

# Theoretische Informatik II

## Einheit 5

### Theorie der Berechenbarkeit



1. Turing-Berechenbarkeit
2. Rekursive Funktionen
3. Funktionale und logische Programme
4. Elementare Berechenbarkeitstheorie
5. Unlösbare Probleme

# KERNFRAGEN ZUR BERECHENBARKEIT

- **Welche Berechnungsmethoden sind denkbar?**

- Es gibt weit mehr Modelle als nur die Standard PC Architektur
  - Lisp Maschinen, Parallelrechner, Neuronale Netze, (Quantencomputer)
- Sind die Modelle miteinander vergleichbar?

- **Welche allgemeingültigen Zusammenhänge gibt es?**

- Eigenschaften, die nicht vom Berechnungsmodell abhängen?
- Beweismethoden wie Abschlußeigenschaften und Problemtransformation

- **Gibt es Grenzen für den Einsatz von Computern?**

- Funktionen, die prinzipiell **nicht berechenbar** sind?
- Eigenschaften, die **unentscheidbar** sind?
- Sprachen, die nicht vollständig **aufgezählt** werden können?

**Mit welchen Techniken kann man dies beweisen?**

# ES GIBT VIELE MODELLE FÜR BERECHENBARKEIT ... SCHON LANGE VOR DEN ERSTEN COMPUTERN

- **Turingmaschine\*** (Rechnen mit Papier und Bleistift)
- **Nichtdeterministische Turingmaschine\*** (Parallelismus/Quantenrechner)
- **$\mu$ -rekursive Funktionen\*** (Mathematisches Rechnen)
- **$\lambda$ -Kalkül\*** (Funktionale Sprachen, LISP)
- **Logische Repräsentierbarkeit\*** (Logikprogrammierung, PROLOG)
- **Markov-Algorithmen (Typ-0 Grammatiken)** (Regelbasierte Sprachen)
- **Abakus** (Das älteste mechanische Hilfsmittel)
- **PASCAL-reduziert** (Imperative höhere Sprachen)
- **Registermaschine** (Assembler-/Maschinenprogrammierung)

