

**Vorlesung**  
**Grundlagen der Theoretischen Informatik /**  
**Einführung in die Theoretische Informatik I**

**Bernhard Beckert**

**Institut für Informatik**



**Sommersemester 2007**

Diese Vorlesungsmaterialien basieren ganz wesentlich auf den Folien zu den Vorlesungen von

**Katrin Erk** (gehalten an der Universität Koblenz-Landau)

**Jürgen Dix** (gehalten an der TU Clausthal)

Ihnen beiden gilt mein herzlicher Dank.

*– Bernhard Beckert, April 2007*

## Inhalt von Teil V

- Was ist eine **berechenbare** Funktion?
- Determinierte Turing-Maschinen (DTMs)?
- Modifikationen von DTMs:  
(mehrere) Halbbänder, zweiseitig unbeschränkte Bänder
- Indeterminierte Turing-Maschinen (NTMs)
- **Gödelisieren**: Programme als Wörter in  $\Sigma^*$ .
- **Aufzählbar** vs. **entscheidbar**
- **Unentscheidbarkeit**, Reduktionen von Problemen aufeinander.

## Turing Maschinen

- 1** **Determinierte Turing-Maschinen (DTMs)**
- 2 Varianten von Turing-Maschinen
- 3 Indeterminierte Turing-Maschinen (NTMs)
- 4 Universelle determinierte Turing-Maschinen
- 5 Entscheidbar/Aufzählbar
- 6 Determinierte Turing-Maschinen entsprechen Typ 0
- 7 Unentscheidbarkeit

## Frage: Berechenbarkeit?

Betrachtet werden Abbildungen über den natürlichen Zahlen  $\mathbb{N}$ :

**Welche davon sollen berechenbar genannt werden?**

## Frage: Komplexität?

Um die Komplexität eines Algorithmus' zu messen braucht man ein Maschinenmodell zum Vergleich!

**Welches Modell wird gewählt?**

**Robustheit:** Das Modell soll nicht von einfachen Modifikationen abhängig sein.

## Turing-Maschine: Die Maschine unter den Maschinen

*One ring to rule them all.*

## Alan Turing ★ 1912, † 1954

- Mathematiker und Logiker
- Einer der Begründer der Informatik
- 1936: Definition des Berechenbarkeitsmodells „Turing-Maschine“
- 1938: Promotion bei Church in Princeton
- Während des zweiten Weltkriegs: Kriegsentscheidender(?) Beitrag zur Entschlüsselung deutscher Funkprüche
- Dozent an der Universität Manchester
- Beiträge zur KI („Turing-Test“)
- Tragischer Tod:  
Strafverfolgung wegen Homosexualität; (vermutlich) Selbstmord
- Nach ihm benannt: Turing-Award



## Erinnerung: Endliche Automaten

- akzeptieren reguläre Sprachen
- Einziger Speicher: der **Zustand** (endlich).
- Die Zustandsmenge kann groß sein, ist aber endlich.
- Das Eingabewort wird nur einmal gelesen, von links nach rechts.



## Erinnerung: Pushdown-Automaten

- akzeptieren kontextfreie Sprachen
- Erster Speicher: der Zustand (endlich)
- Zweiter Speicher: der **Keller**  
(unbeschränkte Größe, beschränkte Zugriffsart)
- Das Eingabewort wird nur einmal gelesen, von links nach rechts.

## Ausblick: Turing-Maschinen

- akzeptieren Sprachen **vom Typ 0**.
- Erster Speicher: der Zustand (endlich)
- Zweiter Speicher: Band  
(unbeschränkte Größe, **Zugriff an beliebiger Stelle**)
- Turing-Maschine hat einen Schreib-/Lesekopf, den sie über diesem Band in einem Rechenschritt um ein Feld nach rechts oder links bewegen kann.
- Das Eingabewort steht (am Anfang) auf dem Band.  
Die Maschine kann es **beliebig oft lesen**.

## Definition 9.1 (Turing-Maschine (DTM))

Eine **determinierte Turing-Maschine (DTM)**  $\mathcal{M}$  ist ein Tupel

$$\mathcal{M} = ( K, \Sigma, \delta, s )$$

Dabei ist

- $K$  eine endliche Menge von Zuständen mit  $h \notin K$ , ( $h$  ist der **Haltezustand**)
- $\Sigma$  ein Alphabet mit  $L, R \notin \Sigma$ ,  $\# \in \Sigma$ ,
- $\delta : K \times \Sigma \rightarrow (K \cup \{h\}) \times (\Sigma \cup \{L, R\})$  eine Übergangsfunktion
- $s \in K$  ein Startzustand.

Anzahl der Zustände:  $|K| - 1$   
(Startzustand wird nicht mitgezählt).

## Arbeitsschritt einer Turing-Maschine

Übergang

$$\delta(q, a) = (q', x)$$

bedeutet:

In Abhängigkeit

- vom aktuellen Zustand  $q \in K$
- von dem Zeichen  $a \in \Sigma$ , das unter dem Schreib-/Lesekopf steht

geschieht folgendes:

- entweder ein **Schritt nach links**, falls  $x = L$  ist
- oder ein **Schritt nach rechts**, falls  $x = R$  ist
- oder **das Zeichen  $a$** , das momentan unter dem Schreib-/Lesekopf steht, wird **durch  $b \in \Sigma$  überschreiben**, falls  $x = b \in \Sigma$
- der **Zustand** wird zu  $q' \in K \cup \{h\}$  **geändert**,

## Leerzeichen

Das spezielle Zeichen # (*blank*) ist das Leerzeichen.

Es ist nie Teil des Eingabeworts; man kann es u.a. dazu benutzen, Wörter voneinander abzugrenzen.

## Begrenzung des Bandes

Das Band einer DTM ist **einseitig unbeschränkt**:

- Nach rechts ist es unendlich lang.
- Nach links hat es ein Ende.
- Wenn eine DTM versucht, das Ende zu überschreiten, bleibt sie „hängen“.

In diesem Fall **hält sie nicht**.

## Anfangskonfiguration

- Ganz links auf dem Band steht ein Blank
- Direkt rechts davon steht das Eingabewort
- Wenn eine DTM mehrere Eingabewörter hintereinander bekommt, sind sie durch Blanks getrennt.
- Rechts vom letzten Eingabewort stehen nur noch Blanks.
- Der Schreib-/Lesekopf der DTM steht auf dem Blank direkt rechts neben dem (letzten) Eingabewort.

## Merke

Das Band enthält immer nur endlich viele Symbole, die keine Blanks sind.

## Beispiel 9.2 ( $\mathcal{R}(a)$ ): $a$ 's durch $b$ 's ersetzen)

Die folgende Turing-Maschine  $\mathcal{R}(a)$  erwartet *ein* Eingabewort. Sie liest es von rechts nach links einmal durch und macht dabei jedes  $a$  zu einem  $b$ .

Es ist

$$\mathcal{R}(a) = ( \{q_0, q_1\}, \{a, b, \#\}, \delta, q_0 )$$

mit folgender  $\delta$ -Funktion:

$$\begin{array}{ll} q_0, \# \mapsto q_1, L & q_1, \# \mapsto h, \# \\ q_0, a \mapsto q_0, a & q_1, a \mapsto q_1, b \\ q_0, b \mapsto q_0, b & q_1, b \mapsto q_1, L \end{array}$$



## Beispiel 9.3 ( $L_{\#}$ )

Die folgende Turing-Maschine  $\mathcal{L}_{\#}$  läuft zum ersten Blank links von der momentanen Position.

Es ist  $\mathcal{L}_{\#} = ( \{q_0, q_1\}, \{a, b, \#\}, \delta, q_0 )$  mit folgender  $\delta$ -Funktion:

$$\begin{array}{ll} q_0, \# \mapsto q_1, L & q_1, \# \mapsto h, \# \\ q_0, a \mapsto q_1, L & q_1, a \mapsto q_1, L \\ q_0, b \mapsto q_1, L & q_1, b \mapsto q_1, L \end{array}$$

$q_0$ : Anfangsposition

$q_1$ : Anfangsposition verlassen

## Beispiel 9.4 (Copy)

Die folgende DTM  $\mathcal{C}$  erhält als Eingabe einen String Einsen.

Dieser String wird kopiert:

Falls  $n$  Einsen auf dem Band stehen, stehen nach Ausführung von  $\mathcal{C}$   $2n$  Einsen auf dem Band stehen, (getrennt durch ein Blank #).

state	#	1	c
$q_0$	$\langle q_1, c \rangle$	—	—
$q_1$	$\langle q_2, R \rangle$	$\langle q_1, L \rangle$	$\langle q_1, L \rangle$
$q_2$	—	$\langle q_3, \# \rangle$	$\langle q_7, \# \rangle$
$q_3$	$\langle q_4, R \rangle$	—	—
$q_4$	$\langle q_5, 1 \rangle$	$\langle q_4, R \rangle$	$\langle q_4, R \rangle$
$q_5$	$\langle q_6, 1 \rangle$	$\langle q_5, L \rangle$	$\langle q_5, L \rangle$
$q_6$	—	$\langle q_2, R \rangle$	—
$q_7$	$\langle q_8, R \rangle$	—	—
$q_8$	$\langle h, \# \rangle$	$\langle q_8, R \rangle$	—

## Übergangsfunktion $\delta$ nicht überall definiert

Wir erlauben ab jetzt auch, daß  $\delta$  nicht überall definiert ist.

Falls die DTM dann in einen solchen nichtdefinierten Zustand kommt, sagen wir **die DTM hängt**. Sie **hält** also nicht.

Dies wird z.T. in der Literatur anders gehandhabt.

