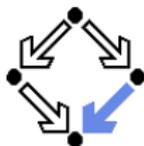


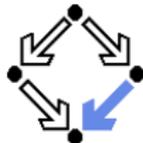
# Berechenbarkeit und Komplexität Turing-Maschinen

Wolfgang Schreiner  
Wolfgang.Schreiner@risc.uni-linz.ac.at

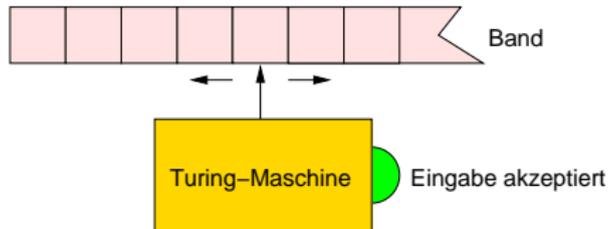
Research Institute for Symbolic Computation (RISC)  
Johannes Kepler University, Linz, Austria  
<http://www.risc.uni-linz.ac.at>



# Turing-Maschine

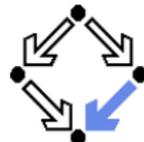


Alan M. Turing, 1937: *On computable numbers, with an application to the Entscheidungsproblem.*



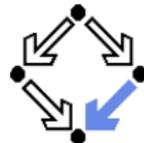
- Eine Kontrolleinheit mit einer **endlichen Anzahl von Zuständen**.
  - Ein Anfangszustand, eine Menge von Endzuständen.
- Ein in einer Richtung **unendlich langes Band**.
  - In jedem Schritt liest der Automat das Symbol in der aktuellen Zelle,
  - bestimmt daraus (und aus seinem Zustand) seinen neuen Zustand,
  - schreibt ein neues Symbol in die aktuelle Zelle und
  - bewegt (optional) den Lese/Schreib-Kopf nach links oder nach rechts.
  - Berechnung endet, wenn kein neuer Zustand bestimmt werden kann.
    - Maschine ist in einem Endzustand: die Eingabe wurde **akzeptiert**.

# Beispiel



Turing-Maschine  $M_1$ , die alle Wörter der Form  $0^n 1^n$  ( $n \in \mathbb{N}$ ) akzeptiert.

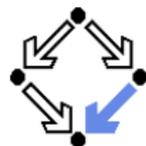
- Am Anfang steht das Wort in den ersten Zellen des Bandes.
  - Gefolgt von einer unendlichen Anzahl von "Leersymbolen"  $\square$ .
- $M_1$  führt wiederholt folgende Schritte durch:
  - $M_1$  ersetzt die am weitesten links stehende 0 durch X.
  - $M_1$  bewegt den L/S-Kopf nach rechts bis zur ersten 1.
  - $M_1$  ersetzt die 1 durch Y.
  - $M_1$  bewegt den L/S-Kopf nach links bis zum ersten X und dann um eins nach rechts (zur jetzt am weitesten links stehenden 0).
- Findet  $M_1$  bei der Suche nach einer 1 stattdessen ein  $\square$ :
  - $M_1$  stoppt, die Eingabe wurde nicht akzeptiert.
- Kann  $M_1$  keine weitere 0 mehr ersetzen:
  - $M_1$  überprüft, ob noch 1en vorhanden sind.
  - Wenn ja, stoppt  $M_1$  und die Eingabe wurde nicht akzeptiert.
  - Wenn nein, stoppt  $M_1$  und die Eingabe wurde akzeptiert.



- **Turing-Maschine**  $M = (Q, \Sigma, \Gamma, q_0, F, \delta)$ 
  - $Q$  ... eine endliche Menge von **Zuständen**.
  - $\Sigma$  ... eine endliche Menge von Symbolen (das **Eingabealphabet**).
  - $\Gamma$  ... eine endliche Menge von Symbolen (das **Bandalphabet**).
    - $\Sigma \subset \Gamma, \sqcup \in \Gamma \setminus \Sigma$
    - $\sqcup$  ... das "Leersymbol".
  - $q_0 \in Q$  ... der **Startzustand**.
  - $F \subseteq Q$  ... die Menge der **Endzustände**.
  - $\delta : Q \times \Gamma \xrightarrow{\text{partiell}} Q \times \Gamma \times \{L, R, S\}$ 
    - $\delta(q, s) = (q', s', r)$  ... der Automat liest im Zustand  $q$  das Symbol  $s$ , geht in den Zustand  $q'$  über, schreibt das Symbol  $s'$  und bewegt den Lese/Schreib-Kopf in die durch  $r$  angezeigte Richtung.
      - $r = L$  ... L/S-Kopf wird um eine Position nach links bewegt.
      - $r = R$  ... L/S-Kopf wird um eine Position nach rechts bewegt.
      - $r = S$  ... L/S-Kopf bleibt stationär (bewegt sich nicht).

