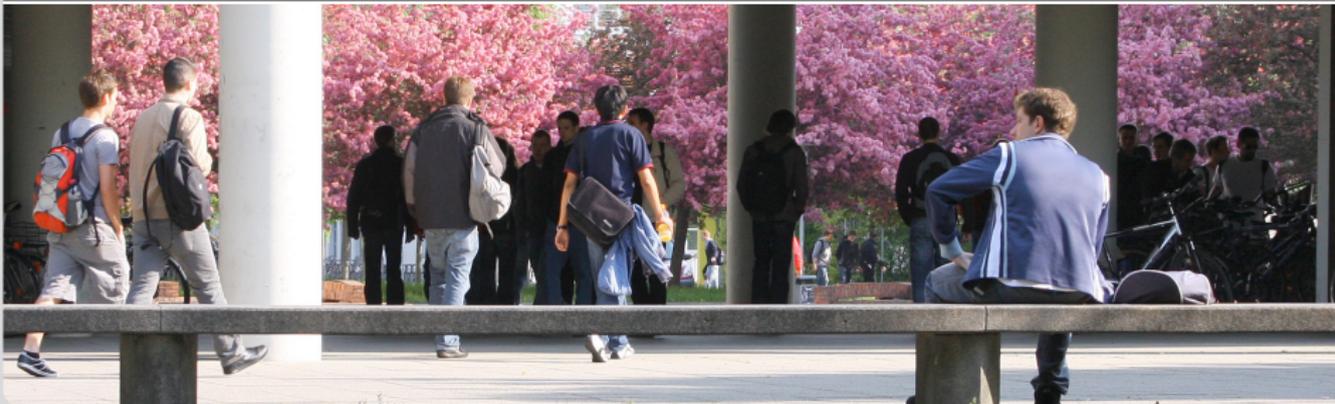


# Theoretische Grundlagen der Informatik

## Vorlesung am 14. November 2017

INSTITUT FÜR THEORETISCHE INFORMATIK

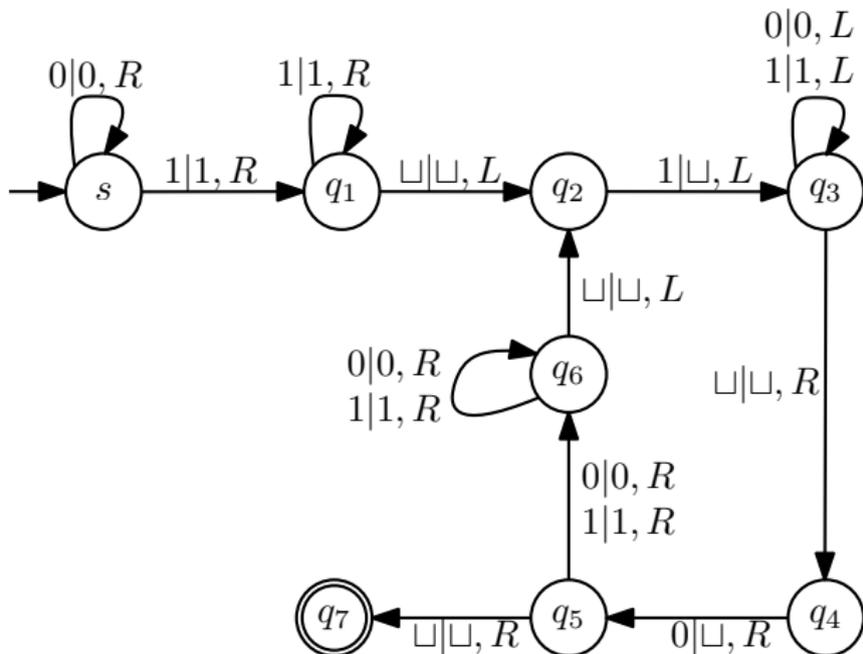


- Eine Turing-Maschine **akzeptiert** eine Eingabe  $w \in \Sigma^*$ , wenn sie nach Lesen von  $w$  in einem Zustand aus  $F$  stoppt.
- Sie **akzeptiert** eine Sprache  $L$  genau dann, wenn sie ausschließlich Wörter aus  $w \in L$  als Eingabe akzeptiert.
- Eine Sprache  $L \subseteq \Sigma^*$  heißt **rekursiv** oder **entscheidbar**, wenn es eine Turing-Maschine gibt, die auf allen Eingaben stoppt und eine Eingabe  $w$  genau dann akzeptiert, wenn  $w \in L$  gilt.
- Eine Sprache  $L \subseteq \Sigma^*$  heißt **rekursiv-aufzählbar** oder **semi-entscheidbar**, wenn es eine Turing-Maschine gibt, die genau die Eingaben  $w$  akzeptiert für die  $w \in L$ .