## Beispiel 9.5 (Print $n$ )

Für jedes  $n \in \mathbb{N}$  konstruieren wir eine Maschine, die genau  $n$  Einsen auf das leere Band schreibt (**mit möglichst wenig Zuständen**):

- 1 schreibe  $\lfloor \frac{n}{2} \rfloor$  viele Einsen auf das Band (höchstens  $\lfloor \frac{n}{2} \rfloor$  Zustände)
- 2 kopiere diesen String (8 Zustände)
- 3 ersetze das trennende # durch eine 1
- 4 falls  $n$  gerade ist, ersetze die letzte 1 durch # (2 neue Zustände)

Insgesamt können wir  $n$  Einsen mit höchstens  $\lfloor \frac{n}{2} \rfloor + 10$  Zuständen konstruieren

## Begriff der Konfigurationen

- **Konfiguration** beschreibt die *komplette* aktuelle Situation der Maschine in einer Rechnung.
- Eine **Rechnung** ist eine Folge von Konfigurationen, wobei immer von einer Konfiguration zu einer Nachfolgekongfiguration übergegangen wird.

## Konfiguration einer DTM

Besteht aus 4 Elementen:

- das aktuellen Zustand  $q$ ,
- das Wort  $w$  links vom Schreib-/Lesekopf,
- das Zeichen  $a$ , auf dem der Kopf gerade steht,
- das Wort  $u$  rechts von der aktuellen Kopfposition.

## Konfigurationen sind endlich

- $w$  enthält das Anfangsstück des Bandes vom linken Ende bis zur aktuellen Kopfposition.
- **Links ist das Band endlich!**  
 $w = \varepsilon$  bedeutet, daß der Kopf ganz links steht
- $u$  enthält den Bandinhalt rechts vom Schreib-/Lesekopf bis zum letzten Zeichen, das kein Blank ist.
- **Nach rechts ist das Band unendlich, aber es enthält nach rechts von einer bestimmten Bandposition an nur noch Blanks.**  
 $u = \varepsilon$  bedeutet, daß rechts vom Schreib-/Lesekopf nur noch Blanks stehen.

## Definition 9.6 (Konfiguration einer DTM)

Eine **Konfiguration**  $C$  einer DTM  $\mathcal{M} = (K, \Sigma, \delta, s)$  ist ein Wort der Form  $C = q, w\underline{a}u$ . Dabei ist

- $q \in K \cup \{h\}$  der aktuelle Zustand,
- $w \in \Sigma^*$  der Bandinhalt links des Kopfes,
- $a \in \Sigma$  das Bandzeichen unter der Schreib-/Lesekopf.

**Notation:** Die Position des Schreib-/Lesekopfes ist durch einen Unterstrich gekennzeichnet.

- $u \in \Sigma^*(\Sigma - \{\#\}) \cup \{\varepsilon\}$  der Bandinhalt rechts des Kopfes.

## Definition 9.7 (Nachfolgekonfiguration)

Eine Konfiguration  $C_2$  heißt **Nachfolgekonfiguration** von  $C_1$ , in Zeichen

$$C_1 \vdash_{\mathcal{M}} C_2$$

falls gilt:

- $C_i = q_i, w_i \underline{a_i} u_i$  für  $i \in \{1, 2\}$ , und
- es gibt einen Übergang  $\delta(q_1, a_1) = (q_2, b)$  wie folgt:

**Fall 1:**  $b \in \Sigma$ . Dann ist  $w_1 = w_2$ ,  $u_1 = u_2$ ,  $a_2 = b$ .

**Fall 2:**  $b = L$ . Dann gilt für  $w_2$  und  $a_2$ :  $w_1 = w_2 a_2$ .

Für  $u_2$  gilt: Wenn  $a_1 = \#$  und  $u_1 = \varepsilon$  ist, so ist  $u_2 = \varepsilon$ ,  
sonst ist  $u_2 = a_1 u_1$ .

**Fall 3:**  $b = R$ . Dann ist  $w_2 = w_1 a_1$ .

Für  $a_2$  und  $u_2$  gilt: Wenn  $u_1 = \varepsilon$  ist, dann ist  $u_2 = \varepsilon$   
und  $a_2 = \#$ , ansonsten ist  $u_1 = a_2 u_2$ .



## Definition 9.8 (Eingabe)

$w$  heißt **Eingabe** (*input*) für  $\mathcal{M}$ , falls  $\mathcal{M}$  mit der **Startkonfiguration**

$$C_0 = s, \#w\underline{\#}$$

startet.

$(w_1, \dots, w_n)$  heißt **Eingabe** für  $\mathcal{M}$ , falls  $\mathcal{M}$  mit der **Startkonfiguration**

$$C_0 = s, \#w_1\# \dots \#w_n\underline{\#}$$

startet.

## Definition 9.9 (Halten, Hängen)

Sei  $\mathcal{M}$  eine Turing-Maschine.

- $\mathcal{M}$  **hält** in  $C = q, w\underline{a}u$  **gdw.**  $q = h$ .
- $\mathcal{M}$  **hängt** in  $C = q, w\underline{a}u$  **gdw.** es keine Nachfolgekonfiguration gibt  
**Insbesondere:** wenn  $w = \varepsilon \wedge \exists q' \delta(q, a) = (q', L)$ .

## Definition 9.10 (Rechnung)

Sei  $\mathcal{M}$  eine Turing-Maschine. Man schreibt

$$C \vdash_{\mathcal{M}}^* C'$$

gdw.:

es gibt eine Reihe von Konfigurationen

$$C_0, C_1, \dots, C_n \quad (n \geq 0)$$

so daß

- $C = C_0$  und  $C' = C_n$
- für alle  $i < n$  gilt:  $C_i \vdash_{\mathcal{M}} C_{i+1}$

Dann heißt  $C_0, C_1, \dots, C_n$  eine **Rechnung** der Länge  $n$  von  $C_0$  nach  $C_n$ .

# Turing-Maschine können Funktionen berechnen

## Definition 9.11 (TM-berechenbare Funktion)

Sei  $\Sigma_0$  ein Alphabet mit  $\# \notin \Sigma_0$ .

Eine (partielle) Funktion

$$f : (\Sigma_0^*)^m \rightarrow (\Sigma_0^*)^n$$

heißt **DTM-berechenbar**, falls:

Es existiert eine determinierte Turing-Maschine  $\mathcal{M} = (K, \Sigma, \delta, s)$

- mit  $\Sigma_0 \subseteq \Sigma$ ,
- so daß für alle  $w_1, \dots, w_m, u_1, \dots, u_n \in \Sigma_0^*$  gilt:
  - $f(w_1, \dots, w_m) = (u_1, \dots, u_n)$  gdw  
 $s, \#w_1\#\dots\#w_m\# \vdash_{\mathcal{M}}^* h, \#u_1\#\dots\#u_n\#$
  - $f(w_1, \dots, w_m)$  ist undefiniert gdw  
 $\mathcal{M}$  gestartet mit  $s, \#w_1\#\dots\#w_m\#$  hält nicht (läuft unendlich oder hängt)

# Turing-Maschine können Funktionen berechnen

## Vorsicht

Wir betrachten Turing-Maschinen hier unter einem anderen Aspekt als alle bisherigen Automaten:

- Bei endlichen Automaten und Pushdown-Automaten haben wir untersucht, **welche Sprachen sie akzeptieren**.
- Bei Turing-Maschinen untersuchen wir,
  - welche Sprachen sie akzeptieren und
  - **welche Funktionen sie berechnen**.

**Akzeptieren ist Spezialfall von Berechnen**

## Definition 9.12 (Von einer DTM akzeptierte Sprache)

Ein Wort  $w$  wird **akzeptiert von einer DTM  $\mathcal{M}$** ,  
falls  $\mathcal{M}$  auf Eingabe von  $w$  hält  
(wobei am Ende der Kopf auf dem ersten Blank rechts von  $w$  steht).

Eine Sprache  $L \subseteq \Sigma^*$  **wird akzeptiert von einer DTM  $\mathcal{M}$** , wenn genau die  
Wörter aus  $L$  aus  $\mathcal{M}$  und keine anderen akzeptiert werden.

## Achtung

Bei nicht akzeptierten Wörtern muss die DTM nicht halten

**Sie darf es sogar nicht!**

## Funktionen auf natürlichen Zahlen

- Wir verwenden die **Unärdarstellung**  
Eine Zahl  $n$  wird auf dem Band der Maschine durch  $n$  senkrechte Striche dargestellt.
- Eine Turing-Maschine  $\mathcal{M}$  berechnet eine Funktion

$$f: \mathbb{N}^k \rightarrow \mathbb{N}^n$$

in Unärdarstellung wie folgt:

- Wenn  $f(i_1, \dots, i_k) = (j_1, \dots, j_n)$  ist, dann rechnet  $\mathcal{M}$

$$s, \#|^{i_1}\# \dots \#|^{i_k}\# \vdash_{\mathcal{M}}^* h, \#|^{j_1}\# \dots \#|^{j_n}\#$$

- Ist  $f(i_1, \dots, i_k)$  undefiniert, dann hält  $\mathcal{M}$  bei Input  $\#|^{i_1}\# \dots \#|^{i_k}\#$  nicht.

## Definition 9.13

- **TM<sup>part</sup>** ist die Menge der partiellen TM-berechenbaren Funktionen  
 $f : \mathbb{N}^k \rightarrow \mathbb{N}$
- **TM** ist die Menge der totalen TM-berechenbaren Funktionen  
 $f : \mathbb{N}^k \rightarrow \mathbb{N}$



## Achtung: Einschränkung

In der Definition von TM und  $TM^{part}$  haben wir uns eingeschränkt:

- nur Funktionen über natürliche Zahlen
- nur Funktionen mit einstelligem Wertebereich

## Das ist keine echte Einschränkung

Elemente (Wörter) aus anderen Definitions- und Wertebereiche können als natürliche Zahlen kodiert werden.