Die Kontrolle über das Band und seine Beschreibbarkeit machen den wesentlichen Unterschied zu einem endlichen Automaten aus.

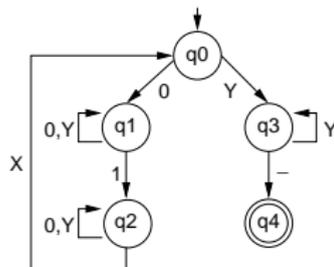
# Beispiel



$$M_1 = (Q, \Sigma, \Gamma, q_0, F, \delta)$$

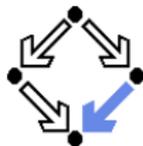
$$Q = \{q_0, q_1, q_2, q_3, q_4\}, \Sigma = \{0, 1\}, \Gamma = \{\sqcup, 0, 1, X, Y\}, F = \{q_4\}$$

$\delta$	$\sqcup$	0	1	X	Y
$q_0$	—	$(q_1, X, R)$	—	—	$(q_3, Y, R)$
$q_1$	—	$(q_1, 0, R)$	$(q_2, Y, L)$	—	$(q_1, Y, R)$
$q_2$	—	$(q_2, 0, L)$	—	$(q_0, X, R)$	$(q_2, Y, L)$
$q_3$	$(q_4, \sqcup, R)$	—	—	—	$(q_3, Y, R)$
$q_4$	—	—	—	—	—



- $q_0$  ... Startzustand, ersetze 0 durch X bzw. starte Suche nach  $\sqcup$ .
- $q_1$  ... Gehe nach rechts bis zur ersten 1 und ersetze es durch Y.
- $q_2$  ... Gehe nach links bis zum ersten X, dann eins nach rechts.
- $q_3$  ... Alle 0en ersetzt, gehe nach rechts auf der Suche nach  $\sqcup$ .
- $q_4$  ...  $\sqcup$  gefunden, Endzustand.

# Konfigurationen der Turing-Maschine



Sei  $M$  eine Turing-Maschine.

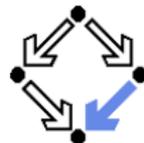
- Eine **Konfiguration**  $\alpha_1 \dots \alpha_k q \alpha_{k+1} \dots \alpha_m$  von  $M$ :
  - Bandymbole  $\alpha_1 \dots \alpha_k$  links vom L/S-Kopf.
  - Aktueller Zustand  $q$ .
  - Bandsymbole  $\alpha_{k+1} \dots \alpha_m$  unter dem bzw. rechts vom L/S-Kopf.
    - $\alpha_{k+1} \dots$  aktuelles Bandsymbol.
    - Rechts von  $\alpha_m$  ausschließlich Leersymbole  $\sqcup$ .

(eine Beschreibung der aktuellen Situation von  $M$ ).

- $\alpha_1 \dots \alpha_k q \alpha_{k+1} \dots \alpha_m \vdash \beta_1 \dots \beta_l p \beta_{l+1} \dots \beta_m$ :
  - $\alpha_i = \beta_i$  für alle  $i \neq k + 1$  und
    - $l = k$  und  $\delta(q, \alpha_{k+1}) = (p, \beta_{l+1}, S)$ , oder
    - $l = k + 1$  und  $\delta(q, \alpha_{k+1}) = (p, \beta_l, R)$ , oder
    - $l = k - 1$  und  $\delta(q, \alpha_{k+1}) = (p, \beta_{l+2}, L)$ .

$(\alpha_1 \dots \alpha_k q \alpha_{k+1} \dots \alpha_m$  geht in  $\beta_1 \dots \beta_l p \beta_{l+1} \dots \beta_m$  über).

