Turingmaschinen und Registermaschinen

Wie man eine Registermaschine in eine Turingmaschine übersetzt und welche Erkenntnis man daraus gewinnt

Florian Loch

Theoriesemniar bei Prof. Dr. Heinrich Braun und Dr. Martin Holzer

17.03.2015

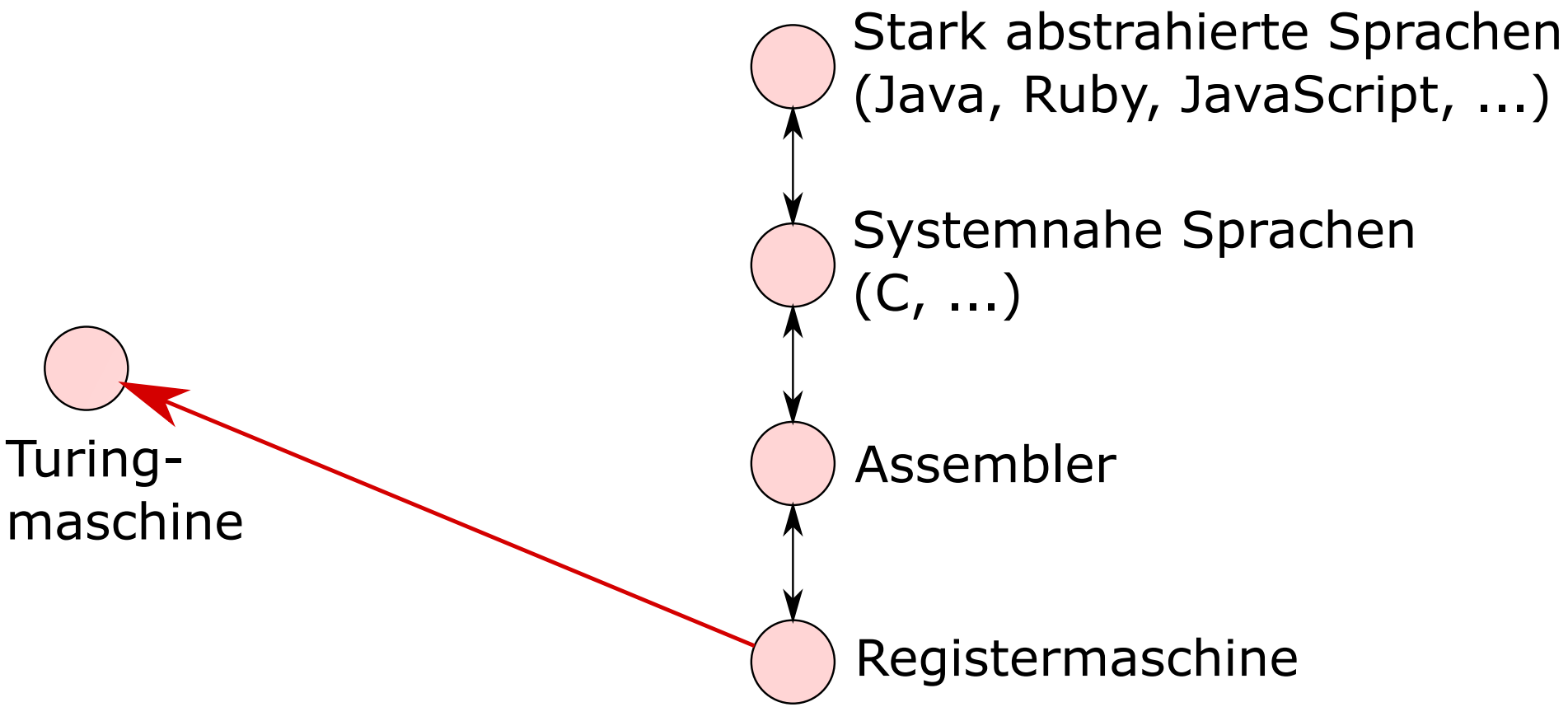
Inhaltsverzeichnis

Motivation

- Turingmaschine ist Teil des Fundaments der theoretischen Informatik
- Algorithmen werden jedoch für registerorientierte Maschinen entwickelt
- Durch Simulation einer Registermaschine innerhalb einer TM ist der Beweis möglich, ...
  - ...dass eine TM *mindestens* so mächtig ist wie eine Registermaschine
  - ...dass ein Algorithmus ebenfalls auf einer TM umgesetzt werden kann [1]
  - ...dass ein Algorithmus und dessen Zeitkomplexität prinzipiell unabhängig von der Rechner-Architektur ist (ARM, x86, etc.)
  - ...dass er in jeder Turing-Vollständigen Sprache implementiert werden kann

[1] i. Allg. mit einer polynomiellen Zeitkomplexitätsverschlechterung, bspw. wird aus $T(n) \Rightarrow T^3(n)$)

Motivation (II)



Überblick

Wiederholung: Deterministische Turing Maschine

Eine DTM M ist wie folgt definiert:

$$M = \{Q, \Gamma, \Sigma, B, q_0, \delta, [F]\}$$

- Q: Menge aller Zustände, die M annehmen kann
- Γ: Bandalphabet
- Σ: Eingabealphabet, $\Sigma \subseteq \Gamma$
- B: Blank-Zeichen, $B \in \Gamma \setminus \Sigma$
- q_0: Startzustand von M, $q_0 \in Q$
- δ: Zustandsüberführungsfunktion, $\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, L, N\}$
- T: Implizite Menge aller Endzustände. Umfasst q, die bei beliebigem a keine Zustandsänderung oder Bewegung des Kopfes bewirken: $\delta(q, a) = (q, a, N)$
- F: Menge der Finalzustände, $F \subseteq T$, nur bei binär antwortendem M

