https://matse.paddel.xyz/spicker

Theoretische Grundlagen der Informatik

Patrick Gustav Blaneck

Letzte Änderung: 17. Juni 2021

Inhaltsverzeichnis

1 Random Access Machine (RAM)

Definition: Algorithmus

Ein *Algorithmus* ist eine **Verarbeitungsvorschrift**, die angibt wie Eingabedaten schrittweise in Ausgabedaten umgewandelt werden.

Wichtig für einen Algorithmus sind insbesondere:

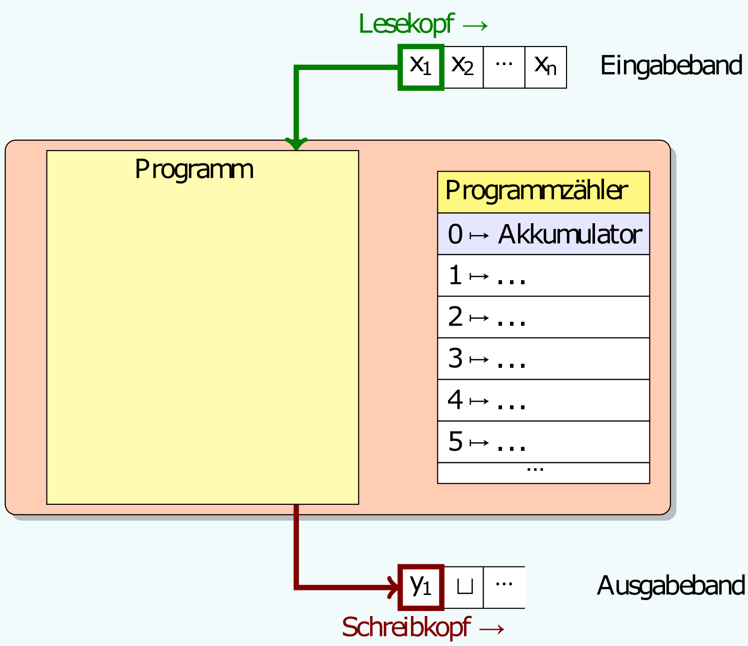
- **Korrektheit:** berechnet der Algorithmus das gewünschte?
- **Termination:** terminiert der Algorithmus immer?
- **Geschwindigkeit:** wie lange läuft der Algorithmus?
- **Speicherverbrauch:** wie viel Speicher verbraucht der Algorithmus?

Definition: Random Access Machine (RAM), informal

Die *Random Access Machine (RAM)* ist ein axiomatisch definiertes Rechnermodell.

Die RAM besteht aus:

- Programmspeicher (lesen)
- Befehlszähler
- Hauptspeicher (lesen und schreiben)
  - Speicherzelle 0 als *Akkumulator*
  - Speicherzellen nehmen ganze Zahlen auf
  - keine Größenbeschränkung
- Ein- und Ausgabeband
  - beliebig viele ganze Zahlen
  - Zugriff nicht wahlfrei



Definition: Befehlssatz der RAM

Zugriff auf die Bänder:

- `READ` n:
  liest den Wert unter dem Lesekopf, schreibt ihn an Speicherstelle n und bewegt den Lesekopf um eine Stelle nach rechts
- `WRITE` n:
  schreibt den Wert aus Speicherstelle n an die Position des Schreibkopfes auf das Ausgabeband und bewegt den Schreibkopf um eine Stelle nach rechts

Akkumulator:

- `LOAD` op_r:
  beschreibt einen *Wert*
  Arten von Operanden op_r:
  - z: *unmittelbarer* Operand ($z \in \mathbb{Z}$)
  - $[n]$: *direkt adressierter* Operand ($\sigma(n)$ - Inhalt an Speicheradresse $n \in \mathbb{N}_0$)
  - $[*n]$: *indirekt adressierter* Operand ($\sigma(\sigma(n))$ - Inhalt der Speicheradresse $\sigma(n)$)
- `STORE` op_w:
  beschreibt eine *Speicheradresse*
  Arten von Operanden op_w:
  - n: *unmittelbarer* Operand ($n \in \mathbb{N}_0$)
  - $[n]$: *direkt adressierter* Operand ($\sigma(n)$ - Inhalt an Speicheradresse $n \in \mathbb{N}_0$)

Arithmetik:

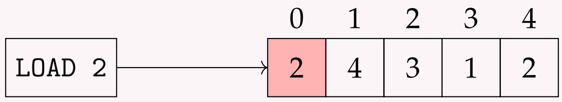
- `ADD` op_r:
  addiert den Operanden op_r zum Akkumulator
- `SUB` op_r:
  subtrahier den Operanden op_r vom Akkumulator
- `MUL` op_r:
  multipliziert den Akkumulator mit dem Operanden op_r
- `DIV` op_r:
  dividiert den Akkumulator durchden Operanden op_r (Ganzzahldivision)

Sprungbefehle:

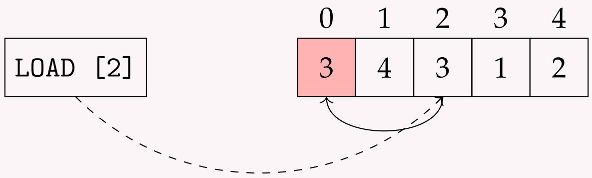
- `GOTO` p:
  die Ausführung wird in Zeile p fortgeführt
- `JZ` p (Jump Zero):
  falls der Akkumulator 0 enthält, wird die Ausführung in Zeile p fortgeführt, ansonsten bei der folgenden Zeile
- `JGTZ` p (Jump Greater Than Zero):
  falls der Akkumulator einen Wert größer als 0 enthält, wird die Ausführung in Zeile p fortgeführt, ansonsten bei der folgenden Zeile
- `HALT`:
  RAM stoppt die Ausführung

Bonus: Adressierungsarten der RAM

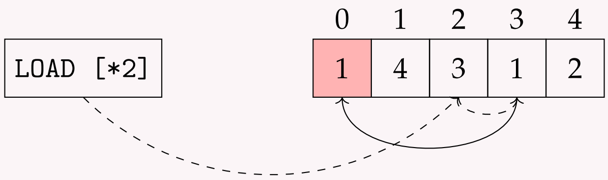
Unmittelbar:



Direkt:



Indirekt:



Definition: Speicher der RAM

Ein *Speicher* ist eine totale Funktion $\sigma : \mathbb{N}_0 \to \mathbb{Z}$, wobei Adresse 0 *Akkumulator* genannt wird.

Die Funktion σ_0 mit $\forall i \in \mathbb{N}_0 : \sigma_0(i) = 0$ nennen wir den *initialen Speicher*.

Seien $n \in \mathbb{N}_0$ eine Speicheradresse und $c \in \mathbb{Z}$, dann ist $\sigma[n \to c] : \mathbb{N}_0 \to \mathbb{Z}$ definiert durch[a]

$$\boxed{\sigma[n \to c](x) = \begin{cases} c & \text{falls } x = n \\ \sigma(x) & \text{sonst} \end{cases}}$$

Notation: Betrachte $\sigma[n_1 \to c_1][n_2 \to c_2] \dots [n_k \to c_k]$

- Wir schreiben stattdessen auch $\sigma[n_1 \to c_1, n_2 \to c_2, \dots, n_k \to c_k]$.
- Für ein n_i notieren wir immer nur das letzte Paar $n_i \to c_j$, da dieses alle vorangegangenen überschreibt (z.B. $\sigma[0 \to 1, 1 \to 3]$ statt $\sigma[1 \to 7, 0 \to 1, 1 \to 3]$)

[a] Verbal: Der Speicheradresse n wird der Wert c zugeordnet. $\sigma(x)$ gibt dann den Wert an der Stelle x zurück.

Definition: Bänder der RAM

Sei $N = \{1, \ldots, n\} \in \mathbb{N}$ eine endliche Menge, dann ist ein *RAM-Band* eine Folge von n ganzen Zahlen, die wir als Funktion $\alpha : N \to \mathbb{Z}$ modellieren.

Bandoperationen:

- $\text{read}(\alpha) : (N \to \mathbb{Z}) \to (N - \{n\} \to \mathbb{Z}) \times \mathbb{Z}$ ist definiert durch[a]

$$\boxed{\text{read}(\alpha) = (\alpha', \alpha(1))} \quad \text{mit} \quad \forall i \in \{1, \ldots, n-1\} : \alpha'(i) = \alpha(i+1)$$

- $\text{write}(\alpha, v) : (N \to \mathbb{Z}) \times \mathbb{Z} \to (N \cup \{n+1\} \to \mathbb{Z})$ ist definiert durch

$$\boxed{\text{write}(\alpha, v) = \alpha'} \quad \text{mit} \quad \forall i \in \{1, \ldots, n+1\} : \alpha'(i) = \begin{cases} v & \text{falls } i = n+1 \\ \alpha(i) & \text{sonst} \end{cases}$$

Beachte: read entfernt das erste Element einer Folge, write hängt ein Element an das Ende einer Folge an.