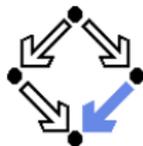
# Die Sprache einer Turingmaschine



Sei  $M = (Q, \Sigma, \Gamma, q_0, F, \delta)$  eine Turing-Maschine.

- Eine **Berechnung** von  $M$ :
  - Eine Folge von Konfigurationen  $C_0, C_1, \dots$ 
    - $C_0 = q_0x$  für ein Eingabewort  $x \in \Sigma^*$ .
    - $C_0 \vdash C_1, C_1 \vdash C_2, \dots$
  - Die Berechnung **endet** in einer Konfiguration  $C_n$ , wenn keine mögliche Konfiguration  $D$  existiert, sodass  $C_n \vdash D$ .
    - Ansonsten ist die Berechnung unendlich lang.
- $M$  **akzeptiert** ein Eingabewort  $x \in \Sigma^*$ :
  - Die von der Konfiguration  $q_0x$  ausgehende Berechnung endet in einer Konfiguration  $\alpha_1 \dots \alpha_k q \alpha_{k+1} \dots \alpha_m$  mit  $q \in F$ .
- Die von  $M$  **akzeptierte Sprache**  $L(M) := \{x \mid M \text{ akzeptiert } x\}$ .
  - Eine Sprache  $L \subseteq \Sigma^*$  heißt **rekursiv aufzählbar**, wenn es eine Turing-Maschine  $M$  gibt, sodass  $L(M) = L$ .
  - Eine Sprache  $L \subseteq \Sigma^*$  heißt **rekursiv**, wenn es eine Turing-Maschine  $M$  mit  $L(M) = L$  gibt, sodass jede Berechnung von  $M$  endet.

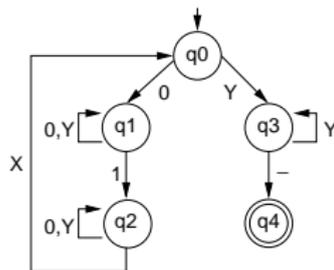
# Beispiel



Eingabe 0011 für Turing-Maschine  $M_1 = (Q, \Sigma, \Gamma, q_0, F, \delta)$ .

$Q = \{q_0, q_1, q_2, q_3, q_4\}$ ,  $\Sigma = \{0, 1\}$ ,  $\Gamma = \{\sqcup, 0, 1, X, Y\}$ ,  $F = \{q_4\}$

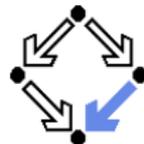
$\delta$	$\sqcup$	0	1	X	Y
$q_0$	—	$(q_1, X, R)$	—	—	$(q_3, Y, R)$
$q_1$	—	$(q_1, 0, R)$	$(q_2, Y, L)$	—	$(q_1, Y, R)$
$q_2$	—	$(q_2, 0, L)$	—	$(q_0, X, R)$	$(q_2, Y, L)$
$q_3$	$(q_4, \sqcup, R)$	—	—	—	$(q_3, Y, R)$
$q_4$	—	—	—	—	—



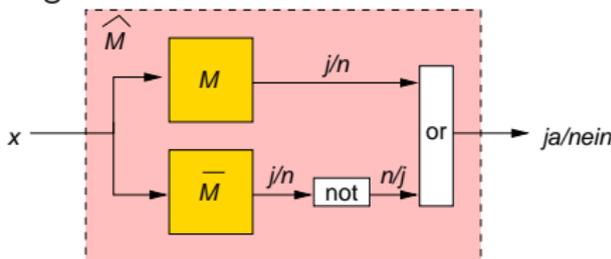
$q_0 0011 \vdash Xq_1 011 \vdash X0q_1 11 \vdash Xq_2 0Y1$   
 $\vdash q_2 X0Y1 \vdash Xq_0 0Y1 \vdash XXq_1 Y1 \vdash XXYq_1 1$   
 $\vdash XXq_2 YY \vdash Xq_2 XYY \vdash XXq_0 YY \vdash XXYq_3 Y$   
 $\vdash XXYq_3 \vdash XXY \sqcup q_4$

Das Wort 0011 wird von  $M_1$  akzeptiert.

