

# **Automatentheorie und Formale Sprachen**

## **– LV 4110 –**

### **Entscheidbarkeit und Berechenbarkeit**

- 
- Verwendung von unterschiedlichen Rechnermodellen
  - Kennenlernen der Begriffe: Berechenbarkeit und Entscheidbarkeit
  - Definitionen für Entscheidungsverfahren und Aufzählverfahren
  - Klärung, was man unter einer **berechenbaren Funktion** versteht
  - Definitionen für **abzählbare** und **überabzählbare** Mengen
  - Beispiele für Turing-berechenbare Funktionen
  - Kennenlernen, was die **Church-Turing'sche These** ausdrückt
  - Herausstellen von **nicht entscheidbaren** Problemen und Sprachen
  - Erweiterungen der Turing-Maschine
-

## VI. Entscheidbarkeit und Berechenbarkeit

1. Hauptfrage in diesem Kapitel
2. Vergleich zwischen Register- und Turingmaschine
3. Definition für einen Algorithmus
4. Definitionen für Entscheidungs- und Aufzählverfahren
5. Berechenbare Funktionen und entscheidbare Mengen
6. Definition der Turing-Berechenbarkeit
7. Die Church-Turing'sche These
8. Das Post'sche Korrespondenzproblem
9. Erweiterungen der Turing-Maschine

## **VI. Entscheidbarkeit und Berechenbarkeit**

### **1. Hauptfrage in diesem Kapitel**

2. Vergleich zwischen Register- und Turingmaschine
3. Definition für einen Algorithmus
4. Definitionen für Entscheidungs- und Aufzählverfahren
5. Berechenbare Funktionen und entscheidbare Mengen
6. Definition der Turing-Berechenbarkeit
7. Die Church-Turing'sche These
8. Das Post'sche Korrespondenzproblem
9. Erweiterungen der Turing-Maschine

**Besitzt jedes Problem, das  
mathematisch exakt formulierbar  
ist, eine algorithmische Lösung**

*oder*

**gibt es Grenzen der  
Berechenbarkeit ?**

Im Jahre **1900** präsentierte **David Hilbert** auf einem Mathematiker-Kongress eine Liste mit **23** ungelösten Problemen:

## Problem Nr. 10:

Gebe einen Algorithmus an, der für jede diophantische Gleichung feststellt, ob sie eine ganzzahlige Lösung besitzt oder nicht!

Beispiele für diophantische Gleichungen:

$$x^2 + y^2 - z^2 = 0$$

Lösung:  $x = 3$  ,  $y = 4$  ,  $z = 5$

$$6x^{18} - x + 3 = 0$$

hat keine ganzzahlige Lösung, da

$$\forall x \in \mathbf{Z} : 6x^{18} > x - 3$$

## VI. Entscheidbarkeit und Berechenbarkeit

1. Hauptfrage in diesem Kapitel
- 2. Vergleich zwischen Register- und Turingmaschine**
3. Definition für einen Algorithmus
4. Definitionen für Entscheidungs- und Aufzählverfahren
5. Berechenbare Funktionen und entscheidbare Mengen
6. Definition der Turing-Berechenbarkeit
7. Die Church-Turing'sche These
8. Das Post'sche Korrespondenzproblem
9. Erweiterungen der Turing-Maschine

