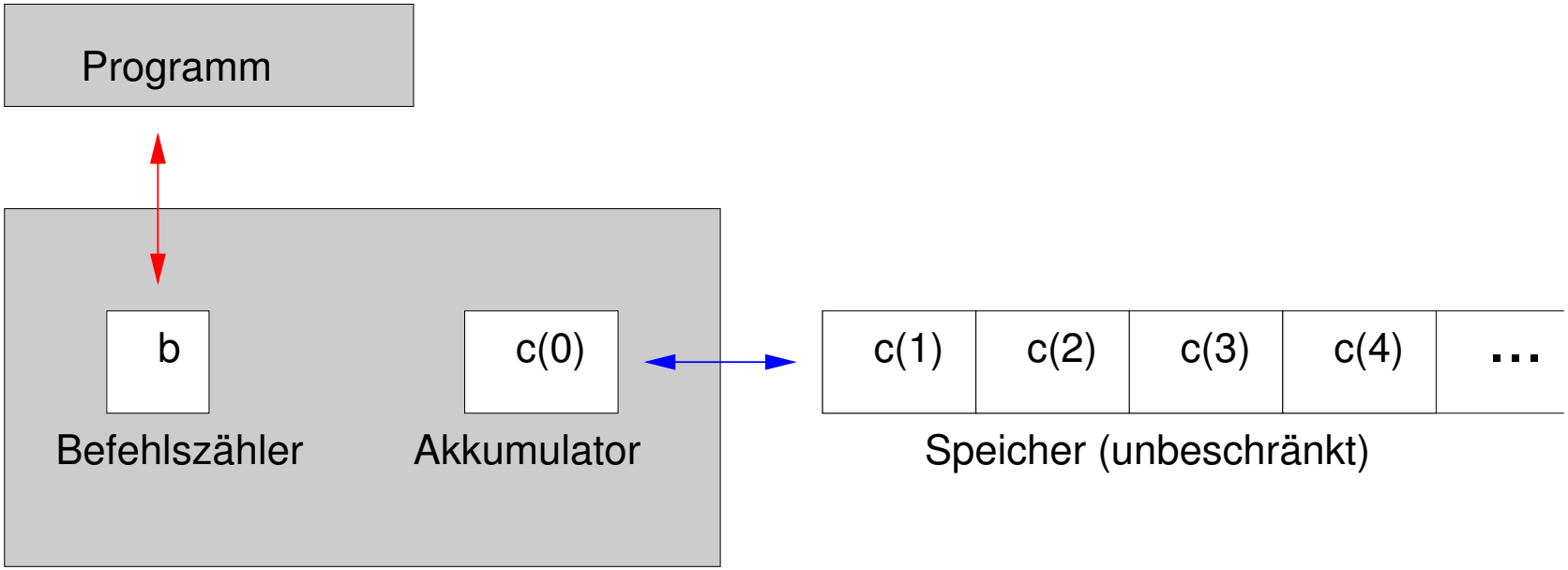
Registermaschine (RAM), Church-Turing-These

Prof. Dr. Berthold Vöcking
Lehrstuhl Informatik 1
Algorithmen und Komplexität
RWTH Aachen

Oktober 2011

Registermaschinen (RAM)



Befehlsatz:
LOAD,STORE,ADD,SUB,DIV
INDLOAD,INDSTORE,INDADD,INDSUB,INDDIV
CLOAD, CADD, CSUB, CDIV
GOTO, IF c(0)?x THEN GOTO j (wobei ? aus {=,<,<=,>,>=}), END

Erläuterung einiger ausgewählter RAM-Befehle

- LOAD i: $c(0) := c(i)$, $b := b + 1$;
- INDLOAD i: $c(0) := c(c(i))$, $b := b + 1$;
- CLOAD i: $c(0) := i$, $b := b + 1$;
- STORE i: $c(i) := c(0)$, $b := b + 1$;
- INDSTORE i: $c(c(i)) := c(0)$, $b := b + 1$;
- ADD i: $c(0) := c(0) + c(i)$, $b := b + 1$;
- CADD i: $c(0) := c(0) + i$, $b := b + 1$;
- INDADD i: $c(0) := c(0) + c(c(i))$, $b := b + 1$;