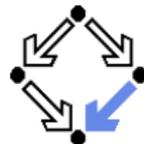
# Rekursive Sprachen



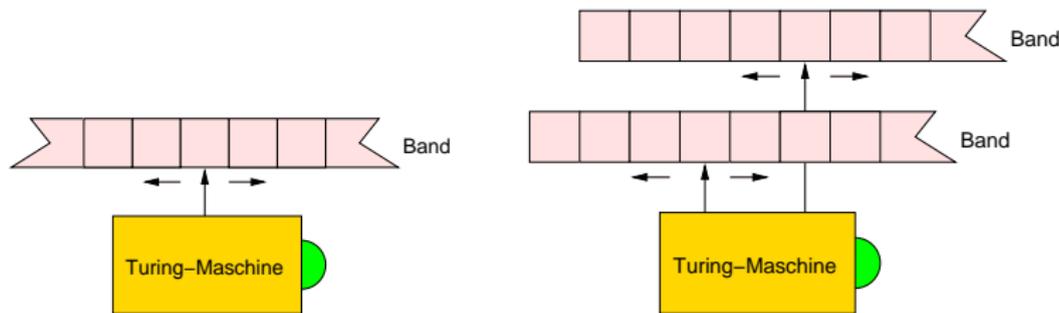
- **Satz:** Die rekursiven Sprachen sind genau diejenigen Sprachen  $L$ , für die sowohl  $L$  als auch das Komplement  $\bar{L}$  rekursiv aufzählbar sind.
- **Beweis von  $\Rightarrow$ :**
  - Sei  $L$  rekursiv und  $M$  eine Turing-Maschine, die  $L$  akzeptiert und deren Berechnungen immer mit “ja/nein” enden. Invertiert man die Antwort von  $M$ , erhält man eine Turing-Maschine  $\bar{M}$ , die  $\bar{L}$  akzeptiert.
- **Beweis von  $\Leftarrow$ :**
  - Seien  $L$  und  $\bar{L}$  rekursiv aufzählbar und  $M$  und  $\bar{M}$  Maschinen, die  $L$  und  $\bar{L}$  akzeptieren (deren Berechnungen also für Wörter aus diesen Sprachen mit “ja” enden und sonst möglicherweise nicht enden).
  - Man kann eine Turingmaschine  $\hat{M}$  konstruieren, die die folgende “parallele” Zusammenschaltung von  $M$  und  $\bar{M}$  simuliert:



# Varianten von Turing-Maschinen



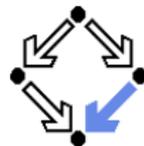
Verschiedenste Variationen sind möglich.



- In beide Richtungen unendliches Band.
  - Simulation durch Maschine mit einseitig unendlichem Band möglich.
- Mehrere Bänder.
  - In jedem Schritt können alle LS-Köpfe unabhängig voneinander schreiben und bewegt werden.
  - Simulation durch Maschine mit nur einem Band möglich.

Die Mächtigkeit des Konzepts wird dadurch nicht vergrößert.

# Nichtdeterministische Turing-Maschinen

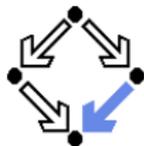


Mehr als eine mögliche Nachfolger-Konfiguration.

- Nichtdeterministische Turing-Maschine  $M = (Q, \Sigma, \Gamma, q_0, F, \delta)$ 
  - ...
  - $\delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R, S\})$ 
    - $(q', s', r) \in \delta(q, s)$  ... der Automat **kann** im Zustand  $q$  das Symbol  $s$  lesen, in den Zustand  $q'$  übergehen, das Symbol  $s'$  schreiben und den Lese/Schreib-Kopf in die durch  $r$  angezeigte Richtung bewegen.

Erweiterung analog zu den nichtdeterministischen endlichen Automaten.

# Beispiel

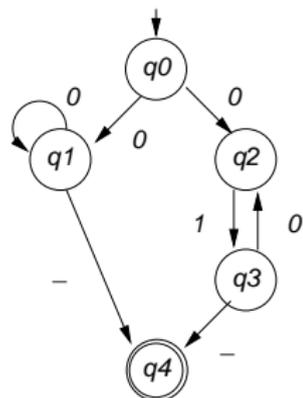


$$L = \{0^n \mid n \geq 1\} \cup \{(01)^n \mid n \geq 1\}$$

$$M = (Q, \Sigma, \Gamma, q_0, F, \delta)$$

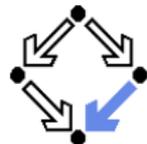
$$Q = \{q_0, q_1, q_2, q_3, q_4\}, \Sigma = \{0, 1\}, \Gamma = \{\sqcup, 0, 1\}, F = \{q_4\}$$