Das Verhalten der Turing-Maschine für Eingaben  $w \notin L$  ist damit nicht definiert. D.h., die Turing-Maschine stoppt entweder nicht in einem Endzustand oder aber stoppt gar nicht.

- Situation in der sich eine TM  $\mathcal{M} := (Q, \Sigma, \Gamma, \delta, s, F)$  befindet wird durch die Angabe der **Konfiguration** codiert.
- Eine Konfiguration hat die Form  $w(q)av$ , wobei
  - $w, v \in \Gamma^*$
  - $a \in \Gamma$
  - $q \in Q$
- Bedeutung:
  - $\mathcal{M}$  befindet sich gerade im Zustand  $q$ .
  - Der Lesekopf steht auf dem Zeichen  $a$ .
  - Links vom Lesekopf steht das Wort  $w$  auf dem Rechenband.
  - Rechts vom Lesekopf steht das Wort  $v$  auf dem Rechenband.

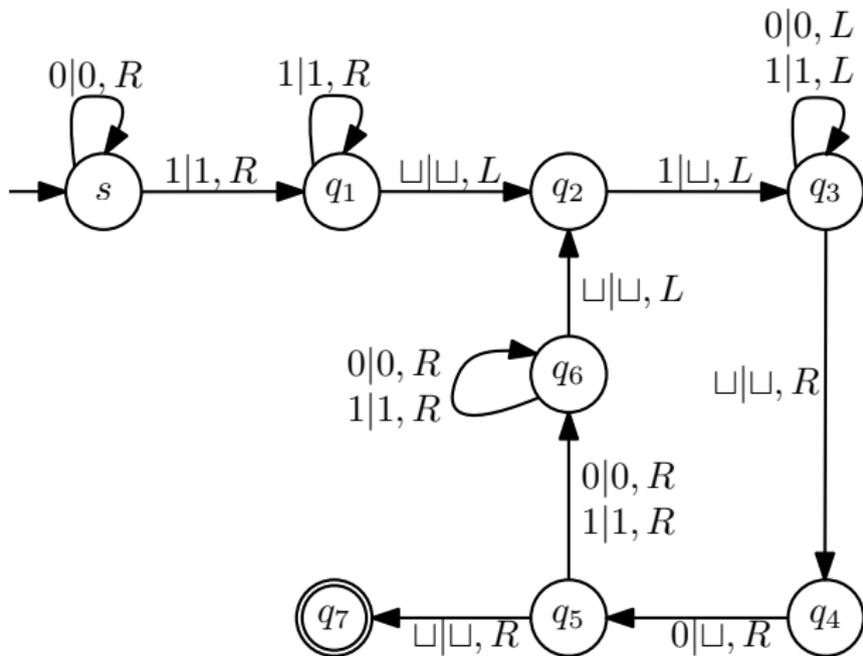
# Beispiel: Konfiguration



TM akzeptiert  
 $\{0^n 1^n : n \geq 1\}$ .

Eingabe: 0011

# Beispiel: Konfiguration

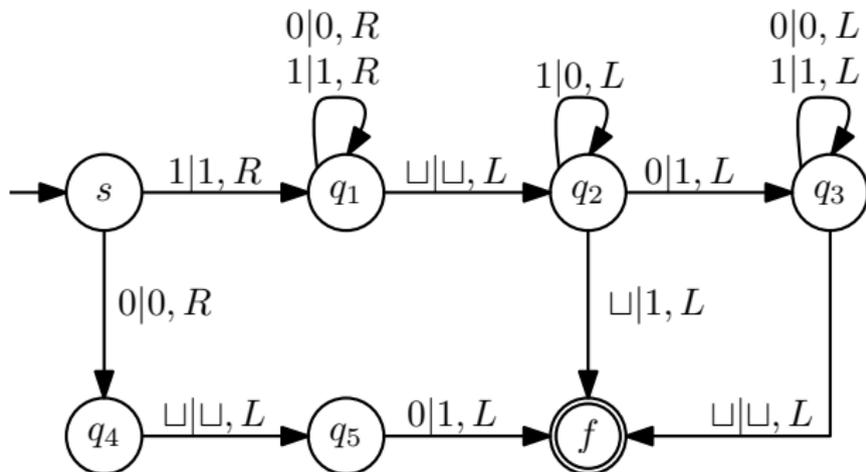


$\sqcup(s)0011$   
 $0(s)011$   
 $00(s)11$   
 $001(q_1)1$   
 $0011(q_1)\sqcup$   
 $001(q_2)1$   
 $00(q_3)1$   
 $0(q_3)01$   
 $\sqcup(q_3)001$   
 $\sqcup(q_3)\sqcup001$   
 $\sqcup(q_4)001$   
 $\sqcup(q_5)01$   
 $0(q_6)1$   
 $01(q_6)\sqcup$   
 $0(q_2)1$   
 $\sqcup(q_3)0$   
 $\sqcup(q_3)\sqcup0$   
 $\sqcup(q_4)0$   
 $\sqcup(q_5)\sqcup$   
 $\sqcup(q_7)\sqcup$

