

# Berechenbarkeit und Komplexität

Dozent: Mathias Weller (Skript adaptiert von Rolf Niedermeier)

Betreuer: Leon Kellerhals, Vincent Froese und Philipp Zschoche

Sekretariat: Christlinde Thielcke

Viele Fleißige Tutorinnen und Tutoren

Fachmentorin: Niloofar Nazemi

TU Berlin

Fakultät IV

Fachgebiet Algorithmik und Komplexitätstheorie

<https://www.akt.tu-berlin.de>

# Informatikstudium an der TU Berlin

1. Semester 27 LP	Rechner- organisation (6 LP)	Einführung in die Programmierung (6 LP)	Informatik Propädeutikum (3 LP)	Analysis I und Lineare Algebra für Ingenieurwissenschaften (12 LP)	
2. Semester 30 LP	System- programmierung (6 LP)	Algorithmen und Datenstrukturen (6 LP)	Informationssys- teme und Daten- analyse (6 LP)	Formale Sprachen und Automaten (6 LP)	Diskrete Strukturen (6 LP)
3. Semester 30 LP	Rechnernetze und Verteilte Systeme (6 LP)	Softwaretechnik und Program- mierparadigmen (6 LP)	Wissenschaft- liches Rechnen (6 LP)	Berechenbarkeit und Komplexität (6 LP)	Logik (6 LP)
4.-6. Semester 93 LP	Wahlpflicht Technische Informatik (6 LP)	Wahlpflicht Programmier- praktikum (6-9 LP)	Wahlpflicht Theoretische Informatik (6 LP)		Stochastik für Informatik (9 LP)
	Wahlpflichtbereich Katalog Informatik (27-33 LP)				Informatik und Gesellschaft (6 LP)
	Wahlbereich (15-18 LP)			Bachelorarbeit (12 LP)	

LP = Leistungspunkte nach dem ECTS-System (1 LP entspricht etwa 30 Zeitstunden)

- Technische Grundlagen der Informatik
  Methodisch-praktische Grundlagen der Informatik
- Theoretische Informatik
  Grundlagen des wiss. Arbeitens/Informatik in gesellschaftlicher Relevanz
- Grundlagen der Mathematik
  Wahlpflichtbereich
  Wahlbereich
  Bachelorarbeit

# Organisation

## **Vorlesungsbetrieb:**

- ▶ Vorlesung: Screencasts und PDF-Folien verfügbar über ISIS
- ▶ “Freie Großübung” + “Modulkonferenz”

## **Tutorien:**

- ▶ Tutorien: siehe ISIS und MOSES
- ▶ Tutor\*innensprechstunde: TBA

## **Prüfungen:** Portfolioprüfung

- ▶ Multiple-Choice-Test: 25 PP (ca. Mitte Dezember)
- ▶ Hausaufgabe in Dreiergruppen 25 PP (im Januar)
- ▶ Schriftlicher Test: 50 PP (Termin wird über ISIS bekanntgegeben)

# Ergänzendes Material

## Literatur:

- ▶ Uwe Schöning. *Theoretische Informatik–kurz gefasst*. Spektrum Akademischer Verlag 2008 (5. Auflage).
- ▶ Elaine Rich. *Automata, Computability, and Complexity*. Pearson 2008.
- ▶ Cristopher Moore, Stephan Mertens. *The Nature of Computation*. Oxford University Press 2011.

## Weiteres Material:

YouTube-Kanal NLogSpace ([https://www.youtube.com/channel/UCMWYg3eBFp5bbqj111Uku\\_w](https://www.youtube.com/channel/UCMWYg3eBFp5bbqj111Uku_w))

# Gliederung

1. Einführung
2. Berechenbarkeitsbegriff
3. LOOP-, WHILE-, und GOTO-Berechenbarkeit
4. Primitive und partielle Rekursion
5. Die Ackermannfunktion
6. (Un-)Entscheidbarkeit, Halteproblem und Reduzierbarkeit
7. Das Postsche Korrespondenzproblem
8. Komplexität – Einführung
9. NP-Vollständigkeit
10. PSPACE

# Das „hello, world“-Problem

**Ziel:** Entwicklung von Programm  $E$  mit folgender Spezifikation:

**Input:** Programm  $P$

**Output:** „Top“, wenn  $P$  den string „hello, world“ ausgibt, „Flop“, sonst

**Bemerkung:**  $E$  hat „Typ höherer Ordnung“ (d.h. Eingabe ist (Text eines) Programms  $P$ ).

## Beispiel für Eingabe $P$

```
main(){  
    printf("hello, world");  
}
```

↪ Existiert ein Programm  $E$  für diese spezielle Eingabe  $P$ ?