[a] Verbal: $\text{read}(\alpha)$ entfernt die erste Position des Bandes (α' entspricht α „um eins nach links verschoben") und gibt das erste Element des alten Bandes $\alpha(1)$ zurück

Definition: Random Access Machine (RAM), formal

Eine *Random Access Machine (RAM)* ist definiert durch eine endliche Folge von RAM-Befehlen $\mathcal{R}_{am} = (s_1, \ldots, s_n)$, wobei für jedes Sprungziel gilt, dass es im Bereich $\{1, \ldots, n\}$ liegt.

Definition: Konfiguration der RAM

Sei $\mathcal{R}_{am} = (s_1, \ldots, s_n)$ eine RAM. Eine *Konfiguration* von $\mathcal{R}_{am}$ ist ein Quadrupel $(\pi, \alpha, \beta, \sigma)$, bestehend aus:

- π, dem Programmzähler mit $\pi \in \{0, \ldots, n\}$
- α, dem Eingabeband,
- β, dem Ausgabeband
- σ, dem Speicher

Für ein beliebiges α bezeichnet $1, \alpha, (), \sigma_0)$ die *Startkonfiguration* einer RAM.

Konfigurationen der Form $(0, (), \beta, \sigma)$ nennen wir *Endkonfiguration* und mit $\text{Conf}(\mathcal{R}_{am})$ bezeichnen wir die Menge aller Konfigurationen zu einer RAM $\mathcal{R}_{am}$.

Definition: Operandenfunktion

Sei $\gamma = (\pi, \alpha, \beta, \sigma)$ eine RAM-Konfiguration, dann ist die *Operandenfunktion* eval definiert durch:

- $\text{eval}(\gamma, z) = z$ für $z \in \mathbb{Z}$
- $\text{eval}(\gamma, [n]) = \sigma(n)$ für $n \in \mathbb{N}_0$
- $\text{eval}(\gamma, [*n]) = \sigma(\sigma(n))$ für $n \in \mathbb{N}_0$

Anstelle von $\text{eval}(\gamma, \chi) = z$ schreiben wir auch[a]

$$\boxed{\gamma \vdash \chi = z}$$

[a] Verbal: Unter der Konfiguration γ hat der Operand χ den Wert z.

Definition: Deduktionssystem

Die Rechenregeln der RAM werden wir in Form eines *Deduktionssystems* angeben. Darin haben die Regeln die Form:

$$\boxed{\frac{\text{Prämisse}_1 \cdots \text{Prämisse}_n}{\gamma \vdash \gamma'} \quad (\text{NAME})}$$

Die Regel mit Namen NAME beschreibt den

- den *Konfigurationsübergang* von γ nach γ', der nur dann möglich ist, wenn
- die Bedingungen aller Prämissen erfüllbar sind.

Das Deduktionssystem beschreibt somit eine Relation

$$\vdash : \text{ Conf}(\mathcal{R}_{am}) \times \text{Conf}(\mathcal{R}_{am})$$

die wir *Schrittrelation* nennen.

Beispiel: Schrittrelationen (Bandoperationen)

$$\frac{s_\pi = \text{READ } n \quad \text{read}(\alpha) = (\alpha', z)}{(\pi, \alpha, \beta, \sigma) \vdash (\pi + 1, \alpha', \beta, \sigma[n \to z])} \quad (\text{READ})$$

$$\frac{s_\pi = \text{WRITE } n \quad \sigma(n) = z}{(\pi, \alpha, \beta, \sigma) \vdash (\pi + 1, \alpha, \text{write}(\beta, z), \sigma)} \quad (\text{WRITE})$$

$$\frac{s_\pi = \text{LOAD } op_r \quad (\pi, \alpha, \beta, \sigma) \vdash op_r = z}{(\pi, \alpha, \beta, \sigma) \vdash (\pi + 1, \alpha, \beta, \sigma[0 \to z])} \quad (\text{LOAD})$$

$$\frac{s_\pi = \text{STORE } op_w \quad (\pi, \alpha, \beta, \sigma) \vdash op_w = n \quad n \in \mathbb{N}}{(\pi, \alpha, \beta, \sigma) \vdash (\pi + 1, \alpha, \beta, \sigma[n \to \sigma(0)])} \quad (\text{STORE})$$