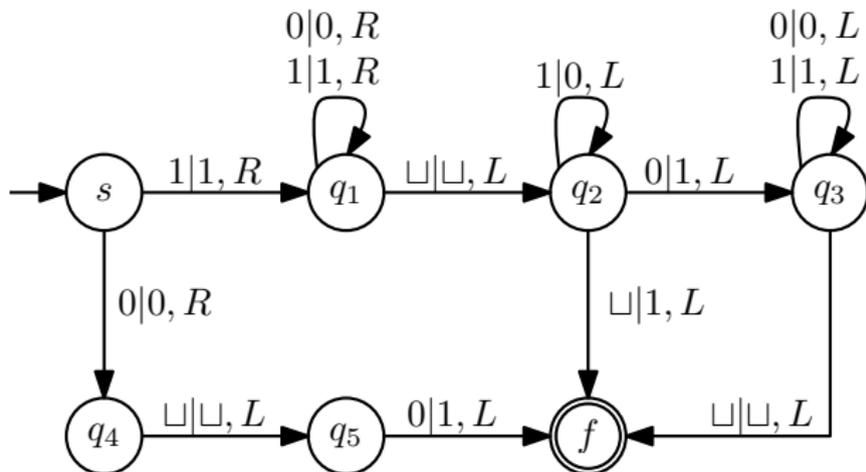
- Eine Funktion  $f: \Sigma^* \rightarrow \Gamma^*$  heißt **(Turing-)berechenbar** oder **totalrekursiv**, wenn es eine Turing-Maschine gibt, die bei Eingabe von  $w \in \Sigma^*$  den Funktionswert  $f(w) \in \Gamma^*$  ausgibt.

- Eine Turing-Maschine **realisiert** eine Funktion  $f: \Sigma^* \rightarrow \Gamma^*$ , falls gilt:

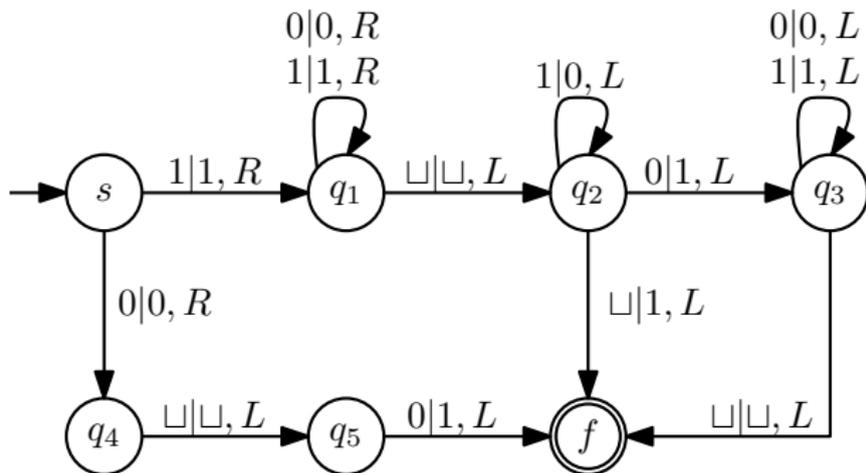
$$f(w) = \begin{cases} \text{Ausgabe der Turing-Maschine, wenn sie bei Eingabe } w \text{ stoppt} \\ \text{undefiniert sonst} \end{cases}$$



- Fasse die Eingabe  $w$  als binäre Zahl auf.
- Es sollen nur Eingaben ohne führende Nullen und die Null selbst akzeptiert werden.
- Addiere zur Eingabe  $w \in (0 \cup 1)^*$  eine Eins.