$\delta$	$\sqcup$	0	1
$q_0$	—	$\{(q_1, 0, R), (q_2, 0, R)\}$	—
$q_1$	$\{(q_4, \sqcup, R)\}$	$\{(q_1, 0, R)\}$	—
$q_2$	—	—	$\{(q_3, 1, R)\}$
$q_3$	$\{(q_4, \sqcup, R)\}$	$\{(q_2, 0, R)\}$	—
$q_4$	—	—	—



Die nichtdeterministische Turing-Maschine  $M$  akzeptiert  $L$ .

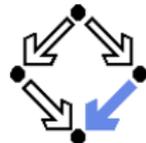
# Nichtdeterministische Turing-Maschinen



- **Satz:** Zu jeder nichtdeterministischen Turing-Maschine  $M$  gibt es eine deterministische Turing-Maschine  $M'$  sodass  $L(M) = L(M')$ .
- **Beweis:** Sei  $r$  die größte Anzahl der Wahlmöglichkeiten von  $M$  für jedes Paar  $(q, x)$  von Zustand  $q$  und Bandsymbol  $x$ .
  - Jede Wahlmöglichkeit ist repräsentiert durch eine Zahl  $1 \dots r$ .
    - Jede Folge von Wahlen in einer Berechnung von  $M'$  ist repräsentiert durch eine endliche Folge der Zahlen  $1 \dots r$ .
  - Deterministische Turing-Maschine  $M'$  mit 3 Bändern:
    - Band 1 ist das ursprüngliche Eingabeband (wird nur gelesen).
    - Auf Band 2 werden alle endlichen Folgen aus  $1 \dots r$  erzeugt. Zuerst alle Folgen der Länge 1, dann alle der Länge 2, etc.
    - Nach Erzeugen der ersten Folge auf Band 2 wird Band 1 auf 3 kopiert.
  - Jetzt wird die Ausführung von  $M$  auf Band 3 simuliert. Jede nichtdeterministische Auswahl wird durch den Inhalt von Band 2 bestimmt. Führt die Simulation zu einem akzeptierenden Endzustand, beendet  $M$  seine Berechnung; ansonsten wird die nächste Folge auf Band 2 erzeugt.

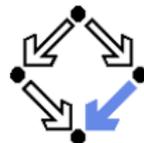
**Auch Nichtdeterminismus vergrößert die Mächtigkeit des Konzepts nicht.**

# Die Erzeugung von Sprachen



- Die von  $M$  erzeugte (generierte) Sprache  $G(M)$ :
  - $M$  sei eine Turing-Maschine, die ihr Band nur zur Ausgabe benutzt.
    - $M$  schreibt nur und bewegt L/S-Kopf niemals nach links.
    - Ausgezeichnetes Bandsymbol  $\#$ .
  - $G(M)$  ist die Menge aller Wörter, die  $M$  jeweils zwischen zwei Vorkommen von  $\#$  auf das Ausgabeband schreibt.
- **Satz:** Eine Sprache ist genau dann rekursiv aufzählbar, wenn sie von einer Turing-Maschine erzeugt werden kann.
  - Beweis  $\Rightarrow$ : Sei  $M$  so dass  $L = L(M)$ . Wir konstruieren  $M'$  zur Erzeugung von  $L$ .  $M'$  konstruiert alle Paare von positiven ganzen Zahlen. Nach Konstruktion eines Paares  $(i, j)$  konstruiert  $M'$  das  $i$ -te Wort  $x_i \in \Sigma^*$  und simuliert maximal  $j$  Schritte der Berechnung von  $M$  mit Eingabe  $x_i$ . Wird dabei  $x_i$  akzeptiert, schreibt  $M'$  das Wort  $x_i$  hinaus. Also gilt  $L = G(M')$ .
  - Beweis  $\Leftarrow$ : Sei  $M$  so dass  $L = G(M)$ . Wir konstruieren  $M'$  zur Erkennung von  $L$  mit einem zusätzlichen Band, auf dem  $M'$  die Berechnung von  $M$  simuliert: immer, wenn darauf ein neues Wort geschrieben wird, vergleicht es  $M'$  mit dem Eingabewort. Stimmen Sie überein, akzeptiert  $M'$  das Wort. Also gilt  $L = L(M')$ .