**Viele Formalisierungen eines intuitiven Begriffes**

# Theoretische Informatik II

## Einheit 5.1

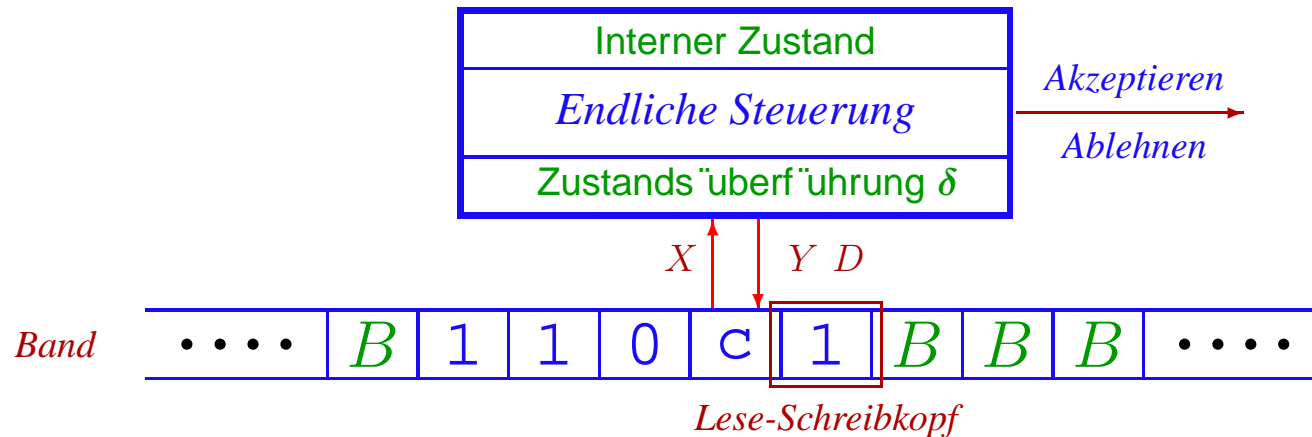
### Turing-Berechenbarkeit



1. Rückblick: Turingmaschinen und Sprachen
2. Turing-berechenbare Funktionen
3. Berechnen vs. Akzeptieren

# RÜCKBLICK: TURINGMASCHINEN

## DAS EINFACHSTE IMPERATIVE COMPUTERMODELL



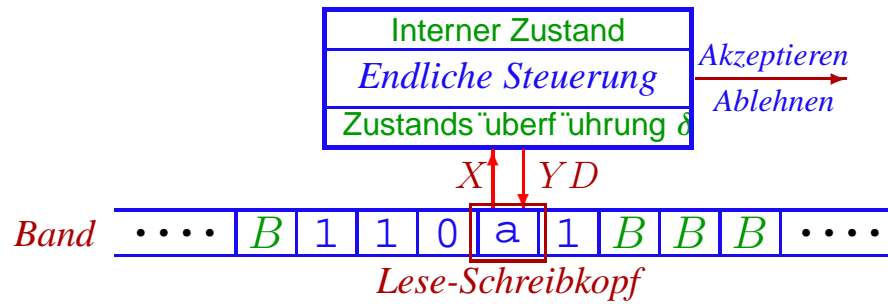
### ● Endlicher Automat + lineares Band

- Endliche Steuerung liest Bandsymbol unter **Lese-Schreibkopf**
- **Keine separate Eingabe**: Eingabewort steht zu Anfang auf Band

### ● Einfacher Verarbeitungsmechanismus

- Bandsymbol  $X$  wird gelesen
- Interner Zustand  $q$  wird zu  $q'$  verändert
- Neues Symbol  $Y$  wird auf das Band geschrieben
- **Kopf** wird in eine Richtung  $D$  (rechts oder links) bewegt

# RÜCKBLICK: TURINGMASCHINEN MATHEMATISCH



Deterministische **Turingmaschine**: 7-Tupel  $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

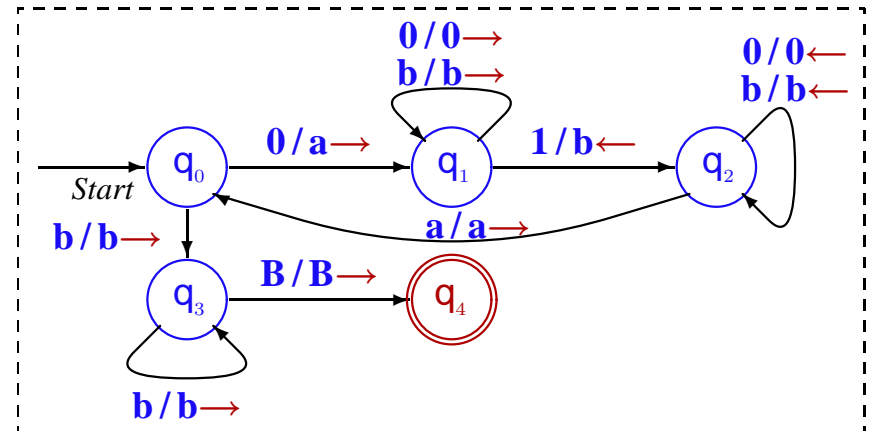
- $Q$  nichtleere endliche **Zustandsmenge**
- $\Sigma$  endliches **Eingabealphabet**
- $\Gamma \supseteq \Sigma$  endliches **Bandalphabet**
- $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$  (partielle) **Überföhrungsfunktion**
- $q_0 \in Q$  **Startzustand**
- $B \in \Gamma \setminus \Sigma$  **Leersymbol des Bands** (“blank”)
- $F \subseteq Q$  Menge von **akzeptierenden** (End-) **Zuständen**