### Modellbildung:

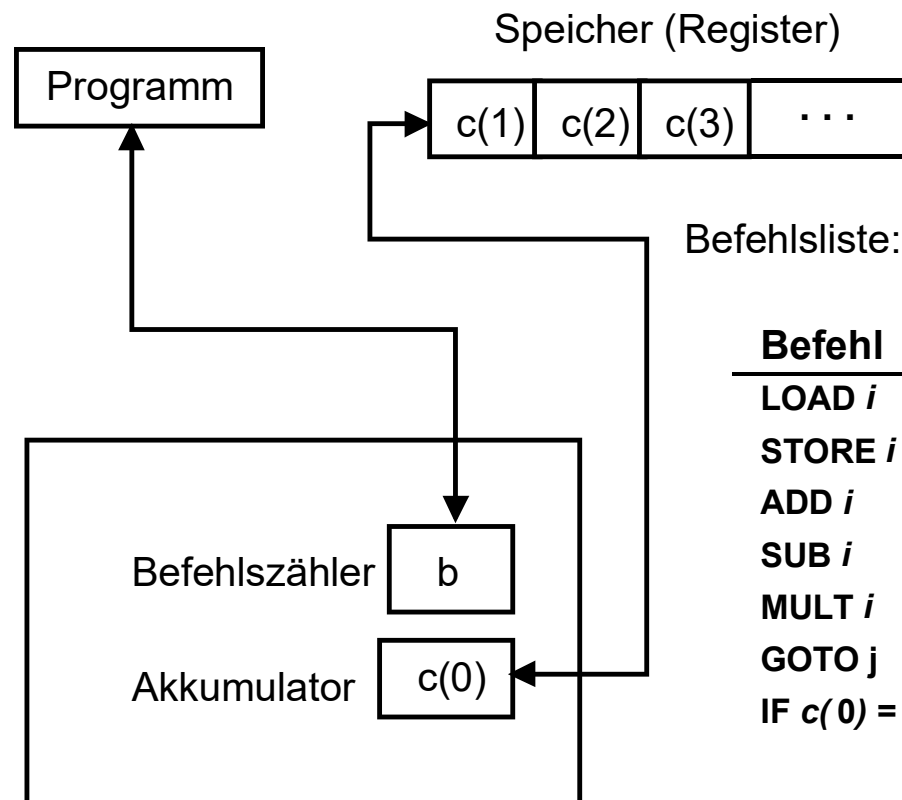
- Ein sehr realistisches Rechnermodell (nach der Vorstellung eines wirklichen Rechners) ist die **Registermaschine** (**R**andom **A**ccess **M**aschine, kurz **RAM**).
- Zwar scheint die **Turing-Maschine** (kurz **TM**) nicht besonders realistisch zu sein, dient sie jedoch als Rechnermodell, das sich für „allgemeine“ theoretische Aussagen hervorragend eignet.
- Man kann zeigen (tun wir hier nicht!), dass die **TM** „gleichwertig“ zur **RAM** ist, die – wie deren Arbeitsweise demonstrieren wird – wiederum einen wirklichen Rechner modelliert.



### Arbeitsweise:

- Die RAM besteht aus einem **Befehlszähler**, einem **Akkumulator**, aus **Registern** und aus einem **Programm**.
- Die Inhalte von Befehlszähler, Akkumulator und Registern sind natürliche Zahlen. In den ersten Registern steht die Eingabe.
- Die Register bilden den (unendlichen) Speicher der RAM und haben alle eine eindeutige Adresse. Der Inhalt (***content***) des Registers ***i*** sei mit ***c(i)*** bezeichnet.
- Das Programm besteht aus einer Folge von Befehlen, wobei die Programmzeilen durchnummeriert (0, 1, 2, ...) sind.
- Der Befehlszähler startet bei **Null**, und enthält die Nummer des nächsten auszuführenden Befehls.

## Schematische Darstellung der RAM:



Befehl	Wirkung
LOAD $i$	$c(0) := c(i); \quad b := b + 1$
STORE $i$	$c(i) := c(0); \quad b := b + 1$
ADD $i$	$c(0) := c(0) + c(i); \quad b := b + 1$
SUB $i$	$c(0) := \max\{0, c(0) - c(i)\}; \quad b := b + 1$
MULT $i$	$c(0) := c(0) * c(i); \quad b := b + 1$
GOTO $j$	$b := j$
IF $c(0) = k$ GOTO $j$	$b := j \quad \text{falls } c(0) = k$ $b := b + 1 \quad \text{sonst}$
END	$b := b$

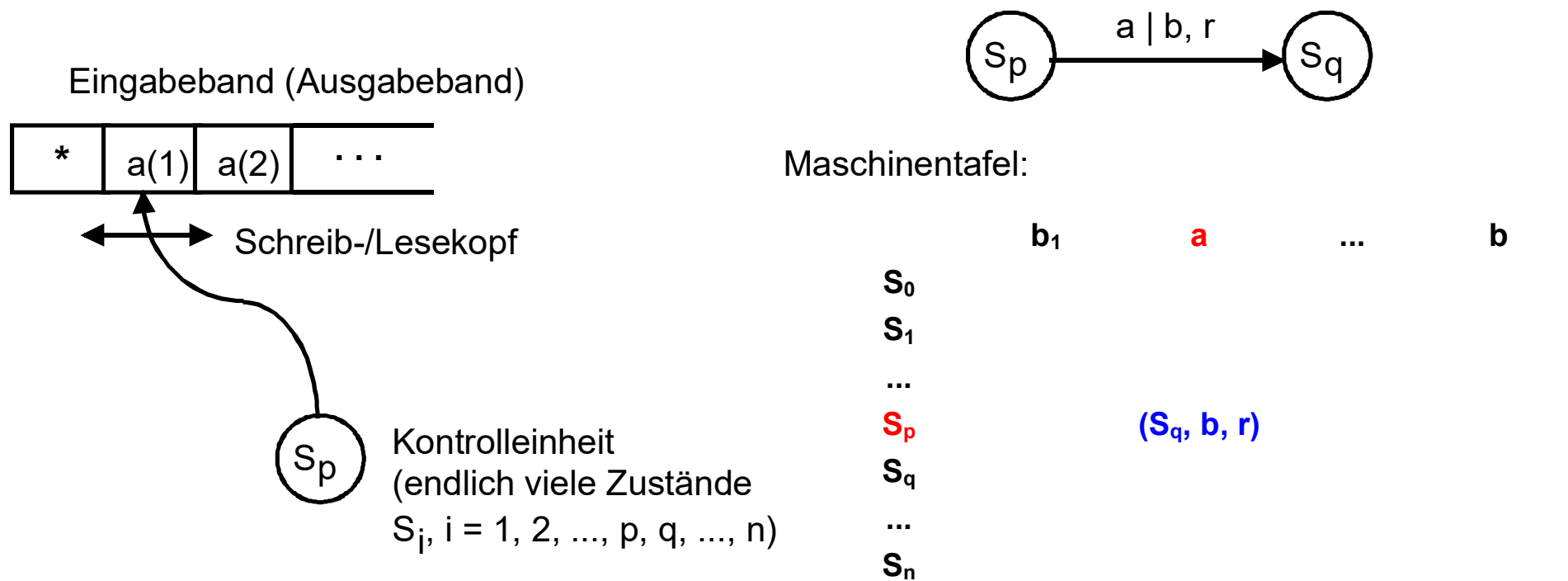
## Beispiel: $f(x) = 0$ , falls $x$ gerade, sonst Endlosschleife