# Die Berechnung von Funktionen



Seien  $\Sigma_1$  und  $\Sigma_2$  zwei Alphabete.

- Eine Turing-Maschine  $M$  **berechnet eine partielle Funktion**

$$f : (\Sigma_1^*)^k \xrightarrow{\text{partiell}} \Sigma_2^*$$

wenn folgendes gilt:

- $\Sigma_1 \cup \Sigma_2$  ist eine Teilmenge des Bandalphabets von  $M$ .

- Für jedes Tupel von Wörtern  $(x_1, \dots, x_k) \in (\Sigma_1^*)^k$  mit

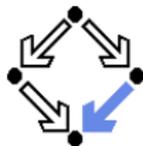
$$f(x_1, \dots, x_k) = y$$

endet die mit der Eingabe (Bandinhalt)  $x_1 \sqcup \dots \sqcup x_k$  begonnene Berechnung von  $M$  mit der Ausgabe (dem Bandinhalt)  $y$ .

- Ist  $f(x_1, \dots, x_k)$  undefiniert, dann terminiert die mit der Eingabe  $x_1 \sqcup \dots \sqcup x_k$  begonnene Berechnung von  $M$  nicht.

- Eine (partielle) Funktion ist **Turing-berechenbar**, wenn es eine Turing-Maschine gibt, die die Funktion berechnet.

- **Satz:**  $f : (\Sigma_1^*)^k \xrightarrow{\text{partiell}} \Sigma_2^*$  ist genau dann Turing-berechenbar, wenn  $L_f = \{(x_1, \dots, x_k, y) \mid f(x_1, \dots, x_k) = y\}$  rekursiv aufzählbar ist.



# Beispiel

Berechnung von  $x \dot{-} y := \begin{cases} x - y, & \text{wenn } x \geq y \\ 0 & \text{sonst} \end{cases}$

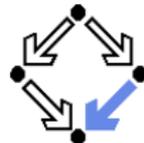
Eingabe 5, 3: 0 0 0 0 0  $\sqcup$  0 0 0  $\sqcup$

Ausgabe 2:  $\sqcup$   $\sqcup$   $\sqcup$  0 0  $\sqcup$   $\sqcup$   $\sqcup$   $\sqcup$   $\sqcup$

- Für jedes Vorkommen einer 0 in der Codierung von  $y$ :
  - Überschreibe das Vorkommen durch  $\sqcup$ .
  - Überschreibe ein führendes Vorkommen von 0 in  $x$  durch  $\sqcup$ .
    - Sofern ein solches Vorkommen noch existiert.

Turing-Maschinen-Berechnung der Negation auf  $\mathbb{N}$ .

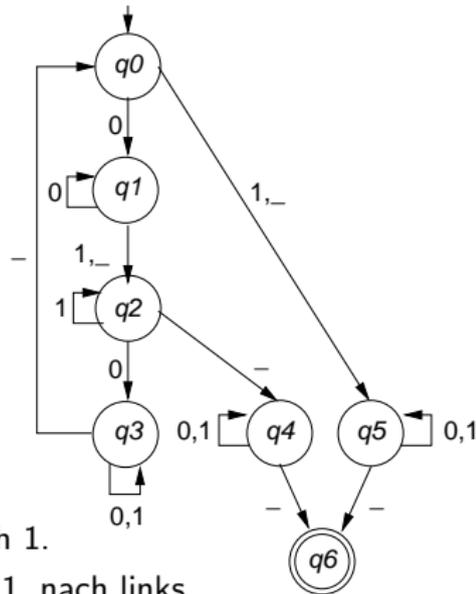
# Beispiel



$$M_2 = (Q, \Sigma, \Gamma, F, \delta)$$

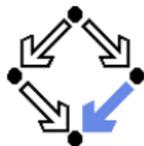
$$Q = \{q_0, \dots, q_6\}, \Sigma = \{0\}, \Gamma = \{0, 1, \sqcup\}, F = \{\}$$

