Ein alternativer Rechenmechanismus: Register Maschinen

k-Register Maschine:

Speicher: $x_1, \cdots, x_k$ k "Register", jeder speichert eine natürliche Zahl

Operationen: $x_i = x_j$ op x_k mit op $\in \{ +, \div, *$ div , mod $\}$

$x_i = x_j$ op c

$x_i = c$

(tunix)

$a \div b = \max\{a-b,0\}$

a div $b = \lfloor a/b \rfloor$

c Konstante

Prädikate: $x_i =? 0$

mit der üblichen Semantik

Ein alternativer Rechenmechanismus: Register Maschinen

Programm: endliche Menge von Befehlen folgender Form

Label: Operation **goto** Label

Label: **if** Prädikat **then** Operation **goto** Label
else Operation **goto** Label

mit der üblichen Semantik

Es gibt zwei besondere Label: *Start, Halt*

Programm: endliche Menge von Befehlen folgender Form

Label: Operation **goto** Label

Label: **if** Prädikat **then** Operation **goto** Label
else Operation **goto** Label

mit der üblichen Semantik

Es gibt zwei besondere Label: *Start, Halt*

Ein Programm Π berechnet Funktion $f_\Pi: \mathbb{N}^m \to \mathbb{N}^n$

Registerinhalte
vor der Durchführung von Π: $x_1=a_1, \cdots x_m=a_m, x_{m+1}=0 \cdots x_k=0$
nach der Durchführung von Π: $x_1=b_1, \cdots x_m=b_n$

$f_\Pi(a_1,\cdots,a_m) = (b_1,\cdots,b_n)$

Beispielprogramm für $f(x) = \begin{cases} 1 & \text{falls x eine Primzahl} \\ 0 & \text{sonst} \end{cases}$

Start: $x_1 = x_1 \div 1$
L1: **if** $x_1 =? 0$ **then** $x_1 = 0$ **goto** *Halt*
else $x_1 = x_1 + 1$ **goto** *L2*
L2: $x_2 = x_1 \div 1$ **goto** *Loop*
Loop: $x_3 = x_2 \div 1$ **goto** *L3*
L3: **if** $x_3 =? 0$ **then** $x_1 = 1$ **goto** *Halt*
else goto *L4*
L4: $x_4 = x_1$ mod x_2 **goto** *L5*
L5: **if** $x_4 =? 0$ **then** $x_1 = 0$ **goto** *Halt*
else $x_2 = x_2 \div 1$ **goto** *Loop*
Halt:

```
if x < 2 then return 0
d = x – 1
while d >1 do
      if d teilt x then return 0
                   else d=d-1
return 1
```

x_1 .... x (oder Ausgabe)
x_2 .... d
x_3 .... d-1

Additive Register Maschinen

k-Register Maschine:

Speicher: $x_1, \cdots, x_k$ k "Register", jeder speichert eine natürliche Zahl

Operationen: $x_i = x_j \text{ op } x_k$ mit $op \in \{ +, \dot{-}, \sout{*}, \sout{div}, \sout{mod} \}$

$x_i = x_j \text{ op } c$

$x_i = c$

(tunix)

$a \dot{-} b = \max\{a-b,0\}$

~~$a \text{ div } b = \lfloor a/b \rfloor$~~

c Konstante

Prädikate: $x_i =?\ 0$

mit der üblichen Semantik

Einfache k-Register-Maschinen

Speicher: $x_1, \cdots, x_k$ k "Register", jeder speichert eine natürliche Zahl

Operationen: $x_i = x_i + 1$ (inkrementieren)

$x_i = x_i \dot{-} 1$ (dekrementieren)

(tunix)

Prädikate: $x_i =?\ 0$

mit der üblichen Semantik

"Satz:" Alles, was von einer k-Register-Maschine berechnet werden kann, kann auch von einer **additiven** (k+3)-Register-Maschine berechnet werden.

"Satz:" Alles, was von einer k-Register-Maschine berechnet werden kann, kann auch von einer **additiven** (k+3)-Register-Maschine berechnet werden.

"Satz:" Alles, was von einer **additiven** m-Register-Maschine berechnet werden kann, kann auch von einer **einfachen** (m+3)-Register-Maschine berechnet werden.

"Satz:" Alles, was von einer k-Register-Maschine berechnet werden kann, kann auch von einer **additiven** (k+3)-Register-Maschine berechnet werden.

"Satz:" Alles, was von einer **additiven** m-Register-Maschine berechnet werden kann, kann auch von einer **einfachen** (m+3)-Register-Maschine berechnet werden.

"Satz:" Alles, was von einer **einfachen** k-Register-Maschine berechnet werden kann, kann auch von einer **einfachen** **2**-Register-Maschine berechnet werden.

"Satz:" Alles, was von einer k-Register-Maschine berechnet werden kann, kann auch von einer **additiven** (k+3)-Register-Maschine berechnet werden.

"Satz:" Alles, was von einer **additiven** m-Register-Maschine berechnet werden kann, kann auch von einer **einfachen** (m+3)-Register-Maschine berechnet werden.

"Satz:" Alles, was von einer **einfachen** k-Register-Maschine berechnet werden kann, kann auch von einer **einfachen** **2**-Register-Maschine berechnet werden.

"Satz:" Alles, was von einer **einfachen** 2-Register-Maschine berechnet werden kann, kann auch von einer Turingmaschine berechnet werden.

"Satz:" Alles, was von einer k-Register-Maschine berechnet werden kann, kann auch von einer **additiven** (k+3)-Register-Maschine berechnet werden.

"Satz:" Alles, was von einer **additiven** m-Register-Maschine berechnet werden kann, kann auch von einer **einfachen** (m+3)-Register-Maschine berechnet werden.

"Satz:" Alles, was von einer **einfachen** k-Register-Maschine berechnet werden kann, kann auch von einer **einfachen** **2**-Register-Maschine berechnet werden.

"Satz:" Alles, was von einer **einfachen** 2-Register-Maschine berechnet werden kann, kann auch von einer Turingmaschine berechnet werden.

Das gilt alles modulo geeigneter Kodierungen der Ein- und Ausgabe.