## Turing Maschinen

- 1 Determinierte Turing-Maschinen (DTMs)
- 2 Varianten von Turing-Maschinen**
- 3 Indeterminierte Turing-Maschinen (NTMs)
- 4 Universelle determinierte Turing-Maschinen
- 5 Entscheidbar/Aufzählbar
- 6 Determinierte Turing-Maschinen entsprechen Typ 0
- 7 Unentscheidbarkeit

# Variationen von Turing-Maschinen

## Standard-DTM

Die Turing-Maschine, die wir bisher kennen, ...

- ist determiniert
- hat ein einseitig unbeschränktes Band (**Halbband**).

Ab jetzt nennen wir sie auch:

**Standard-Turing-Maschine (Standard-DTM oder kurz DTM)**

## Variationen

- **zweiseitig** unbeschränktes Band (kein Hängen)
- **mehrere** Bänder
- **indeterminierte** Turing-Maschinen

## Turing-Maschinen, die nie hängen

### Gegeben:

Eine Turing-Maschine  $\mathcal{M}$ , mit Eingabe  $\#w\#$

Daraus konstruieren wir eine DTM  $\mathcal{M}'$ , die

- dasselbe berechnet wie  $\mathcal{M}$
- **nie hängt.**

## Konstruktion der TM, die nie hängt

Das Bandende ist am Anfang ein Zeichen links vom Eingabewort

DTM  $\mathcal{M}'$  rechnet so:

- Sie verschiebt die Eingabe ein Zeichen nach rechts.
- Dann druckt sie ganz links ein **Sonderzeichen  $\alpha$** , das das **Bandende** anzeigt.
- Ab dann rechnet sie wie  $\mathcal{M}$ .
- Aber:  
Wenn sie  $\alpha$  erreicht, bleibt sie dort stehen und druckt immer wieder  $\alpha$ .

## Eigenschaften der nicht-hängenden DTM

- $\mathcal{M}'$  hält für Eingabe  $w$  gdw  $\mathcal{M}$  hält für Eingabe  $w$ .
- $\mathcal{M}'$  hängt nie.  
Wenn  $\mathcal{M}$  hängt, rechnet  $\mathcal{M}'$  unendlich lang.

**O.B.d.A. sollen alle Turing-Maschinen, die wir von jetzt an betrachten, nie hängen.**

## DTM mit zweiseitig unbeschränktem Band

- Die Definition der Maschine bleibt gleich.
- Die Definition der **Konfiguration** ändert sich.
- Sie hat immer noch die Form  $q, w\underline{a}u$ , aber:
  - $w$  umfasst analog zu  $u$  alle Zeichen bis zum letzten nicht-Blank links vom Schreib-/Lesekopf.
  - $w = \varepsilon$  bzw.  $u = \varepsilon$  bedeutet, daß links bzw. rechts vom Schreib-/Lesekopf nur noch Blanks stehen.

# DTM mit zweiseitig unbeschränktem Band (zw-DTM)

## Definition 10.1 (DTM mit zweiseitig unbeschränktem Band, zw-DTM)

Eine **Turing-Maschine mit zweiseitig unbeschränktem Band (zw-DTM)** ist eine DTM, für die die Begriffe der Konfiguration und der Nachfolgekonfiguration wie folgt definiert sind:

Eine **Konfiguration**  $C$  einer zw-DTM  $\mathcal{M} = (K, \Sigma, \delta, s)$  ist von der Form

$$C = q, w\underline{a}u$$

Dabei ist

- $q \in K \cup \{h\}$  der aktuelle Zustand,
- $w \in (\Sigma - \{\#\})\Sigma^* \cup \{\varepsilon\}$  der Bandinhalt links des Kopfes,
- $a \in \Sigma$  das Zeichen unter dem Kopf, und
- $u \in \Sigma^*(\Sigma - \{\#\}) \cup \{\varepsilon\}$  der Bandinhalt rechts des Kopfes.



## Definition (Forts.)

$C_2 = q_2, w_2 \underline{a_2} u_2$  heißt **Nachfolgekonfiguration** von  $C_1 = q_1, w_1 \underline{a_1} u_1$ ,  
in Zeichen  $C_1 \vdash_{\mathcal{M}} C_2$ ,

falls es einen Übergang  $\delta(q_1, a_1) = (q_2, b)$  gibt, mit:

**Fall 1:**  $b \in \Sigma$ . Dann  $w_1 = w_2, u_1 = u_2$  und  $a_2 = b$ .

**Fall 2:**  $b = L$ . Für  $u_2$ : Wenn  $a_1 = \#$  und  $u_1 = \varepsilon$  ist, dann  $u_2 = \varepsilon$ , sonst  
 $u_2 = a_1 u_1$ .

Für  $a_2$  und  $w_2$ : Wenn  $w_1 = \varepsilon$  ist, dann  $w_2 = \varepsilon$  und  $a_2 = \#$ ; sonst  
 $w_1 = w_2 a_2$ .

**Fall 3:**  $b = R$ . Für  $w_2$ : Wenn  $a_1 = \#$  und  $w_1 = \varepsilon$  ist, dann  $w_2 = \varepsilon$ , sonst  
 $w_2 = w_1 a_1$ .

Für  $a_2$  und  $u_2$ : Wenn  $u_1 = \varepsilon$  ist, dann  $u_2 = \varepsilon$  und  $a_2 = \#$ ;  
ansonsten  $u_1 = a_2 u_2$ .

# DTM mit zweiseitig unbeschränktem Band (zw-DTM)

## Theorem 10.2 (Simulation von zw-DTM durch DTM)

*Zu jeder zw-DTM  $\mathcal{M}$ , die eine Funktion  $f$  berechnet oder eine Sprache  $L$  akzeptiert, existiert eine Standard-DTM  $\mathcal{M}'$ , die ebenfalls  $f$  berechnet bzw.  $L$  akzeptiert.*

## Beweis

Sei  $w = a_1 \dots a_n$  die Eingabe für  $M = (K, \Sigma, \delta, s)$ .

Dann sieht das beidseitig unendliche Band zu Beginn der Rechnung so aus:

$\dots \### a_1 \dots a_n \underline{\quad} \#\#\dots$

## Beweis (Forts.)

### Idee:

- $\mathcal{M}$  hat quasi zwei unendlich lange Halbbänder.
- Ziel ist, den Inhalt beider Halbbänder von  $\mathcal{M}$  auf einem unterzubringen.
- Dazu: Den Teil des Bandes, der zwei Zeichen links vom Input  $w$  beginnt, umklappen:

Spur 1 ## ...# #

Spur 2 #  $a_1$  ...  $a_n$  # ...

- Die DTM  $\mathcal{M}'$  hat zwei **Spuren**, d.h. zwei Reihen von Zeichen, die auf demselben Band untergebracht (kodiert) sind.
- Das Bandalphabet von  $\mathcal{M}'$  ist  $\Sigma' \supseteq \Sigma \times \Sigma$ .

## Beweis (Forts.)

Sei  $\mathcal{M}' = (K', \Sigma', \delta', s)$ .  $\mathcal{M}'$  rechnet so:

- $\mathcal{M}'$  legt zunächst eine zweite Spur an,
- simuliert dann die Arbeit von  $\mathcal{M}$ , und
- transformiert dann das Ergebnis wieder auf nur eine Spur herunter.

## Beweis (Forts.)

### Erste Phase der Rechnung:

$\mathcal{M}'$  rechnet

$$s, \#a_1 \dots a_n \# \vdash_{\mathcal{M}'}^* q, \$ \begin{array}{c} \#\# \dots \# \# \\ \# a_1 \dots a_n \# \end{array} \# \dots$$

Die zweite Spur wird nur so weit wie nötig angelegt.

Mit dem Symbol \$ markiert  $\mathcal{M}'$  das Ende des Halbbands, damit sie nicht hängenbleibt.

## Beweis (Forts.)

### Zweite Phase der Rechnung:

$\mathcal{M}'$  simuliert  $\mathcal{M}$ .

Dabei muß sie sich immer merken, auf welcher der beiden Spuren sie gerade arbeitet.

Deshalb definieren wir  $K' \supseteq K \times \{1, 2\}$ .

$(q, i)$  bedeutet, daß die simulierte Maschine  $\mathcal{M}$  im Zustand  $q$  ist und  $\mathcal{M}'$  auf Spur  $i$  arbeitet.

# DTM mit zweiseitig unbeschränktem Band (zw-DTM)

## Beweis (Forts.)

Für die Simulation von  $\mathcal{M}$  durch  $\mathcal{M}'$  soll nun gelten:

$\mathcal{M}$  erreicht von  $s, \# \dot{ : } \# w \#$  aus eine Konfiguration  $q, u_1 b \dot{ : } a u_2$

gdw

$\mathcal{M}'$  rechnet  $p, \# \# \dots \# \# \# \vdash_{\mathcal{M}'}^* p', \# \# \begin{matrix} b & u_1^R & \# \\ a & u_2 & \# \end{matrix} \dots \# \#$

( $\dot{ : }$  steht in beiden Konf. zwischen denselben zwei Bandpositionen (an denen das Band "umklappt"))

## Beweis (Forts.)

$\mathcal{M}'$  simuliert  $\mathcal{M}$  wie folgt:

- Wenn  $\mathcal{M}'$  das Zeichen \$ erreicht, wechselt sie die Spur.
- Wenn die simulierte Maschine  $\mathcal{M}$  nach rechts (links) geht, geht  $\mathcal{M}'$  nach rechts (links) auf Spur 2 und nach links (rechts) auf Spur 1.
- Wenn  $\mathcal{M}'$  ein # erreicht (d.h. sie erreicht den Bandteil, wo noch nicht zwei Spuren angelegt sind), macht sie daraus  $\begin{matrix} \# \\ \# \end{matrix}$ .