$\delta$	0	1	$\sqcup$
$q_0$	$(q_1, \sqcup, R)$	$(q_5, \sqcup, R)$	$(q_5, \sqcup, R)$
$q_1$	$(q_1, 0, R)$	$(q_2, 1, R)$	$(q_2, 1, R)$
$q_2$	$(q_3, 1, L)$	$(q_2, 1, R)$	$(q_4, \sqcup, L)$
$q_3$	$(q_3, 0, L)$	$(q_3, 1, L)$	$(q_0, \sqcup, R)$
$q_4$	$(q_4, 0, L)$	$(q_4, \sqcup, L)$	$(q_6, 0, R)$
$q_5$	$(q_5, \sqcup, R)$	$(q_5, \sqcup, R)$	$(q_6, \sqcup, R)$
$q_6$	—	—	—



- $q_0$ : Startzustand, ersetze führende 0 durch  $\sqcup$ .
- $q_1$ : Suche nach nächstem  $\sqcup$  und ersetze es durch 1.
- $q_2$ : Suche nach nächster 0 und ersetze es durch 1, nach links.
- $q_3$ : Suche nach vorigem  $\sqcup$ , nach rechts, beginne von vorne.
- $q_4$ :  $\sqcup$  statt 0 gefunden, ersetze alle vorigen 1 durch  $\sqcup$ .
- $q_5$ : x ist 0, lösche Rest des Bandes.
- $q_6$ : Endzustand.

# Beispiel



- Berechnung von  $2 \dot{-} 1$ :

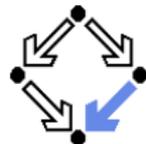
$$\begin{aligned} & q_0 00 \sqcup 0 \vdash \sqcup q_1 0 \sqcup 0 \vdash \sqcup 0 q_1 \sqcup 0 \vdash \sqcup 01 q_2 0 \\ & \vdash \sqcup 0 q_3 11 \vdash \sqcup q_3 0 11 \vdash q_3 \sqcup 0 11 \vdash \sqcup q_0 0 11 \\ & \vdash \sqcup \sqcup q_1 11 \vdash \sqcup \sqcup 1 q_2 1 \vdash \sqcup \sqcup 11 q_2 \vdash \sqcup \sqcup 1 q_4 1 \\ & \vdash \sqcup \sqcup q_4 1 \vdash \sqcup q_4 \vdash \sqcup 0 q_6. \end{aligned}$$

- Berechnung von  $1 \dot{-} 2$ :

$$\begin{aligned} & q_0 0 \sqcup 00 \vdash \sqcup q_1 \sqcup 00 \vdash \sqcup 1 q_2 00 \vdash \sqcup q_3 110 \\ & \vdash q_3 \sqcup 110 \vdash \sqcup q_0 110 \vdash \sqcup \sqcup q_5 10 \vdash \sqcup \sqcup \sqcup q_5 0 \\ & \vdash \sqcup \sqcup \sqcup \sqcup q_5 \vdash \sqcup \sqcup \sqcup \sqcup \sqcup q_6. \end{aligned}$$

Ebenso sind alle üblichen arithmetischen Funktionen Turing-berechenbar.

# Simulator für Turing-Maschinen



Zum Beispiel auf <http://ironphoenix.org/tril/tm>

State: 1

Start Stop Resume Step Speed: Fast

1 1 1 1 1 1 - 1 1 1 1 =

Load new program: Subtractor

Machine name Subtractor

Initial tape position 0

Initial characters on tape 111111-1111=

Programming

```
1, 1, >
1, 1, 1, >
1, -, 1, >
1, =, 2, <
2, 1, 3, =, <
2, -, H, <
3, 1, 3, 1, <
3, -, 4, -, <
4, -, 4, <
4, 1, >
```

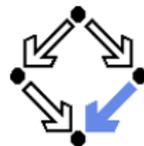
Clear Program Install Program

Entering program...  
Machine programmed successfully.

Program description:  
Subtracts numbers in unary.  
Input format: '<num>-<num>='

Clear Message Box

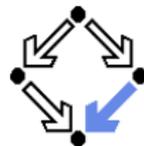
# Turing-Maschinen und RAMs



- **Satz:** jede Turing-Maschine kann durch eine RAM simuliert werden.
- Beweis: Wir simulieren eine Turing-Maschine  $M$  durch eine RAM  $R$ .
  - $R$  verwendet  $c$  Register  $0 \dots c - 1$  für eigene Zwecke, speichert in Zelle  $c$  die Position des L/S-Kopfes von  $M$  und in den Registern  $c + 1, c + 2, \dots$  die Zellen  $1, 2, \dots$  des Bands von  $M$  ab. Die aktuelle Bandzelle von  $M$  kann von  $R$  über Register  $c$  durch indirekte Adressierung gelesen werden.