**NTM** analog mit mengenwertigem  $\delta: Q \times \Gamma \rightarrow \mathcal{P}_e(Q \times \Gamma \times \{L, R\})$

# RÜCKBLICK: BESCHREIBUNG VON TURINGMASCHINEN

## • Übergangsdiagramme

- Zustände durch **Knoten** dargestellt
- $q_0$  markiert durch *Start-Pfeil*, Endzustände durch **doppelte Kreise**
- Für  $\delta(q, X) = (p, Y, D)$  hat das Diagramm eine **Kante**  $q \xrightarrow{X/YD} p$
- $\Sigma$  und  $\Gamma$  implizit durch Diagramm bestimmt, **Leersymbol** heißt  $B$



## • Übergangstabellen

- Funktionstabelle für  $\delta$   
— heißt “ $\delta$  nicht definiert”
- Pfeil  $\rightarrow$  kennzeichnet  $q_0$
- Stern  $*$  kennzeichnet  $F$
- $\Sigma$ ,  $\Gamma$  und  $B$  implizit bestimmt

$Q \setminus \Gamma$	0	1	a	b	B
$\rightarrow q_0$	$(q_1, a, R)$	—	—	$(q_3, b, R)$	—
$q_1$	$(q_1, 0, R)$	$(q_2, b, L)$	—	$(q_1, b, R)$	—
$q_2$	$(q_2, 0, L)$	—	$(q_0, a, R)$	$(q_2, b, L)$	—
$q_3$	—	—	—	$(q_3, b, R)$	$(q_4, B, R)$
$* q_4$	—	—	—	—	—

- **Konvention:**  $\delta(q, X)$  undefiniert für Endzustände  $q \in F$

# RÜCKBLICK: ARBEITSWEISE VON TURINGMASCHINEN

## ● **Konfiguration** $\hat{=}$ **Zustand** + **Bandinhalt** + **Kopfposition**

– Formal dargestellt als Tripel  $K = (u, q, v) \in \Gamma^* \times Q \times \Gamma^+$

·  $u, v$ : String links/rechts vom Kopf  $q$  Zustand

– Nur der bereits ‘besuchte’ Teil des Bandes wird betrachtet

**Blanks** am Anfang von  $u$  oder am Ende von  $v$  **entfallen, wo möglich**

*Achtung: im Buch wird das Tripel als ein (!) String  $uqv$  geschrieben*

## ● **Konfigurationsübergangsrelation** $\vdash^*$

–  $(uZ, q, Xv) \vdash (u, p, ZYv)$ , falls  $\delta(q, X) = (p, Y, L)$

–  $(u, q, Xv) \vdash (uY, p, v)$ , falls  $\delta(q, X) = (p, Y, R)$

Sonderfälle für Verhalten am Bandende

–  $(\epsilon, q, Xv) \vdash (\epsilon, p, BYv)$ , falls  $\delta(q, X) = (p, Y, L)$

–  $(uZ, q, X) \vdash (u, p, Z)$ , falls  $\delta(q, X) = (p, B, L)$

–  $(u, q, X) \vdash (uY, p, B)$ , falls  $\delta(q, X) = (p, Y, R)$

–  $(\epsilon, q, Xv) \vdash (\epsilon, p, v)$ , falls  $\delta(q, X) = (p, B, R)$

$K_1 \vdash^* K_2$ , falls  $K_1 = K_2$  oder es gibt ein  $K$  mit  $K_1 \vdash K$  und  $K \vdash^* K_2$

Definition analog für nichtdeterministische Maschinen



# RÜCKBLICK: SPRACHE EINER TURINGMASCHINE

- **Akzeptierte Sprache**

- Menge der Eingaben, für die  $\vdash^*$  zu akzeptierendem Zustand führt

$$L(M) = \{w \in \Sigma^* \mid \exists p \in F. \exists u, v \in \Gamma^*. (\epsilon, q_0, w) \vdash^* (u, p, v)\}$$

- Bei Einhalten der Konvention hält  $M$  im akzeptierenden Zustand an

Definition identisch für nichtdeterministische Maschinen

DTMs akzeptieren dieselben Sprachen wie NTMs (exponentielle Simulation)

- **Semi-entscheidbare Sprache**  $\hat{=}$  **Typ-0 Sprache**

- Sprache, die von einer Turingmaschine  $M$  akzeptiert wird

- Alternative Bezeichnung: **(rekursiv) aufzählbare Sprache**

- **Entscheidbare Sprache** (auch: **rekursive Sprache**)