↪ Existiert ein Programm  $E$  auch für **beliebige** Programme  $P$ ?

# Wiederholung: Endliche Automaten

## Definition (Endlicher Automat)

- ▶ Ein **(deterministischer) endlicher Automat** (kurz **DFA**) ist ein Quintupel  $M = (Z, \Sigma, \delta, z_0, E)$  mit
  - ▶  $Z$  ist eine nichtleere, endliche Menge von **Zuständen**,
  - ▶  $\Sigma$  ist ein nichtleeres, endliches Alphabet von **Eingabezeichen** mit  $Z \cap \Sigma = \emptyset$ ,
  - ▶  $\delta: Z \times \Sigma \rightarrow Z$  ist die **partielle Überföhrungsfunktion**,
  - ▶  $z_0 \in Z$  ist der **Startzustand** und
  - ▶  $E \subseteq Z$  ist die Menge der **Endzustände**.
- ▶ Zu  $M$  definieren wir die partielle Funktion  $\hat{\delta}: Z \times \Sigma^* \rightarrow Z$  induktiv für alle  $z \in Z$ :

$$\begin{aligned} \hat{\delta}(z, \epsilon) &:= z \\ \forall_{x \in \Sigma^*} \quad \hat{\delta}(z, ax) &:= \hat{\delta}(\delta(z, a), x) && \text{falls } \delta(z, a) \neq \perp \end{aligned}$$

- ▶ Ein DFA  $M = (Z, \Sigma, \delta, z_0, E)$  **akzeptiert** ein Wort  $w \in \Sigma^*$  falls  $\hat{\delta}(z_0, w) \in E$ 
  - ▶ Die von  $M$  **akzeptierte Sprache** ist  $T(M) := \{x \in \Sigma^* \mid \hat{\delta}(z_0, x) \in E\}$ .

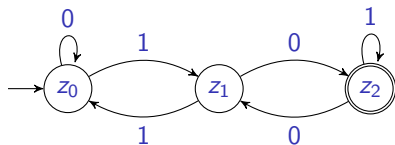
# Wiederholung: Endliche Automaten II

## Beispielautomat

$M = (\{z_0, z_1, z_2\}, \{0, 1\}, \delta, z_0, \{z_2\})$  mit

$\delta$	$z_0$	$z_1$	$z_2$
0	$z_0$	$z_2$	$z_1$
1	$z_1$	$z_0$	$z_2$

## Zustandsgraph



$z_i \leadsto$  das bisher gelesene Wort ist die Binärkodierung einer Zahl  $n$  mit Rest  $i$  modulo 3.

$$T(M) = \{w \in \{0, 1\}^* \mid w \text{ ist Binärdarstellung einer Zahl } n \text{ mit } n \bmod 3 = 2\}.$$

**Frage:** Sind die Binärdarstellungen der Zahlen  $n$  mit  $n \bmod 4 = 1$  von einem DFA erkennbar?



# Grenzen endlicher Automaten

Gibt es jeweils einen endlichen Automaten zur Erkennung folgender Sprachen?

- ▶  $\{w \in \{0,1\}^* \mid w \text{ ist Binärdarstellung einer geraden Zahl}\}$  ✓

Letztes Zeichen muss eine 0 sein.

- ▶  $\{a^n b^n \mid 0 \leq n \leq 1000\}$  ✓

Sprache enthält nur 1001 Wörter. Jede endliche Sprache ist regulär.

- ▶  $\{a^n b^n \mid n \geq 0\}$  ✗

Mit endlich vielen Zuständen können wir uns nicht „merken“ wieviele  $a$ 's wir schon gelesen haben.  
Erkennung mit Kellerautomaten möglich.

- ▶  $\{(abc)^n \mid n \geq 0\}$  ✓

ählich zur " $n \bmod 3 = 2$ " Sprache.

- ▶  $\{a^n b^m c^k \mid n, m, k \geq 0\}$  ✓

Müssen uns nur „merken“ welche Buchstaben noch folgen dürfen.

- ▶  $\{a^n b^n c^n \mid n \geq 0\}$  ✗

Siehe  $a^n b^n$ . Erkennung mittels Turing-Maschinen.

- ▶  $\{a^i b^j c^i d^j \mid i, j \geq 0\}$  ✗

Siehe oben. Erkennung mittels Turing-Maschinen.