Wiederholung: Funktionsweise der Turingmaschine

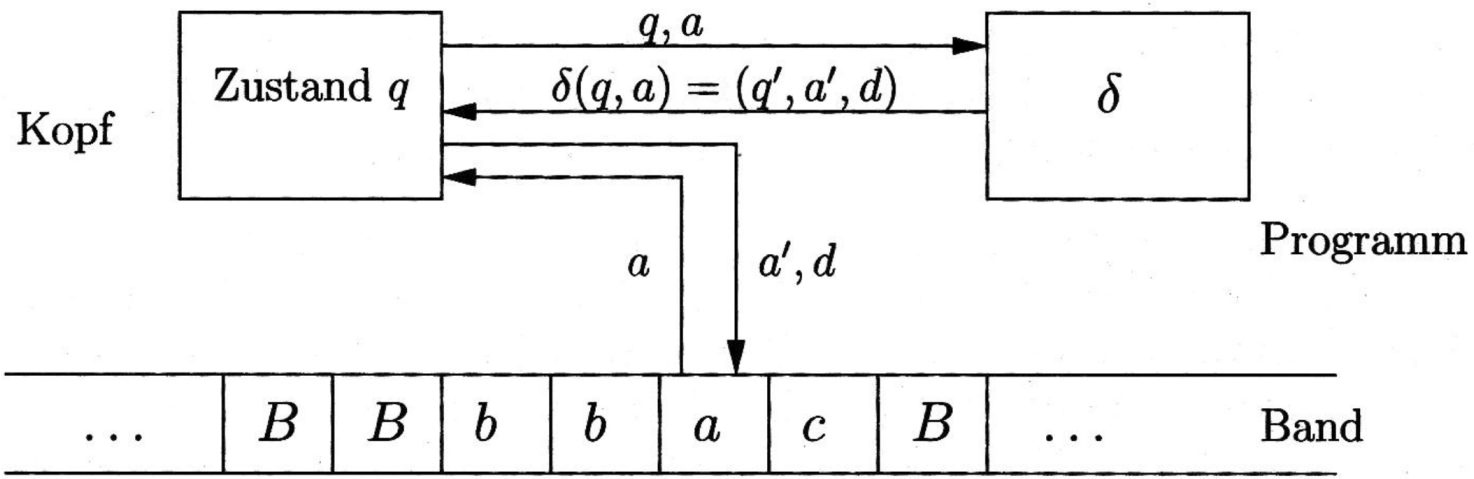


Abbildung: Aufbau einer TM [1, S. 10]

Überblick

Beschreibung einer Registermaschine

- Random Access Machine, kurz RAM
- An reale Computer angelehntes Modell eines "Prozessors", mit Assembler programmierten CPUs nachempfunden
- Reale Probleme wie Overflow etc. werden nicht berücksichtigt

Beschreibung einer Registermaschine (II)

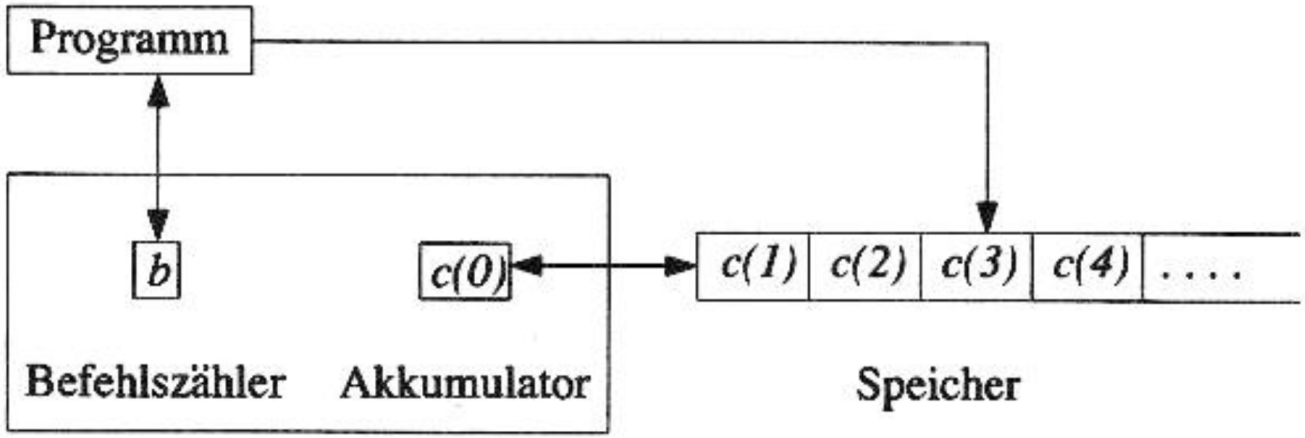


Abbildung: Aufbau einer RAM [1, S. 7]

- Verfügt über eine unbegrenzte Menge an Registern, R, welche beliebige $x \in \mathbb{N}$ enthalten können
- Für jedes Register R_i in R gilt, dass es *direkt* oder *indirekt* adressiert werden kann
- R_0 ist per Definition der Akkumulator
- Programm P mit endlicher Länge, $p = |P|$
- Im Befehlszähler b wird die aktuelle Programmzeile vermerkt

Befehle einer Registermaschine

$LOAD\ i:$	$c(0) := c(i),\ b := b+1.$
$STORE\ i:$	$c(i) := c(0),\ b := b+1.$
$ADD\ i:$	$c(0) := c(0)+c(i),\ b := b+1.$
$SUB\ i:$	$c(0) := \max\{c(0)-c(i),0\},\ b := b+1.$
$MULT\ i:$	$c(0) := c(0)*c(i),\ b := b+1.$
$DIV\ i:$	$c(0) := \lfloor c(0)/c(i) \rfloor,\ b := b+1.$
$GO\ TO\ j$	$b := j.$
$IF\ c(0)?\ l\ GO\ TO\ j$	$b := j$ falls $c(0)?\ l$ wahr ist, und
	$b := b+1$ sonst.
	(Dabei ist $? \in \{=, <, \leq, >, \geq\}$).
END	$b := b.$

Abbildung: Standard-Befehle einer RAM [1, S. 8]

Daneben gibt es noch Befehlsvarianten für die Arbeit mit konstanten Werten und für indirekte Adressierung.

