

Da Machine für Primzahlen

Thomas Pajor
thomas.pajor@logn.de

17. Februar 2005

- 1 Das Problem
- 2 Beschreibung der TM
- 3 Definition der TM

Definition

Sei \mathbb{N} die Menge der natürlichen Zahlen.

Eine Zahl $n \in \mathbb{N}$ mit $n > 1$ heißt *prim*, wenn es keine Zahlen $p, q \in \mathbb{N}$ gibt mit $p, q > 1$ so, dass gilt:

$$n = p \cdot q$$

- Gegeben: Eine natürliche Zahl $n \in \mathbb{N}$ mit $n > 1$.

Formulierung als Entscheidungsproblem

- Gegeben: Eine natürliche Zahl $n \in \mathbb{N}$ mit $n > 1$.
- Frage: Ist n eine Primzahl?

Formulierung als Entscheidungsproblem

- Gegeben: Eine natürliche Zahl $n \in \mathbb{N}$ mit $n > 1$.
- Frage: Ist n eine Primzahl?
- Sprache: $\mathcal{L} := \{n \mid n \text{ prim}\}$.

- Deterministisch

- Deterministisch
- Eingabe w soll in binärer Kodierung vorliegen, also $\Sigma = \{0, 1\}$

- Deterministisch
- Eingabe w soll in binärer Kodierung vorliegen, also $\Sigma = \{0, 1\}$
- Akzeptierende Endzustände:
 - 1 q_J , falls w eine Primzahl ist
 - 2 q_N , sonst

Algorithmus in Pseudocode

```
w = leseZahl(); divisor = 2;
```

```
while (divisor < w) {  
    k = w;  
    while(k > 0)  
        k = k - divisor;  
    if (k == 0)  
        halte(w ist nicht prim);  
    divisor = divisor + 1;  
}
```

```
halte(w ist prim);
```

Algorithmus in Pseudocode

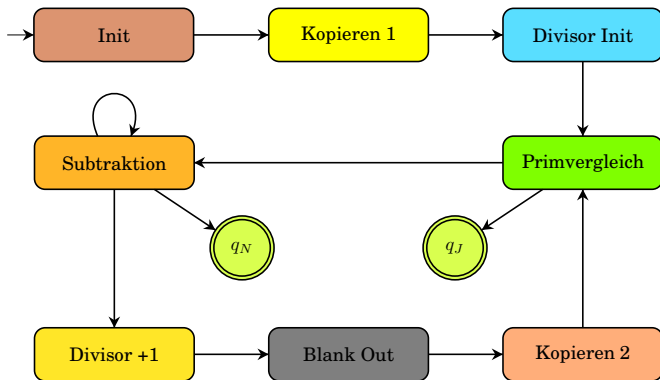
```
w = leseZahl(); divisor = 2;
```

```
while (divisor < w) {  
    k = w;  
    while(k > 0)  
        k = k - divisor;  
    if (k == 0)  
        halte(w ist nicht prim);  
    divisor = divisor + 1;  
}
```

```
halte(w ist prim);
```

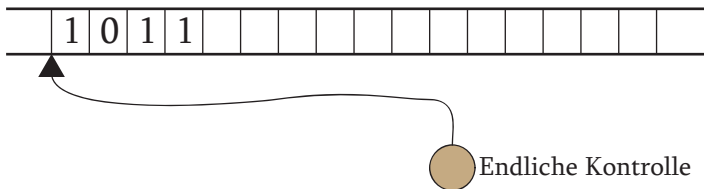
Aufwand leider **exponentiell**.

