Algorithmen und Datenstrukturen

Theoretische Grundlagen der Informatik

Prof. Dr. rer. nat. Jörg Striegnitz

Fachhochschule Aachen
striegnitz@fh-aachen.de

19. April 2024

Algorithmen und Datenstrukturen

Die Random Access Machine (RAM)

Prof. Dr. rer. nat. Jörg Striegnitz

Fachhochschule Aachen
striegnitz@fh-aachen.de

19. April 2024

Analyse von Algorithmen

- RAM bietet elementaren Operationen und (darauf aufbauend) klar definierte **Semantik**
- **Damit**: Fundierte Analyse von Algorithmen möglich

Zentrale Fragen der Analyse von Algorithmen

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

- Die ersten beiden Aspekte werden nicht (immer) im Detail behandelt.
- Geschwindigkeit und Speicherverbrauch
  - relative Angaben
    z.B. Zahl der RAM-Instruktionen, Anzahl der RAM-Speicherzellen
  - in Abhängigkeit von der Größe der Eingabedaten
    z.B. Anzahl Daten, Größe der Zahlen etc.

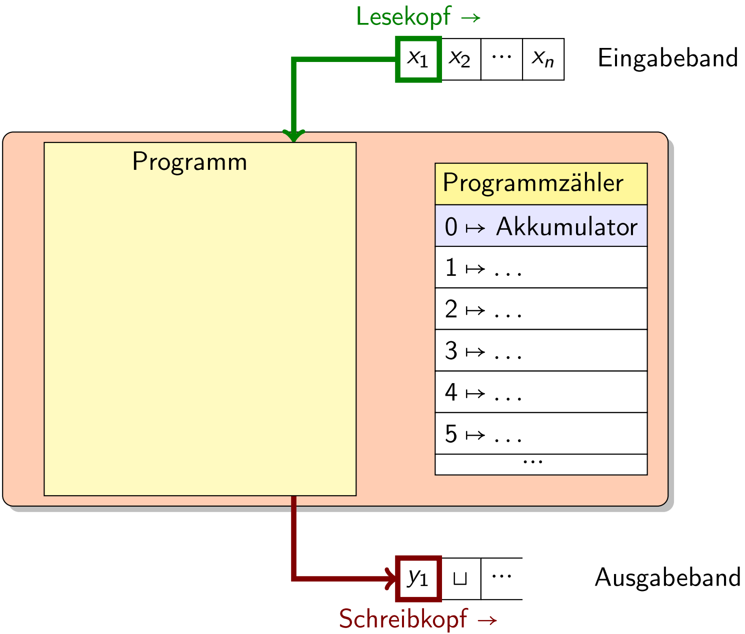
Die Random Access Machine

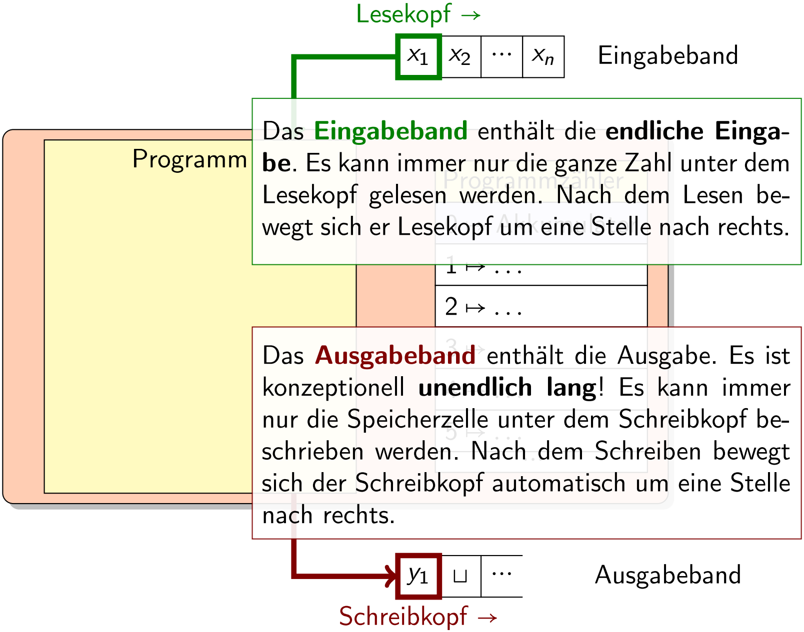
- Axiomatisch definiertes Rechnermodell
  - Registermaschine
    in Anlehnung an **Harvard-Architektur** ⇒ Programm auf Lochstreifen; Programmspeicher und Datenspeicher physikalisch getrennt (Im Gegensatz zur **von Neumann Architektur**, wo Programm und Daten sich Speicher teilen)
  - Befehlssatz definiert elementare Operationen