Funktionsweise der Registermaschine: Beispiel

```
1 LOAD 1      //Wert aus Register 1 laden
2 ADD 2       //Wert aus Register 2 aufaddieren
3 STORE 1     //Wert in Register 1 zurueckschreiben
```

Überblick

Definitionen

- Simulieren $\Rightarrow$ Entwurf einer von M', welche das Verhalten einer Maschine M schrittweise nachahmt
- k ist ein konstanter, endlicher Wert aus $\mathbb{N}$

Satz: Überführung von RAM zu TM

Satz

Jede logarithmisch $t(n)$-zeitbeschränkte Registermaschine kann für ein Polynom q durch eine $O(q(n + t(n)))$-zeitbeschränkte Turingmaschine simuliert werden. [1, S. 17s]

Idee und Ansatz

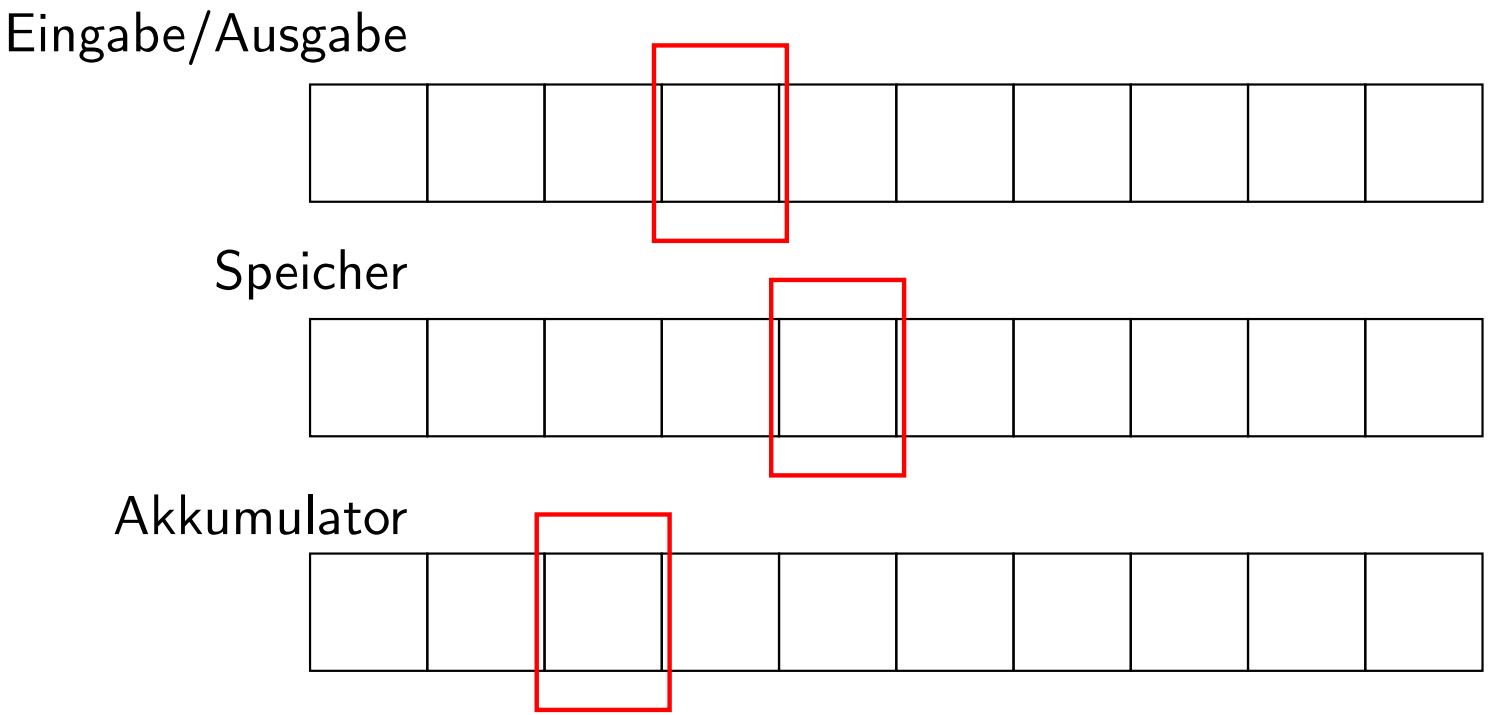
Idee: Schrittweises überführen der Registermaschine in Turingmaschine:

RAM $\rightarrow$ **Mehrband-TM** $\rightarrow$ **Mehrspur-TM** $\Rightarrow$ **(Einspur-)TM**

Die Mehrband-Turingmaschine

- Verfügt über k unbegrenzte Bänder
- Pro Band existiert ein eigener LS-Kopf
- LS-Köpfe können unabhängig von einander bewegt werden
- Das pro Schritt gelesene Wort $w \in \Gamma^k$ setzt sich aus den Zellinhalten an den Positionen der LS-Köpfe zusammen
- Zustandsüberführungsfunktion: $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{R, L, N\}^k$

Die Mehrband-Turingmaschine



Überführung hin zu Mehrband-Turingmaschine

Treffen wir nun zunächst folgende Vereinbarungen:

- $c(i)$ liefert den Inhalt von Register i; jedoch nur für tatsächlich im Laufe der Ausführung verwendete Register
  - Register in RAM kann beliebigen Wert ($c(i) \in \mathbb{N}$ annehmen)
  - Turingmaschine hat fixes Bandalhphabet Γ
- $bin\,(i)$ liefert die Binärdarstellung des Wertes i
- $bin\,(c\,(i))$ liefert die Binärdarstellung des in R_i enthaltenen Wertes

Überführung hin zu Mehrband-Turingmaschine (II)

Wir wollen nun ein Schema zur Übertragung aufstellen:

- Der Programmzähler b wird durch die Zustände realisiert
- Ein- und Ausgabe werden auf Band 1 geschrieben
- Band 2 wird zur Simulation des Speichers/Register der RAM verwendet: $\&bin(i_1)\#bin(c(i_1))\& \ldots \&bin(i_m)\#bin(c(i_m))\#\#$

- Der besseren Übersicht wegen verwenden wir für den Akkumulator ein eigenes Band, Band 3

Überführung hin zu Mehrband-Turingmaschine (III)

- Definieren von $p + 2$ Unterprogrammen in U
  - U_0 zur Übertragung der Eingabe (Band 1) in den entsprechenden Bereich des Speichers (die Register) (Band 2)
  - $U_1 \ldots U_p$ für die p Programmzeilen
  - U_{p+1} zur Ausgabe auf Band 1

$\Rightarrow$ Anschließend sequentielles Ausführen von U_0 bis U_{p+1}

Überführung hin zu Mehrband-Turingmaschine (III)

- Definieren von $p + 2$ Unterprogrammen in U
  - U_0 zur Übertragung der Eingabe (Band 1) in den entsprechenden Bereich des Speichers (die Register) (Band 2)
  - $U_1 \ldots U_p$ für die p Programmzeilen
  - U_{p+1} zur Ausgabe auf Band 1

$\Rightarrow$ Anschließend sequentielles Ausführen von U_0 bis U_{p+1}

Überführung hin zu Mehrband-Turingmaschine (III)

- Definieren von $p + 2$ Unterprogrammen in U
  - U_0 zur Übertragung der Eingabe (Band 1) in den entsprechenden Bereich des Speichers (die Register) (Band 2)
  - $U_1 \ldots U_p$ für die p Programmzeilen
  - U_{p+1} zur Ausgabe auf Band 1

$\Rightarrow$ Anschließend sequentielles Ausführen von U_0 bis U_{p+1}

Überführung hin zu Mehrband-Turingmaschine (III)

- Definieren von $p + 2$ Unterprogrammen in U
  - U_0 zur Übertragung der Eingabe (Band 1) in den entsprechenden Bereich des Speichers (die Register) (Band 2)
  - $U_1 \ldots U_p$ für die p Programmzeilen
  - U_{p+1} zur Ausgabe auf Band 1

$\Rightarrow$ Anschließend sequentielles Ausführen von U_0 bis U_{p+1}

Überführung hin zu Mehrband-Turingmaschine (III)

- Definieren von $p + 2$ Unterprogrammen in U
  - U_0 zur Übertragung der Eingabe (Band 1) in den entsprechenden Bereich des Speichers (die Register) (Band 2)
  - $U_1 \ldots U_p$ für die p Programmzeilen
  - U_{p+1} zur Ausgabe auf Band 1

$\Rightarrow$ Anschließend sequentielles Ausführen von U_0 bis U_{p+1}

Funktionsweise der Registermaschine: Beispiel

```
1 LOAD 1      //Wert aus Register 1 laden
2 ADD 2       //Wert aus Register 2 aufaddieren
3 STORE 1     //Wert in Register 1 zurueckschreiben
```

Überführung hin zu Mehrband-Turingmaschine: Beispiel

Notation in Pseudo-Code, da als Turingprogramm bereits jetzt nicht mehr handhabbar (Zustandsexplosion!)

U_0 (unter der Annahme, dass die Eingabe in R_1 abgelegt wird):

1. Auf Band 2 Markierung für R_1 anlegen
2. Zeichenweises Übertragen der Eingabe von Band 1 auf in R_1
3. Zu Initialposition auf Band zurücklaufen
4. Fertig. In Startzustand von U_1 übergehen

Überführung hin zu Mehrband-Turingmaschine: Beispiel (II)

U_1 alias LOAD:

1. Speicherband nach rechts ablaufen bis (nach bitweisem Vergleich der Binärdarstellung) R_1 gefunden wurde
2. $c(1)$ zeichenweise in R_0 übertragen
3. Zu Initialposition auf Band zurücklaufen
4. Fertig. In Startzustand von U_2 übergehen

U_2 bis U_4 verhalten sich analog.

Überführung hin zu Mehrband-Turingmaschine: Beispiel (II)

U_1 alias LOAD:

1. Speicherband nach rechts ablaufen bis (nach bitweisem Vergleich der Binärdarstellung) R_1 gefunden wurde
2. $c(1)$ zeichenweise in R_0 übertragen
3. Zu Initialposition auf Band zurücklaufen
4. Fertig. In Startzustand von U_2 übergehen

U_2 bis U_4 verhalten sich analog.

Überführung hin zu Mehrband-Turingmaschine: Beispiel (II)

U_1 alias LOAD:

1. Speicherband nach rechts ablaufen bis (nach bitweisem Vergleich der Binärdarstellung) R_1 gefunden wurde
2. $c(1)$ zeichenweise in R_0 übertragen
3. Zu Initialposition auf Band zurücklaufen
4. Fertig. In Startzustand von U_2 übergehen

U_2 bis U_4 verhalten sich analog.

Überführung hin zu Mehrband-Turingmaschine: Beispiel (II)

U_1 alias LOAD:

1. Speicherband nach rechts ablaufen bis (nach bitweisem Vergleich der Binärdarstellung) R_1 gefunden wurde
2. $c(1)$ zeichenweise in R_0 übertragen
3. Zu Initialposition auf Band zurücklaufen
4. Fertig. In Startzustand von U_2 übergehen

U_2 bis U_4 verhalten sich analog.

Überführung hin zu Mehrband-Turingmaschine: Beispiel (II)

U_1 alias LOAD:

1. Speicherband nach rechts ablaufen bis (nach bitweisem Vergleich der Binärdarstellung) R_1 gefunden wurde
2. $c(1)$ zeichenweise in R_0 übertragen
3. Zu Initialposition auf Band zurücklaufen
4. Fertig. In Startzustand von U_2 übergehen

U_2 bis U_4 verhalten sich analog.