0	:	READ	1	// Wert x in Reg. c(1) ablegen
1	:	LOAD	1	// c(1) = x in Akkumulator laden
2	:	DIV	=2	// dividiere Akku durch 2 (integer!)
3	:	MULT	=2	// multipliziere Akku mit 2 (integer!)
4	:	STORE	2	// Akkumulator in c(2) ablegen
5	:	LOAD	1	// c(1) in Akkumulator laden
6	:	SUB	2	// davon nun c(2) subtrahieren
7	:	IF c(0) = 0 GOTO	10	// bedingter Sprung zu 10 (Ausgabe)
8	:	SUB	=1	// Akkumulator wird bzw. bleibt 0
9	:	IF c(0) = 0 GOTO	8	// Rücksprung zu 8 (Endlosschleife)
10	:	WRITE	=0	// c(0) war 0 und somit 0 ausgeben
11	:	END		// Befehlszähler b bleibt 11

## Arbeitsweise:

- Die TM besteht aus einem einseitig unendlichen **Eingabe- und Ausgabeband** mit einem freibeweglichen **Schreib-/Lesekopf**.
- Der Schreib-/Lesekopf wird von einer endlichen **Kontrolleinheit** (endlich viele Zustände) gesteuert.
- Das Eingabe- und Ausgabeband enthält eine Folge von Symbolen, welche zu Beginn der Berechnung die Eingabe darstellt.
- Die Kontrolleinheit entspricht dem Befehlszähler der RAM und die Zellen des Eingabe- und Ausgabebandes den Registern der RAM.
- Die Zustandsübergänge der TM werden anhand einer Maschinentafel vollzogen. Dabei bewegt sich der Schreib-/Lesekopf eine Stelle nach recht, nach links oder überhaupt nicht.

## Schematische Darstellung der TM:



Beispiel:  $L(TM) = \{ w \mid w = a^n b^n c^n \text{ für } n = 1, 2, \dots \}$

$\Sigma = \{a, b, c\}$ ;  $\mathbf{B} = \{a, b, c, *, A, B, C\}$ ;  $\mathbf{S} = \{S_0, S_1, \dots, S_7\}$ ;  $\mathbf{F} = \{S_7\}$

	<b>a</b>	<b>b</b>	<b>c</b>	<b>A</b>	<b>B</b>	<b>C</b>	<b>*</b>
<b>S<sub>0</sub></b>	–	–	–	–	–	–	(S <sub>1</sub> , *, r)
<b>S<sub>1</sub></b>	(S <sub>2</sub> , A, r)	–	–	–	(S <sub>5</sub> , B, r)	–	–
<b>S<sub>2</sub></b>	(S <sub>2</sub> , a, r)	(S <sub>3</sub> , B, r)	–	–	(S <sub>2</sub> , B, r)	–	–
<b>S<sub>3</sub></b>	–	(S <sub>3</sub> , b, r)	(S <sub>4</sub> , C, l)	–	–	(S <sub>3</sub> , C, r)	–
<b>S<sub>4</sub></b>	(S <sub>4</sub> , a, l)	(S <sub>4</sub> , b, l)	–	(S <sub>1</sub> , A, r)	(S <sub>4</sub> , B, l)	(S <sub>4</sub> , C, l)	–
<b>S<sub>5</sub></b>	–	–	–	–	(S <sub>5</sub> , B, r)	(S <sub>5</sub> , C, r)	(S <sub>6</sub> , *, l)
<b>S<sub>6</sub></b>	–	–	–	(S <sub>6</sub> , A, l)	(S <sub>6</sub> , B, l)	(S <sub>6</sub> , C, l)	(S <sub>7</sub> , *, h)