Zusammenhang der Unterprogramme



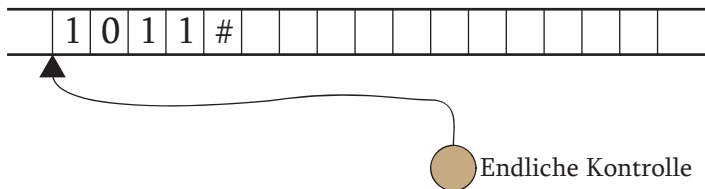
Ausgangssituation

Funktionsweise am Beispiel von $1011_2 = 11_{10}$



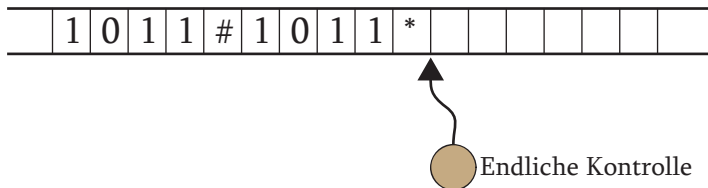
Unterprogramm „Init“

Schreibe # hinter das Eingabewort.



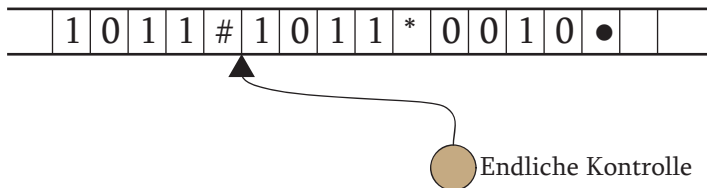
Unterprogramm „Kopieren 1“

Kopiere das Eingabewort w und setze ein $*$ an das Ende. Die Kopie sei mit k bezeichnet.



Unterprogramm „Divisor Init“

Setze die Zahl $d = 10_2$ an das Ende und fülle mit so viel Nullen auf, dass $|d| = |w|$.



Unterprogramm „Primvergleich“

Vergleiche d mit k und führe folgende Aktion aus:

Unterprogramm „Primvergleich“

Vergleiche d mit k und führe folgende Aktion aus:

- $d = k$? Dann gehe in Zustand q_J und stoppe.

Unterprogramm „Primvergleich“

Vergleiche d mit k und führe folgende Aktion aus:

- $d = k$? Dann gehe in Zustand q_J und stoppe.
- $d \neq k$? **Führe nächstes Unterprogramm aus.**

Ziehe Divisor d von k ab. Folgende Fälle können nach der Subtraktion eintreten:

Ziehe Divisor d von k ab. Folgende Fälle können nach der Subtraktion eintreten:

- $k < 0$. Die „Division“ ist beendet. TM geht zum nächsten Unterprogramm.

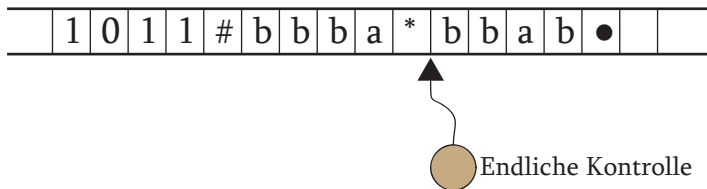
Unterprogramm „Subtraktion“

Ziehe Divisor d von k ab. Folgende Fälle können nach der Subtraktion eintreten:

- $k < 0$. Die „Division“ ist beendet. TM geht zum nächsten Unterprogramm.
- $k > 0$. Die Subtraktion wird wiederholt.

Ziehe Divisor d von k ab. Folgende Fälle können nach der Subtraktion eintreten:

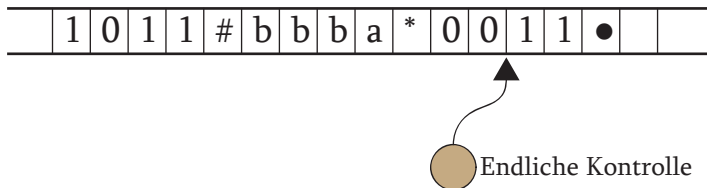
- $k < 0$. Die „Division“ ist beendet. TM geht zum nächsten Unterprogramm.
- $k > 0$. Die Subtraktion wird wiederholt.
- $k = 0$. Die „Division“ ging auf, w ist nicht prim. Wechsle also in q_N und stoppe.



