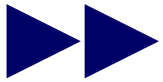
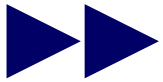
Theoretische Informatik
Kap 2: Berechnungstheorie



Gliederung der Vorlesung

0. Grundbegriffe
1. Formale Sprachen/Automatentheorie
    - 1.1. Grammatiken
    - 1.2. Reguläre Sprachen
    - 1.3. Kontextfreie Sprachen
2. Berechnungstheorie
    - **2.1. Berechenbarkeitsmodelle**
    - 2.2. Die Churchsche These
    - 2.3. Unentscheidbarkeit
3. Komplexitätstheorie
    - 3.1. Nicht-deterministische Turing Maschinen
    - 3.1 Komplexitätsmaße
    - 3.2. Das P=NP? Problem

Theoretische Informatik
Kap 2: Berechnungstheorie



Berechenbarkeitsmodelle

konzeptioneller Aspekt

Präzisierung des Algorithmen-Begriffs und dadurch die Festlegung einer Klasse von berechenbaren Funktionen

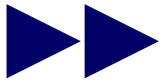
oberstes Ziel

nachweisen, daß bestimmte Funktionen nicht berechenbar sind

... natürlich nicht irgendwelche Funktionen, sondern Funktionen, die zu wichtigen, interessanten, praktisch relevanten ... Problemen gehören

Hinweis: ein Berechenbarkeitsmodell entspricht einer Präzisierung des Algorithmen-Begriffs

Theoretische Informatik
Kap 2: Berechnungstheorie



Berechenbarkeitsmodelle

Qualitätskriterien

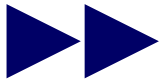
(1) jede im präzisierten Sinn berechenbare Funktion f sollte auch „intuitiv“ berechenbar sein

... man sollte sich davon überzeugen können, daß man einen Algorithmus A angeben kann, der f berechnet

(2) jede „intuitiv“ berechenbare Funktion f sollte auch im präzisierten Sinn berechenbar sein sein

... das ist der ungleich schwerere Teil, den man nicht so richtig in den Griff bekommt

Theoretische Informatik
Kap 2: Berechnungstheorie



Berechenbarkeitsmodelle

Fahrplan

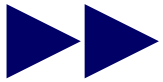
- wir werden uns zwei Berechenbarkeitsmodelle genauer ansehen

  ... ein besseres Gefühl für mögliche Ansätze bekommen

- nachweisen, daß die beiden Berechenbarkeitsmodelle äquivalent sind, d.h. für jede Funktion f gilt: f ist in dem einem Modell berechenbar gdw. f ist in dem anderen Modell berechenbar

  ... Ideen ableiten, wie man mit Qualitätskriterium (2) umgehen kann

Theoretische Informatik
Kap 2: Berechnungstheorie



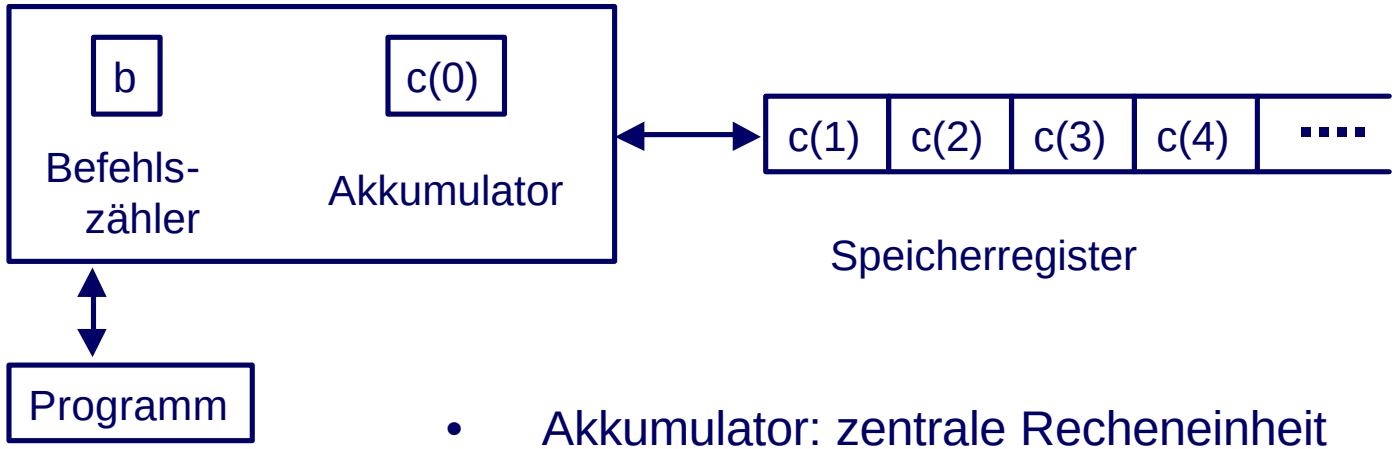
Random-Access-Maschinen

ein Berechenbarkeitsmodell, das de facto eine für theoretische Zwecke vereinfachte Version eines von-Neumann-Rechners darstellt

