zahlen $\mathbb{N}$. Eine Zahl $n \in \mathbb{N}$, $n > 1$ ist genau dann eine Primzahl, wenn es keine $p, q \in \mathbb{N}$, $p, q > 1$ gibt mit

$$n = p \cdot q$$

Eines der bedeutensten Probleme der Mathematik ist nun die Frage ob eine gegebene Zahl $n \in N$, $n > 1$ eine Primzahl ist oder nicht. Wie man leicht sieht, lässt sich das Problem als *Entscheidungsproblem* definieren. Wir definieren uns hierfür eine Sprache $\mathcal{L}_P$ wie folgt:

$$\mathcal{L}_P := \{n \in \mathbb{N} \mid n \text{ prim}\}$$

Dies ist also die Menge aller natürlichen Zahlen, die Primzahlen sind.

1.2 Funktionsweise

Die Turingmaschine soll als 1-Band-Turingmaschine realisiert werden indem sie ein Eingabewort in binärer Kodierung erhält, und überprüft ob das Wort in $\mathcal{L}_P$ ist. Offensichtlich ist das Problem entscheidbar, die Turingmaschine soll also die folgenden zwei Enduzustände besitzen

1. q_J - hält die Turingmaschine in diesem Zustand, so ist das Wort eine Primzahl
2. q_N - hält die Turingmaschine in diesem Zustand, so ist das Wort keine Primzahl

und nach endlich vielen Übergängen immer einen dieser Endzustände erreichen. Also: $F := \{q_J, q_N\}$.

Da die Zahl binär kodiert werden soll, lässt sich das Alphabet schon festlegen, nämlich $\Sigma := \{0, 1\}$.

2 Der Algorithmus

Der Algorithmus, der auf der Turingmaschine implementiert werden soll, lässt sich abstrakt in Pseudokot folgendermaßen beschreiben

```
1   p = leseZahl();
2   divisor = 2;
3
4   while (divisor < p) {
5           q = p;
6
7           while(q > 0)
8                   q = q - divisor;
9
10          if (q == 0)
11                  halte("p ist nicht prim");
12
13          divisor = divisor + 1;
```

```
14  }
15
16  halte("p ist prim");
```

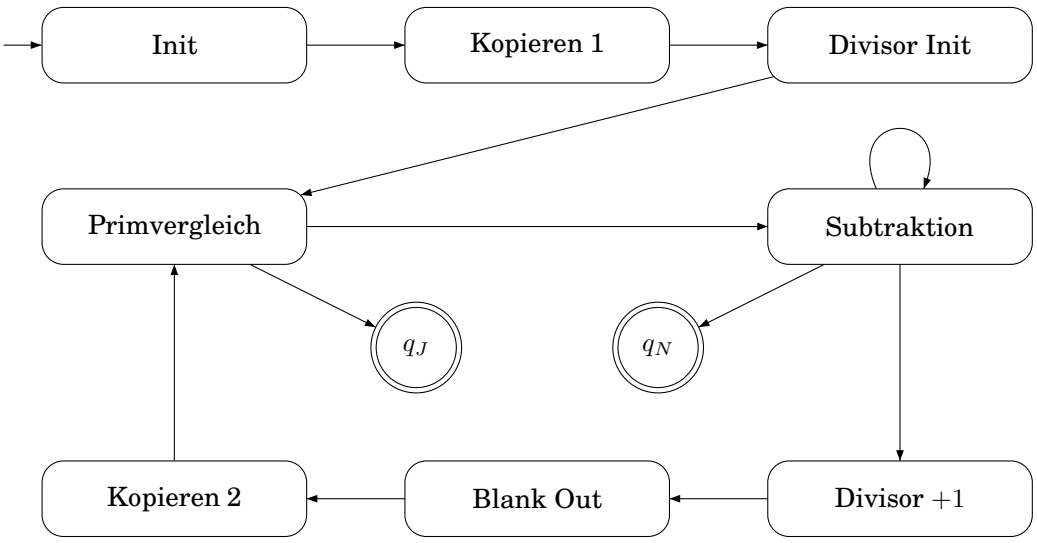
Der Algorithmus entspricht quasi dem naiven Ansatz alle Teiler d mit $d > 1$ durchzuprobieren, ob die Division p/d keinen Rest liefert. Ist der Rest 0, so weiß man, dass die Zahl p nicht prim ist, da sie ja offensichtlich durch d teilbar ist. Die Operation der Division wird der Einfachheit halber durch sukzessives Subtrahieren des Divisors von der Primzahl realisiert, da sich dies leichter auf einer Turingmaschine implementieren lässt.

Liefert kein Divisor $d < p$ einen Rest gleich 0, so handelt es sich bei p um eine Primzahl.