Überführung hin zu Mehrband-Turingmaschine: Erkenntnisse

Folgende Feststellungen bzgl. Laufzeitkomplexität und Speicherverbrauch können getroffen werden:

- Offenkundiger Nachteil: Auf der Suche nach einem Register muss womöglich der gesamte Speicher durchlaufen werden
  $\Rightarrow$ Aus Speicherzugriffszeit $O(1)$ wird $O(s(n))$!
- Speicherverbrauchsklasse ändert sich nicht
- Bei Zeitkomplexität $O(t(n))$ und Speicherkomplexität $O(s(n))$ folgt $O(t\,(n)^2)$

Die Mehrspur-Turingmaschine

- Verfügt über 1 unendliches Turingband
- Band hat n Spuren
- Es existieren n LS-Köpfe
- Allerdings sind alle LS-Köpfe abhängig voneinander und können nur zusammen verschoben werden
- Das pro Schritt gelesene Wort $w \in \Gamma^k$ setzt sich aus den Zellinhalten an den Positionen der LS-Köpfe zusammen
- Zustandsüberführungsfunktion: $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{R, L, N\}$

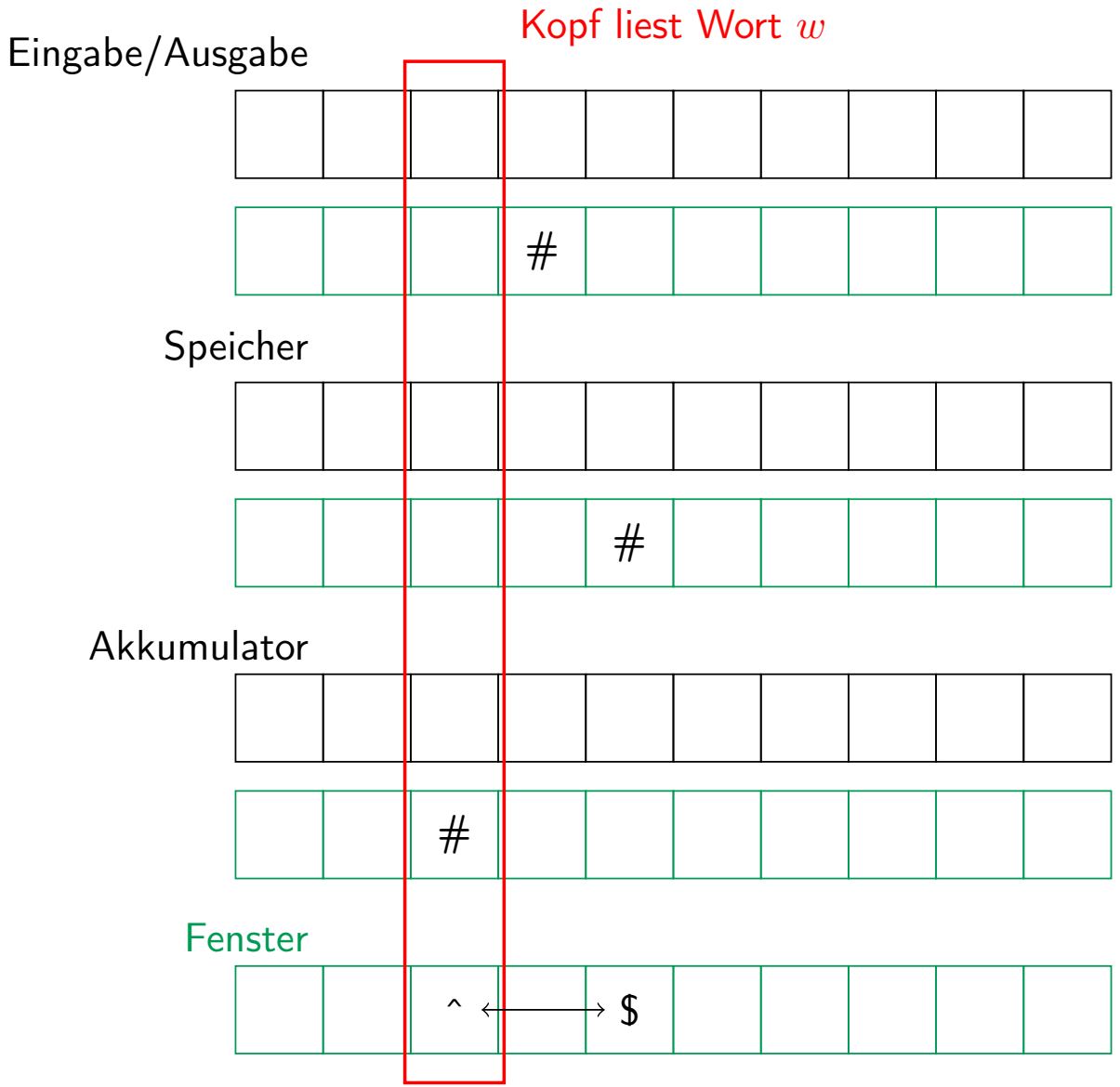
- $\Rightarrow$ Entscheidender Unterschied zu Mehrband-TM: Kann keine Informationen durch die Position der Köpfe speichern