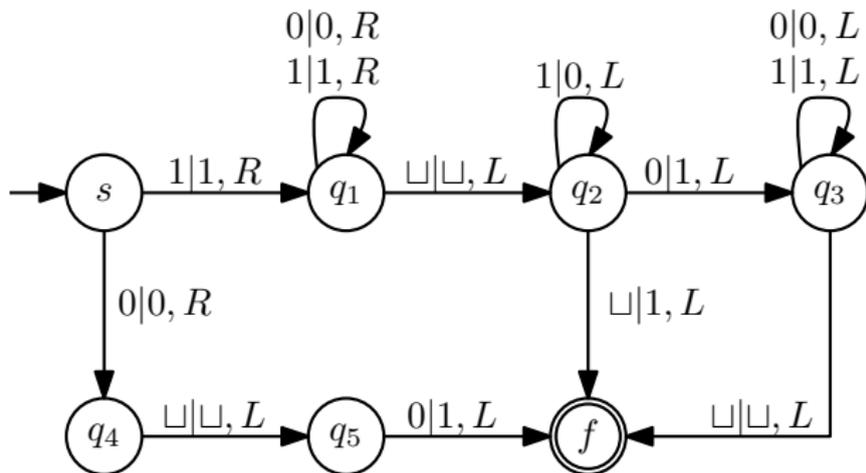


$$\text{Es gilt: } f(w) = \begin{cases} w + 1 & \text{falls } w \in 0 \cup 1(0 \cup 1)^*, \\ & w \text{ interpretiert als Binärzahl} \\ \text{undefiniert} & \text{sonst} \end{cases}$$



Dabei sind die Zustände jeweils für die folgenden Aufgaben verantwortlich:

- $q_1$  Bewegung des Lese-/Schreibkopfes nach rechts bis zum Eingabeende,
- $q_2$  Bildung des Übertrages, der durch die Addition von Eins zu einer bereits vorhandenen Eins entsteht,



Dabei sind die Zustände jeweils für die folgenden Aufgaben verantwortlich:

- $q_3$  Bewegung des Lese-/Schreibkopfes nach links, nachdem die Aufsummierung abgeschlossen ist (kein Übertrag mehr),
- $q_4, q_5$  Sonderbehandlung für den Fall der Eingabe 0, und
- $f$  Endzustand.

Entscheidbarkeit von Sprachen und Berechenbarkeit von Funktionen sind verwandt:

- Eine Turing-Maschine akzeptiert eine Sprache  $L$ , wenn sie genau auf den Eingaben  $w \in L$  in einem ausgezeichneten Endzustand stoppt.
- $L$  ist entscheidbar, wenn es eine Turing-Maschine gibt, die auf allen Eingaben stoppt und  $L$  akzeptiert.
- Die Funktion  $f$  heißt berechenbar, wenn eine Turing-Maschine existiert, die  $f$  realisiert.

Entscheidbarkeit von Sprachen und Berechenbarkeit von Funktionen sind verwandt:

- Man kann eine Turing-Maschine  $\mathcal{M}$ , die auf allen Eingaben stoppt, so modifizieren, dass es zwei ausgezeichnete Zustände  $q_J$  und  $q_N$  gibt und dass  $\mathcal{M}$  stets in einem der Zustände  $q_J$  oder  $q_N$  hält.
- Dabei stoppt sie bei der Eingabe  $w$  genau dann in  $q_J$ , wenn sie  $w$  akzeptiert, ansonsten in  $q_N$ .
- Damit ist die Sprache  $L$  genau dann entscheidbar, wenn es eine Turing-Maschine gibt, die immer in einem der Zustände  $\{q_J, q_N\}$  stoppt, wobei sie gerade für  $w \in L$  in  $q_J$  hält.

- Eine Sprache  $L \subseteq \Sigma^*$  ist **entscheidbar** genau dann, wenn ihre **charakteristische Funktion**  $\chi_L$  berechenbar ist, wobei gilt:

$$\chi_L: \Sigma^* \rightarrow \{0, 1\} \quad \text{mit} \quad \chi_L(w) = \begin{cases} 1 & \text{falls } w \in L \\ 0 & \text{sonst} \end{cases}$$

- Eine Sprache  $L$  ist **semi-entscheidbar** genau dann, wenn die Funktion  $\chi_L^*$  berechenbar ist, wobei gilt:

$$\chi_L^*(w) = \begin{cases} 1 & \text{falls } w \in L \\ \text{undefiniert} & \text{sonst} \end{cases}$$

## Church'sche These

Die Menge der (Turing-)berechenbaren Funktionen ist genau die Menge der im intuitiven Sinne überhaupt berechenbaren Funktionen.

## Church'sche These

Die Menge der (Turing-)berechenbaren Funktionen ist genau die Menge der im intuitiven Sinne überhaupt berechenbaren Funktionen.

## Interpretation

- Turing-Maschinen sind formale Modelle für Algorithmen.
- Kein Berechnungsverfahren kann algorithmisch genannt werden, wenn es nicht von einer Turing-Maschine ausführbar ist.