- $a, b \in \Gamma$, wobei $a \hat{=} 1$ und $b \hat{=} 0$.
- Die nächste Subtraktion würde einen Unterlauf produzieren!

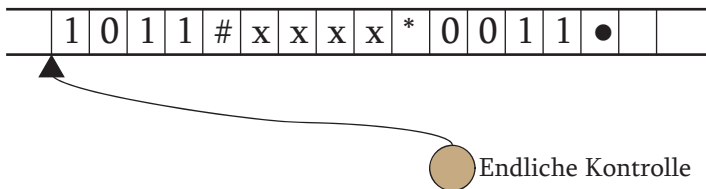
Unterprogramm „Divisor +1“

Der Divisor d wird binär um den Wert 1 inkrementiert.



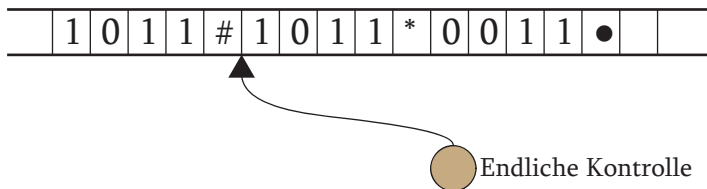
Unterprogramm „Blank Out“

Überschreibe k durch ein Hilfssymbol aus Γ , z.B. X .



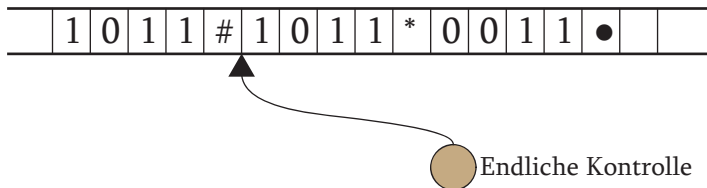
Unterprogramm „Kopieren 2“

Kopiere Eingabewort w an die Stelle der X s.



Unterprogramm „Kopieren 2“

Kopiere Eingabewort w an die Stelle der X s.



Fahre fort mit Unterprogramm „Primvergleich“.

$\mathcal{M} := (Q, \Sigma, \Gamma, \sqcup, \delta, s, F)$ wobei:

$\mathcal{M} := (Q, \Sigma, \Gamma, \sqcup, \delta, s, F)$ wobei:

- $\Sigma := \{0, 1\}$

$\mathcal{M} := (Q, \Sigma, \Gamma, \sqcup, \delta, s, F)$ wobei:

- $\Sigma := \{0, 1\}$
- $\Gamma := \{X, a, b, \#, *, \bullet\}$

$\mathcal{M} := (Q, \Sigma, \Gamma, \sqcup, \delta, s, F)$ wobei:

- $\Sigma := \{0, 1\}$
- $\Gamma := \{X, a, b, \#, *, \bullet\}$
- $F := \{q_J, q_N\}$

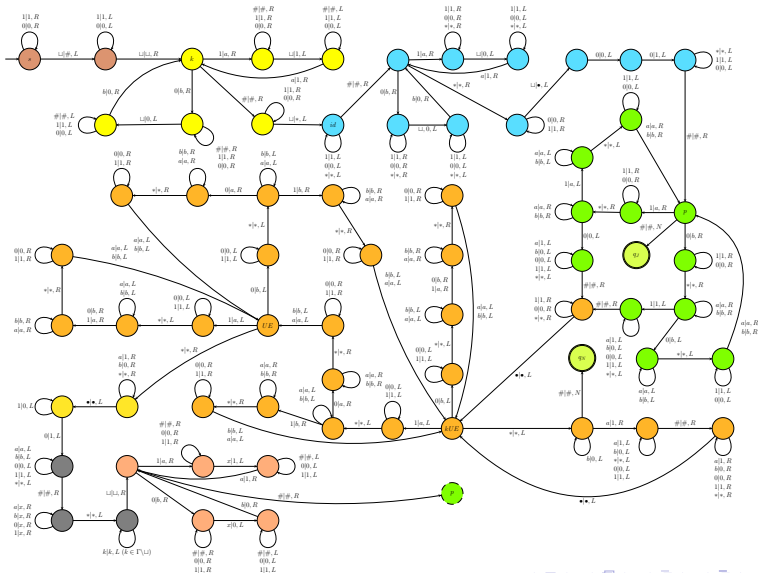
$\mathcal{M} := (Q, \Sigma, \Gamma, \sqcup, \delta, s, F)$ wobei:

