Maschinenmodell RAM

Random-Access-Maschine (RAM):

Eingabe:	$w = a_1 a_2 \cdots \in \Sigma^*$ (oder Σ^ω) wird sequentiell gelesen
Ausgabe:	$f(w) = b_1 b_2 \cdots \in \Sigma^*$ wird sequentiell geschrieben
Speicher:	Register $i \in \mathbb{N}$, in denen ein Wert $!i \in \mathbb{N}$ steht
	Akkumulator: = Register 0
Programm:	$P = p_1; \cdots ; p_k \in \mathit{Instruktion}^*$
Instr.zähler:	$c \in \{0, \ldots, \lvert P \rvert\}$

Berechnung durch Konfigurationsübergänge: $\kappa \vdash_P \kappa'$.

Konfiguration:	$\kappa = (w, r, c, a) \in (\mathbb{N}^*, \mathbb{N}^\omega, \lvert P \rvert, \mathbb{N}^*)$
	(Eingabe,Register,Instr.zähler,Ausgabe)
Anfangskonfig.:	!i = 0 für alle Register i,
	c = 1,
	Eingabe w, Ausgabe ϵ
Endkonfig.:	c = 0, Ausgabe $f(w)$

Berechnungsaufwand (Einheitskostenmodell)

- Jede Instruktion braucht 1 Schritt
- Jedes Register braucht Platz der Größe 1
- Nur realistisch, wenn Registerinhalte in ein Maschinenwort passen und die Anzahl gebrauchter Register die Eingabegröße nicht wesentlich übersteigt.

Instruktionen

Instruktion	*Bedeutung*
LOAD i	$0 := !i$
STORE i	$i := !0$
ADD i	$0 := !0+!i$
...	...
DIV i	$0 := !0/!i$
READ	$0 := a_{next}$
WRITE i	$b_{next} := !i$
LOAD $\underline{i}$	$0 := i$
...	...
WRITE $\underline{i}$	$b_{next} := i$
GOTO $\underline{i}$	$c := i$
JUMP0 $\underline{i}$	$c := \begin{cases} i, & \text{if } !0 = 0, \\ c := !c + 1 & \text{else,} \end{cases}$
HALT	$c := 0$

Berechnung durch Konfigurationsübergänge, $\kappa \vdash_P \kappa'$.

z.B. für $P = (READ; GOTO\ 4; READ; WRITE\ 2; HALT)$

$$(a :: w, r, 3, b) \vdash_P (w, r', 4, b)$$

mit

$$r'(i) = \begin{cases} a, & \text{falls i} = 0, \\ r(i), & \text{sonst.} \end{cases}$$

Speicher

Wir rechnen statt mit Zahlen und arithmetischen Funktionen meist mit Buchstaben, Wörtern, und Grundfunktionen wie Verkettung von Wörtern.

Dann ist es zweckmäßig, komplexe *Werte* in Registen speichern zu können.

Ein *Speicher* $S = (N, V)$ ist eine endliche Menge N von *Adressen* mit einer Abbildung $V : N \to V[N]$ in die Menge $V[N]$ der *Werte* über N zuordnet, die wie folgt aufgebaut sind:

$$
\begin{array}{lllll}
v & := & & & \mathit{Werte} \\
& & & c & \mathit{atomare Werte, z.B. Buchstaben} \\
& & \mid & n & \mathit{Adressen} \\
& & \mid & (v_1, v_2) & \mathit{Paare} \\
& & \mid & inl(v) & \mathit{Summanden} \\
& & \mid & inr(v) & \mathit{Summanden} \\
& & \mid & \ldots &
\end{array}
$$

Der Bereich der Werte $V[N]$ ist ein Bereich U mit

$$U \simeq N + V_a + (U \times U) + (U + U) + \ldots \qquad (1)$$

d.h. jedes Objekt $u \in U$ ist entweder eine Adresse, oder ein atomarer Wert, oder ein Paar, oder Element eines Datentyps wie

$$a\ \texttt{list} := \texttt{nil} \mid (a :: (a\ \texttt{list})).$$

Durch $n := v$ wird der Wert v bei der Adresse n gespeichert.

Durch $!n$ wird der bei der Adresse n gespeicherte Wert gelesen.