## Bemerkung

- The Church'sche These ist ohne eine präzise Definition von *intuitiv berechenbar* nicht beweisbar.
- Sie ist aber in der Informatik allgemein akzeptiert.

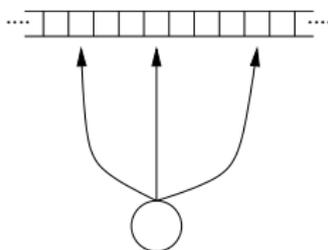
## Church'sche These

Die Menge der (Turing-)berechenbaren Funktionen ist genau die Menge der im intuitiven Sinne überhaupt berechenbaren Funktionen.

## Begründung

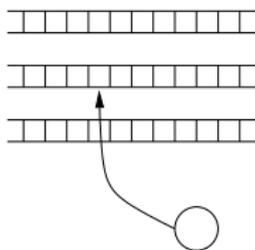
- Es existieren keine Beispiele von Funktionen, die als intuitiv berechenbar angesehen werden, aber nicht Turing-berechenbar sind.
- Alle Versuche, realistische Modelle aufzustellen, die mächtiger sind als Turing-Maschinen, schlugen fehl.
- Eine Reihe von völlig andersartigen Ansätzen, den Begriff der Berechenbarkeit formal zu fassen, wie zum Beispiel die Registermaschine, haben sich als äquivalent erwiesen.

## Mehrere Lese-/Schreibköpfe



- Mehrere Lese-/Schreibköpfe ( $n \in \mathbb{N}$ ) greifen auf das eine Eingabeband zu und werden von der endlichen Kontrolle gesteuert.
- Die Übergangsfunktion ist dann vom Typ  $\delta: Q \times \Gamma^n \rightarrow Q \times \Gamma^n \times \{L, N, R\}^n$ .
- Die Zustände  $q \in Q$  kann man als  $n$ -Tupel auffassen.
- Es ist nötig eine Prioritätenregel für die einzelnen Köpfe anzugeben, falls mehrere auf einem Feld des Eingabebandes stehen.

## Mehrere Bänder



- Ein Lese-/Schreibkopf kann auf mehrere Eingabebänder ( $n \in \mathbb{N}$ ) zugreifen.
- Die Übergangsfunktion ist dann vom Typ

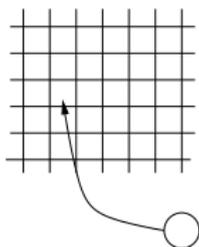
$$\delta: Q \times \Gamma \times \{1, \dots, n\} \rightarrow Q \times \Gamma \times \{L, N, R\} \times \{1, \dots, n\}.$$

## Mehrere Lese-/Schreibköpfe für mehrere Bänder

- Wir haben jetzt  $m$  Bänder und  $n$  Lese-/Schreibköpfe.
- Die Übergangsfunktion ist dann vom Typ

$$\delta: Q \times \Gamma^n \times \{1, \dots, m\}^n \rightarrow Q \times \Gamma^n \times \{L, N, R\}^n \times \{1, \dots, m\}^n.$$

## Mehrdimensionale Bänder



- Das Eingabeband ist nun mehrdimensional und hat hier im Beispiel die Dimension zwei.
- Wir sprechen dann von einem Arbeitsfeld.
- Dabei ist

$$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L(eft), U(p), R(ight), D(own), N(othing)\}$$

## Bemerkungen

- Fragestellungen der Art:
  - Wann stoppt eine Mehrkopf-Maschine?
  - Welcher Kopf ‚gewinnt‘, wenn mehrere Köpfe (verschiedene) Symbole an dieselbe Stelle schreiben wollen?müssen bei solchen Modifikationen noch geklärt werden.
  
- Es hat sich allerdings gezeigt, dass keine dieser Erweiterungen mehr leistet, als eine normale Turing-Maschine.
  
- **Alle angegebenen Modifikationen können durch eine normale 1–Band Turing-Maschine simuliert werden.**

## Ziel

- Bisher: Nur Turing-Maschinen, die eine bestimmte Aufgabe erfüllen.
- Jetzt: Konstruktion einer Turing-Maschine, die als Eingabe
  - ein Programm und
  - eine spezielle Eingabeerhält.
- Die Aufgabe besteht darin, das gegebene Programm auf der gegebenen speziellen Eingabe auszuführen.

Wir betrachten dazu eine normierte Turing-Maschine, d.h.

- $Q := \{q_1, \dots, q_n\}$
  - $\Sigma := \{a_1, \dots, a_k\}$
  - $\Gamma := \{a_1, \dots, a_k, a_{k+1}, \dots, a_l\}$
  - $s := q_1$
  - $F := \{q_2\}$
- 
- Dies bedeutet keine Einschränkung in der Mächtigkeit der Turing-Maschinen:
    - Jede beliebige Turing-Maschine kann durch eine derart normiert Turing-Maschine der obigen Form simuliert werden.
    - Jede normierte Turing-Maschine  $\mathcal{M}$  lässt sich eindeutig als Wort aus  $(0 \cup 1)^*$  kodieren.