- $\Sigma := \{0, 1\}$
- $\Gamma := \{X, a, b, \#, *, \bullet\}$
- $F := \{q_J, q_N\}$
- $|Q| = 68$

$\mathcal{M} := (Q, \Sigma, \Gamma, \sqcup, \delta, s, F)$ wobei:

- $\Sigma := \{0, 1\}$
- $\Gamma := \{X, a, b, \#, *, \bullet\}$
- $F := \{q_J, q_N\}$
- $|Q| = 68$
- $|\delta| = 243$

Übergangsgraph



TM Simulator: <http://www.ncc.up.pt/fado/node2.html>

The screenshot shows a window titled "TM Simulator <2>". The interface includes a menu bar (File, Help), a toolbar with navigation icons, and a control panel. The control panel shows the current state as 36, the current symbol as 'a', and the current head position as 172. Below the control panel is a table for the Turing Machine's state transitions.

State	Symbol	New State	Write Symbol	Move	
0	0	30	.	w	
27	32	+	33	+	RIGHT
28	32	0	32	0	LEFT
29	32	1	32	1	LEFT
30	32	a	32	1	LEFT
31	32	b	32	0	LEFT
32	33	+	33	+	RIGHT
33	0	33	0	RIGHT	
34	33	1	33	1	RIGHT
35	33	a	33	1	RIGHT
36	33	b	33	0	RIGHT
37	33	0	29	0	LEFT
38	34	+	34	+	RIGHT
39	34	0	34	0	RIGHT
100	34	1	34	1	RIGHT
101	34	0	29	0	LEFT
102	35	+	36	+	LEFT
103	35	0	35	0	LEFT
104	35	1	35	1	LEFT
105	36	0	37	b	RIGHT
106	36	1	37	a	RIGHT
107	36	a	36	a	LEFT
108	36	b	36	b	LEFT

The main simulation area shows a tape with the input string "01011+10ba*0baba" and a series of state transitions. The current state is 36, and the current symbol is 'a'. The tape is being processed from left to right, with the head moving across the symbols. The simulation shows the machine's internal state and the symbols written on the tape at each step.

Simulation mit verschiedenen Eingaben w :

- $w = 1011_2 = 11_{10} \rightarrow$ **2586** Abarbeitungsschritte.

Simulation mit verschiedenen Eingaben w :

- $w = 1011_2 = 11_{10} \rightarrow$ **2586** Abarbeitungsschritte.
- $w = 1100001_2 = 97_{10} \rightarrow$ bereits **87274** Schritte.

Simulation mit verschiedenen Eingaben w :

- $w = 1011_2 = 11_{10} \rightarrow$ **2586** Abarbeitungsschritte.
- $w = 1100001_2 = 97_{10} \rightarrow$ bereits **87274** Schritte.
- $w = 1111100101_2 = 997_{10} \rightarrow$ Überlauf des Simulators...

Simulation mit verschiedenen Eingaben w :

- $w = 1011_2 = 11_{10} \rightarrow$ **2586** Abarbeitungsschritte.
- $w = 1100001_2 = 97_{10} \rightarrow$ bereits **87274** Schritte.
- $w = 1111100101_2 = 997_{10} \rightarrow$ Überlauf des Simulators...

PS: Die Gödelnummer hat **21461** Stellen!

Folien online unter: www.logn.de