- Die RAM besteht aus
  - Programmspeicher (nur lesender Zugriff)
  - 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!

Achtung: Viele Variationen in der Literatur!

Schematischer Aufbau der RAM





Das **Eingabeband** enthält die **endliche Eingabe**. Es kann immer nur die ganze Zahl unter dem Lesekopf gelesen werden. Nach dem Lesen bewegt sich er Lesekopf um eine Stelle nach rechts.

Das **Ausgabeband** enthält die Ausgabe. Es ist konzeptionell **unendlich lang**! Es kann immer nur die Speicherzelle unter dem Schreibkopf beschrieben werden. Nach dem Schreiben bewegt sich der Schreibkopf automatisch um eine Stelle nach rechts.

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

Befehle der RAM: Zugriff auf den Speicher

Akkumulator: Laden und Speichern

- `LOAD` op_r
  wobei op_r ein unmittelbarer, direkt- oder indirekt adressierter Operand ist, der einen **Wert** beschreibt.

- `STORE` op_w
  wobei op_w ein unmittelbar oder direkt adressierter Operand ist, der eine **Speicheradresse** beschreibt.

Die RAM unterstützt für `LOAD` drei Arten von Operanden:

- z: **unmittelbarer** Operand; die Zahl $z \in \mathbb{Z}$
- $*n$: **direkt adressierter** Operand; $\sigma(n)$ - der Inhalt an Speicheradresse n (mit $n \in \mathbb{N}_0$)
- $**n$: **indirekt adressierter** Operand; $\sigma(\sigma(n))$ - der Inhalt an Speicheradresse $\sigma(n)$ (mit $n \in \mathbb{N}_0$)

Beispiel: Adressierungsarten

Beispiel 1.3 (Adressierungsarten der RAM)

LOAD 2 // unmittelbar

0	1	2	3
2	4	3	1

LOAD *2 // direkt

0	1	2	3
3	4	3	1

LOAD **2 // indirekt

0	1	2	3
1	4	3	1

op_r vs. op_w

Beim Schreiben mit `STORE` beschreibt der Operand eine Adresse:

- op_w: **schreibender** Zugriff:
  - n mit $n \in \mathbb{N}_0$ und
  - $*n$ mit $n \in \mathbb{N}_0$.
- op_r: **lesender** Zugriff:
  - z mit $z \in \mathbb{Z}$,
  - $*n$ mit $n \in \mathbb{N}_0$ oder
  - $**n$ mit $n \in \mathbb{N}_0$.

!

Beachte die unterschiedliche Bedeutung von z.B. der 2 in

- **LOAD 2**: hier ist die **ganze Zahl** 2 gemeint
- **STORE 2**: hier ist die 2 eine **Speicheradresse**

Befehle der RAM: Arithmetische Operationen

Die RAM kennt die folgenden **arithmetischen Operationen**, wobei der erste Operand immer der Inhalt des Akkumulators und op_r ein unmittelbar, direkt- oder indirekt adressierter Operand ist. Das Ergebnis wird immer im **Akkumulator** abgelegt.