Sei  $\mathcal{M} := (Q, \Sigma, \Gamma, \delta, s, F)$  eine Turing-Maschine.

Die Gödelnummer von  $\mathcal{M}$ , bezeichnet als  $\langle \mathcal{M} \rangle$ , ist definiert durch folgende Kodierungsvorschrift:

- Kodiere

$$\delta(q_i, a_j) = (q_r, a_s, d_t) \text{ durch } 0^j 1 0^j 1 0^r 1 0^s 1 0^t,$$

wobei  $d_t \in \{d_1, d_2, d_3\}$  und

- $d_1$  für  $L$ ,
  - $d_2$  für  $R$  und
  - $d_3$  für  $N$  steht.
- Die Turing-Maschine wird dann kodiert durch:

$$111\text{code}_1 11\text{code}_2 11 \dots 11\text{code}_z 111,$$

wobei  $\text{code}_i$  für  $i = 1, \dots, z$  alle Funktionswerte von  $\delta$  in beliebiger Reihenfolge beschreibt.

- Die eigentlichen Werte der Turing-Maschine werden also (unär) durch Nullen beschrieben und die Einsen dienen als Begrenzung der Eingabewerte.
- Jede Turing-Maschine kann also durch ein Wort aus  $(0 \cup 1)^*$  kodiert werden.
- Umgekehrt beschreibt jedes Wort aus  $(0 \cup 1)^*$  (höchstens) eine Turing-Maschine.
- Wir vereinbaren, dass ein Wort, das keine Turing-Maschine in diesem Sinne beschreibt, (zum Beispiel  $\varepsilon$ , 0, 000) eine Turing-Maschine kodiert, die  $\emptyset$  akzeptiert.
- Eine *universelle Turing-Maschine* erhält als Eingabe ein Paar  $(\langle \mathcal{M} \rangle, w)$ , wobei  $w \in \{0, 1\}^*$  ist, und sie simuliert  $\mathcal{M}$  auf  $w$ .

# Die Gödelnummer - Beispiel

Sei  $\mathcal{M} = (\{q_1, q_2, q_3\}, \{0, 1\}, \sqcup, \{0, 1, \sqcup\}, \delta, q_1, \{q_2\})$ , mit

$$\delta(q_1, 1) = (q_3, 0, R)$$

$$\delta(q_3, 0) = (q_1, 1, R)$$

$$\delta(q_3, 1) = (q_2, 0, R)$$

$$\delta(q_3, \sqcup) = (q_3, 1, L)$$

$\mathcal{M}$  zusammen mit der Eingabe 1011 ist dann:

```
111010010001010011000101010010011000
100100101001100010001000100101111011
```

# Definition

Zu  $w \in \{0, 1\}^*$  sei  $T_w$

- die Turing-Maschine mit der Gödelnummer (GN)  $w$ , bzw.
- die Turing-Maschine, die  $\emptyset$  akzeptiert.

Es sei  $L(T_w)$  die Sprache, die von  $T_w$  akzeptiert wird.

Wir konstruieren die sogenannte **Diagonalsprache**  $L_d$ , wie folgt:

- Betrachte die Wörter aus  $\{0, 1\}^*$  in *kanonischer* Reihenfolge, d.h.  $w_i$  steht vor  $w_j$  ( $i < j$ ), falls
  - $|w_i| < |w_j|$ , oder
  - $|w_i| = |w_j|$  und  $w_i$  lexikographisch vor  $w_j$  steht.
- $\mathcal{M}_j$  sei die TM, die durch die Gödelnummer  $w_j$  kodiert ist.
- Wir konstruieren eine unendliche Tabelle,
  - an deren Position  $(i, j)$  für  $1 \leq i, j < \infty$  eine Null oder eine Eins steht, und
  - welche beinhaltet, ob  $w_i$  in  $L(\mathcal{M}_j)$  ist.
- Damit gilt für die Einträge

$$(i, j) = \begin{cases} 1 & \text{falls } \mathcal{M}_j \text{ } w_i \text{ akzeptiert} \\ 0 & \text{sonst} \end{cases}$$

- Damit gilt für die Einträge

$$(i, j) = \begin{cases} 1 & \text{falls } \mathcal{M}_j \text{ } w_i \text{ akzeptiert} \\ 0 & \text{sonst} \end{cases}$$

- Definiere dazu

$$L_d := \{w_i \mid \mathcal{M}_i \text{ akzeptiert } w_i \text{ nicht}\}.$$

- $L_d$  enthält also alle  $w_i$ , für die auf der Diagonalen an der Stelle  $(i, i)$  eine Null steht.
- Dies führt später zu einem Diagonalbeweis (Cantor).