Gilt etwa  $\delta_{\mathcal{M}}(q, a) = (q', L)$ , so muß in  $\mathcal{M}$  gelten:

- $\delta_{\mathcal{M}'}((q, 2), \begin{matrix} x \\ a \end{matrix}) = ((q', 2), L)$  für alle möglichen  $x$ ,
- $\delta_{\mathcal{M}'}((q, 1), \begin{matrix} a \\ x \end{matrix}) = ((q', 1), R)$  (auf der oberen Spur ist der Inhalt des “linken Halbbandes” revers notiert, deshalb muß hier die Laufrichtung entgegengesetzt sein).



## Beweis (Forts.)

Außerdem gilt immer:

- $\delta_{\mathcal{M}'}((q, 1), \$) = ((q, 2), R)$
- $\delta_{\mathcal{M}'}((q, 2), \$) = ((q, 1), R)$   
Spurwechsel beim Überschreiten von \$
- $\delta_{\mathcal{M}'}((q, i), \#) = (q, i), \#)$   
Erzeugen eines neuen Doppelspurstücks
- etc.

## Beweis (Forts.)

Wenn dann  $\mathcal{M}$  mit  $h, u\#$  hält, dann erreicht  $\mathcal{M}'$  eine Konfiguration, die eine der folgenden Formen hat:

$$(i) \quad (h, 1), \$ \# \dots \overline{\#} u^R \# \dots \# \text{ oder}$$

$$(ii) \quad (h, 2), \$ \# \dots \# \# \dots \# \# \dots \# \text{ oder}$$

$$(iii) \quad (h, 2), \$ \begin{matrix} u_1^R \\ u_2\# \end{matrix} \# \dots \# \text{ mit } u_1 u_2 = u.$$

Bei Konfigurations-Form (iii) kann entweder das  $u_1^R$  über das  $u_2$  "hinausragen" oder umgekehrt.

## Beweis (Forts.)

### Dritte Phase der Rechnung:

Die Simulation von  $\mathcal{M}$  ist abgeschlossen.

$\mathcal{M}'$  muß nun den Bandinhalt von zwei Spuren auf nur eine heruntertransformieren, um danach die Konfiguration  $h, \#u\#$  zu erreichen.

## Beweis (Forts.)

- $\mathcal{M}'$  macht zunächst alle  $\#$  rechts vom beschriebenen Bandteil zu  $\#$ . Für Fall (i) und (ii) löscht sie die  $\#$  links von  $u^R$  bzw.  $u$ .
- Für Fall (iii) schiebt  $\mathcal{M}'$  dann die untere Spur nach links, bis sie eine Konfiguration  $q, \# \dots \# u_1^R u_2 \#$  erreicht.
- Für Fall (i) und (iii) muß  $\mathcal{M}'$  jetzt  $u_1^R$  bzw.  $u^R$  auf nur eine Spur transformieren und zugleich invertieren, sie muß also für den allgemeineren Fall (iii)  $q, \$ \# \dots \# u_1^R u_2 \# \vdash_{\mathcal{M}'}^* q', \$ u_1 u_2 \#$  rechnen.
- Danach muß  $\mathcal{M}'$  nur noch das  $\$$  links löschen und nach rechts neben  $u$  laufen.

# DTM mit zweiseitig unbeschränktem Band (zw-DTM)

## Beweis (Forts.)

Damit hat die Standard-DTM  $\mathcal{M}'$  die Arbeitsweise der zw-DTM  $\mathcal{M}$  vollständig simuliert.

**Also kann man mit zw-Turing-Maschinen nicht mehr berechnen als mit Standard DTM'n.**



## Definition 10.3 (DTM mit $k$ Halbbändern, $k$ -DTM)

Eine **Turing-Maschine**  $\mathcal{M} = (K, \Sigma_1, \dots, \Sigma_k, \delta, s)$  **mit  $k$  Halbbändern** (mit je einem Kopf) ist eine Turing-Maschine mit einer Übergangsfunktion

$$\delta : K \times \Sigma_1 \times \dots \times \Sigma_k \rightarrow (K \cup \{h\}) \times (\Sigma_1 \cup \{L, R\}) \times \dots \times (\Sigma_k \cup \{L, R\})$$

Eine **Konfiguration** einer  $k$ -Turing-Maschine hat die Form

$$C = q, w_1 \underline{a_1} u_1, \dots, w_k \underline{a_k} u_k.$$

## DTM mit $k$ Halbbändern

- Die Köpfe einer  $k$ -DTM können sich **unabhängig** bewegen (sonst hätten wir nur eine DTM mit  $k$  Spuren).
- Die Definition der Nachfolgekonfiguration verläuft analog zu der Definition bei Standard-DTM.
- Für eine  $k$ -DTM, die eine Funktion  $f : \Sigma_0^m \rightarrow \Sigma_0^n$  berechnet, legen wir fest, daß sowohl die  $m$  Eingabewerte als auch – nach der Rechnung – die  $n$  Ergebniswerte auf dem ersten Band stehen sollen.
- Es übertragen sich alle Begriffe wie *berechenbar*, *entscheidbar* etc. kanonisch auf  $k$ -DTM.

## Theorem 10.4 (Simulation von $k$ -DTM durch DTM)

*Zu jeder  $k$ -DTM  $\mathcal{M}$ , die eine Funktion  $f$  berechnet (resp. eine Sprache  $L$  akzeptiert), existiert eine DTM  $\mathcal{M}'$ , die ebenfalls  $f$  berechnet (resp.  $L$  akzeptiert).*



## Beweis (Skizze)

Wir arbeiten mit einer Turing-Maschine mit mehreren Spuren.

Um eine  **$k$ -DTM** zu simulieren, verwenden wir  **$2k$  Spuren**, also Bandzeichen, die aus  $2k$  übereinander angeordneten Buchstaben bestehen.

- In den Spuren mit ungerader Nummer stehen die Inhalte der  $k$  Bänder von  $\mathcal{M}$ .
- Die Spuren mit gerader Nummer verwenden wir, um die Positionen der Köpfe von  $\mathcal{M}$  zu simulieren:  
Die  $2i$ -te Spur enthält an genau einer Stelle ein  $\wedge$ , nämlich da, wo  $\mathcal{M}$  gerade seinen  $i$ -ten Kopf positioniert hätte, und ansonsten nur Blanks.

$\mathcal{M}'$  kodiert zunächst die Eingabe von  $\mathcal{M}$ . Dann simuliert  $\mathcal{M}'$  die Maschine  $\mathcal{M}$ . Am Ende der Rechnung wird noch die Ausgabe dekodiert.

## Turing Maschinen

- 1 Determinierte Turing-Maschinen (DTMs)
- 2 Varianten von Turing-Maschinen
- 3 Indeterminierte Turing-Maschinen (NTMs)**
- 4 Universelle determinierte Turing-Maschinen
- 5 Entscheidbar/Aufzählbar
- 6 Determinierte Turing-Maschinen entsprechen Typ 0
- 7 Unentscheidbarkeit

## Definition 11.1 (Indeterminierte Turing-Maschine, NTM)

Eine **indeterminierte Turing-Maschine**  $\mathcal{M}$  ist ein Tupel

$$M = (K, \Sigma, \Delta, s)$$

Dabei sind  $K$ ,  $\Sigma$ ,  $s$  definiert wie bei determinierten Turing-Maschinen.

Übergangs**relation**:

$$\Delta \subseteq (K \times \Sigma) \times ((K \cup \{h\}) \times (\Sigma \cup \{L, R\}))$$

## Mehrere Nachfolgekongfigurationen

**Kongfigurationen** sind definiert wie bei DTMs.

**Nun kann eine Kongfiguration aber mehrere mögliche Nachfolgekongfigurationen haben.**

## Definition 11.2 (NTM: Halten, Hängen, Akzeptieren)

Sei  $\mathcal{M} = (K, \Sigma, \Delta, s_0)$  eine indeterminierte Turing-Maschine.

- $\mathcal{M}$  **hält** bei Input  $w$ , falls es **unter den möglichen Rechnungen**, die  $\mathcal{M}$  wählen kann, **eine gibt**, so daß  $\mathcal{M}$  eine Haltekonfiguration erreicht.
- $\mathcal{M}$  **hängt** in einer Konfiguration, wenn es keine (durch  $\Delta$  definierte) Nachfolgekonfiguration gibt.
- $\mathcal{M}$  **akzeptiert** ein Wort  $w$ , falls sie **von  $s, \#w\#$  aus einen Haltezustand erreichen kann**, und  $\mathcal{M}$  akzeptiert eine Sprache  $L$ , wenn sie genau alle Wörter  $w \in L$  akzeptiert.

## Bemerkung

Wenn es nicht nur darauf ankommt, ob die Maschine hält, sondern auch mit welchem Bandinhalt:

**Welche der vielen Haltekonfigurationen sollte dann gelten?**

Um dies Problem zu umgehen, übertragen wir die Begriffe des *Entscheidens* und *Aufzählens* **nicht** auf NTM. Im Allgemeinen verwendet man NTM auch nicht dazu, Funktionen zu berechnen.