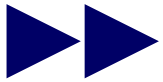
Fokus: Qualitätskriterium (1)

Theoretische Informatik
Kap 2: Berechnungstheorie

Random-Access-Maschinen



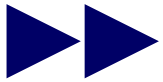
- Akkumulator: zentrale Recheneinheit
- beliebige viele Register
- der Befehlszähler b, der Akkumulator c(0) und die Register c(i) können beliebig große natürliche Zahlen (/* in Binärdarstellung */) speichern
- ein RAM-Programm besteht aus einer fortlaufend numerierten Folge von Befehlen



Random-Access-Maschinen

- zu Beginn ist b = 1 und in den Registern c(1), ..., c(k) (/* für die Eingaben */) stehen die Eingaben (/* alle anderen Register haben den Wert 0 */)
- das Ergebnis steht nach Beendigung der Berechnung (/* Ausführen des Befehls End */) im Register c(k+1)
- in jedem Schritt wird der Befehl mit der Nummer b abgearbeitet
- unterschiedliche Typen von Befehlen
  - Arithmetik
  - Sprünge
  - Speicherzugriffe

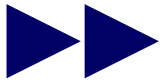
Theoretische Informatik
Kap 2: Berechnungstheorie



Random-Access-Maschinen

Befehl	Bedeutung	Auswirkung auf b
Add i	c(0) := c(0) + c(i)	b := b+1
Sub i	c(0) := c(0) - c(i), falls c(0) ≥ c(i) c(0) := 0, sonst	b := b+1
cAdd z, cSub z	c(0) := c(0) + z, c(0) := c(0) - z	b := b+1
Div 2	c(0) := ⌊½ c(0)⌋	b := b+1
Goto j		b := j
If c(0) = 0 Goto j		b := j, falls c(0) = 0 b := b+1, sonst
If c(0) > 0 Goto j		b := j, falls c(0) > 0 b := b+1, sonst
End	Programm hält	
Load i	c(0) := c(i)	b := b+1
Store i	c(i) := c(0)	b := b+1

Theoretische Informatik
Kap 2: Berechnungstheorie



Random-Access-Maschinen

Beispiel: f(x,y) = min { x, y }

Eingabe: c(1), c(2)
Ausgabe: c(3)

1. Load 1
2. Sub 2
3. If c(0) = 0 Goto 7
4. Load 2
5. Store 3
6. Goto 9
7. Load 1
8. Store 3
9. End

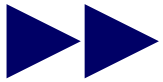
Random-Access-Maschinen

Beispiel: f(x,y) = x * y

Eingabe: c(1), c(2)
Ausgabe: c(3)

1. Load 2
2. If c(0) = 0 Goto 9
3. cSub 1
4. Store 2
5. Load 1
6. Add 3
7. Store 3
8. Goto 1
9. End

Theoretische Informatik
Kap 2: Berechnungstheorie



Random-Access-Maschinen

es sei R eine RAM

es seien c(1), ... c(k) die Eingaberegister und c(k+1) das Ausgaberegister

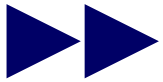
es seien x_1, ..., x_k natürliche Zahlen

Für die von R berechnete Funktion f_R gilt $f_R(x_1,...,x_k) = y$, falls

R gestartet mit x_1, ..., x_k in den Registern c(1), ..., c(k) nach Abarbeitung von endlichen vielen Rechenschritten den Befehl „End“ ausführt und im Register c(k+1) die Zahl y steht

$f_R(x_1,x_2,...,x_k)$ ist undefiniert, falls

- R gestartet mit x_1, ..., x_k in den Registern c(1), ..., c(k) nie den Befehl „End“ ausführt



Random-Access-Maschinen

es sei f: $N^k \rightarrow N$ eine Funktion

f ist RAM-berechenbar, falls es eine RAM R gibt, so daß für alle $x_1 \in N, ..., x_k \in N$ gilt: $f_R(x_1,...,x_k) = f(x_1,...,x_k)$.