Jede Sprache bzw. Funktion die von einer Turing-Maschine akzeptiert bzw. berechnet werden kann, kann auch von einer RAM akzeptiert bzw. berechnet werden.

# Turing-Maschinen und RAMs

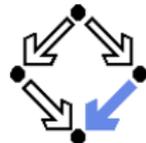


- **Satz:** Jede RAM kann durch eine Turing-Maschine simuliert werden.
- Beweis: Wir simulieren eine RAM  $R$  durch eine Turing-Maschine  $M$  mit 5 Bändern  $B_1, \dots, B_5$ .
  - $B_1$  ist das Eingabeband.
  - $B_2$  ist das Ausgabeband.
  - $B_3$  dient zur Darstellung des Speichers von  $R$ :

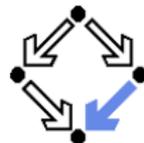


- $(i_1, \dots, i_k)$  ... die Indizes aller Register von  $R$ , die nicht 0 enthalten.
- $(c_1, \dots, c_k)$  ... die entsprechenden Speicherinhalte  $c(i_1), \dots, c(i_k)$ .  
Indizes und Speicherinhalte in Binärdarstellung.
- $B_4$  speichert den Inhalt des Akkumulators (in Binärdarstellung).
- $B_5$  dient als Arbeitsband.

Jeder Rechenschritt von  $R$  wird durch eine endliche Folge von Zuständen von  $M$  simuliert.



- Simulation von ADD \* $i$ :
  - Suche auf  $B_3$  nach der Markierung  $i$ .
    - Folge  $##i'$  mit Binärdarstellung  $i'$  von  $i$ .
    - Ist Markierung nicht vorhanden, beende die Berechnung.
    - Ansonsten kopiere die der Markierung folgende Zahl auf  $B_5$ .
  - Suche auf  $B_3$  nach der Markierung, dessen Nummer auf  $B_5$  steht.
    - Ist Eintrag nicht vorhanden, schreibe 0 auf  $B_5$ .
    - Ansonsten kopiere die der Markierung folgende Zahl auf  $B_5$ .
  - Addiere die Zahl auf  $B_5$  zum Inhalt von  $B_4$  (dem Akkumulator).
- Simulation von STORE  $i$ :
  - Suche auf  $B_3$  nach der Markierung  $i$ .
  - Ist Markierung vorhanden:
    - Kopiere den der folgenden Zahl nachfolgenden Bandinhalt auf  $B_5$ .
    - Schreibe auf  $B_3$  nach der Markierung den Inhalt von  $B_4$ .
    - Schreibe auf  $B_3$  danach den Bandinhalt von  $B_5$ .
  - Ist Eintrag nicht vorhanden:
    - Schreibe  $i'\#$  auf  $B_3$  unmittelbar nach dem letzten von  $\sqcup$  verschiedenen Symbol, anschließend den Inhalt von  $B_4$ , danach  $##$ .



- **Church'sche These** (Alonzo Church):

*Eine Funktion ist genau dann berechenbar, wenn sie Turing-berechenbar ist.*

- Nicht beweisbar.

- "Berechenbarkeit" ist nur ein intuitiver Begriff.

- Sehr glaubhaft.

- Viele verschiedene Berechnungsmodelle wurden entworfen.

- Keines von ihnen ist mächtiger als die Turing-Maschine.

- Ein **Algorithmus** ist eine Turing-Maschine, deren Berechnungen für alle möglichen Eingaben enden.

- Alle Elemente der Ein-/Ausgabemengen müssen sich als Wörter über einem Alphabet schreiben lassen.

- Dies trifft insbesondere auf die natürlichen Zahlen zu.

- Natürliche Zahlen lassen sich als Folgen von Ziffern schreiben.

- Damit umschließt der Begriff (aber beschränkt sich nicht auf) die Berechnung zahlentheoretischer Funktionen.