# Die Diagonalsprache - Veranschaulichung

$w \in \{0, 1\}^*$	Gödelnummer							
	$w_i$	$w_j$	$w_k$					
$\vdots$	$\vdots$							
$w_i$	1	<b>0</b>	1	0	1	0	0	$w_i \in L_d$
$w_j$	0	0	<b>1</b>	0	0	1	1	$w_j \notin L_d$
$w_k$	1	0	0	<b>1</b>	1	0	1	$w_k \notin L_d$
$\vdots$	$\vdots$							

**Satz (Unentscheidbarkeit der Diagonalsprache):**  
Die Sprache  $L_d$  ist nicht entscheidbar.

**Satz (Unentscheidbarkeit der Diagonalsprache):**  
Die Sprache  $L_d$  ist nicht entscheidbar.

## Beweis:

Falls  $L_d$  entscheidbar ist, gibt es eine Turing-Maschine  $\mathcal{M}_i$ , die

- (1) stets hält,
- (2) genau die  $w \in L_d$  akzeptiert.

Wende nun  $\mathcal{M}_i$  auf  $w_i$  an:

- Falls  $w_i \in L_d$ , dann wird  $w_i$ , wegen (2) von  $\mathcal{M}_i$  akzeptiert.
- Falls  $w_i \notin L_d$ , dann akzeptiert  $\mathcal{M}_i$  das Wort  $w_i$  wegen (2) nicht.

Beides ist ein Widerspruch zur Definition von  $L_d$ .

## Korollar

Die Sprache  $L_d^c := \{0, 1\}^* \setminus L_d$  ist nicht entscheidbar.

## Korollar

Die Sprache  $L_d^c := \{0, 1\}^* \setminus L_d$  ist nicht entscheidbar.

## Beweis:

- Falls  $L_d^c$  entscheidbar ist, gibt es eine Turing-Maschine, die  $L_d^c$  entscheidet.
- Diese kann aber leicht zu einer Turing-Maschine modifiziert werden, die  $L_d$  entscheidet.
- Dies ist ein Widerspruch zur Unentscheidbarkeit der Diagonalsprache.

## Korollar

Die Sprache  $L_d^c := \{0, 1\}^* \setminus L_d$  ist nicht entscheidbar.

## Beweis:

- Falls  $L_d^c$  entscheidbar ist, gibt es eine Turing-Maschine, die  $L_d^c$  entscheidet.
- Diese kann aber leicht zu einer Turing-Maschine modifiziert werden, die  $L_d$  entscheidet.
- Dies ist ein Widerspruch zur Unentscheidbarkeit der Diagonalsprache.

Der Barbier von Hintertupfingen rasiert genau die Männer im Dorf, die sich nicht selbst rasieren.

**Wer rasiert den Barbier?**

Der Barbier von Hintertupfingen rasiert genau die Männer im Dorf, die sich nicht selbst rasieren.

**Wer rasiert den Barbier?**

Daniel Düsentrieb behauptet, eine allwissende Maschine erfunden zu haben.

Man stellt eine Ja/Nein-Frage und die Antwort leuchtet auf.

Dagobert Duck kauft die Maschine.

Will aber nur bei korrekter Antwort zahlen.

Er stellt der Maschine die Frage: Wirst du mit **Nein** antworten?