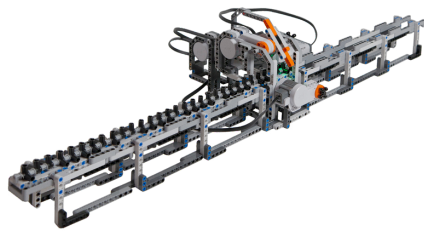
# Die Turing Maschine



Alan Mathison Turing, 1912-1954.



Inspiration: „Menschliche Computer“ 1890.



LEGO Turing Maschine

Informell

endliche Kontrolle + unendliches Band

Quellen:

<http://therunnereclectic.files.wordpress.com/2014/11/alan-turing-running.jpg>

[http://en.wikipedia.org/wiki/Harvard\\_Computers](http://en.wikipedia.org/wiki/Harvard_Computers)

[http://cs.cmu.edu/~soonhok/images/20120718\\_LegoTM/legotm.png](http://cs.cmu.edu/~soonhok/images/20120718_LegoTM/legotm.png)

# Definition Turing-Maschinen

## Definition ((deterministische) Turing-Machine)

Eine **Turing-Maschine (kurz DTM)** ist ein Septupel  $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$  mit

- ▶  $Z$ , einer nicht-leeren, endlichen Menge von **Zuständen**,
- ▶  $\Sigma$ , dem **Eingabealphabet**,
- ▶  $\Gamma \supseteq \Sigma$ , dem **Arbeits- oder Bandalphabet** mit  $\Gamma \cap Z = \emptyset$ ,
- ▶  $\delta: (Z \setminus E) \times \Gamma \rightarrow Z \times \Gamma \times \{L, R, N\}$ , die **partielle Überföhrungsfunktion**
- ▶  $z_0 \in Z$ , dem **Startzustand**,
- ▶  $\square \in \Gamma \setminus \Sigma$ , dem **Blanksymbol** und
- ▶  $E \subseteq Z$ , der Menge der **Endzustände**.

**Interpretation:** Wenn  $M$  im Zustand  $z$  das Zeichen  $a$  liest und  $\delta(z, a) = (z', a', p)$ , so

- ▶ geht  $M$  in Zustand  $z'$  über,
- ▶ **überschreibt** das  $a$  durch ein  $a'$
- ▶ bewegt den Lese/Schreibkopf gemäß  $p$  (nach **Links**, **Rechts**, oder gar **Nicht**)

# Konfigurationen

## Definition (Konfiguration, Folgekonfiguration)

Sei  $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$  eine TM. Eine **Konfiguration** von  $M$  ist ein Wort  $azb$  mit  $a, b \in \Gamma^*$  und  $z \in Z$ .  
(überflüssige  $\square$ -Symbole an den Rändern der Konfiguration weglassen)

Die **Startkonfiguration** zu einem Wort  $x \in \Sigma^*$  ist  $z_0x$ .

Sei  $k = a_1 \dots a_m \mathbf{z} b_1 \dots b_n$  eine Konfiguration (falls  $n = 0$ , dann  $b_1 := \square$ ). Dann

$$k \vdash_M^0 k$$

$$k \vdash_M^1 a_1 \dots a_m \mathbf{z}' c b_2 \dots b_n \quad \text{falls } \delta(z, b_1) = (z', c, N)$$

$$k \vdash_M^1 a_1 \dots a_m c \mathbf{z}' b_2 \dots b_n \quad \text{falls } \delta(z, b_1) = (z', c, R)$$

$$k \vdash_M^1 a_1 \dots a_{m-1} \mathbf{z}' a_m c b_2 \dots b_n \quad \text{falls } \delta(z, b_1) = (z', c, L) \text{ und } m > 0$$

$$k \vdash_M^1 \mathbf{z}' \square c b_2 \dots b_n \quad \text{falls } \delta(z, b_1) = (z', c, L) \text{ und } m = 0.$$

$k$  ist **haltend** (d.h.  $k$  hat keine Folgekonfiguration) falls  $\delta(z, b_1) = \perp$

$k$  ist **akzeptierend** falls  $z \in E$

Weiter sei  $k \vdash_M^{i+1} k' \iff \exists q k \vdash_M^1 q \vdash_M^i k'$  für alle  $i$  und  $k \vdash_M^* k' \iff \exists i \in \mathbb{N} k \vdash_M^i k'$

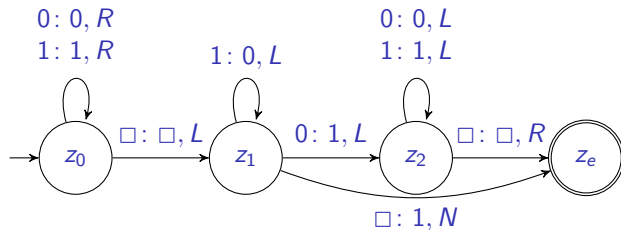
**Frage:** gibt es akzeptierende Konfigurationen, die nicht haltend sind?

