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 $\text{op} \in \{ +, \dot{-}, \cancel{*, \text{div}, \text{mod}} \}$

$x_i = x_j \text{ op } c$

$x_i = c$

(tunix)

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

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

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.

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

"Satz:" Alles, was von einer Turingmaschine berechnet werden kann, das kann auch von einer 3-Register-Maschine berechnet werden.

Turingmaschinen haben genau die gleiche Rechenkraft wie **Registermaschinen.**

* Turingmaschinen
  (1-Band, k-Band, deterministisch, nicht-deterministisch)
* Registermaschinen
* Grammatiken
* λ–Kalkül
* Markov-Maschinen

....

Alle bekannten starken Rechenmechanismen haben die gleiche Rechenkraft

Church-Turing These:

"Alles, was sich überhaupt berechnen lässt, lässt sich von einer Turingmaschine berechnen."

Achtung: Dies ist nicht beweisbar, da "überhaupt berechnen" kein formaler Begriff ist.

Church-Turing These:

"Alles, was sich überhaupt berechnen lässt, lässt sich von einer Turingmaschine berechnen."

Achtung: Dies ist nicht beweisbar, da "überhaupt berechnen" kein formaler Begriff ist.

Church-Turing These:

"Alles, was von einer Turingmaschine nicht berechnet werden kann, lässt sich **überhaupt** nicht berechnen."

partielle Funktion $f : \Sigma^* \to \Gamma^*$ **berechenbar :**

Es gibt Turing Maschine M, die für jedes $x \in \Sigma^*$,
bei Eingabe x
mit Bandinhalt $f(x) \in \Gamma^*$ stoppt, falls $f(x)$ definiert ist,
und nicht stoppt, falls $f(x)$ nicht definiert ist.

Funktion $f : \Sigma^* \to \Gamma^*$ **"überall berechenbar"**

$f(x)$ für jedes $x \in \Sigma^*$, definiert, und f berechenbar.

Anmerkung: $f : A \to B$ berechenbar, wenn A und B jeweils als Strings kodiert werden können, sodass die entsprechende Funktion über den Strings berechenbar ist.

Die folgenden sind äquivalent für $A \subset \Sigma^*$:

- A ist Turing akzeptierbar
- Es gibt TM M mit $L(M) = A$
- A ist semi-entscheidbar
- χ_A^+ ist berechenbar
- A wird von Typ-0 Grammatik generiert
- A ist rekursiv aufzählbar
- A ist Definitionsbereich einer berechenbaren Funktion
- A ist Wertebereich einer überall berechenbaren Funktion

$A \subset \Sigma^*$ heißt **semi-entscheidbar**, wenn die
positive charakteristische Funktion von A berechenbar ist.

$$\chi_A^+ : \Sigma^* \to \{0,1\} : \quad \chi_A^+(x) = \begin{cases} 1 & x \in A \\ \text{undef} & x \notin A \end{cases}$$

Die folgenden sind äquivalent für $A \subset \Sigma^*$:

- A ist Turing akzeptierbar
- Es gibt TM M mit $L(M) = A$
- A ist semi-entscheidbar
- χ_A^+ ist berechenbar
- A wird von Typ-0 Grammatik generiert
- A ist rekursiv aufzählbar
- A ist Definitionsbereich einer berechenbaren Funktion
- A ist Wertebereich einer überall berechenbaren Funktion

$A \subset \Sigma^*$ heißt **rekursiv aufzählbar**, wenn es eine überall
berechenbare Funktion $F : \mathbb{N} \to \Sigma^*$ gibt , sodass

$$A = \{ F(0), F(1), F(2), \cdots \} = \{ F(n) \mid n \in \mathbb{N} \}$$

Die Funktion F "zählt die Menge A auf"

Die folgenden sind äquivalent für $A \subset \Sigma^*$:

- A ist Turing akzeptierbar
- Es gibt TM M mit $L(M) = A$
- A ist semi-entscheidbar
- χ_A^+ ist berechenbar
- A wird von Typ-0 Grammatik generiert
- A ist rekursiv aufzählbar
- A ist Definitionsbereich einer berechenbaren Funktion
- A ist Wertebereich einer überall berechenbaren Funktion