## VI. Entscheidbarkeit und Berechenbarkeit

1. Hauptfrage in diesem Kapitel
2. Vergleich zwischen Register- und Turingmaschine
- 3. Definition für einen Algorithmus**
4. Definitionen für Entscheidungs- und Aufzählverfahren
5. Berechenbare Funktionen und entscheidbare Mengen
6. Definition der Turing-Berechenbarkeit
7. Die Church-Turing'sche These
8. Das Post'sche Korrespondenzproblem
9. Erweiterungen der Turing-Maschine

---

### Definition (Algorithmus):

Es sei **P** eine **Problemklasse** und **A** die Menge der konkreten **Problemausprägungen**, d. h. die Menge derjenigen Daten, die ein Problem beschreiben.

Dann verstehen wir unter einem **Algorithmus  $A_P$**  zu der Klasse **P** ein **allgemeines, deterministisches Verfahren**, welches – auf richtige Anfangsdaten  $a \in A$  angewendet – nach endlich vielen Schritten hält und die Lösung des Problems liefert (d. h. ein Element der Lösungsmenge **B**).

Wir schreiben:  $A_P : A \rightarrow B$ ,

d. h.  $\forall a \in A : \exists b \in B : A_P(a) = b$



## VI. Entscheidbarkeit und Berechenbarkeit

1. Hauptfrage in diesem Kapitel
2. Vergleich zwischen Register- und Turingmaschine
3. Definition für einen Algorithmus
- 4. Definitionen für Entscheidungs- und Aufzählverfahren**
5. Berechenbare Funktionen und entscheidbare Mengen
6. Definition der Turing-Berechenbarkeit
7. Die Church-Turing'sche These
8. Das Post'sche Korrespondenzproblem
9. Erweiterungen der Turing-Maschine

### Definition (Entscheidungs- und Aufzählverfahren):

Einen Algorithmus  $\mathbf{A_P} : \mathbf{A} \rightarrow \mathbf{B}$  nennen wir auch

- ein **Entscheidungsverfahren**, falls
$$\mathbf{B} = (\text{True}, \text{False}) \text{ bzw. auch allgemeiner falls}$$
$$|\mathbf{B}| = 2$$
- ein **Aufzählverfahren**, falls
$$\mathbf{A} = \mathbb{IN}$$

### Beispiel 1 (Primzahlen):

geg: Zahl  $Z \in \mathbb{IN}$     ges: Algorithmus, der entscheiden kann, ob  $Z$  eine Primzahl ist oder nicht d. h.  $\{2, 3, 5, 7, 11, 13, \dots\}$  ?

$\Rightarrow$  **Entscheidungsverfahren**

---

Beispiel 2 (Primzahlen):

geg:  $A = \{ \forall \text{ Primzahlen} \}$

$= \{ p \mid p \text{ ist nicht das Vielfache einer ganzen Zahl mit Ausnahme der } 1 \text{ und } p \text{ selbst} \}$

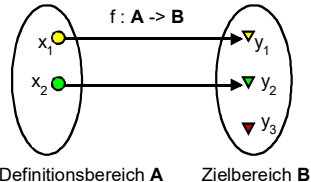
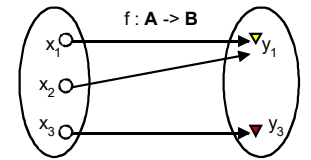
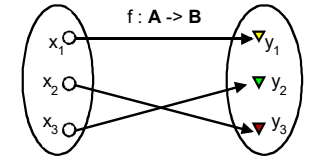
ges:  $A_P(1), A_P(2), A_P(3), \dots$

$\Rightarrow$  Aufzählverfahren

## VI. Entscheidbarkeit und Berechenbarkeit

1. Hauptfrage in diesem Kapitel
2. Vergleich zwischen Register- und Turingmaschine
3. Definition für einen Algorithmus
4. Definitionen für Entscheidungs- und Aufzählverfahren
- 5. Berechenbare Funktionen und entscheidbare Mengen**
6. Definition der Turing-Berechenbarkeit
7. Die Church-Turing'sche These
8. Das Post'sche Korrespondenzproblem
9. Erweiterungen der Turing-Maschine

Definition (injektiv, surjektiv, bijektiv):