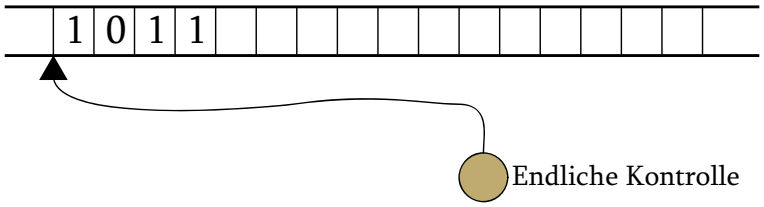
3 Da Machine

3.1 Unterprogramme

Obwohl der Algorithmus sehr einfach aussieht, ist es nicht trivial diesen auf einer Turingmaschine mit nur einem Band und einem Lese/Schreibkopf zu implementieren. Deshalb sei hier erst einmal eine graphische Darstellung der Turingmaschine, bei der Zustände noch zu Unterprogrammen zusammengefasst werden:

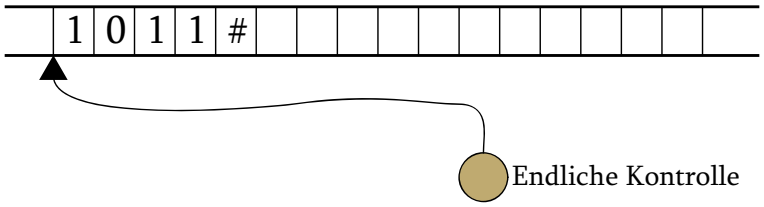


Die einzelnen Unterprogramme seien in den folgenden Sektionen beschrieben. Außerdem sei am Beispielwort 1011 die Funktionsweise der Turingmaschine illustriert. Unsere Anfangssituation auf dem Band sieht also folgendermaßen aus:



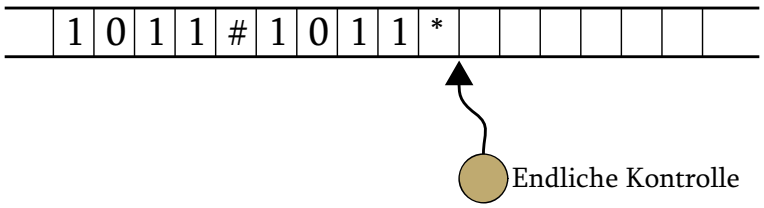
3.1.1 Init

Das Unterprogramm *Init* ist sehr einfach. Es wird hier einfach hinter das Wort eine Raute (♯) gesetzt, um das Ende des Wortes zu markieren. Der Kopf soll am Ende des Unterprogramms wieder zum Anfang des Wortes zurückfahren. Die Situation auf dem Band ist also die Folgende:



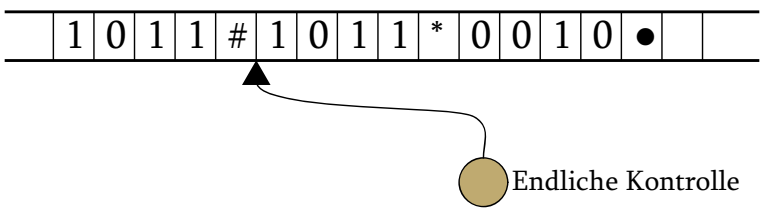
3.1.2 Kopieren 1

Kopieren 1 bewirkt, dass das Eingabewort, hinter die soeben eingefügte Raute kopiert wird. Abschließend wird hinter das kopierte Wort noch ein * gesetzt, um auch hier den Abschluss zu kennzeichnen. Der Schreib/Lesekopf soll sich am Ende des Unterprogramms hinter dem * befinden:



3.1.3 Divisor Init

Dieses Programm initalisiert den Divisor mit dem Wert 2 in Dualkodierung. Dabei wird das Wort vorne mit so vielen Nullen aufgefüllt, dass es gerade die gleiche Länge wie das Eingabewort hat. Am Ende dieser Operation soll der Schreib/Lesekopf an den Anfang des kopierten Eingabewortes zurückfahren:



3.1.4 Primvergleich

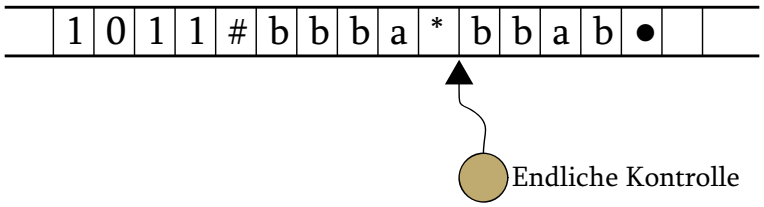
Hier wird das Eingabewort (bzw genauer gesagt die Kopie des Eingabewortes) mit dem Divisor verglichen. Sind die beiden Strings identisch, so geht die Turingmaschine in den Endzustand q_J über, andernfalls fährt der Schreib/Lesekopf wieder vor das kopierte Eingabewort und die Bandsituation bleibt unverändert.