  $\vdots$
- DIV i: $c(0) := \lfloor c(0)/c(i) \rfloor$, $b := b + 1$;
- CDIV i: $c(0) := \lfloor c(0)/i \rfloor$, $b := b + 1$;
- INDDIV i: $c(0) := \lfloor c(0)/c(c(i)) \rfloor$, $b := b + 1$;
- GOTO j: $b := j$
- IF $c(0) = x$ GOTO j: $b := j$ falls $c(0) = x$, sonst $b := b + 1$;
- END.

Funktionsweise der RAM

- Der Speicher der RAM ist unbeschränkt und besteht aus den Registern $c(0), c(1), c(2), c(3), \ldots$
- Die Inhalte der Register sind natürliche Zahlen, die beliebig groß sein können.
- Die Eingabe sind ebenfalls natürliche Zahlen, die initial in den ersten Registern abgespeichert sind.
- Der Befehlszähler startet mit dem Wert 1. Ausgeführt wird jeweils der Befehl in derjenigen Zeile auf den der Befehlszähler verweist.
- Die Rechnung stoppt sobald der Befehl END erreicht ist.
- Die Ausgabe befindet sich nach dem Stoppen ebenfalls in den ersten Registern.

Beispielprogramm für die RAM: Potenzierung

Eingabe: Zahl $m \in \mathbb{N}$ in Register 1, Zahl $k \in \mathbb{N}$ in Register 2
Ausgabe: Zahl m^k in Register 3 (genannt *erg*)

1:	CLOAD 1	$akku := 1$
2:	STORE 3	$erg := akku$
3:	LOAD 2	$akku := k$
4:	If c(0) = 0 THEN GOTO 11	falls $akku = 0$ dann END
5:	CSUB 1	$akku := akku - 1$
6:	STORE 2	$k := akku$
7:	LOAD 3	$akku := erg$
8:	MULT 1	$akku := akku \cdot m$
9:	STORE 3	$erg := akku$
10:	GOTO 3	zurück zu Zeile 3
11:	END	erg enthält nun m^k

Disclaimer: Wir behaupten nicht, dieses Programm sei effizient.

Bemerkungen zur RAM

Auf einer RAM können wir offensichtlich alle Befehle wie beispielsweise Schleifen und Rekursionen, die wir von höheren Programmiersprachen gewohnt sind, realisieren.

Modelle für die Rechenzeit

- *Uniformes Kostenmaß:* Jeder Schritt zählt eine Zeiteinheit.
- *Logarithmisches Kostenmaß:* Die Laufzeitkosten eines Schrittes sind proportional zur binären Länge der Zahlen in den angesprochenen Registern.

Simulation RAM durch TM

Satz:

Jede im logarithmischen Kostenmaß $t(n)$-zeitbeschränkte RAM kann für ein Polynom q durch eine $O(q(n + t(n)))$-zeitbeschränkte TM simuliert werden.

Simulation RAM durch TM – Vorbemerkung zum Beweis

Im Beweis können wir für die Simulation eine 2-Band-TM statt einer (1-Band) TM verwenden. Warum?

Simulation RAM durch TM – Vorbemerkung zum Beweis

- Seien $\alpha, \beta, \gamma \in \mathbb{N}$ geeignet gewählte Konstanten.
- Wir werden zeigen, die Laufzeit der Simulation der RAM mit Laufzeitschranke $t(n)$ durch eine 2-Band TM ist nach oben beschränkt durch $t'(n) = \alpha(n + t(n))^\beta$.
- Die 2-Band TM mit Laufzeitschranke $t'(n)$ kann nun wiederum mit quadratischem Zeitverlust durch eine (1-Band) TM simuliert werden, also mit einer Laufzeitschranke der Form $t''(n) = \gamma(t'(n))^2$ simulieren.
- Für die Simulation der RAM auf der (1-Band) TM ergibt sich somit eine Laufzeitschranke von

$$t''(n) \;=\; \gamma(t'(n))^2 \;=\; \gamma\left(\alpha(n + t(n))^\beta\right)^2 \;=\; \gamma\alpha^2 \cdot (n+t(n))^{2\beta}.$$

- Diese Laufzeitschranke ist polynomiel in $n + t(n)$, weil sowohl der Term $\gamma\alpha^2$ als auch der Term 2β konstant sind.