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

Befehle der RAM: Sprungbefehle

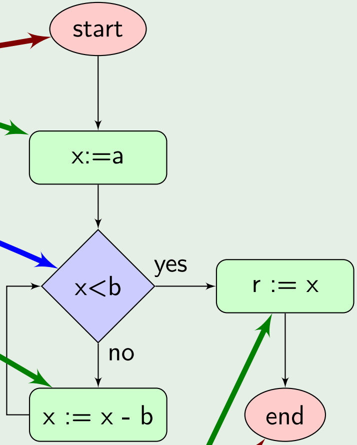
Die RAM kennt drei Sprungbefehle. $p \in \mathbb{N}$ ist das **Sprungziel** und gibt die Programmzeile an, zu der verzweigt werden soll.

- `GOTO` p
  die Ausführung wird in Zeile p fortgeführt
- `JZ` p
  Jump **Z**ero: falls der Akkumulator 0 enthält, wird die Ausführung in Zeile p fortgeführt, ansonsten bei der folgenden Zeile.
- `JGTZ` p
  Jump **G**reater **T**han **Z**ero: falls der Akkumulator einen Wert größer als 0 enthält, wird die Ausführung in Zeile p fortgeführt, ansonsten bei der folgenden Zeile.
- `HALT`
  Die RAM stoppt die Ausführung

mod auf der RAM

Lösung 1.5 (Algorithmus 'mod' auf der RAM)

```
 1: READ 1      a ≅ σ(1)
 2: READ 2      b ≅ σ(2)
 3: LOAD *1     a→Akku
 4: STORE 3     x ≅ σ(3) = a
 5: LOAD *2     b→Akku
 6: SUB *3      b-x→Akku
 7: JGTZ 12     (b-x)>0?
 8: LOAD *3     x→Akku
 9: SUB *2      x-b→Akku
10: STORE 3     x=x-b
11: GOTO 5      LOOP
12: LOAD *3     x->Akku
13: WRITE 0     x auf Band
14: HALT        Done.
```



Modellierung der RAM: Speicher

Wir wollen die Funktionsweise der RAM nicht textuell beschreiben - wir wollen sie **mathematisch modellieren**.

Speicher und Bänder stellen wir als Funktionen dar:

- Ein **Speicher** ist eine totale Funktion $\sigma : \mathbb{N}_0 \to \mathbb{Z}$ $\Rightarrow$ Speicher unendlich groß!
- Adresse 0 nennen wir **Akkumulator**
- 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 $\sigma[n \mapsto c] : \mathbb{N}_0 \to \mathbb{Z}$ definiert durch

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

$\longrightarrow$ damit können wir den Speicher "punktweise" ändern

Notation: Betrachte $\sigma[n_1 \mapsto c_1][n_2 \mapsto c_2]\cdots[n_k \mapsto c_k]$,

1. dann schreiben wir auch $\sigma[n_1 \mapsto c_1, n_2 \mapsto c_2, \cdots, n_k \mapsto c_k]$ und
2. für ein n_i notieren wir immer nur das letzte Paar $n_i \mapsto c_j$, da dieses alle vorangegangenen überschreibt (z.B. $\sigma[0 \mapsto 1, 1 \mapsto 3]$ anstelle von $\sigma[1 \mapsto 7, 0 \mapsto 1, 1 \mapsto 3]$)

Modellierung der RAM: Bänder

- Sei $N = \{1, \ldots, n\} \subseteq \mathbb{N}$ eine **endliche** Menge, dann ist ein RAM-**Band** eine Folge von n ganzen Zahlen, die wir als Funktion $\alpha : N \to \mathbb{Z}$ modellieren
- Wir führen zwei Operationen auf Bändern ein:
  - $\text{read}(\alpha) : (N \to \mathbb{Z}) \to (N - \{n\} \to \mathbb{Z}) \times \mathbb{Z}$ ist definiert durch

    $$\text{read}(\alpha) = \big(\alpha', \alpha(1)\big)$$

    wobei

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

    $$\text{write}(\alpha, v) = \alpha'$$

    wobei

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

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

Formale Definition der RAM

Definition 1.2 (RAM)

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.