- Sprache, die von einer Turingmaschine  $M$  akzeptiert wird,  
die bei jeder Eingabe terminiert

## RÜCKBLICK: PROGRAMMIERTECHNIKEN FÜR TURINGMASCHINEN

- **Datenregister speichern Werte aus Menge  $\Delta$** 
  - Simulation durch erweiterte Zustandsmenge  $Q' := Q \times \Delta^k$
- **Mehrspur-Maschinen mit  $k$  Datenspuren**
  - Simulation durch erweitertes Bandalphabet  $\Sigma' := \Sigma^k$
- **Mehrband-Maschinen mit  $k$  unabhängigen Bändern**
  - Simulation mit  $2k+1$  Spuren: Inhalt, Kopfmarker + Endmarker
- **Unterprogramme**
  - Simulation wie bei Unterprogrammen in Assemblersprachen
- **Beschränkte Modelle für Beweise**
  - Halbseitig unendliches Band kann beidseitiges Band simulieren
  - Binäres Bandalphabet  $\Gamma = \{1, B\}$  kann jedes Alphabet codieren
  - 2 Stacks können jede Konfiguration einer Turingmaschine simulieren

**Genauso leistungsfähig wie konventionelle Computer**

# NEU: ZEIT- UND PLATZBEDARF VON TURINGMASCHINEN

- **Rechenzeit**  $t_M(w)$

- Anzahl der Konfigurationsübergänge bis  $M$  bei Eingabe  $w$  anhält

- **Speicherbedarf**  $s_M(w)$

- Anzahl der Bandzellen, die  $M$  während der Berechnung aufsucht

- **Komplexität: Bedarf relativ zur Größe**

- $T_M(n) = \max\{t_M(w) \mid |w|=n\}$

Maximaler Bedarf relativ zur Länge  
eines Eingabewortes (worst-case)

- $S_M(n) = \max\{s_M(w) \mid |w|=n\}$

- Die **Größenordnung** der Funktionen (linear, quadratisch, kubisch,...)

ist **aussagekräftiger** als die genauen Werte

→ **Komplexitätstheorie (§6)**

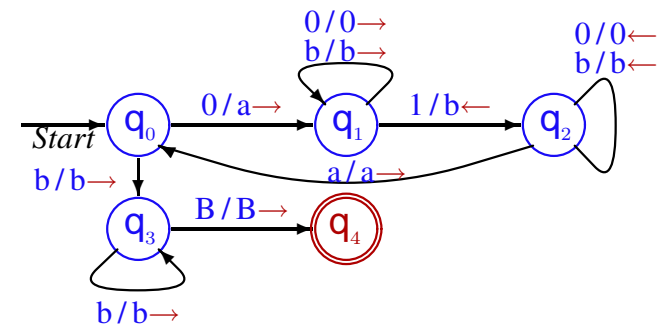
- **Komplexität der Turingmaschine für  $\{0^n 1^n \mid n \geq 1\}$**

- Zeitaufwand für Schleife  $q_0, q_1, q_2, q_0$ :  $2n$

- Gesamter **Zeitaufwand quadratisch** ( $2n^2$ )

- Platzbedarf nicht größer als die Eingabe

- **Lineare Speicherplatzkomplexität**



# DIE BERECHNETE FUNKTION EINER TURINGMASCHINE

- **Turingmaschinen berechnen Funktionen auf  $\Sigma^*$**

- Eingabe der Funktion wird aufs Band geschrieben
- Bandinhalt wird durch Abarbeitung des Programms verändert
- Wenn Maschine anhält, kann Bandinhalt ausgegeben werden
- Akzeptierende Endzustände werden irrelevant (üblicherweise  $F = \emptyset$ )