Simulation RAM durch TM – Vorbemerkung zum Beweis

Beobachtung:

Die Klasse der Polynome ist gegen Hintereinanderausführung abgeschlossen.

Deshalb können wir eine *konstante Anzahl* von Simulationen, deren Zeitverlust jeweils polynomiell nach oben beschränkt ist, ineinander schachteln und erhalten dadurch wiederum eine Simulation mit polynomiell beschränktem Zeitverlust.

Simulation RAM durch TM – Beweis

Beweis des Satzes:

- Wir verwenden eine 2-Band-TM, die die RAM schrittweise simuliert.
- Das RAM-Programm P bestehe aus p Programmzeilen.
- Für jede Programmzeile schreiben wir ein TM-Unterprogramm. Sei M_i das Unterprogramm für Programmzeile i, $0 \leq i \leq p$.
- Außerdem spezifizieren wir ein Unterprogramm M_0 für die Initialisierung der TM und M_{p+1} für die Aufbereitung der Ausgabe des Ergebnisses.

Simulation RAM durch TM – Beweis

Abspeichern der *RAM-Konfiguration* auf der TM:

- Den Befehlszähler kann die TM im Zustand abspeichern, da die Länge des RAM-Programms konstant ist.
- Die Registerinhalte werden wie folgt auf Band 2 abgespeichert:

$$\#\#0\#bin(c(0))\#\#bin(i_1)\#bin(c(i_1))\#\#\ldots$$

$$\ldots\#\#bin(i_m)\#bin(c(i_m))\#\#\#\ ,$$

wobei $0, i_1, \ldots, i_m$ die Indizes der benutzten Register sind.

Beobachtung:

Der Platzbedarf auf Band 2 ist durch $O(n+t(n))$ beschränkt, weil die RAM für jedes neue Bit, das sie erzeugt, mindestens eine Zeiteinheit benötigt.

Simulation RAM durch TM – Beweis

Rechenschritt für Rechenschritt simuliert die TM nun die Konfigurationsveränderungen der RAM.

Dazu ruft die TM das im Programmzähler b angegebene Unterprogramm M_b auf.

Das Unterprogramm M_b

- kopiert den Inhalt der in Programmzeile b angesprochenen Register auf Band 1,
- führt die notwendigen Operationen auf diesen Registerinhalten durch,
- kopiert dann das Ergebnis in das in Zeile b angegebene Register auf Band 2 zurück, und
- aktualisiert zuletzt den Programmzähler b.

Simulation RAM durch TM – Beweis

Laufzeitanalyse:

Die Initialisierung erfordert Zeit $O(n)$.

Alle anderen Unterprogramme haben eine Laufzeit, die polynomiell in der Länge der Bandinschrift auf Band 2 beschränkt ist, also eine polynomielle Laufzeit in $n + t(n)$.

Somit ist auch die Gesamtlaufzeit der Simulation polynomiell in $n + t(n)$ beschränkt. □

Simulation TM durch RAM

Satz:

Jede $t(n)$-zeitbeschränkte TM kann durch eine RAM simuliert werden, die zeitbeschränkt ist durch

- $O(t(n) + n)$ im uniformen Kostenmaß und
- $O((t(n) + n) \cdot \log(t(n) + n))$ im logarithmischen Kostenmaß.