3.1.5 Subtraktion

Dies ist das wohl komplizierteste Unterprogramm. Hier wird einmal von der Kopie (ich möchte sie ab jetzt mit k bezeichnen) der Divisor d abgezogen. Die Turingmaschine nimmt sich dabei einige Zeichen aus dem Bandalphabet zur Hilfe. Dabei steht ein a für 1 und b für 0. Es können nun folgende Fälle eintreten:

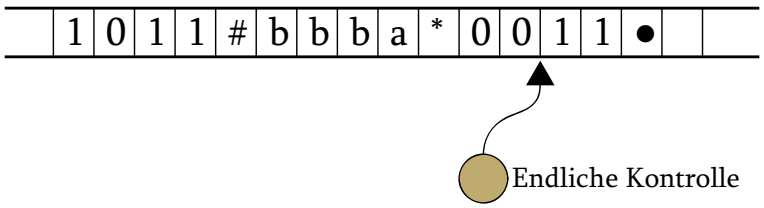
1. Die Subtraktion scheitert, weil $k < d$. In dem Fall geht die Turingmaschine zum nächsten Unterprogramm über.
2. Die Subtraktion geht auf und $k \neq 0$. In dem Fall wiederholt die Turingmaschine die Subtraktion.
3. Die Subtraktion geht auf und $k = 0$ nach der Subtraktion. In dem Fall ist das Eingabewort keine Primzahl und die Turingmaschine geht in den Zustand q_N über.

Die Bandsituation sei hier für den Fall 1. dargestellt, nachdem sie mit den Subtraktionen fertig ist und zum nächsten Unterprogramm wechselt. Das Wort 0010 wurde also so oft von 1011 subtrahiert, bis 0001 dasteht.



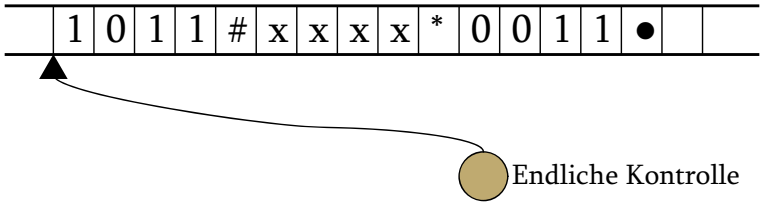
3.1.6 Divisor $+1$

Zunächst wird hier der Divisor wiederhergestellt (a durch 1 und b durch 0 ersetzt). Danach wird $+1$ binär addiert. Zu einem Überlauf kann es hier niemals kommen, denn die Wortlänge von d entspricht der Länge des Eingabwortes. Des Weiteren wird nach jedem Inkrementieren geprüft ob $k = d$ und in den Zustand q_J übergegangen falls das der Fall sein sollte. Die Bandsituation ist also die Folgende:



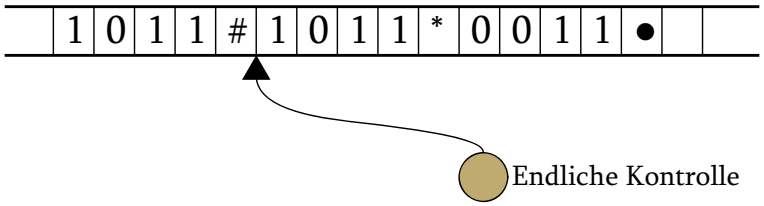
3.1.7 Blank Out

Dies ist nur ein kleines Unterprogramm, welches das Wort k durch „X“e ersetzt, damit später das Eingabewort wieder an die Stelle kopiert werden kann. Am Ende soll der Schreib/Lesekopf ganz zum Anfang des Bandes zurückfahren:



3.1.8 Kopieren 2

Hier wird ähnlich wie bei *Kopieren 1* das Eingabewort wieder an die Stelle hinter der Raute kopiert. Der Platz zwischen ♯ und $*$ reicht in jedem Fall genau aus, da sich die Länge des Wortes nicht verändert hat. Am Schluss soll der Kopf hinter das ♯ Zeichen, um mit dem Primvergleich fortfahren zu können:



Dies war nun eine rein informelle Beschreibung der Turingmaschine. Im folgenden Abschnitt ist die formale Definition der einzelnen Zustände sowie Übergänge angegeben.

3.2 Formale Definition

Die formale bzw Graphische Definition der Turingmaschine entnehme man dem Poster.

3.3 Gödelnummer

Zum Abschluss sei hier noch die 21461 stellige Gödelnummer der Turingmaschine angegeben.

```
1110001000010000100001110100101001001101010101001100000100100000100101100000101
0000101011010001000001000001011000001000100000010001001100000001000001000000100
0001001100000000100100000001001001100000000101000000010100110000000010000010000000
0100000101100000000010010000000010010110000000001010000000010101100000000010000010
0000000010000010011000000000100100000000010010011000000000101000000000101001100 0
0000000100000100000000001000001011000000000010010000000000100101100000000000101 00
0000000010101100000010010000000100000010011000000010001000000010010110000000101 0
0000000010000000100110000000001000100000000001010110000001000001000000000010000
0100110000000000100000001000000101001100000000100000010000001001001100000000000 1
```