Konvention: Sofern $s_n \neq \texttt{HALT}$, ergänzen wir implizit $s_{n+1} = \texttt{HALT}$ (setzen also $\mathcal{R}_{am} = (s_1, \ldots, s_n, \texttt{HALT})$).

Modellierung der RAM: Konfiguration

Die RAM führt Befehle **schrittweise** aus. Ein Befehl hat dabei Auswirkungen auf die aktuelle **Konfiguration** der RAM:

Definition 1.3 (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 und
- 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 $Conf(\mathcal{R}_{am})$ bezeichnen wir die Menge aller Konfigurationen zu einer RAM $\mathcal{R}_{am}$.

Formale Semantik der RAM: Operandenfunktion

Die Operandenfunktion ist eine Hilfsfunktion mit der wir die Operanden der Lade- und Speicheroperationen sowie der arithmetischen Operationen unter einer bestimmten Konfiguration der RAM ausrechnen können:

Definition 1.4 (Operandenfunktion)

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

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

Konvention: Anstelle von $\mathrm{eval}(\gamma, \chi) = z$ schreiben wir auch

$$\gamma \vdash \chi = z$$

lies: *"Unter der Konfiguration γ hat der Operand χ den Wert z"*

Deduktionssysteme

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

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

Diese Regel mit Namen **NAME** beschreibt den

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

Das Deduktionssystem beschreibt somit eine Relation

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

die wir **Schrittrelation** nennen.

Formale Semantik der RAM: Schrittrelation (1)

Definition 1.5 (Schrittrelation)

Sei $\mathcal{R}_{am} = (s_1, \dots, s_n)$ eine RAM. Ein **Konfigurationsübergang** (ein Schritt) von $\mathcal{R}_{am}$ ist eine Relation $\vdash : Conf(\mathcal{R}_{am}) \times Conf(\mathcal{R}_{am})$, die definiert ist durch:

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

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

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

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

Formale Semantik der RAM: Schrittrelation (2)

Definition 1.5 (Schrittrelation - Fortsetzung)

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

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

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

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

Formale Semantik der RAM: Schrittrelation (3)

Definition 1.5 (Schrittrelation - Fortsetzung)

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

$$\frac{s_\pi = \texttt{JZ}\ p \quad \sigma(0) = 0}{(\pi,\alpha,\beta,\sigma) \vdash (p,\alpha,\beta,\sigma)}\ (\textbf{JZ}) \quad \frac{s_\pi = \texttt{JZ}\ p \quad \sigma(0) \neq 0}{(\pi,\alpha,\beta,\sigma) \vdash (\pi+1,\alpha,\beta,\sigma)}\ (\textbf{JZ}_2)$$

$$\frac{s_\pi = \texttt{JGTZ}\ p \quad \sigma(0) > 0}{(\pi,\alpha,\beta,\sigma) \vdash (p,\alpha,\beta,\sigma)}\ (\textbf{JGTZ})$$

$$\frac{s_\pi = \texttt{JGTZ}\ p \quad \sigma(0) \leq 0}{(\pi,\alpha,\beta,\sigma) \vdash (\pi+1,\alpha,\beta,\sigma)}\ (\textbf{JGTZ}_2)$$