Überführung hin zu Mehrspur-Turingmaschine

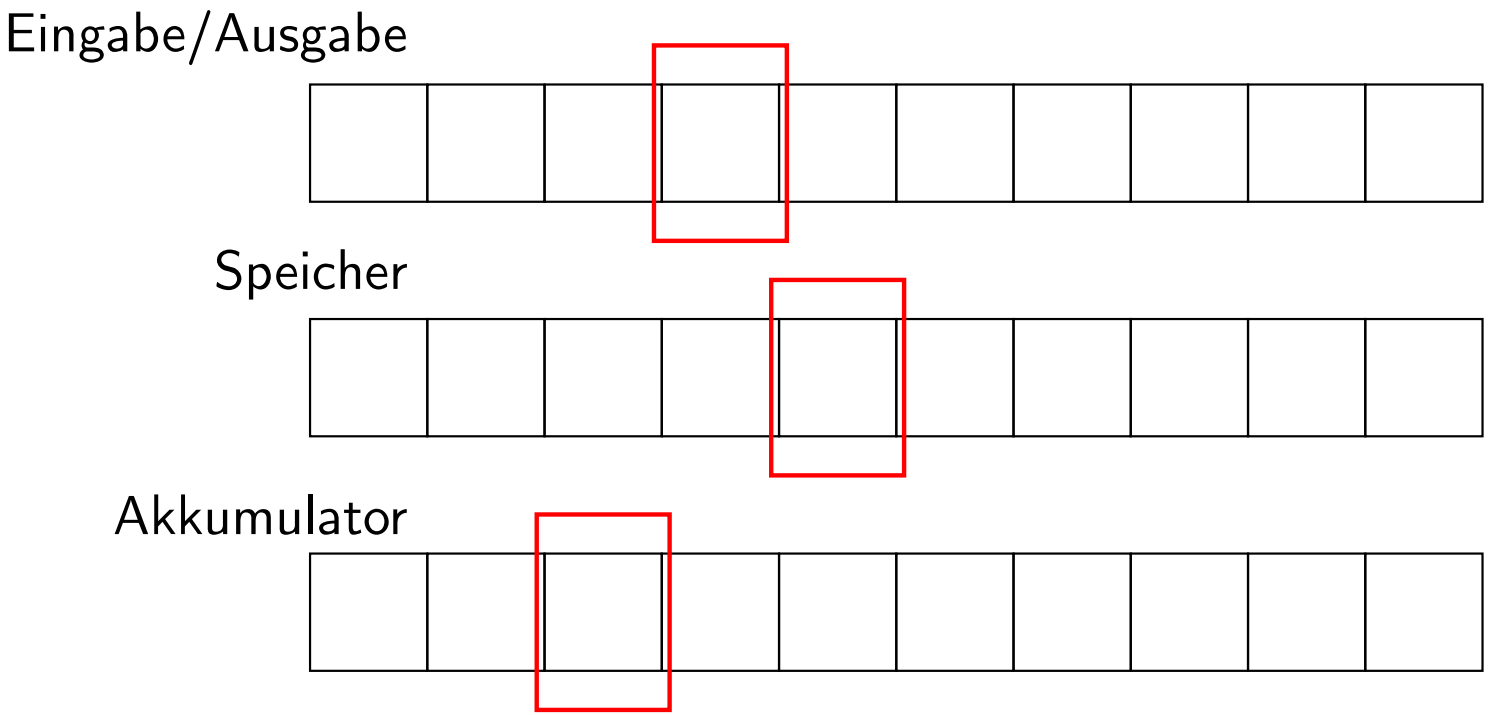
Idee zur alternativen Speicherung der Kopfpositionen:

- Bei k Bändern Verwendung von $2k + 1$ Spuren
- Die geraden Spuren sind äquivalent den einzelnen Bändern
- Die ungeraden Spuren repräsentieren Kopfpositionen der Mehrband-TM (Marker in entsprechender Zelle)
- S_k markiert das Fenster, in dem sich alle Kopfpositionen befinden: ^ . . . $
- Das aktuelle Wort w erhält man, indem man der Reihe nach die jeweils markierten Zellen "anfährt" und (von oben nach unten) einliest
  $\Rightarrow$ Dadurch Aktion bekannt, welche die Mehrband-TM ausführen würde
- Zu Beginn des t-ten Rechenschritts steht der Kopf an der Position von ^. Zustand von M' muss Zustand von M entsprechen

Überführung hin zu Mehrspur-Turingmaschine (II)



Die Mehrband-Turingmaschine



Überführung hin zu Mehrspur-Turingmaschine: Erkenntnisse

Satz

Eine k-Band-Turingmaschine M, die mit Rechenzeit $t(n)$ und Speicherplatz $s(n)$ auskommt, kann von einer Turingmaschine M' mit Zeitbedarf $O(t^2(n))$ und Speicherplatz $O(s(n))$ simuliert werden. [1, S. 15]

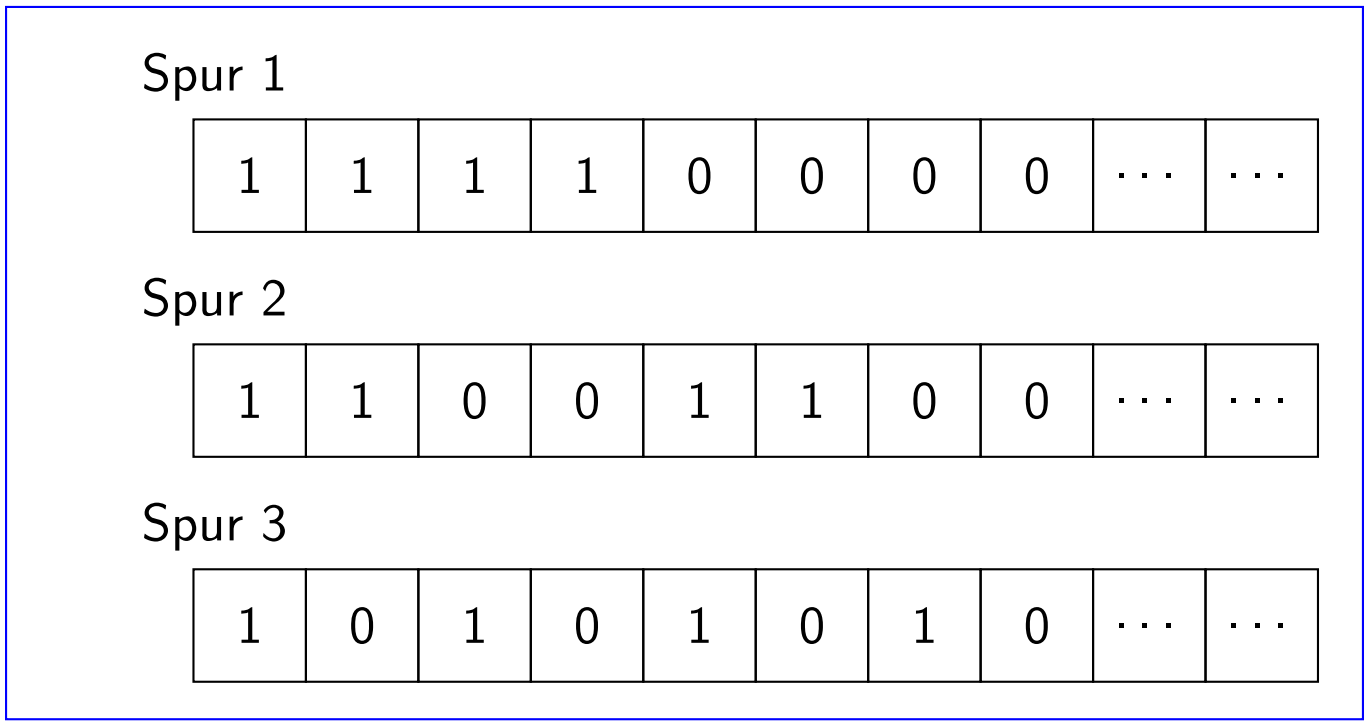
- Zustandszahl explodiert: $Q \times (a \in \Gamma)^k \times \{R, L, N\}^k$