Simulation TM durch RAM – Beweis

Beweis des Satzes:

- O.B.d.A. nehmen wir an, es handelt sich um eine TM mit einseitig beschränktem Band, deren Zellen mit $0, 1, 2, 3, \ldots$ durchnummeriert sind. (vgl. Übung)
- Die Zustände und Zeichen werden ebenfalls durchnummeriert und mit ihren Nummern identifiziert, so dass sie in den Registern abgespeichert werden können.
- Register 1 speichert den Index der Kopfposition.
- Register 2 speichert den aktuellen Zustand.
- Die Register $3, 4, 5, 6, \ldots$ speichern die Inhalte der jemals besuchten Bandpositionen $0, 1, 2, 3, \ldots$

Simulation TM durch RAM – Beweis

Die TM wird nun Schritt für Schritt durch die RAM simuliert.

Auswahl des richtigen TM-Übergangs

Die RAM verwendet eine zweistufige *if*-Abfrage:

- Auf einer ersten Stufe von $|Q|$ vielen *if*-Abfragen wird der aktuelle Zustand selektiert.
- Für jeden möglichen Zustand, gibt es dann eine zweite Stufe von $|\Gamma|$ vielen *if*-Abfragen, die das gelesene Zeichen selektieren.

Durchführung des TM-Übergangs

Je nach Ausgang der *if*-Abfragen aktualisiert die RAM

- den TM-Zustand in Register 2,
- die TM-Bandinschrift in Register $c(1)$ und
- die TM-Bandposition in Register 1.

Simulation TM durch RAM – Beweis

Laufzeitanalyse im uniformen Kostenmodell:

- Die Initialisierung kann in Zeit $O(n)$ durchgeführt werden.
- Die Simulation jedes einzelnen TM-Schrittes hat konstante Laufzeit.
- Insgesamt ist die Simulationszeit somit $O(n + t(n))$.

Simulation TM durch RAM – Beweis

Laufzeitanalyse im logarithmischen Kostenmodell:

- Die in den Registern gespeicherten Zahlen repräsentieren Zustände, Zeichen und Bandpositionen.
- Zustände und Zeichen haben eine konstante Kodierungslänge.
- Die Bandpositionen, die während der Simulation angesprochen werden, sind durch $\max\{n, t(n)\} \leq n + t(n)$ beschränkt. Die Kodierungslänge dieser Positionen ist also $O(\log(t(n) + n))$.
- Damit kann die Simulation jedes einzelnen TM-Schrittes in Zeit $O(\log(t(n) + n))$ durchgeführt werden.
- Insgesamt ergibt sich somit eine Simulationszeit von $O((t(n) + n)\log(t(n) + n))$.

□

Zusammenfassung

- Die Mehrband-TM kann mit quadratischem Zeitverlust durch eine (1-Band) TM simuliert werden.
- TM und RAM (mit logarithmischen Laufzeitkosten) können sich gegenseitig mit polynomiellem Zeitverlust simulieren.
- Wenn es uns also „nur“ um Fragen der Berechenbarkeit von Problemen (oder um ihre Lösbarkeit in polynomieller Zeit) geht, können wir wahlweise auf die TM, die Mehrband-TM oder die RAM zurückgreifen.

Die Church-Turing-These

Kein jemals bisher vorgeschlagenes „vernünftiges“ Rechnermodell hat eine größere Mächtigkeit als die TM.

Diese Einsicht hat Church zur Formulierung der folgenden These veranlasst.

Church-Turing-These

Die Klasse der TM-berechenbaren Funktionen stimmt mit der Klasse der “intuitiv berechenbaren” Funktionen überein.

Wir werden deshalb nicht mehr von *TM-berechenbaren* Funktionen sprechen, sondern allgemein von *berechenbaren* Funktionen.

Gleichbedeutend verwenden wir den Begriff *rekursive* Funktion bzw. *rekursive* oder auch *entscheidbare* Sprache.

Zum Inhalt der Vorlesung

Jetzt sind wir bereit, die zentrale Fragestellung des ersten Teils dieser Vorlesung formaler zu fassen.

In der *Berechenbarkeitstheorie* ...

... wird untersucht welche Probleme rekursiv sind, d.h. welch Probleme durch einen Algorithmus – ohne jegliche Einschränkungen bzgl. Rechenzeit und Speicherplatz – gelöst werden können.