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

5 3 Eingabeband

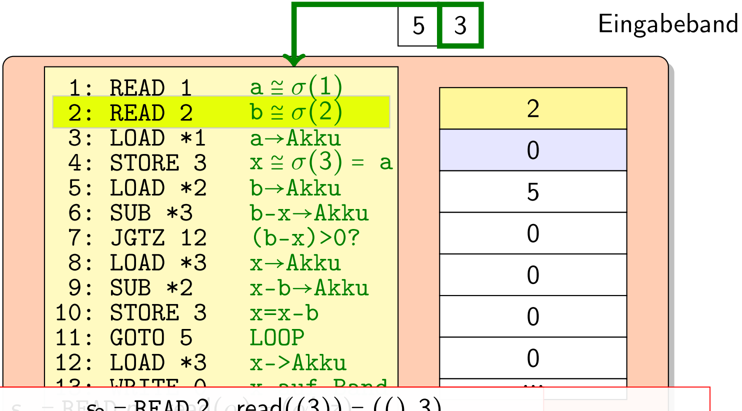
```
 1: READ 1      a ≅ σ(1)
 2: READ 2      b ≅ σ(2)
 3: LOAD *1     a→Akku
 4: STORE 3     x ≅ σ(3) = a
 5: LOAD *2     b→Akku
 6: SUB *3      b-x→Akku
 7: JGTZ 12     (b-x)>0?
 8: LOAD *3     x→Akku
 9: SUB *2      x-b→Akku
10: STORE 3     x=x-b
11: GOTO 5      LOOP
12: LOAD *3     x->Akku
13: WRITE 0     x auf Band
```

1
0
0
0
0
0
0
...

$$\frac{s_1 = \mathtt{READ}\,1 \quad \mathrm{read}((5,3)) = ((3),5)}{(1,(5,3),(),\sigma_0) \vdash (2,(3),(),\sigma_0[1 \mapsto 5])}\ (\textbf{READ})$$

Ausgabeband

$$(1,(5,3),(),\sigma_0)$$



$$\frac{s_2 = \mathrm{READ}\ 2 \qquad \mathrm{read}((3)) = ((),3)}{(2,(3),(),\sigma_0[1 \mapsto 5]) \vdash (3,(),(),\sigma_0[1 \mapsto 5][2 \mapsto 3])}\ (\mathbf{READ})$$

$$(2,(3),(),\sigma_0[1 \mapsto 5])$$

5	3	

Eingabeband

```
 1: READ 1      a ≅ σ(1)
 2: READ 2      b ≅ σ(2)
 3: LOAD *1     a→Akku
 4: STORE 3     x ≅ σ(3) = a
 5: LOAD *2     b→Akku
 6: SUB *3      b-x→Akku
 7: JGTZ 12     (b-x)>0?
 8: LOAD *3     x→Akku
 9: SUB *2      x-b→Akku
10: STORE 3     x=x-b
11: GOTO 5      LOOP
12: LOAD *3     x->Akku
13: WRITE 0     x auf Band
```

3
0
5
3
0
0
0
...

$$\frac{s_3 = \texttt{LOAD } *1 \quad (3,(),(),\sigma_0[1\mapsto 5][2\mapsto 3]) \vdash [1] = 5}{(3,(),(),\sigma_0[1\mapsto 5][2\mapsto 3]) \vdash (4,(),(),\sigma_0[1\mapsto 5][2\mapsto 3][0\mapsto 5])}\ (\textbf{LOAD})$$

Ausgabeband

$$(3,(),(),\sigma_0[1\mapsto 5][2\mapsto 3])$$

Beispiel: Berechnung einer RAM (4)

5	3		Eingabeband

```
 1: READ 1      a ≅ σ(1)
 2: READ 2      b ≅ σ(2)
 3: LOAD *1     a→Akku
 4: STORE 3     x ≅ σ(3) = a
 5: LOAD *2     b→Akku
 6: SUB *3      b-x→Akku
 7: JGTZ 12     (b-x)>0?
 8: LOAD *3     x→Akku
 9: SUB *2      x-b→Akku
10: STORE 3     x=x-b
11: GOTO 5      LOOP
12: LOAD *3     x->Akku
13: WRITE 0     x auf Band
```

4
5
5
3
0
0
0
...

$$\frac{s_4 = \mathtt{STORE}\ 3 \qquad (4,(),(),\sigma_0[1\mapsto 5][2\mapsto 3][0\mapsto 5]) \vdash 3 = 3 \qquad 3 \in \mathbb{N}}{(4,(),(),\sigma_0[\cdots]) \vdash (5,(),(),\sigma_0[1\mapsto 5][2\mapsto 3][0\mapsto 5][3\mapsto 5])}\ (\mathbf{STORE})$$