Die ursprünglich vorgesehene Verwendung von Turingmaschinen

- **Formale Beschreibung mittels Konfigurationen**

- Anfangskonfiguration:  $\alpha(w) := (\epsilon, q_0, w)$
  - Rechenzeit:  $t_M(w) := \max\{j \mid \alpha(w) \vdash^j (u, q, Xv) \wedge \delta(q, X) \text{ undefiniert}\}$   
Undefiniert falls dieses Maximum nicht existiert, d.h.  $M$  hält nicht
  - Ausgabefunktion:  $\omega(u, q, v) := v|_{\Sigma}$  (längster Präfix von  $v$ , der zu  $\Sigma^*$  gehört)  
Ausgabe beginnt unter dem Kopf bis ein Symbol nicht aus  $\Sigma$  erreicht wird
  - Berechnete Funktion:  $f_M(w) := \{\omega(\kappa) \mid \alpha(w) \vdash^{t_M(w)} \kappa\}$   
Genau dann definiert, wenn  $M$  auf  $w$  anhält
- Für DTMs ist  $f_M(w) = \omega(\kappa)$  für das eindeutig bestimmte  $\kappa$  mit  $\alpha(w) \vdash^{t_M(w)} \kappa$

# BERECHNUNG MIT TURING-MASCHINEN AM BEISPIEL

- $M_1 = (\{q_0, q_1, q_2\}, \{1\}, \{1, B\}, \delta, q_0, B, \{q_2\})$  mit

$\delta_1$	1	B
→ $q_0$	( $q_0, 1, R$ )	( $q_1, 1, L$ )
$q_1$	( $q_1, 1, L$ )	( $q_2, B, R$ )
* $q_2$	—	—

Abarbeitungsbeispiel:  $(\epsilon, q_0, 111) \vdash^8 (\epsilon, q_2, 1111)$

Fügt am Ende eines Wortes  $w \in \{1\}^*$  eine 1 an (“Bierdeckelmaschine”)

- **Mathematische Analyse:**

- Anfangskonfiguration:  $\alpha(1^n) = (\epsilon, q_0, 1^n)$
- Nachfolgekonfigurationen:  $\alpha(1^n) \vdash (1, q_0, 1^{n-1}) \vdash^{n-1} (1^n, q_0, B)$   
 $\vdash (1^{n-1}, q_1, 1) \vdash^n (\epsilon, q_1, B1^{n+1}) \vdash (\epsilon, q_2, 1^{n+1})$
- Terminierung:  $\max\{j \mid \alpha(w) \vdash^j (u, q, Xv) \wedge \delta(q, X) \text{ undefiniert}\}$   
 $= 2n+2$
- Ergebnis:  $\alpha(1^n) \vdash^{2n+2} (\epsilon, q_2, 1^{n+1})$
- Ausgabefunktion:  $\omega(\epsilon, q_2, 1^{n+1}) = 1^{n+1}$

$f_{M_1}(1^n) = 1^{n+1}$  für alle  $n$ , Definitionsbereich  $\{1\}^*$ , Wertebereich  $\{1\}^+$

# BEISPIELE FÜR TURING-MASCHINEN

- $M_2 = (\{q_0, q_1\}, \{1\}, \{1, B\}, \delta_2, q_0, B, \{q_1\})$

mit

$\delta_2$	1	B
$\rightarrow q_0$	$(q_0, B, R)$	$(q_1, B, L)$
$* q_1$	—	—

Abarbeitungsbeispiel:

$$(\epsilon, q_0, 111) \vdash^4 (\epsilon, q_1, B)$$

Löscht ein Wort vom Band:  $f_{M_2}(w) = \epsilon$  für alle  $w \in \{1\}^*$

- $M_3 = (\{q_0, q_1, q_2\}, \{1\}, \{1, B\}, \delta_3, q_0, B, \{q_2\})$

mit

$\delta_3$	1	B
$\rightarrow q_0$	$(q_1, 1, R)$	$(q_2, B, R)$
$q_1$	$(q_0, 1, R)$	$(q_1, B, R)$
$* q_2$	—	—