Theoretische Informatik
Kap 2: Berechnungstheorie

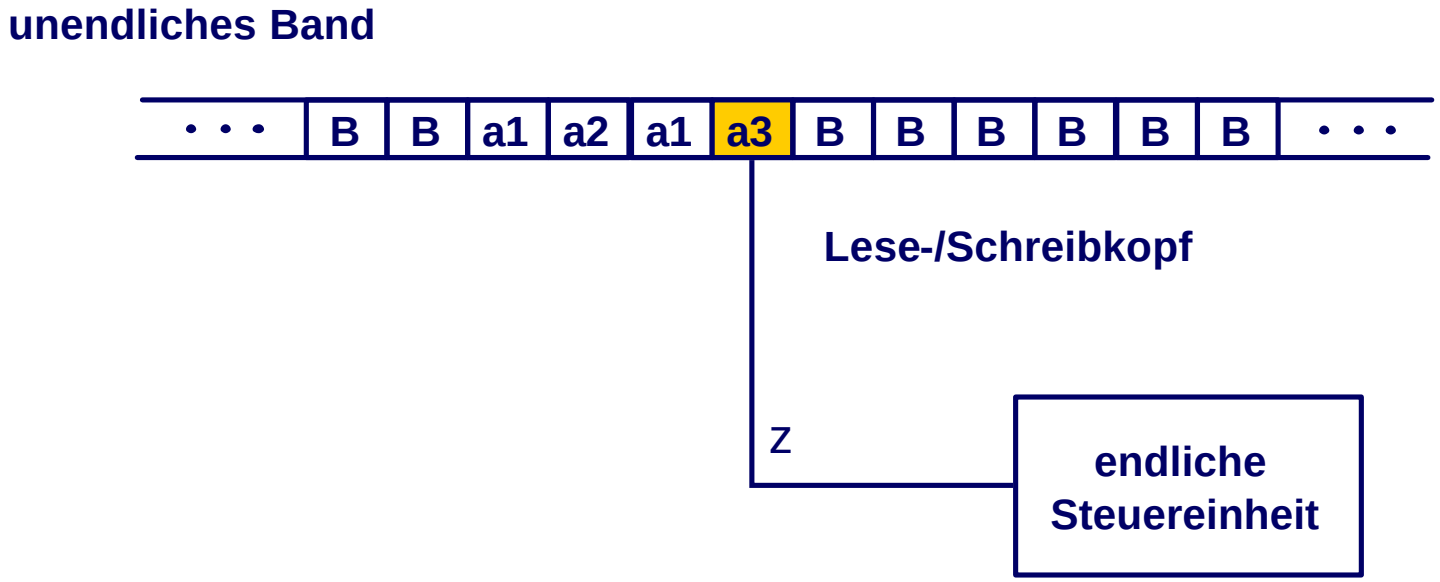
Turing-Maschinen

Turing A., On computable numbers with an application to the Entscheidungsproblem, Proc. London Math. Soc., Vol. 42 (1936), 230-265.

- historisch die erste Formalisierung des Begriffs „Algorithmus“ (/* formal: Berechenbarkeitsmodell */)
- für theoretische Untersuchungen zweckmäßig, weil mathematisch einfach „handhabbar“

Theoretische Informatik
Kap 2: Berechnungstheorie

Turing-Maschinen



Turing-Maschinen

- Speicher ist linear angeordnet (unendliches Band); in Felder unterteilt; jedes Feld enthält ein Zeichen
- endliche Steuereinheit; mit dem Lese-/Schreibkopf verbunden
  - bekommt Informationen über den Inhalt einer Speicherzelle
  - befindet sich in einem von endlich vielen Zuständen
  - legt in Abhängigkeit vom aktuellen Zustand und vom Inhalt der aktuell gesehenen Speicherzelle fest, was im nächsten Arbeitsschritt passiert
- ein Arbeitsschritt besteht aus:
  - Drucken eines Zeichens in die aktuell gesehene Speicherzelle
  - Bewegung des Lese-/Schreibkopfs nach links, rechts bzw. gar nicht
  - Festlegung des nächsten Zustands

Turing-Maschinen

Bestimmungsstücke einer Turing-Maschine

- endliche Menge Z von Zuständen
- Anfangszustand z_0, Endzustand z_e
- endliches Bandalphabet Γ mit Blanksymbol B
- endliches Eingabealphabet $\Sigma \subseteq \Gamma \setminus \{ B \}$
- Zustandsüberführungsfunktion

$$\delta: (Z \setminus \{ z_e \}) \times \Gamma \rightarrow Z \times \Gamma \times \{ L,R,N \}$$

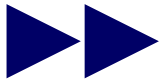
Turing-Maschinen

Beispiel: Turing-Maschine, die bei Eingabe der Binärdarstellung von x die Binärdarstellung von 2x ausgibt

	0	1	B
z_0	z_0,0,R	z_0,1,R	z_1,0,L
z_1	z_1,0,L	z_1,1,L	z_e,B,R

$(z_0,0) \rightarrow (z_0,0,R)$
$(z_0,1) \rightarrow (z_0,1,R)$
$(z_0,B) \rightarrow (z_1,0,L)$