Ausgabeband

$$(4,(),(),\sigma_0[1\mapsto 5][2\mapsto 3][0\mapsto 5])$$

Beispiel: Berechnung einer RAM (5)

			Eingabeband
5	3		

```
 1: READ 1      a ≅ σ(1)
 2: READ 2      b ≅ σ(2)
 3: LOAD *1     a→Akku
 4: STORE 3     x ≅ σ(3) = a
 5: LOAD *2     b→Akku
 6: SUB *3      b-x→Akku
 7: JGTZ 12     (b-x)>0?
 8: LOAD *3     x→Akku
 9: SUB *2      x-b→Akku
10: STORE 3     x=x-b
11: GOTO 5      LOOP
12: LOAD *3     x->Akku
13: WRITE 0     x auf Band
```

7
-2
5
3
5
0
0
...

$$\frac{s_7 = \text{JGTZ } 12 \quad -2 \leq 0}{(7, (), (), \sigma_0[0 \mapsto -2, \cdots]) \vdash (8, (), (), \sigma_0[0 \mapsto -2, \cdots])}\ (\textbf{JGTZ}_2)$$

Ausgabeband

$$(7, (), (), \sigma_0[0 \mapsto -2, 1 \mapsto 5, 2 \mapsto 3, 3 \mapsto 5])$$

Beispiel: Berechnung einer RAM - Übersicht

Insgesamt ergibt sich:

$$
\begin{array}{rcl}
(1,(5,3),(),\sigma_0) & & \\
/*\ \texttt{READ 1}\ */ & \vdash & (2,(3),(),\sigma_0[1\mapsto 5]) \\
/*\ \texttt{READ 2}\ */ & \vdash & (3,(),(),\sigma_0[1\mapsto 5, 2\mapsto 3]) \\
/*\ \texttt{LOAD *1}\ */ & \vdash & (4,(),(),\sigma_0[0\mapsto 5, 1\mapsto 5, 2\mapsto 3]) \\
/*\ \texttt{STORE 3}\ */ & \vdash & (5,(),(),\sigma_0[0\mapsto 5, 1\mapsto 5, 2\mapsto 3, 3\mapsto 5]) \\
/*\ \texttt{LOAD *2}\ */ & \vdash & (6,(),(),\sigma_0[0\mapsto 3, 1\mapsto 5, 2\mapsto 3, 3\mapsto 5]) \\
/*\ \texttt{SUB *3}\ */ & \vdash & (7,(),(),\sigma_0[0\mapsto -2, 1\mapsto 5, 2\mapsto 3, 3\mapsto 5]) \\
/*\ \texttt{JGTZ 12}\ */ & \vdash & (8,(),(),\sigma_0[0\mapsto -2, 1\mapsto 5, 2\mapsto 3, 3\mapsto 5]) \\
/*\ \texttt{LOAD *3}\ */ & \vdash & (9,(),(),\sigma_0[0\mapsto 5, 1\mapsto 5, 2\mapsto 3, 3\mapsto 5]) \\
/*\ \texttt{SUB *2}\ */ & \vdash & (10,(),(),\sigma_0[0\mapsto 2, 1\mapsto 5, 2\mapsto 3, 3\mapsto 5]) \\
/*\ \texttt{STORE 3}\ */ & \vdash & (11,(),(),\sigma_0[0\mapsto 2, 1\mapsto 5, 2\mapsto 3, 3\mapsto 2]) \\
/*\ \texttt{GOTO 5}\ */ & \vdash & (5,(),(),\sigma_0[0\mapsto 2, 1\mapsto 5, 2\mapsto 3, 3\mapsto 2]) \\
/*\ \texttt{LOAD *2}\ */ & \vdash & (6,(),(),\sigma_0[0\mapsto 3, 1\mapsto 5, 2\mapsto 3, 3\mapsto 2]) \\
/*\ \texttt{SUB *3}\ */ & \vdash & (6,(),(),\sigma_0[0\mapsto 1, 1\mapsto 5, 2\mapsto 3, 3\mapsto 2]) \\
/*\ \texttt{JGTZ 12}\ */ & \vdash & (12,(),(),\sigma_0[0\mapsto 1, 1\mapsto 5, 2\mapsto 3, 3\mapsto 2]) \\
/*\ \texttt{LOAD *3}\ */ & \vdash & (13,(),(),\sigma_0[0\mapsto 2, 1\mapsto 5, 2\mapsto 3, 3\mapsto 2]) \\
/*\ \texttt{WRITE}\ */ & \vdash & (14,(),(2),\sigma_0[0\mapsto 2, 1\mapsto 5, 2\mapsto 3, 3\mapsto 2]) \\
/*\ \texttt{HALT}\ */ & \vdash & (0,(),(2),\sigma_0[0\mapsto 2, 1\mapsto 5, 2\mapsto 3, 3\mapsto 2])
\end{array}
$$