## Wie rechnet eine indeterminierte Turing-Maschine?

- Die Regeln einer determinierten DTM kann man sich als Programm (aus sehr einfachen Schritten) vorstellen.
- **Bei NTM ist das anders!**
- **Eine NTM ist nicht einfach eine Maschine, die immer richtig rät!**

Dieselbe Diskussion hatten wir bei indeterminierten endlichen Automaten.

## Vorstellung von einer NTM

- Korrekte Vorstellung:
  - Übergänge von Konfiguration zu Nachfolgekonfiguration entspr. Regelmenge
  - **plus Suchverfahren!**oder: Eine NTM beschreitet alle möglichen Rechenwege parallel.
- **Eine NTM akzeptiert ein Wort, wenn es mindestens einen Berechnungsweg gibt, der in einer Haltekonfiguration endet.**
- Sprechweise **“Die NTM rät”**: Wir verwenden diese Sprechweise, sie ist aber mit Vorsicht zu genießen!



## Beispiel 11.3

Eine indeterminierte Turing-Maschine, die

$$L = \{w \in \{a, b\}^* \mid w \text{ besitzt } aba \text{ als Teilwort}\}$$

akzeptiert.

Siehe Tafel.

## Beispiel 11.4

Sei

$$L = \{|^n \mid n \text{ ist nicht prim und } n \geq 2\}$$

Eine NTM kann diese Sprache wie folgt akzeptieren:

- 1 Eine Zahl „raten“ und (nach rechts) aufs Band schreiben.
- 2 Noch eine Zahl „raten“ und daneben schreiben.
- 3 Die beiden Zahlen miteinander multiplizieren.
- 4 Das Ergebnis mit der Eingabe vergleichen.
- 5 Genau dann, wenn beide gleich sind, anhalten

## Theorem 11.5 (Simulation von NTM durch DTM)

*Jede Sprache, die von einer indeterminierten Turing-Maschine akzeptiert wird, wird auch von einer Standard-DTM akzeptiert.*

## Beweis (Anfang)

Sei

- $L$  eine Sprache über  $\Sigma_0^*$  mit  $\# \notin \Sigma_0$ ;
- $\mathcal{M} = (K, \Sigma, \Delta, s)$  eine indeterminierte Turing-Maschine, die  $L$  akzeptiert.

## Beweis (Fortsetzung)

Wir konstruieren zu  $\mathcal{M}$  eine Standard-DTM  $\mathcal{M}'$ , die so rechnet:

- $\mathcal{M}'$  durchläuft systematisch **alle** Rechnungen von  $\mathcal{M}$ , und sucht dabei nach einer Haltekonfiguration.
- $\mathcal{M}'$  genau dann, wenn sie eine Haltekonfiguration von  $\mathcal{M}$  findet.

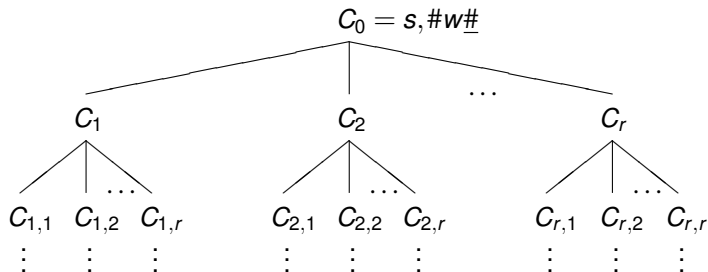
# Indeterminierte Turing-Maschine

## Beweis (Fortsetzung)

### Suchbaum, Rechnungsbaum:

Stelle alle Rechnungen von  $\mathcal{M}$  von einer Startkonfiguration  $C_0$  dar als einen Baum mit Wurzel  $C_0$ .

Ein Ast ist eine mögliche Rechnung von  $\mathcal{M}$ .



## Beweis (Fortsetzung)

### Problem:

Es kann zur Startkonfiguration

$$C_0 = s, \# \underline{w} \#$$

unendlich viele Rechnungen von  $\mathcal{M}$  geben,  
und jede einzelne von ihnen kann unendlich lang sein.

Wir können also nicht erst einen Ast ganz durchlaufen und dann den nächsten Ast durchsuchen.

## Beweis (Fortsetzung)

### Die Lösung:

Breitensuche

Durchlaufe den Rechnungsbaum nicht depth-first, sondern per **iterative deepening**.

- Untersuchen alle möglichen Rechnungen bis zum ersten Schritt.
- Untersuchen alle möglichen Rechnungen bis zum zweiten Schritt.
- Untersuchen alle möglichen Rechnungen bis zum dritten Schritt.
- usw.

## Beweis (Fortsetzung)

Können wir damit denn in endlicher Zeit eine Haltekonfiguration finden, falls es eine gibt?

### Problem:

Kann der Rechnungsbaum nicht nur **unendlich tief**, sondern auch **unendlich breit** werden?

**Nein**, denn:

Maximale Anzahl von Nachfolgekfigurationen

$$r = \max\{|\Delta(q, a)| \mid q \in K, a \in \Sigma\}$$



## Beweis (Fortsetzung)

$\mathcal{M}'$  kann (z.B.) als eine 3-DTM gewählt werden:

- **Auf dem ersten Band steht immer das Eingabewort  $w$ .**  
Da die Rechnung immer wieder neu mit  $s, \#w\#$  von  $\mathcal{M}$  beginnt, wird das Eingabewort immer wieder gebraucht.
- **Auf dem zweiten Band steht, welcher Weg durch den Rechnungsbaum gerade verfolgt wird.**  
Der Einfachheit halber: Wenn eine Konfiguration weniger als  $r$  Nachfolgekonfigurationen hat, soll der zugehörige Knoten trotzdem  $r$  Söhne haben, und die überzähligen Konfigurationen sind leer.

## Beweis (Fortsetzung)

Darstellung des aktuellen Pfades im Rechnungsbaum als Zahl im  $r$ -adischen System.

Eine Zahl  $d_1 \dots d_n$  bedeutet:

- Von der Startkonfiguration  $C_0$  aus ist die  $d_1$ -te der  $r$  möglichen Nachfolgekonfigurationen gewählt worden,  $C_{d_1}$ .
- Von  $C_{d_1}$ , einem Knoten der Tiefe 1, aus wurde die  $d_2$ -te mögliche Nachfolgekonfiguration gewählt,
- usw.

## Beweis (Fortsetzung)

Ausführung des Iterative Deepening:

- Beginne mit 0 auf zweitem Band.
- Jeweils nächste zu betrachtende Rechnung erhöhen der Zahl auf Band 2 um 1
- Auf Band 3 wird eine Rechnung von  $\mathcal{M}$  **determiniert** simuliert. Und zwar entsprechend der Zahl  $d_1 \dots d_n$  auf Band 2. Die Endkonfiguration  $C_{d_1 \dots d_n}$  dieser Rechnung steht im Rechnungsbaum an dem Knoten, der das Ende des Pfades  $d_1 \dots d_n$  bildet.
- Ist die Konfiguration  $C_{d_1 \dots d_n}$  eine Haltekonfiguration, so hält  $\mathcal{M}'$ .
- Sonst Zahl auf Band 2 erhöhen und die nächste Rechnungssimulation beginnen

## Beweis (Ende)

Damit gilt:

$\mathcal{M}'$  hält bei Input  $w$

gdw

es gibt in  $R_{C_0}$  eine Haltekonfiguration.

Das ist genau dann der Fall, wenn

- $\mathcal{M}$  bei Input  $w$  hält,
- $w$  in  $L$  liegt.



## Turing Maschinen

- 1 Determinierte Turing-Maschinen (DTMs)
- 2 Varianten von Turing-Maschinen
- 3 Indeterminierte Turing-Maschinen (NTMs)
- 4 Universelle determinierte Turing-Maschinen**
- 5 Entscheidbar/Aufzählbar
- 6 Determinierte Turing-Maschinen entsprechen Typ 0
- 7 Unentscheidbarkeit

## Vergleich Turing-Maschine / „normaler“ Computer

Turing-Maschinen sind sehr mächtig.

**Wie mächtig sind sie wirklich?**

- Eine Turing-Maschine hat eine vorgegebenes „Programm“ (Regelmenge)
- „Normale“ Computer können beliebige Programme ausführen.

**Tatsächlich geht das mit Turing-Maschinen auch!**

## Turing-Maschine, die andere TMen simuliert

- Universelle TM  $\mathcal{U}$  bekommt als Eingabe:
  - die Regelmenge einer beliebigen Turing-Maschine  $\mathcal{M}$  und
  - ein Wort  $w$ , auf dem  $\mathcal{M}$  rechnen soll.
- $\mathcal{U}$  simuliert  $\mathcal{M}$ , indem sie jeweils nachschlägt, welchen  $\delta$ -Übergang  $\mathcal{M}$  machen würde.

## TM als Eingabe für eine andere TM

### Frage:

In welches Format fasst man die Regeln einer DTM  $\mathcal{M}$  am besten, um sie einer universellen DTM als Eingabe zu geben?

### Was muss man angeben, um eine DTM komplett zu beschreiben?

- das Alphabet,
- die Zustände,
- die  $\delta$ -Übergänge
- den Startzustand.