Beispiel: Schrittrelationen (Arithmetik)

$$\frac{s_\pi = \text{ADD}\ op_r \quad (\pi,\alpha,\beta,\sigma) \vdash op_r = z_b \quad (\pi,\alpha,\beta,\sigma) \vdash [0] = z_a}{(\pi,\alpha,\beta,\sigma) \vdash (\pi+1,\alpha,\beta,\sigma[0 \to z_a + z_b])} \quad (\text{ADD})$$

$$\frac{s_\pi = \text{SUB}\ op_r \quad (\pi,\alpha,\beta,\sigma) \vdash op_r = z_b \quad (\pi,\alpha,\beta,\sigma) \vdash [0] = z_a}{(\pi,\alpha,\beta,\sigma) \vdash (\pi+1,\alpha,\beta,\sigma[0 \to z_a - z_b])} \quad (\text{SUB})$$

$$\frac{s_\pi = \text{MUL}\ op_r \quad (\pi,\alpha,\beta,\sigma) \vdash op_r = z_b \quad (\pi,\alpha,\beta,\sigma) \vdash [0] = z_a}{(\pi,\alpha,\beta,\sigma) \vdash (\pi+1,\alpha,\beta,\sigma[0 \to z_a \cdot z_b])} \quad (\text{MUL})$$

$$\frac{s_\pi = \text{DIV}\ op_r \quad (\pi,\alpha,\beta,\sigma) \vdash op_r = z_b \quad z_b \neq 0 \quad (\pi,\alpha,\beta,\sigma) \vdash [0] = z_a}{(\pi,\alpha,\beta,\sigma) \vdash \left(\pi+1,\alpha,\beta,\sigma[0 \to \left\lfloor \frac{z_a}{z_b} \right\rfloor]\right)} \quad (\text{DIV})$$

Beispiel: Schrittrelationen (konditional)

$$\frac{s_\pi = \text{GOTO}\ p}{(\pi,\alpha,\beta,\sigma) \vdash (p,\alpha,\beta,\sigma)} \quad (\text{GOTO})$$

$$\frac{s_\pi = \text{JZ}_1\ p \quad \sigma(0) = 0}{(\pi,\alpha,\beta,\sigma) \vdash (p,\alpha,\beta,\sigma)} \quad (\text{JZ}_1) \qquad \frac{s_\pi = \text{JZ}_2\ p \quad \sigma(0) \neq 0}{(\pi,\alpha,\beta,\sigma) \vdash (\pi+1,\alpha,\beta,\sigma)} \quad (\text{JZ}_{22})$$

$$\frac{s_\pi = \text{JGTZ}_1\ p \quad \sigma(0) > 0}{(\pi,\alpha,\beta,\sigma) \vdash (p,\alpha,\beta,\sigma)} \quad (\text{JGTZ}_1) \qquad \frac{s_\pi = \text{JGTZ}_2\ p \quad \sigma(0) \leq 0}{(\pi,\alpha,\beta,\sigma) \vdash (p,\alpha,\beta,\sigma)} \quad (\text{JGTZ}_2)$$

$$\frac{s_\pi = \text{HALT}}{(\pi,\alpha,\beta,\sigma) \vdash (0,\alpha,\beta,\sigma)} \quad (\text{HALT})$$

Definition: Abschluss der Schrittrelation

Sei $\mathcal{R}_{am} = (s_1, \ldots, s_n)$ eine RAM. Wir definieren

$$\overset{*}{\vdash}:\ \text{Conf}(\mathcal{R}_{am}) \times \text{Conf}(\mathcal{R}_{am})$$

als reflexiven und transitiven *Abschluss der Schrittrelation*:

$$\boxed{(\pi,\alpha,\beta,\sigma) \overset{*}{\vdash} (\pi_n,\alpha_n,\beta_n,\sigma_n)}$$

falls

$$\exists (\pi_1,\alpha_1,\beta_1,\sigma_1), \ldots, (\pi_n,\alpha_n,\beta_n,\sigma_n) \in \text{Conf}(\mathcal{R}_{am}):$$

$$(\pi,\alpha,\beta,\sigma) \vdash (\pi_1,\alpha_1,\beta_1,\sigma_1) \vdash \ldots \vdash (\pi_n,\alpha_n,\beta_n,\sigma_n)$$

Definition: RAM-Berechenbarkeit

Seien $\mathcal{R}_{am} = (s_1, \dots, s_n)$ eine RAM und $f : \mathbb{Z}^k \to \mathbb{Z}^l$ eine Funktion.