<p><b>Injektion:</b></p>  <p>Definitionsbereich A    Zielbereich B</p>	<p><math>\forall x_1, x_2 \in A : x_1 \neq x_2 \Rightarrow f(x_1) \neq f(x_2)</math>  <math> A  \leq  B </math> (Kardinalität)  auch <b>linkseindeutig</b> genannt  Bsp.: <math>\mathbf{R} \rightarrow \mathbf{R} : f(x) = \arctan(x)</math>  <math>\mathbf{N} \rightarrow \mathbf{N} : f(n) = n^2</math>; aber:  <math>\mathbf{Z} \rightarrow \mathbf{Z} : f(n) = n^2</math> ist nicht injektiv</p>	<p>Zwei verschiedene Elemente von <b>A</b> werden auf verschiedene Elemente von <b>B</b> abgebildet – oder: Jedem y-Wert im Zielbereich besitzt <b>nur genau einen</b> x-Wert im Definitionsbereich; daher wiederholt sich bei einer injektiven Funktion also <u>nie</u> ein Funktionswert.</p>
<p><b>Surjektion:</b></p>  <p>Definitionsbereich A    Zielbereich B</p>	<p><math>\forall y_i \in B : \exists \text{ mind. ein } x_i \in A : f(x_i) = y_i</math>  <math> A  \geq  B </math>  auch <b>rechtstotal</b> genannt  Bsp.: <math>\mathbf{Z} \rightarrow \mathbf{N} : f(x) = \text{abs}(x)</math></p>	<p>Jedes Element der Zielmenge B ist ein Funktionswert, d. h. <b>alle möglichen</b> y-Werte im Zielbereich werden angenommen – oder: Bildbereich von f stimmt mit der Zielmenge von f überein.</p>
<p><b>Bijektion:</b></p>  <p>Definitionsbereich A    Zielbereich B</p>	<p><math>\forall y_i \in B : \exists \text{ genau ein } x_i \in A : f(x_i) = y_i</math>  <math> A  =  B </math>  Bsp.: <math>\{0 \dots 255\} \rightarrow \{0, 1\}^8 :</math>  <math>f(x) = 8\text{-stell. Binärzahl}</math></p>	<p>f heißt bijektiv, falls es sowohl injektiv als auch surjektiv ist.  Genau dann existiert auch eine <b>Umkehrfunktion</b> <math>x = f^{-1}(y)</math></p>

### Definition (berechenbare Funktion):

Eine Funktion  $f : M_1 \rightarrow M_2$  heißt **berechenbar**, falls es einen Algorithmus

$$A_f : M_1 \rightarrow M_2$$

gibt mit

$$A_f(w) = f(w) \text{ für } \forall w \in M_1$$

### Interpretation:

Zu jedem Argument  $w \in M_1$  kann man also den Funktionswert  $f(w) \in M_2$  in endlich vielen Schritten berechnen.

Beispiel (berechenbare Funktion):

Für  $M_1 = \mathbb{IN}$  und  $M_2 = \mathbb{IN}$  ist

$$f : \mathbb{IN} \rightarrow \mathbb{IN} \text{ mit } f(n) = n \cdot (n + 1) / 2$$

eine berechenbare Funktion, da es für diese Funktion einen Algorithmus gibt (Aufgabe 1.3 – vgl. Übungsblatt 1).

Anmerkung:

Es ist keineswegs eine triviale Frage, ob jede gegebene Funktion berechenbar ist, also durch einen Algorithmus realisiert werden kann. **Wir werden sehen, dass die Frage zu verneinen ist.**

### Definition (abzählbare Menge, überabzählbare Menge):

Es sei  $M$  eine Menge. Wir nennen  $M$  **abzählbar**, wenn sich  $M$  **bijektiv** auf die Menge  $\mathbb{N}$  (der natürlichen Zahlen) oder eine Teilmenge von  $\mathbb{N}$  abbilden lässt. Andernfalls nennen wir  $M$  **überabzählbar**.

### Merke:

- Jede endliche Menge ist abzählbar.
- Ist  $A$  ein (endliches) Alphabet, so ist  $A^*$  abzählbar (wegen lexikographischer Ordnung).

### Satz:

Es sei  $\Sigma$  ein Alphabet und somit endlich. Dann gibt es **überabzählbar** viele Funktionen  **$f : \Sigma^* \rightarrow \Sigma^*$** , von denen allerdings nur **abzählbar** viele berechenbar sind. (Beweis durch Widerspruch! – vgl. Übungsblatt 12)

---



### Definition (entscheidbare Menge):

a) Es seien  $M_1 \subseteq M_2 \subseteq \Sigma^*$ .

$M_1$  heißt **entscheidbar relativ zu**  $M_2$ , wenn es einen Algorithmus

$$A_{M_1 M_2} : M_2 \rightarrow \{ \text{True}, \text{False} \}$$

gibt, mit dessen Hilfe man **zu jedem** Element  $w \in M_2$  feststellen kann, ob es zu  $M_1$  gehört oder nicht.

kurz:  $\forall w \in M_2 : A_{M_1 M_2}(w) = „w \in M_1“$

b) Es sei  $M \subseteq \Sigma^*$ .