## Standardisierung von Alphabet, Zustandsmenge, Startzustand

- Unendliches Alphabet  $\Sigma_\infty = \{a_0, a_1, \dots\}$ ,  
so daß das Alphabet jeder DTM eine Teilmenge von  $\Sigma_\infty$  ist.
- Namen der Zustände einer DTM sind egal.  
Sie seien also  $q_1, \dots, q_n$   
( $n$  kann dabei von DTM zu DTM verschieden sein).
- Sei  $q_1$  immer der Startzustand, und  
bezeichne  $q_0$  den Haltezustand

Damit:

**Wir können eine DTM komplett beschreiben,  
indem wir nur ihre  $\delta$ -Übergänge beschreiben.**

## (Mögliche) Kodierung der Übergangsrelation

Die DTM  $\mathcal{L}_\#$  habe die Regeln

$$\begin{aligned} q_1, \# &\mapsto q_2, L & q_2, \# &\mapsto h, \# \\ q_1, | &\mapsto q_2, L & q_2, | &\mapsto q_2, L \end{aligned}$$

Dabei sei:  $\# = a_0$  und  $| = a_1$

Dann kann die DTM  $\mathcal{L}_\#$  so beschrieben werden:

	$a_0$	$a_1$	
$q_1$	$q_2, L$	$q_2, L$	oder kürzer:
$q_2$	$h, a_0$	$q_2, L$	$Z S 2\lambda S 2\lambda$ $Z S 00 S 2\lambda$

## (Mögliche) Kodierung der Übergangsrelation

Dabei steht:

- $Z$  für “nächste Zeile”
- $S$  für “nächste Spalte”
- $\lambda$  für “links”,  $\rho$  für “rechts”
- die Zahl  $n$  für den  $n$ -ten Zustand und für das  $n$ -te Zeichen von  $\Sigma_\infty$ .

Damit ist die DTM insgesamt durch ein einziges Wort beschrieben:

$ZS2\lambda S2\lambda ZS00S2\lambda$

## Gödelisierung

Ein Verfahren, jeder Turing-Maschine eine Zahl oder ein Wort (**Gödelzahl** bzw. **Gödelwort**) so zuzuordnen, daß man aus der Zahl bzw. dem Wort die Turing-Maschine effektiv rekonstruieren kann.

# Kurt Gödel

Kurt Gödel ★ 1906, † 1978

- Bedeutendster Logiker des 20. Jahrhunderts
- Vollständigkeitssatz (1929)  
Promotion in Wien
- Unvollständigkeitssatz (1931)  
Idee der Gödelisierung
- Beweis der Unabhängigkeit der  
Kontinuumshypothese
- Dozent in Princeton,  
befreundet mit Albert Einstein
- Tragischer Tod:  
Verfolgungswahn, Depressionen,  
Tod durch Unterernährung.



## Turing Maschinen

- 1 Determinierte Turing-Maschinen (DTMs)
- 2 Varianten von Turing-Maschinen
- 3 Indeterminierte Turing-Maschinen (NTMs)
- 4 Universelle determinierte Turing-Maschinen
- 5 Entscheidbar/Aufzählbar**
- 6 Determinierte Turing-Maschinen entsprechen Typ 0
- 7 Unentscheidbarkeit

## Akzeptieren

Eine DTM **akzeptiert** eine Sprache  $L$ , wenn sie

- für jedes Eingabe-Wort  $w \in L$  irgendwann hält
- für jedes Wort  $v \notin L$  unendlich lang rechnet oder hängt

## Entscheiden

Eine DTM **entscheidet** eine Sprache  $L$ , wenn sie

- für jedes Eingabe-Wort  $w \in L$  hält mit dem Bandinhalt  $Y$  ("Yes")
- für jedes Wort  $v \notin L$  hält mit dem Bandinhalt  $N$  ("No")

## Definition 13.1 (Entscheidbar)

$L$  sei eine Sprache über  $\Sigma_0$  mit  $\#, N, Y \notin \Sigma_0$ .

$M = (K, \Sigma, \delta, s)$  sei eine DTM mit  $\Sigma_0 \subseteq \Sigma$ .

$M$  **entscheidet**  $L$ , falls für alle  $w \in \Sigma_0^*$  gilt:

$$s, \#w\# \vdash_M^* \begin{cases} h, \#Y\# & \text{falls } w \in L \\ h, \#N\# & \text{sonst} \end{cases}$$

$L$  heißt **entscheidbar**, falls es eine DTM gibt, die  $L$  entscheidet.



## Definition 13.2 (Akzeptierbar)

$L$  sei eine Sprache über  $\Sigma_0$  mit  $\#, N, Y \notin \Sigma_0$ .

$M = (K, \Sigma, \delta, s)$  sei eine DTM mit  $\Sigma_0 \subseteq \Sigma$ .

$\mathcal{M}$  **akzeptiert** ein Wort  $w \in \Sigma_0^*$ ,  
falls  $\mathcal{M}$  bei Input  $w$  hält.

$\mathcal{M}$  **akzeptiert die Sprache  $L$** , falls für alle  $w \in \Sigma_0^*$  gilt:

$\mathcal{M}$  akzeptiert  $w$  *genau dann wenn*  $w \in L$

$L$  heißt **akzeptierbar** (oder auch **semi-entscheidbar**),  
falls es eine DTM gibt, die  $L$  akzeptiert.

## Definition 13.3 (Rekursiv Aufzählbar (recursively enumerable))

$L$  sei eine Sprache über  $\Sigma_0$  mit  $\#, N, Y \notin \Sigma_0$ .

$M = (K, \Sigma, \delta, s)$  sei eine DTM mit  $\Sigma_0 \subseteq \Sigma$ .

$\mathcal{M}$  **zählt  $L$  auf**, falls es einen Zustand  $q_B \in K$  gibt (den **Blinkzustand**), so daß:

$$L = \{w \in \Sigma_0^* \mid \exists u \in \Sigma^* : s, \# \vdash_{\mathcal{M}}^* q_B, \#w\#u\}$$

$L$  heißt **rekursiv aufzählbar**, falls es eine DTM gibt, die  $L$  aufzählt.

**Achtung:** aufzählbar  $\neq$  abzählbar.

## Unterschied

**$M$  abzählbar:** Es gibt eine surjektive Abbildung der natürlichen Zahlen auf  $M$

**$M$  aufzählbar:** Diese Abbildung kann von einer Turing-Maschine berechnet werden.

Wegen Endlichkeit der Wörter und des Alphabets sind alle Sprachen abzählbar.

Aber nicht alle Sprachen sind aufzählbar.

## Beispiel 13.4 (Rekursiv aufzählbar aber nicht entscheidbar)

Folgende Mengen sind rekursiv aufzählbar aber *nicht* entscheidbar:

- Die Menge der Gödelisierungen aller haltenden Turing-Maschinen
- Die Menge aller terminierenden Programme
- Die Menge aller allgemeingültigen prädikatenlogischen Formeln

## Satz 13.5 (Akzeptierbar = Rekursiv Aufzählbar)

*Eine Sprache ist genau dann rekursiv aufzählbar, wenn sie akzeptierbar ist.*

## Beweis

“ $\Rightarrow$ ”

Sei  $L$  rekursiv aufzählbar

Es gibt also eine DTM  $\mathcal{M}_L$ , die  $L$  aufzählt.

Zu zeigen:  $L$  ist akzeptierbar.

## Beweis (Fortsetzung)

Wir konstruieren aus  $\mathcal{M}_L$  eine 2-DTM  $\mathcal{M}$ , die  $L$  akzeptiert:

- $\mathcal{M}$  wird gestartet mit

$$s_0, \quad \#w\underline{\#}$$
$$\quad \underline{\#}$$

- $\mathcal{M}$  simuliert auf Band 2 die Maschine  $\mathcal{M}_L$ .
- Wenn  $\mathcal{M}_L$  den **Blinkzustand**  $q_B$  erreicht, dann enthält Band 2 von  $\mathcal{M}$  ein Wort

$$\#w'\underline{\#}u$$

mit  $w' \in L$ .

## Beweis (Fortsetzung)

- Nach Erreichen des Blinkzustands:  
 $\mathcal{M}$  vergleicht  $w$  und  $w'$ .
  - Falls  $w = w'$ , dann hält  $\mathcal{M}$ :  $w \in L$ .
  - Ansonsten simuliert  $\mathcal{M}$  auf Band 2 weiter die Arbeit von  $\mathcal{M}_L$ .
- Wenn  $\mathcal{M}_L$  hält, ohne das Wort  $w$  auf Band 2 erzeugt zu haben, gerät  $\mathcal{M}$  in eine Endlosschleife.



## Beweis (Fortsetzung)

“ $\Leftarrow$ ”

Sei  $L$  akzeptierbar

Es gebe also eine DTM  $\mathcal{M}_L$ , die  $L$  akzeptiert.

Zu zeigen:  $L$  ist rekursiv aufzählbar.

Wir konstruieren eine DTM  $\mathcal{M}$ , die  $L$  rekursiv aufzählt.

Grundidee:

- die Wörter aus  $\Sigma^*$  der Reihe nach aufzählen
- jedes Wort der Maschine  $\mathcal{M}_L$  vorlegen
- wenn  $\mathcal{M}_L$  das Wort akzeptiert, in den Blinkzustand  $q_B$  gehen

## Beweis (Fortsetzung)

**Problem:**  $\mathcal{M}_L$  akzeptiert, sie entscheidet nicht.

Wenn  $\mathcal{M}_L$  ein Wort nicht akzeptiert, rechnet sie unendlich.

**Lösung:** Wir betrachten die Rechnung von  $\mathcal{M}_L$  zu allen Wörtern aus  $\Sigma^*$   
**gleichzeitig.**

## Beweis (Fortsetzung)

$\Sigma^* = \{w_1, w_2, w_3, \dots\}$  (in lexikalischer Reihenfolge aufgezählt).

Dann rechnet  $\mathcal{M}$  so:

- Im ersten Durchlauf berechne den ersten Rechenschritt von  $\mathcal{M}_L$  für  $w_1$ .
- Im zweiten Durchlauf berechne
  - die ersten zwei Rechenschritte von  $\mathcal{M}_L$  für  $w_1$ ,
  - einen Rechenschritt für  $w_2$ .
- Im dritten Durchlauf berechne drei Rechenschritte für  $w_1$ , zwei für  $w_2$  und einen für  $w_3$  und so weiter.