$\mathcal{R}_{am}$ *berechnet f*, genau dann, wenn

$$\forall (z_1, \dots, z_k) \in \mathbb{Z}^k : (1, (z_1, \dots, z_k), (), \sigma_0) \overset{*}{\vdash} (0, (), f(z_1, \dots, z_k), \sigma')$$

für ein geeignetes σ'.

Eine Funktion heißt *RAM-berechenbar*, wenn es eine RAM gibt, die sie berechnet.

Bonus: Modulo-Operator

Der *Modulo-Operator* $\bmod : \mathbb{Z} \times \mathbb{Z} \setminus \{0\} \to Z$ ist definiert durch

$$a \bmod b = a - \left\lfloor \frac{a}{b} \right\rfloor \cdot b$$

Bonus: Kongruenz

Sei $m \in \mathbb{Z} \setminus \{0\}$. Zwei Zahlen $a, b \in \mathbb{Z}$ heißen *kongruent modulo m*, genau dann, wenn

$$a \bmod m = b \bmod m$$

und wir schreiben

$$a \equiv b \pmod{m}$$

Die Menge

$$[a]_m = \{z \in \mathbb{Z} \mid z \equiv a \bmod m\}$$

nennen wir dann *Kongruenzklasse* (auch Restklasse) von a modulo m.

Definition: Schleifeninvariante

Eine *Schleifeninvariante* ist eine Aussage, die vor und nach einer Schleife und jedem Durchlauf der Schleife gilt. Sie ist damit unabhängig von der Zahl ihrer derzeitigen Durchläufe.

Für die Korrektheit der Schleifeninvatiante muss gezeigt werden, dass

- die Invariante direkt vor Ausführung der Schleife und damit auch am Anfang des ersten Schleifendurchlaufs gilt (*Initialisierung*)
- falls die Invariante am Anfang eines Schleifendurchlaufs erfüllt ist, sie dann auch am Ende erfüllt ist (*Erhaltung*), und
- sie direkt nach Beendigung der Schleife gilt (*Terminierung*).

Definition: Partielle Korrektheit

Seien P eine Vorbedingung und Q eine Nachbedingung. Ein Algorithmus heißt *partiell korrekt*, wenn er für Eingaben, unter denen P erfüllt ist, nur Ausgaben liefert, welche Q erfüllen.

Definition: Totale Korrektheit

Seien P eine Vorbedingung und Q eine Nachbedingung. Ein Algorithmus heißt *total korrekt*, falls er partiell korrekt ist und auf jeder Eingabe, die P erfüllt, terminiert.

Zum Nachweis der Termination wird die *Schleifenvariante* genutzt, für die gilt:[a]

- Ausdruck über Programmvariablen
- liefert Zahl aus $\mathbb{N}_0$
- muss in jeder Iteration verringert werden

[a] analog zu Zählvariablen

Definition: Speicherplatzverbrauch

Der *Speicherplatzverbrauch* M wird in Abhängigkeit zur Größe der Eingabe angegeben.

Einfacher Speicherplatzverbrauch:

- M entspricht Anzahl der gespeicherten Zahlen
- z.B. für mod: $M(A, B) = 3$

Realistischerer Speicherplatzverbrauch:

- M entspricht Anzahl der Bits der gespeicherten Zahlen
- z.B. für mod: $M(A, B) = 2 \cdot \lceil \log_2 A \rceil + \lceil \log_2 B \rceil$

Definition: Laufzeit

Laufzeit ist abhängig von Größe der Eingabe und entspricht der Anzahl der abgearbeiteten RAM-Kommandos.

Einfache Laufzeit:

- M entspricht genau der Anzahl der abgearbeiteten RAM-Kommandos
- z.B. für mod: $T(A, B) = 4 + \underbrace{\left\lfloor \frac{A}{B} \right\rfloor \cdot 7}_{\text{Schleife}} + 3$

Realistischere Laufzeit:

- logarithmisches Kostenmaß für arithmetische Operationen
- z.B. Aufwand für Subtraktion: Anzahl Bits des größeren Operanden
- damit gilt für mod: $T(A, B) = 4 + \lfloor \frac{A}{B} \rfloor \cdot 6 + \underbrace{T_{\text{SUB}}(A, B)}_{\text{Alle Subtraktionen}} + 3$ mit

$$T_{\text{SUB}}(x, y) = \begin{cases} 0 & \text{falls } x < y \\ T_{\text{SUB}}(x - y, y) + \max(\lceil \log_2 x \rceil, \lceil \log_2 y \rceil) & \text{sonst} \end{cases}$$

Index

Beispiele