$M$  heißt **absolut entscheidbar** (oder kurz entscheidbar), wenn  $M$  relativ zu  $\Sigma^*$  entscheidbar ist. **→ Entscheidbarkeit ist eine Mengeneigenschaft und nicht etwa die Eigenschaft eines einzelnen Objektes!**

---

### Beispiele:

a) Die Menge

$$M := \{ n \in \mathbb{N} \mid n \text{ ist eine Primzahl} \}$$

ist **entscheidbar relativ zu**  $\mathbb{N}$ , denn es muss zu einer vorgegebenen Zahl  $n \in \mathbb{N}$  nur getestet werden, ob ein  $m < n$  mit  $m \neq 1$  existiert, das  $n$  teilt („|“). Wenn ja (d.h. es  $\exists$  ein  $m$ )  $\Rightarrow$   $n$  ist **keine** Primzahl; sonst (d. h. es  $\exists$  **kein**  $m$ )  $\Rightarrow$   **$n$  ist eine Primzahl!**

kurz:

$$\text{Für } n \in \mathbb{N} \quad \exists m \quad (m \neq 1 \wedge m < n) \mid (m \text{ „|“ } n) \Rightarrow n \notin \mathbb{P}$$

- b) Jede endliche Menge ist **entscheidbar**; ein entsprechender Algorithmus muss lediglich eine endliche Menge oder Liste durchsuchen, um die Entscheidung treffen zu können.
  
- c) Das allgemeine Wortproblem für **Typ-0-Grammatiken** ist nicht entscheidbar, d. h. es gibt keinen Algorithmus, der zu einer beliebig vorgegebenen Grammatik **G** und einem ebenfalls vorgegebenen Wort **w** entscheiden kann, ob  **$w \in L(G)$**  gilt oder nicht.

### Definition (aufzählbare Menge):

Eine Menge  $M \subseteq \Sigma^*$  heißt **aufzählbar**, wenn es eine Funktion

$$f : \mathbb{N}_0 \rightarrow M$$

gibt, die surjektiv und berechenbar ist. Wir sagen dann: „M wird durch f aufgezählt“, d. h.

$$M = \{ f(0), f(1), f(2), f(1), \dots \}$$

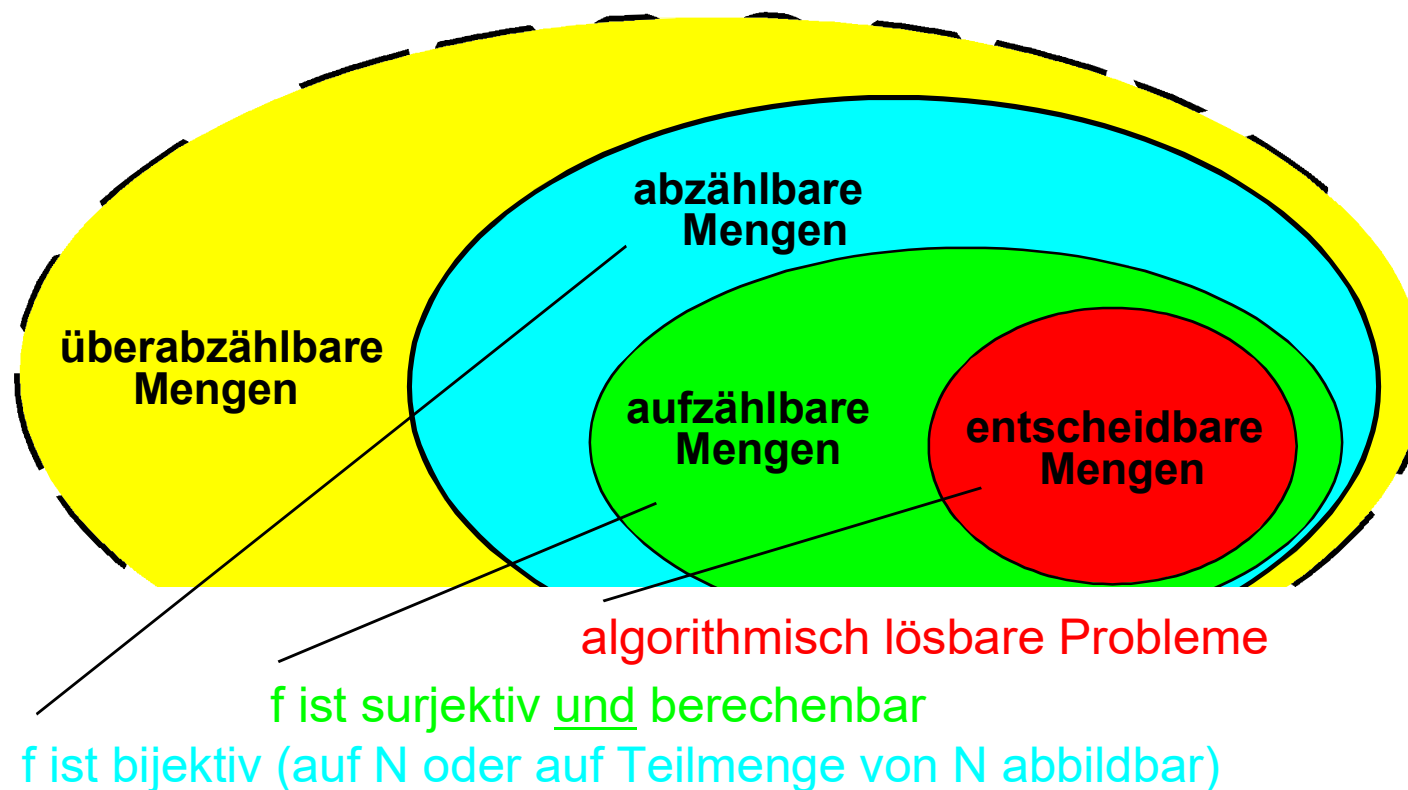
⇒ durch die Funktion f werden die Elemente von M in eine feste Reihenfolge gebracht, wobei nicht ausgeschlossen ist, dass ein Element von M **mehrfach aufgezählt** wird.