Immer wieder bei der Startkonfiguration anfangen: So müssen wir uns den Bandinhalt der Rechnung von  $\mathcal{M}_L$  zu  $w_i$  nach  $j$  Schritten nicht merken.

## Beweis (Fortsetzung)

Wenn  $\mathcal{M}$  so rechnet, dann gilt:

- $\mathcal{M}$  fängt für jedes  $w_i \in \Sigma^*$  in endlicher Zeit (nämlich im  $i$ -ten Durchlauf) an, die Arbeit von  $\mathcal{M}_L$  zu  $w_i$  zu simulieren, und
- falls  $w_i \in L$  und falls  $\mathcal{M}_L$ , gestartet mit  $s, \#w_i\#$ , in  $j$  Schritten einen Haltezustand erreicht, dann erreicht  $\mathcal{M}$  nach endlicher Zeit (nämlich im  $i + j$ -ten Durchlauf) den Haltezustand von  $\mathcal{M}_L$  in der Rechnung zu  $w_i$ .

## Beweis (Ende)

- Wenn  $\mathcal{M}$  bei Simulation von  $\mathcal{M}_L$  zur Eingaben  $w_i$  auf eine Haltekonfiguration trifft, dann ist  $w_i \in L$ .
- $\mathcal{M}$  nimmt dann eine Konfiguration

$$q_B, \#w_i\#u_i$$

ein –  $q_B$  ist der Blinkzustand.

- In der Nebenrechnung  $u_i$  steht, welche Teilrechnung von  $\mathcal{M}_L$  als nächste zu simulieren ist.

Also zählt  $\mathcal{M}$  die  $w \in \Sigma^*$  auf, für die  $\mathcal{M}_L$  hält, und das sind gerade die  $w \in L$ . □

## Satz 13.6 (Entscheidbar und akzeptierbar)

*Jede entscheidbare Sprache ist akzeptierbar.*

## Beweis

Sei  $L$  eine entscheidbare Sprache und  $\mathcal{M}$  eine DTM, die  $L$  entscheidet.

Dann wird  $L$  akzeptiert von der DTM  $\mathcal{M}'$ ,  
die zunächst  $\mathcal{M}$  simuliert und danach in eine Endlosschleife geht, falls  $\mathcal{M}$  mit  $h, \#N\#$  endet.

## Satz 13.7 (Komplement einer entscheidbaren Sprache ist entscheidbar)

*Das Komplement einer entscheidbaren Sprache ist entscheidbar.*

### Beweis

Sei  $L$  eine entscheidbare Sprache und  $\mathcal{M}$  eine DTM, die  $L$  entscheidet.

Dann wird  $\bar{L}$  entschieden von einer DTM  $\mathcal{M}'$ ,

die genau wie  $\mathcal{M}$  rechnet

und nur am Schluß die Antworten  $Y$  und  $N$  vertauscht. □

## Satz 13.8 (Charakterisierung von Entscheidbarkeit)

*Eine Sprache  $L$  ist genau dann entscheidbar, wenn sie und ihr Komplement akzeptierbar sind.*



## Beweis

“ $\Rightarrow$ ”

Sei  $L$  ist entscheidbar.

Zu zeigen:  $L$  und  $\bar{L}$  sind akzeptierbar.

- $L$  ist entscheidbar, also ist  $L$  akzeptierbar
- $L$  ist entscheidbar, also ist  $\bar{L}$  entscheidbar
- $\bar{L}$  ist entscheidbar, also ist  $\bar{L}$  akzeptierbar

## Beweis (Fortsetzung)

“ $\Leftarrow$ ”

Seien  $L$  und  $\bar{L}$  akzeptierbar.

Zu zeigen:  $L$  ist entscheidbar.

- Sei  $\mathcal{M}_1$  eine DTM, die  $L$  akzeptiert.
- Sei  $\mathcal{M}_2$  eine DTM, die  $\bar{L}$  akzeptiert.

Daraus konstruieren wir eine 2-DTM  $\mathcal{M}$ , die  $L$  entscheidet:

- $\mathcal{M}$  wird gestartet mit

$$\begin{array}{l} s_0, \#w\# \\ \# \end{array}$$

- $\mathcal{M}$  kopiert  $w$  auf Band 2.

## Beweis (Ende)

- $M$  simuliert abwechselnd
  - einen Schritt von  $\mathcal{M}_1$  auf Band 1 und
  - einen Schritt von  $\mathcal{M}_2$  auf Band 2.
- Das tut  $\mathcal{M}$ , bis entweder  $\mathcal{M}_1$  oder  $\mathcal{M}_2$  hält.
- Eine von beiden muss halten:  $w$  gehört entweder zu  $L$  oder zu  $\bar{L}$ .
- Wenn  $\mathcal{M}_1$  hält, dann hält  $\mathcal{M}$  mit
  - $\#Y\#$  auf Band 1 und
  - $\#$  auf Band 2.
- Wenn  $\mathcal{M}_2$  hält, dann hält  $\mathcal{M}$  mit
  - $\#N\#$  auf Band 1 und
  - $\#$  auf Band 2.



## Turing Maschinen

- 1 Determinierte Turing-Maschinen (DTMs)
- 2 Varianten von Turing-Maschinen
- 3 Indeterminierte Turing-Maschinen (NTMs)
- 4 Universelle determinierte Turing-Maschinen
- 5 Entscheidbar/Aufzählbar
- 6 Determinierte Turing-Maschinen entsprechen Typ 0**
- 7 Unentscheidbarkeit

## Zur Erinnerung

Formale Sprachen sind vom **Typ 0**, wenn sie durch beliebige Grammatiken (keinerlei Einschränkungen) erzeugt werden können.

## Satz 14.1 (Rekursiv aufzählbar = Typ 0)

*Die rekursiv aufzählbaren Sprachen  
(also die durch DTMn akzeptierbaren Sprachen)  
sind genau die durch beliebige Grammatiken erzeugten Sprachen  
(also die vom Typ 0).*

## Beweisidee

- Zu jeder Turing-Maschine kann eine Grammatik konstruiert werden, deren Ableitungsschritte die Rechenschritte der TM simulieren (spezielle Variable markiert Position des Schreib-/Lesekopfes).
- Zu jeder Grammatik kann eine indeterminierte Turing-Maschine (und damit auch eine DTM) konstruiert werden, deren Rechenschritte den Ableitungsschritten der Grammatik entsprechen.

## Turing Maschinen

- 1 Determinierte Turing-Maschinen (DTMs)
- 2 Varianten von Turing-Maschinen
- 3 Indeterminierte Turing-Maschinen (NTMs)
- 4 Universelle determinierte Turing-Maschinen
- 5 Entscheidbar/Aufzählbar
- 6 Determinierte Turing-Maschinen entsprechen Typ 0
- 7 Unentscheidbarkeit**

# Beispiel einer nicht berechenbaren Funktion: Busy Beaver

## Definition 15.1 (Busy Beaver)

Die Funktion  $BB : \mathbb{N} \rightarrow \mathbb{N}$  sei wie folgt definiert

$n \mapsto f(n) :=$  die maximale Anzahl an Einsen, die eine **haltende** DTM mit maximal  $n$  Zuständen auf einem leeren Band erzeugen kann



# Beispiel einer nicht berechenbaren Funktion: Busy Beaver

## **BB wächst extrem schnell**

Exakte Werte von  $BB(n)$  für  $n \geq 4$  nicht bekannt.

- $BB(4) \geq 4098$
- $BB(5) \geq 1,29 * 10^{865}$

## **Theorem 15.2 (BB ist nicht berechenbar)**

*BB wächst zu stark um berechenbar zu sein:*

*Es gibt keine DTM, die BB berechnet.*

# Beispiel einer nicht berechenbaren Funktion: Busy Beaver

## Beweis (erster Teil)

Man kann immer mindestens ein  $|$  mehr erzeugen, wenn man einen weiteren Zustand zur Verfügung hat:

- man benennt den Haltezustand um in  $q_{neu}$  und geht in den richtigen Haltezustand  $h$  nur, wenn man in  $q_{neu}$  ein Blank  $\#$  gelesen hat. Zusätzlich ersetzt man das Blank durch  $|$ ).
- Wenn man ein  $|$  liest, geht man nach rechts und bleibt in  $q_{neu}$ .

Damit haben wir bewiesen:

***BB wächst streng monoton.***

# Beispiel einer nicht berechenbaren Funktion: Busy Beaver

## Beweis (zweiter Teil)

Angenommen, es gäbe eine DTM  $\mathcal{BB}$ , die  $BB$  berechnet.  
Sie habe  $n_0$  Zustände.

Wir betrachten folgende zusammengesetzten Maschine:

- zuerst schreibt sie  $m$  Einsen auf das leere Band
- dann führt sie  $\mathcal{BB}$  aus

Diese Maschine kommt mit  $n_0 + \lfloor \frac{m}{2} \rfloor + 10$  Zuständen aus

Sei nun  $m$  so groß, dass

$$m > n_0 + \lfloor \frac{m}{2} \rfloor + 10$$

Dann schreibt die neue Maschine  $BB(m)$  Einsen auf das Band und arbeitet mit streng weniger als  $m$  Zuständen: Widerspruch. □

# Beispiel einer nicht berechenbaren Funktion: Busy Beaver

## Intuitive Ursache des Widerspruchs

$\mathcal{BB}$  selbst hat fixe Größe,

Sie muss mehr Einsen schreiben können als jede Maschine ihrer Größe.