# Beispiel Turing-Maschine: Binärzahl inkrementieren

$M = (\{z_0, z_1, z_2, z_e\}, \Sigma = \{0, 1\}, \Gamma = \{0, 1, \square\}, \delta, z_0, \square, \{z_e\})$

$\delta$	0	1	$\square$
$z_0$	$(z_0, 0, R)$	$(z_0, 1, R)$	$(z_1, \square, L)$
$z_1$	$(z_2, 1, L)$	$(z_1, 0, L)$	$(z_e, 1, N)$
$z_2$	$(z_2, 0, L)$	$(z_2, 1, L)$	$(z_e, \square, R)$

## Zustandsgraph



**Frage:** Was macht  $M$  bei leerer Eingabe? Bei Eingabe 000?

## Beispiel: Eingabe 101

$z_0 101$	$\vdash_M^1$
$1 z_0 01$	$\vdash_M^1$
$10 z_0 1$	$\vdash_M^1$
$101 z_0$	$\vdash_M^1$
$10 z_1 1$	$\vdash_M^1$
$1 z_1 00$	$\vdash_M^1$
$z_2 110$	$\vdash_M^1$
$z_2 \square 110$	$\vdash_M^1$
$z_e 110$	

$z_e 110$  haltend & akzeptierend

# Akzeptieren und Halten einer TM

## Definition (Akzeptieren, Halten)

Turing-Maschine  $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ :

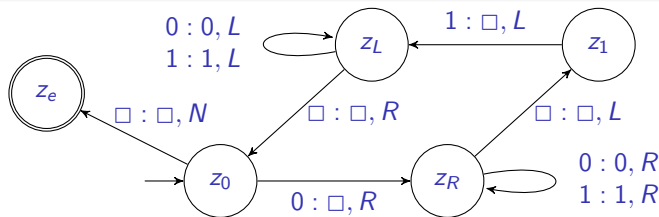
- ▶  $M$  **hält** auf  $w \in \Sigma^*$ , falls eine **haltende** Konfiguration  $k'$  existiert mit  $z_0 w \vdash_M^* k'$ .
- ▶  $M$  **akzeptiert**  $w \in \Sigma^*$ , falls eine **akzeptierende** Konfiguration  $k'$  existiert mit  $z_0 w \vdash_M^* k'$ .
- ▶  $M$  **akzeptiert Sprache**  $T(M)$  enthält genau die Wörter  $w$  die  $M$  akzeptiert. Formal,  
$$T(M) := \{w \in \Sigma^* \mid \exists_{\alpha, \beta \in \Gamma^*} \exists_{z \in E} : z_0 w \vdash_M^* \alpha z \beta\}.$$

# Beispiel Turing-Maschine: akzeptiere $\{0^n 1^n \mid n \in \mathbb{N}\}$

$M = (\{z_0, z_1, z_R, z_L, z_e\}, \Sigma = \{0, 1\}, \Gamma = \{0, 1, \square\}, \delta, z_0, \square, \{z_e\})$

$\delta$	0	1	$\square$
$z_0$	$(z_R, \square, R)$	$\perp$	$(z_e, \square, N)$
$z_1$	$\perp$	$(z_L, \square, L)$	$\perp$
$z_R$	$(z_R, 0, R)$	$(z_R, 1, R)$	$(z_1, \square, L)$
$z_L$	$(z_L, 0, L)$	$(z_L, 1, L)$	$(z_0, \square, R)$

## Zustandsgraph



## Beispiel: Eingabe 0011

$z_0 0011$	$\vdash_M^1$
$0 z_R 11$	$\vdash_M^2$
$011 z_R$	$\vdash_M^1$
$01 z_1 1$	$\vdash_M^1$
$0 z_L 1$	$\vdash_M^2$
$z_L \square 01$	$\vdash_M^1$
$z_0 01$	$\vdash_M^1$
$z_R 1$	$\vdash_M^1$
$1 z_R$	$\vdash_M^1$
$z_1 1$	$\vdash_M^1$
$z_L$	$\vdash_M^1$
$z_0$	$\vdash_M^1$
$z_e$	$\vdash_M^1$