### Satz:

Für eine Menge  $M \subseteq \Sigma^*$  gilt:

- a) M ist **aufzählbar** ⇒ M ist **abzählbar**; die Umkehrung gilt i.a. nicht!
  - b) M ist **entscheidbar** ⇒ M ist **aufzählbar**
-

### Veranschaulichung der Zusammenhänge:



## VI. Entscheidbarkeit und Berechenbarkeit

1. Hauptfrage in diesem Kapitel
2. Vergleich zwischen Register- und Turingmaschine
3. Definition für einen Algorithmus
4. Definitionen für Entscheidungs- und Aufzählverfahren
5. Berechenbare Funktionen und entscheidbare Mengen
- 6. Definition der Turing-Berechenbarkeit**
7. Die Church-Turing'sche These
8. Das Post'sche Korrespondenzproblem
9. Erweiterungen der Turing-Maschine

### Definition (Turing-Berechenbarkeit):

Es sei  $\mathbf{TM} = (S, S_0, F, \Sigma, B, \delta)$  eine **Turing-Maschine** und  $M_1, M_2 \subseteq \Sigma^*$ .

- a) Wir sagen: Eine **TM realisiert** eine Funktion  $f : M_1 \rightarrow M_2$ , wenn folgendes gilt:
- 1) Für  $\forall w \in M_1$  gilt:  $(*, w, s_0) \Rightarrow (*, f(w), s_f)$ , wobei  $(*, f(w), s_f)$  eine sog. Finalkonfiguration ist, d. h. die Maschine hält an, und  $s_f$  ist ein Endzustand.
  - 2) Andernfalls, d. h. für  $w \notin M_1$ , geht die Maschine nie in eine Finalkonfiguration über, und das Verhalten der Maschine ist **unbestimmt**.
- b) Eine Funktion  $f : M_1 \rightarrow M_2$  heißt **Turing-berechenbar**, wenn es eine **TM** gibt, die bei Eingabe von  $w \in \Sigma^*$  den Funktionswert  $f(w) \in B^*$  berechnet.
-

### Beispiele:

- a) Die Funktion  $f : \mathbb{N}_0 \rightarrow \mathbb{N}$  mit  $f(n) = n + 1$  ist Turing-berechenbar.
- b) Die Funktion  $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$  mit  $f(n_1, n_2) = n_1 + n_2$  ist ebenfalls Turing-berechenbar.

zu a)

$* \textcolor{blue}{1} \textcolor{blue}{1} \textcolor{blue}{1} \dots \textcolor{blue}{1} * * \dots$	$\Rightarrow$	$* \textcolor{blue}{1} \textcolor{blue}{1} \textcolor{blue}{1} \dots \textcolor{blue}{1} \textcolor{red}{1} * \dots$
$n$ mal Eins entspricht dem Wert $n$		$n + \textcolor{red}{1}$ mal Eins entspricht dem <b>Wert</b> $n + \textcolor{red}{1}$

zu b)