Mehrspur-Turingmaschine zu Ein-Spur-Turingmaschine

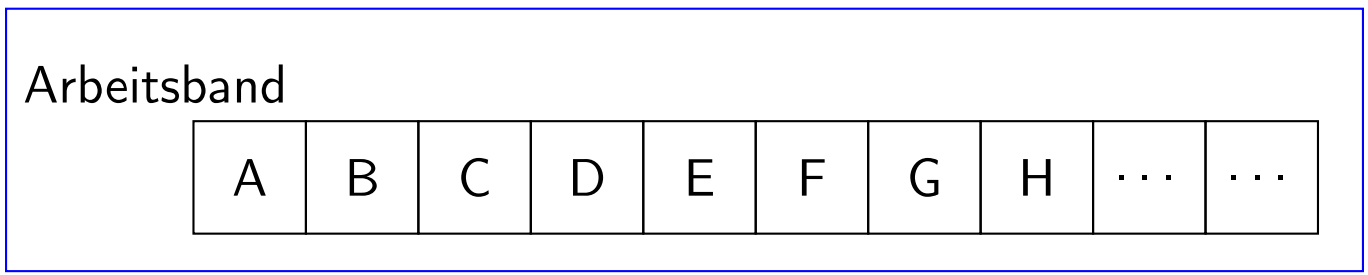
- Ziel: Darstellung mehrerer Spuren auf einer Spur
- Annahme, dass Mehrspur-TM M k Spuren und endliches Bandalphabet Γ hat
- Für aktuelles Wort w von M gilt $w \in \Gamma^k$
- $\Rightarrow$ Jedes mögliche w wird in M' auf ein neues Zeichen abgebildet (resultiert in großem Bandalphabet)

Mehrspur-Turingmaschine zu Ein-Spur-Turingmaschine (II)

Mehrspur-TM



Einspur-TM



Überblick

Umwandlung von Turingmaschine zu Registermaschine

Beweisskizze

Die Übertragbarkeit einer beliebigen Turingmaschine auf eine Registermaschine lässt sich konstruktiv durch Implementierung eines universellen Turingmaschinen-Simulators in einer, für registerbasierte Prozessoren kompilierenden, Sprache beweisen.[a]

[a] Eine solche Programmiersprache könnte Assembler, C, Shell-Skript etc. sein.