Die folgende Definition erlaubt uns die Berechnung einer RAM ohne Angabe von Zwischenschritten zu beschreiben:

Definition 1.6 (Abschluss der Schrittrelation)

Sei $\mathcal{R}_{am} = (s_1, \ldots, s_n)$ eine RAM. Wir definieren $\overset{*}{\vdash}$: $\mathit{Config}(\mathcal{R}_{am}) \times \mathit{Config}(\mathcal{R}_{am})$ als reflexiven und transitiven **Abschluss** der Schrittrelation:

- $(\pi, \alpha, \beta, \sigma) \overset{*}{\vdash} (\pi_n, \alpha_n, \beta_n, \sigma_n)$, falls $\exists(\pi_1, \alpha_1, \beta_1, \gamma_1) \in \mathit{Config}(\mathcal{R}_{am}), \cdots, \exists(\pi_n, \alpha_n, \beta_n, \gamma_n) \in \mathit{Config}(\mathcal{R}_{am})$, so dass

$$(\pi, \alpha, \beta, \sigma) \vdash (\pi_1, \alpha_2, \beta_2, \sigma_2) \vdash \cdots \vdash (\pi_n, \alpha_n, \beta_n, \sigma_n)$$

Für das Beispiel der vorangegangenen Folien gilt dann:

$$(1, (5, 3), (), \sigma_0) \quad \overset{*}{\vdash} \quad (0, (), (2), \sigma_0[0 \mapsto 2, 1 \mapsto 5, 2 \mapsto 3, 3 \mapsto 2])$$

Formale Semantik der RAM: RAM-Berechenbarkeit

Definition 1.7 (Von einer RAM berechnete Funktion, RAM-Berechenbarkeit)

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

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

für ein geeignetes σ'. Eine Funktion heißt **RAM-berechenbar**, wenn es eine RAM gibt, die sie berechnet.

Formale Semantik ist der Schlüssel für

- Korrektheitsbeweise

  Beispiel folgt ...
- Beweise zu Grenzen der RAM-Berechenbarkeit

  d.h. zum Beweis, dass etwas **nicht** RAM-berechenbar ist - wird hier nicht weiter behandelt

Modellierung von Programmiersystemen

Ein Modell für Programmiersystemen (Programmiersprachen, Automaten, Maschinen) erfolgt häufig aus folgenden Komponenten:

- **Syntax / Modell**
  Legt den Aufbau der Sprache / des Systems fest.
- **Konfiguration**
  Dient zur Beschreibung des dynamischen Verhaltens. Darin eingeschlossen sind oft Operationen auf den Komponenten einer Konfiguration.
- **Schrittrelation**
  Sie beschreibt, wie sich syntaktischen Konstrukte auf die Konfiguration auswirken.
- Definition der **Leistungsfähigkeit**
  Man legt fest, was genau das Modell berechnen kann. Damit sind auch die Grenzen des Modells "greifbar".

Wozu?

Lohnt sich der Aufwand?