$* \textcolor{blue}{1} \textcolor{blue}{1} \textcolor{blue}{1} \dots \textcolor{blue}{1} *$	$\textcolor{green}{1} \textcolor{green}{1} \textcolor{green}{1} \dots \textcolor{green}{1} * * \dots$	$\Rightarrow$	$* \textcolor{blue}{1} \textcolor{blue}{1} \textcolor{blue}{1} \dots \textcolor{blue}{1} \textcolor{green}{1} \textcolor{green}{1} \textcolor{green}{1} \dots \textcolor{green}{1} * * \dots$
$\textcolor{blue}{\text{Wort 1}}$ entspricht $\textcolor{blue}{n}_1$	$\textcolor{green}{\text{Wort 2}}$ entspricht $\textcolor{green}{n}_2$		$\textcolor{blue}{\text{Wort 1 und Wort 2 direkt hintereinander}}$ entspricht dem <b>Wert</b> $\textcolor{blue}{n}_1 + \textcolor{green}{n}_2$



## VI. Entscheidbarkeit und Berechenbarkeit

1. Hauptfrage in diesem Kapitel
2. Vergleich zwischen Register- und Turingmaschine
3. Definition für einen Algorithmus
4. Definitionen für Entscheidungs- und Aufzählverfahren
5. Berechenbare Funktionen und entscheidbare Mengen
6. Definition der Turing-Berechenbarkeit
- 7. Die Church-Turing'sche These**
8. Das Post'sche Korrespondenzproblem
9. Erweiterungen der Turing-Maschine

### Satz (Church-Turing'sche These):

Jede (im intuitiven Sinne) berechenbare Funktion ist auch Turing-berechenbar und umgekehrt.

Dieser Satz ist nicht beweisbar; zur Aussage des Satzes hat aber noch niemand ein Gegenbeispiel angeben können.

### Folgerung:

Aufgrund der vorgenommenen Definition und der Church-Turing'sche These gilt nun:

Jede aufzählbare bzw. entscheidbare Menge ist auch Turing-aufzählbar bzw. Turing-entscheidbar und umgekehrt.

## VI. Entscheidbarkeit und Berechenbarkeit

1. Hauptfrage in diesem Kapitel
2. Vergleich zwischen Register- und Turingmaschine
3. Definition für einen Algorithmus
4. Definitionen für Entscheidungs- und Aufzählverfahren
5. Berechenbare Funktionen und entscheidbare Mengen
6. Definition der Turing-Berechenbarkeit
7. Die Church-Turing'sche These
- 8. Das Post'sche Korrespondenzproblem**
9. Erweiterungen der Turing-Maschine

### Definition (PKP):

Beim Post'schen Korrespondenzproblem (PKP) ist eine endliche Folge von Wortpaaren

$$K = ((x_1, y_1), (x_2, y_2), \dots, (x_n, y_n))$$

über einem endlichen Alphabet  $\Sigma$  gegeben.

Es gilt weiter  $x_i \notin \varepsilon$  und  $y_i \notin \varepsilon$ . Gefragt ist nun, ob es eine endliche Folge von Indizes  $i_1, i_2, \dots, i_k \in \{1, 2, \dots, n\}$  gibt, so dass gilt:

$$x_{i_1} x_{i_2} \dots x_{i_k} = y_{i_1} y_{i_2} \dots y_{i_k}$$

### Satz:

**Das Post'schen Korrespondenzproblem (PKP) ist nicht entscheidbar.**

---

### Beispiel 1:

$$K = ((1, 111), (10111, 10), (10, 0))$$

hat die Lösung (2, 1, 1, 3), denn es gilt:

$$x_2 x_1 x_1 x_3 = 10111 1 1 10 = 10 111 111 0 = y_2 y_1 y_1 y_3$$

### Beispiel 2:

$$K = ((10, 101), (011, 11), (101, 011))$$

hat keine Lösung.

### Beispiel 3:

$$K = ((001, 0), (01, 011), (01, 101), (10, 011))$$

hat eine Lösung der Länge 66.

---

## VI. Entscheidbarkeit und Berechenbarkeit

1. Hauptfrage in diesem Kapitel
2. Vergleich zwischen Register- und Turingmaschine
3. Definition für einen Algorithmus
4. Definitionen für Entscheidungs- und Aufzählverfahren
5. Berechenbare Funktionen und entscheidbare Mengen
6. Definition der Turing-Berechenbarkeit
7. Die Church-Turing'sche These
8. Das Post'sche Korrespondenzproblem
- 9. Erweiterungen der Turing-Maschine**

Folgende Erweiterungsmöglichkeiten existieren bei einer TM:

1. Mehrere Schreib-/Leseköpfe
2. Mehrere Bänder
3. Mehrere Schreib-/Leseköpfe für mehrere Bänder
4. Mehrdimensionale Bänder

### Satz:

Es hat sich gezeigt, dass keine dieser Erweiterungen mehr leistet, als eine „normale“ Turing-Maschine. Man kann z. B. beweisen, dass **jede k-Band Turing-Maschine durch eine 1-Band Turing-Maschine simuliert werden kann**. Das gleiche gilt auch für alle anderen angegebenen Modifikationen.

---