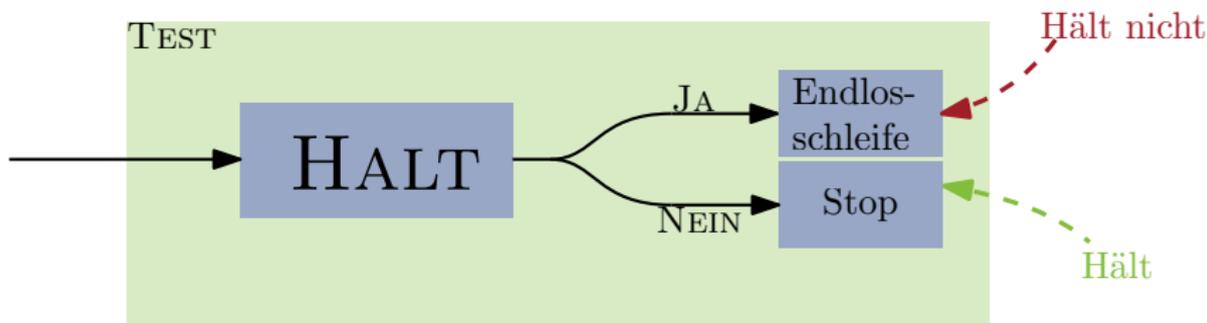
**Was passiert?**

# Halteproblem

## Programm HALT:



## Programm TEST:

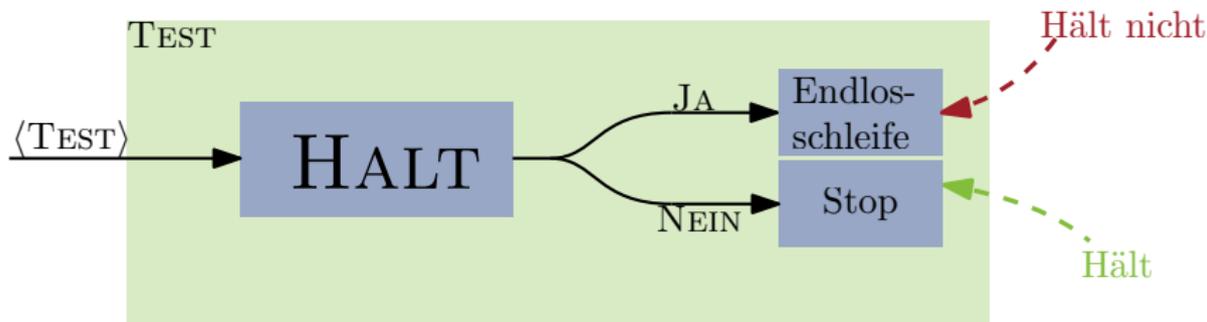


# Halteproblem

## Programm HALT:



## Programm TEST:



Wie verhält sich TEST bei der Eingabe  $\langle \text{TEST} \rangle$ ?

Das **Halteproblem** definiert folgende Sprache

$$\mathcal{H} := \{wv \mid T_w \text{ hält auf der Eingabe } v\}.$$

**Satz (Unentscheidbarkeit des Halteproblems):**

$\mathcal{H}$  ist nicht entscheidbar.

## Interpretation:

- Das Problem, ob eine Turing-Maschine auf einer Eingabe  $w$  stoppt, ist nicht entscheidbar.

## Satz (Unentscheidbarkeit des Halteproblems):

$\mathcal{H} = \{wv \mid T_w \text{ hält auf der Eingabe } v\}$  ist nicht entscheidbar.

### Beweis:

- Angenommen es existiert eine stets haltende Turing-Maschine, die  $\mathcal{H}$  entscheidet.
- Wir konstruieren daraus eine stets haltende Turing-Maschine, die  $L_d^C$  entscheidet, mit Widerspruch zum letzten Korollar.

Sei  $w$  eine Eingabe, für die wir entscheiden wollen, ob  $w \in L_d^C$ .

Wir können wie folgt vorgehen:

- Berechne das  $i$ , so dass  $w = w_i$  ist.
- Betrachte die durch  $w_i$  kodierte Turing-Maschine  $\mathcal{M}_i$ .
- Wende die Turing-Maschine für  $\mathcal{H}$  auf  $\langle \mathcal{M}_i \rangle \cdot w_i$  an.

## Satz (Unentscheidbarkeit des Halteproblems):

$\mathcal{H} = \{wv \mid T_w \text{ hält auf der Eingabe } v\}$  ist nicht entscheidbar.

Sei  $w$  eine Eingabe, für die wir entscheiden wollen, ob  $w \in L_d^c$ .  
Wir können wie folgt vorgehen:

- Berechne das  $i$ , so dass  $w = w_i$  ist.
- Betrachte die durch  $w_i$  kodierte Turing-Maschine  $\mathcal{M}_i$ .
- Wende die Turing-Maschine für  $\mathcal{H}$  auf  $\langle \mathcal{M}_i \rangle \cdot w_i$  an.

Wir machen folgende Fallunterscheidung:

- Falls  $\langle \mathcal{M}_i \rangle \cdot w_i$  nicht akzeptiert wird, dann hält  $\mathcal{M}_i$  nicht auf  $w_i$ .
- Nach Definition von  $\mathcal{H}$  ist also  $w_i \in L_d$  und damit  $w_i \notin L_d^c$ .
- Falls  $\langle \mathcal{M}_i \rangle \cdot w_i$  akzeptiert wird, dann hält  $\mathcal{M}_i$  auf  $w_i$ .
- Dann können wir auf der universellen Turing-Maschine die Berechnung von  $\mathcal{M}_i$  auf  $w_i$  simulieren.