$(z_1,0) \rightarrow (z_1,0,L)$
$(z_1,1) \rightarrow (z_1,1,L)$
$(z_1,B) \rightarrow (z_e,B,R)$



Turing-Maschinen

Beispiel: Turing-Maschine, die bei Eingabe der Binärdarstellung von x die Binärdarstellung von x+1 ausgibt

	0	1	B
z_0	z_0,0,R	z_0,1,R	z_1,B,L
z_1	z_2,1,L	z_3,0,L	
z_2	z_2,0,L	z_2,1,L	z_e,B,R
z_3	z_2,1,L	z_3,0,L	z_e,1,N

Turing-Maschinen

Beispiel: Turing-Maschine, die bei Eingabe der Binärdarstellung von x eine 1 berechnet, falls x gerade ist, und eine 0, sonst

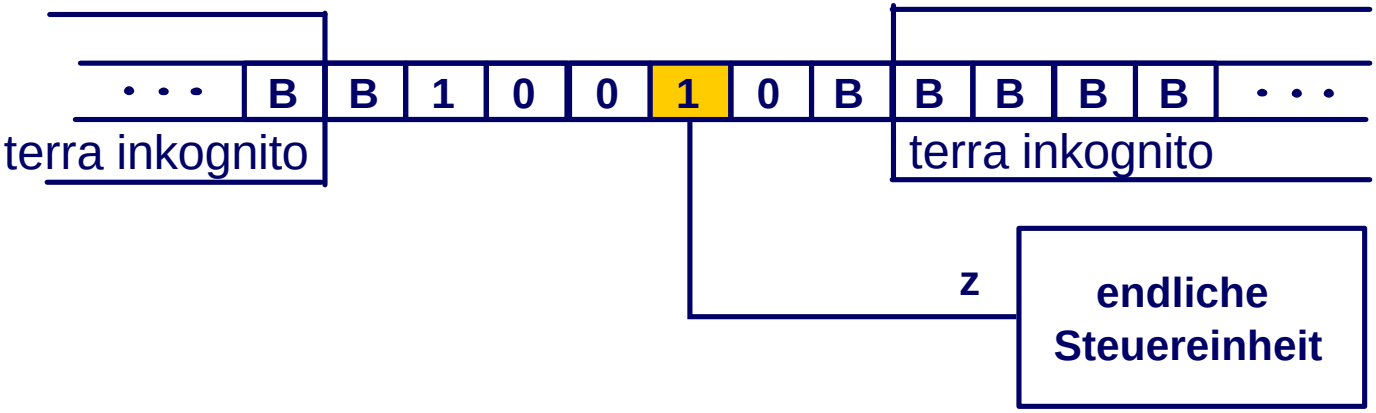
	0	1	B
z_0	z_0,B,R	z_1,B,R	z_2,1,L
z_1	z_0,B,R	z_1,B,R	z_2,0,L
z_2			z_e,B,R

Theoretische Informatik
Kap 2: Berechnungstheorie

Turing-Maschinen

... benutzen Turing-Maschinen zur Berechnung k-stelliger Funktionen über N

→ Hilfsbegriff: Konfiguration



Notation: B100z10B

- Anfangskonfiguration, falls z Anfangszustand ist (/* $z = z_0$ */)
- Endkonfiguration, falls z Endzustand ist (/* $z = z_e$ */)

Turing-Maschinen

es sei M eine Turing-Maschine mit Anfangszustand z_0
und Endzustand z_e

Für die von M berechnete k-stellige Funktion f_M gilt $f_M(x_1,...,x_k) = y$, falls

M die Anfangskonfiguration $z_0 bin(x_1)B...Bbin(x_k)$ in eine Endkonfiguration $wz_e bin(y)Bw'$ transformiert

$f_M(x_1,x_2,...,x_k)$ ist undefiniert, falls

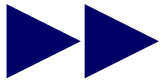
- M die Anfangskonfiguration $z_0 bin(x_1)B...Bbin(x_k)$ nicht in eine Endkonfiguration überführt (/* d.h. M erreicht nie z_e */)

Turing-Maschinen

	0	1	B
z_0	z_0,B,R	z_1,B,R	z_2,1,R
z_1	z_0,B,R	z_1,B,R	z_1,B,L
z_2			z_e,B,R

obige TM M berechnet eine partielle einstellige Funktion f_M

- $f_M(x_1) = 1$, falls x_1 gerade
- $f_M(x_1)$ ist undefiniert, sonst



Turing-Maschinen

es sei f: $N^k \rightarrow N$ eine Funktion

f ist Turing-berechenbar, falls es eine Turing-Maschine gibt, so daß für alle $x_1 \in N, ..., x_k \in N$ gilt: $f_M(x_1,...,x_k) = f(x_1,...,x_k)$.