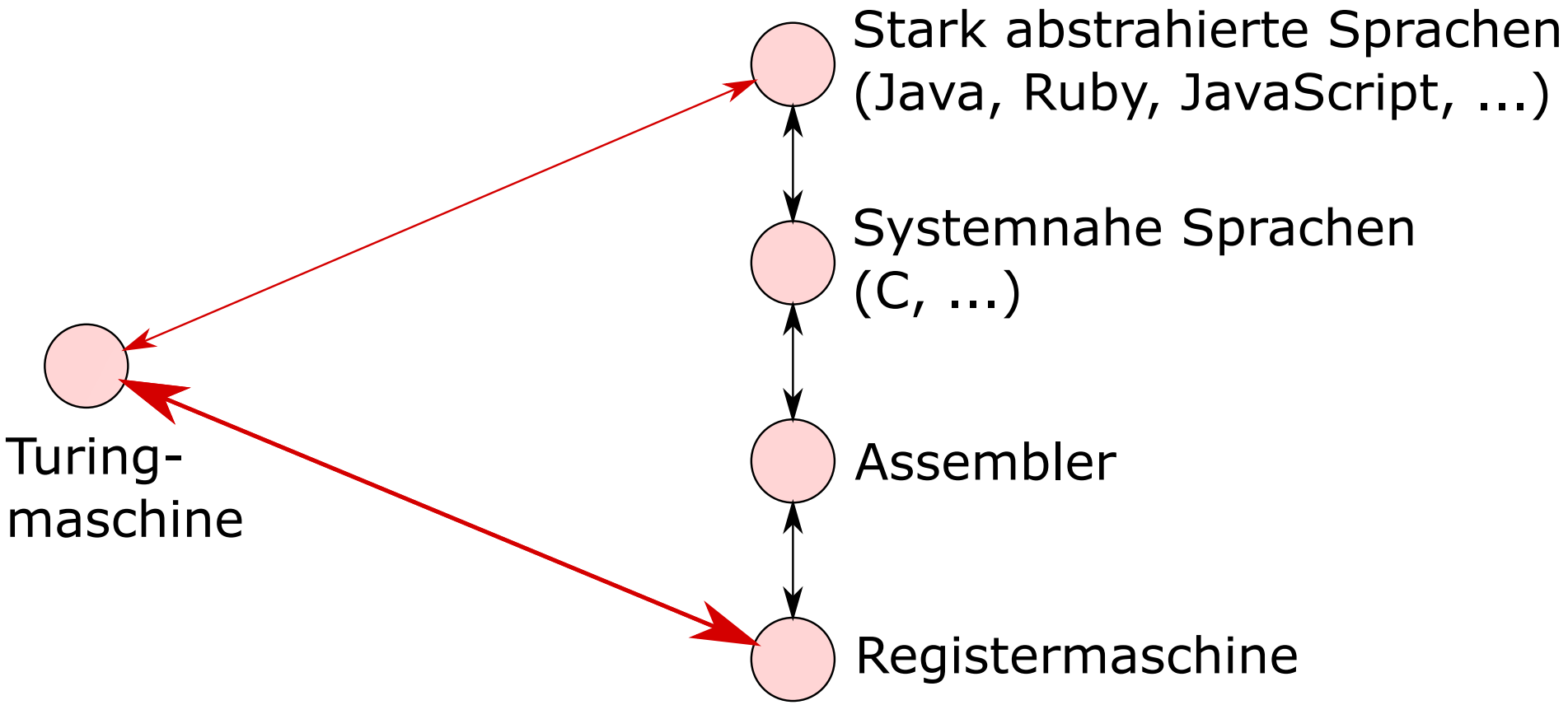
Überblick

Zusammenfassung

Folgendes können wir zusammenfassend feststellen:

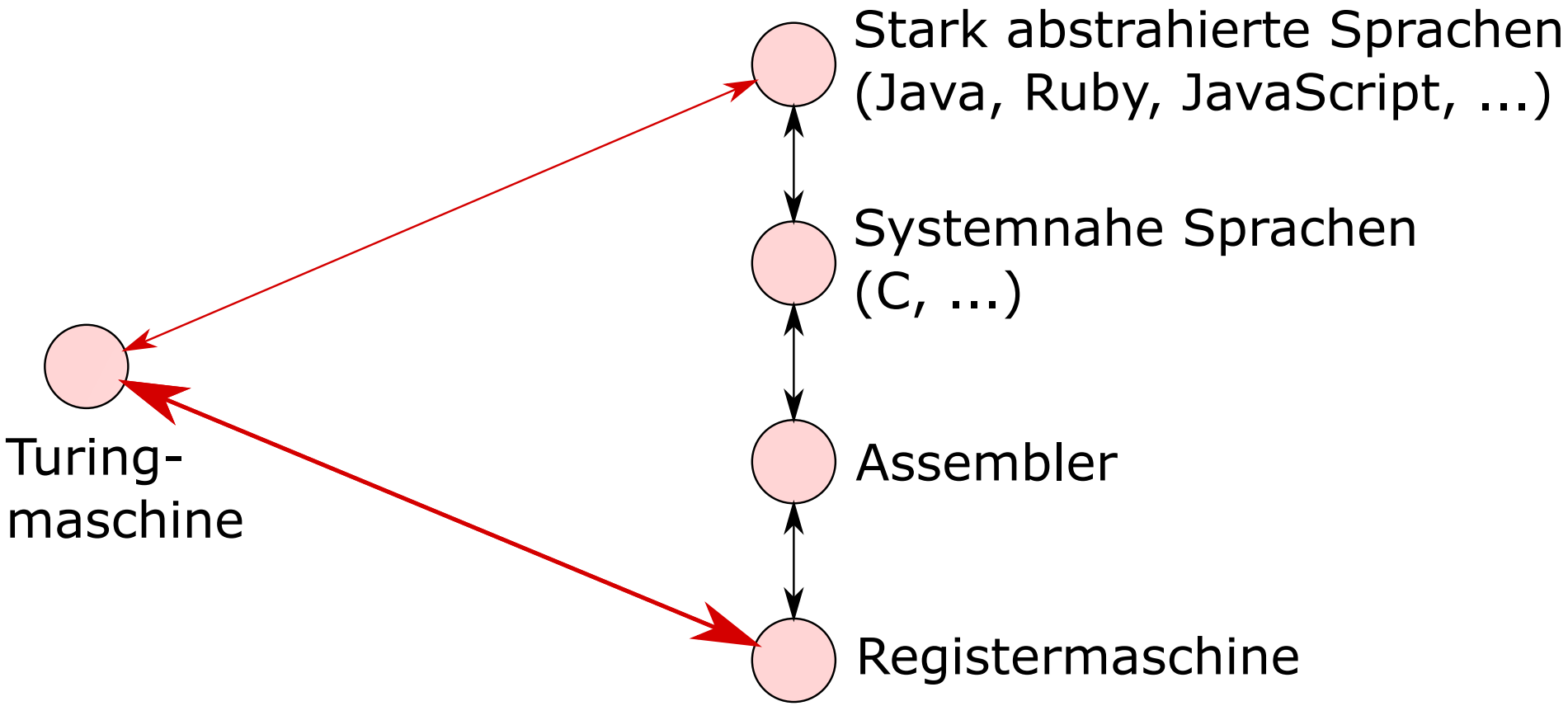
- Die Grundidee gängiger Rechnerarchitekturen kann in Turingmaschinen transformiert bzw. simuliert werden
- **Turingmaschinen und Registermaschinen können ineinander übersetzt werden und daher die gleichen Problemklassen (alle entscheidbaren Probleme) lösen $\Rightarrow$ sprich sind gleich mächtig**

Zusammenfassung (II)



Alle hier gezeigten Konzepte und Sprachen sind gleichermaßen mächtig und ineinander überführbar!

Zusammenfassung (II)



Alle hier gezeigten Konzepte und Sprachen sind gleichermaßen mächtig und ineinander überführbar!

Literatur & Quellen

 Ingo Wegener. *Theoretische Informatik : eine algorithmenorientierte Einführung*. 3., überarb. Aufl. Leitfäden der Informatik. Wiesbaden: Teubner, 2005. ISBN: 3-8351-0033-5.

Vielen Dank für Ihre Aufmerksamkeit!

GitHub: FlorianLoch | fdloch.net | florian.loch@gmail.com

Noch Fragen?