Also muss sie insbesondere mehr Einsen schreiben als sie selbst

Widerspruch

## Definition 15.3 (Gödelnummern von DTMs)

DTMn werden als Gödelzahlen kodiert:

- DTMs können als Gödelwörter dargestellt werden.
- Die Buchstaben der Gödelwörter können in Ziffern kodiert wrden, um Gödelnummern zu bekommen.
- Notation:  $\hat{g}(\mathcal{M})$  für die Gödelnummer der DTM  $\mathcal{M}$ .

## Definition 15.4 (Jede Zahl ist Gödelnummer)

Jede natürliche Zahl  $n$  soll Gödelnummer einer DTM  $\mathcal{M}_n$  sein.

Wir definieren

$$\mathcal{M}_n := \begin{cases} \mathcal{M}, & \text{falls } \hat{g}(\mathcal{M}) = n \\ \mathcal{M}_{halt} & \text{falls } \nexists \mathcal{M} \hat{g}(\mathcal{M}) = n \end{cases}$$

$\mathcal{M}_{halt}$  ist eine TM, die sofort anhält und weiter nichts tut.

## Definition 15.5 (Allgemeines Halteproblem)

Das **allgemeine Halteproblem** ist die Frage, ob die DTM mit Gödelnummer  $n$  bei Eingabe  $i$  hält.

Es entspricht der Sprache

$$\mathcal{H}_{allg} := \{ \langle n, i \rangle \mid \mathcal{M}_n \text{ hält bei Eingabe } i \}.$$

## Definition 15.6 (Spezielles Halteproblem)

Das **spezielle Halteproblem** ist die Frage, ob die DTM mit Gödelnummer  $n$  bei Eingabe  $n$  hält.

Es entspricht der Sprache

$$\mathcal{H} := \{n \mid \mathcal{M}_n \text{ hält bei Eingabe } n\}.$$



## Definition 15.7 (Null-Halteproblem)

Das **Null-Halteproblem** ist die Frage, ob die DTM mit Gödelnummer  $n$  bei leerer Eingabe hält.

Es entspricht der Sprache

$$\mathcal{H}_0 := \{n \mid \mathcal{M}_n \text{ hält bei leerer Eingabe}\}$$

**Manchmal auch:**

“bei Eingabe 0” anstatt “bei leerer Eingabe”.

## Definition 15.8 (Leerheitsproblem)

Das **Leerheitsproblem** ist die Frage, ob die DTM mit Gödelnummer  $n$  bei **keiner** Eingabe aus  $\Sigma^*$  hält.

Es entspricht der Sprache

$$\mathcal{E} := \{n \mid \mathcal{M}_n \text{ hält bei keiner Eingabe aus } \Sigma^*\}.$$

## Definition 15.9 (Totalitätsproblem)

Das **Totalitätsproblem** ist die Frage, ob die DTM mit Gödelnummer  $n$  bei **jeder** Eingabe aus  $\Sigma^*$  hält.

Es entspricht der Sprache

$$\mathcal{T} := \{n \mid \mathcal{M}_n \text{ hält bei jeder Eingabe aus } \Sigma^*\}.$$

## Definition 15.10 (Gleichheitsproblem)

Das **Gleichheitsproblem** ist die Frage, ob die DTM mit Gödelnummer  $n$  die gleiche Sprache über  $\Sigma$  akzeptiert wie die DTM mit Gödelnummer  $m$ .

Es entspricht der Sprache

$$Eq := \{ \langle n, m \rangle \mid \mathcal{M}_n \text{ akzeptiert die gleiche Sprache über } \Sigma \text{ wie } \mathcal{M}_m \}.$$

## Definition 15.11 (Entscheidbarkeitsproblem)

Das **Entscheidbarkeitsproblem** ist die Frage, ob die DTM mit Gödelnummer  $n$  eine entscheidbare Sprache über  $\Sigma$  akzeptiert.

Es entspricht der Sprache

$$\mathcal{E}nt := \{n \mid \mathcal{M}_n \text{ akzeptiert eine} \\ \text{entscheidbare Sprache über } \Sigma\}.$$

## Satz 15.12 (Halteproblem ist unentscheidbar)

*Das spezielle Halteproblem  $\mathcal{H}$  ist unentscheidbar.*

## Beweis (A. Turing)

Beweis durch Widerspruch mit einem **Diagonalisierungsargument**.

Angenommen, es gebe eine DTM  $\mathcal{H}$ ,  
die das spezielle Halteproblem entscheidet.

Konstruiere eine neue Maschine aus  $\mathcal{H}$ :

- Wenn  $\mathcal{H}$  „Y“ antwortet,  
geht sie in eine Endlosschleife (terminiert nicht).
- Wenn  $\mathcal{M}_n$  „N“ antwortet,  
terminiert sie.

Die neue Maschine habe Gödelnummer  $n$ .

## Beweis (A. Turing), Forts.

Was macht  $\mathcal{M}_n$  bei Eingabe  $n$ ?

- Falls  $\mathcal{M}_n$  bei Eingabe  $n$  terminiert, dann antwortet  $\mathcal{H}$  auf Eingabe  $n$  mit „N“, dann terminiert  $\mathcal{M}_n$  auf Eingabe von  $n$  **nicht**  
Widerspruch!
- Falls  $\mathcal{M}_n$  bei Eingabe  $n$  nicht terminiert, dann antwortet  $\mathcal{H}$  auf Eingabe  $n$  mit „Y“, dann terminiert  $\mathcal{M}_n$  auf Eingabe von  $n$   
Widerspruch!



## Satz 15.13 (Akzeptierbarkeit von $\mathcal{H}$ )

Das spezielle Halteproblem  $\mathcal{H}$  ist aufzählbar. D. h. die Sprache

$$\mathcal{H} = \{n \mid \mathcal{M}_n \text{ hält bei Eingabe } n\}$$

ist akzeptierbar.

## Beweis

Akzeptieren durch Simulation von  $\mathcal{M}_n$  mit Hilfe der universellen DTM.

## Korollar

Das Komplement von  $\mathcal{H}$  ist nicht aufzählbar.

## Wie zeigt man, daß ein Problem unentscheidbar ist?

### Reduktion (informell)

Wir geben eine **totale, berechenbare Funktion**  $f$  an, die

- eine Instanz  $p_1$  von  $P_1$
- in eine Instanz  $p_2$  von  $P_2$  umwandelt,
- und zwar so, daß die Antwort zu  $p_1$  „ja“ ist gdw die Antwort zu  $p_2$  „ja“ ist.

## Definition 15.14 (Reduktion)

Seien  $L_1, L_2$  Sprachen über  $\mathbb{N}$ .

$L_1$  wird auf  $L_2$  reduziert,

$$L_1 \preceq L_2$$

gdw

es gibt eine TM-berechenbare Funktion  $f : \mathbb{N} \rightarrow \mathbb{N}$ , so daß gilt:

$$\forall n \in \mathbb{N} \quad (n \in L_1 \text{ gdw } f(n) \in L_2).$$

## Lemma 15.15

Ist  $L_1 \preceq L_2$ , und ist  $L_1$  **unentscheidbar**, so ist auch  $L_2$  **unentscheidbar**.

## Beweis.

- Angenommen,  $L_2$  ist entscheidbar.
- Sei  $\mathcal{M}_2$  eine Turing-Maschine, die  $L_2$  entscheidet.
- Wegen  $L_1 \preceq L_2$  gibt es eine Funktion  $f : \mathbb{N} \rightarrow \mathbb{N} \in TM$  mit  $n \in L_1$  gdw  $f(n) \in L_2$ .
- Sei  $\mathcal{M}_f$  eine DTM, die  $f$  berechnet.
- Dann kann man daraus die Maschine  $\mathcal{M}_1 := \mathcal{M}_f \mathcal{M}_2$  konstruieren, für die gilt:
  - $\mathcal{M}_1$ , gestartet mit Input  $n$ , hält mit  $h, \#Y\#$ , falls  $f(n) \in L_2$ , d.h. wenn  $n \in L_1$  ist.
  - $\mathcal{M}_1$ , gestartet mit Input  $n$ , hält mit  $h, \#N\#$ , falls  $f(n) \notin L_2$ , d.h. wenn  $n \notin L_1$  ist.
- Die Maschine  $\mathcal{M}_1$  entscheidet also  $L_1$ , ein Widerspruch.



## Satz 15.16 (Unentscheidbarkeit von $\mathcal{H}_0$ )

*Das Null-Halteproblem*

$$\mathcal{H}_0 = \{n \mid \mathcal{M}_n \text{ hält bei Eingabe } 0\}$$

*ist unentscheidbar.*

## Beweis

Gegeben eine TM  $\mathcal{M}_n$ .

Kombiniere diese mit einer DTM, die  $n$  aufs Band schreibt.

$f(n)$  sei definiert als die Gödelnummer dieser neuen Maschine.

$\mathcal{M}_{f(n)}$  terminiert auf Eingabe von 0

gdw

$\mathcal{M}_n$  terminiert auf Eingabe von  $n$

Damit:

Reduktion des speziellen Halteproblems auf das Null-Halteproblem.

( $f$  ist total und berechenbar!)

Also:

Unentscheidbarkeit des Null-Halteproblems folgt aus

Unentscheidbarkeit des speziellen Halteproblems

## Weitere unentscheidbare Probleme

Ähnlich kann man per Reduktion die Unentscheidbarkeit der folgenden Probleme zeigen:

- $\mathcal{E}$ , das Leerheitsproblem.
- $\mathcal{T}$ , das Totalitätsproblem.
- $\mathcal{E}q$ , das Gleichheitsproblem.
- $\mathcal{E}nt$ , das Entscheidbarkeitsproblem.