Abarbeitungsbeispiele:

$$(\epsilon, q_0, 1111) \vdash^5 (1111B, q_2, B)$$

$$(\epsilon, q_0, 111) \vdash^n (111BBB\dots BB, q_1, B)$$

Testet, ob Anzahl der Einsen in  $w \in \{1\}^*$  gerade ist

$$f_{M_3}(1^n) = \begin{cases} \epsilon & \text{falls } n \text{ gerade,} \\ \perp & \text{sonst} \end{cases} \quad (\perp \text{ steht für "undefiniert"})$$

## BEISPIELE FÜR TURING-MASCHINEN II

- $M_4 = (\{q_0, q_1, q_2, q_3, q_4\}, \{1\}, \{1, B\}, \delta_4, q_0, B, \{q_4\})$

mit

$\delta_4$	1	B
$\rightarrow q_0$	$(q_0, 1, R)$	$(q_1, B, L)$
$q_1$	$(q_2, B, R)$	$(q_4, B, R)$
$q_2$	$(q_2, 1, R)$	$(q_3, 1, L)$
$q_3$	$(q_3, 1, L)$	$(q_1, 1, L)$
* $q_4$	—	—

Abarbeitungsbeispiel:

$$(\epsilon, q_0, 11) \vdash^{14} (\epsilon, q_4, 1111)$$

Verdoppelt Anzahl der Einsen:  $f_{M_4}(1^n) = 1^{2n}$

- $M_5 = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta_5, q_0, B, \{q_3\})$

mit

$\delta_5$	0	1	B
$\rightarrow q_0$	$(q_0, 0, R)$	$(q_0, 1, R)$	$(q_1, B, L)$
$q_1$	$(q_2, 1, L)$	$(q_1, 0, L)$	$(q_2, 1, L)$
$q_2$	$(q_2, 0, L)$	$(q_2, 1, L)$	$(q_3, B, R)$
* $q_3$	—	—	—

Abarbeitungsbeispiel:

$$(\epsilon, q_0, 10011) \vdash^{12} (\epsilon, q_3, 10100)$$

Addiert 1 auf die Binärdarstellung einer natürlichen Zahl

# TURING-BERECHENBARE FUNKTIONEN

- $f: \Sigma^* \rightarrow \Delta^*$  **Turing-berechenbar**

–  $f = f_M$  für eine Turingmaschine  $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$  mit  $\Delta \subseteq \Gamma$

$\mathcal{T}$ : Menge der Turing-berechenbaren Funktionen

- **Berechenbarkeit auf Zahlen:**  $f: \mathbb{N} \rightarrow \mathbb{N}$

$\hat{=}$  Berechenbarkeit von  $f_r: \Sigma^* \rightarrow \Sigma^*$  mit  $f_r(w) = r(f(r^{-1}(w)))$

wobei  $r: \mathbb{N} \rightarrow \Sigma^*$  **injektive Repräsentation** von Zahlen durch Wörter

· **unäre** Darstellung  $r_u: \mathbb{N} \rightarrow \{1\}^*$  mit  $r_u(n) = 1^n$

· **binäre** Codierung  $r_b: \mathbb{N} \rightarrow \{0, 1\}^*$  (ohne führende Nullen)

**Berechenbarkeit auf anderen Mengen analog**



# BERECHENBARKEIT ARITHMETISCHER FUNKTIONEN

- **Nachfolgerfunktion**  $s:\mathbb{N}\rightarrow\mathbb{N}$  mit  $s(n) = n+1$ 
  - Bei unärer Codierung: berechne  $s_u:\{1\}^*\rightarrow\{1\}^*$  mit  $s_u(1^n) = 1^{n+1}$ 
    - Turingmaschine muß eine 1 anhängen:  $s_u = f_{M_1}$
  - Bei binärer Codierung:  $s_b = f_{M_5}$ 
    - $M$  muß Ziffern von rechts beginnend umwandeln, ggf. mit Übertrag
- **Division durch 2:**  $div_2:\mathbb{N}\rightarrow\mathbb{N}$  mit  $div_2(n) = \lfloor n/2 \rfloor$ 
  - Bei unärer Codierung:  $M$  muß (analog zu  $M_4$ ) je zwei Einsen löschen und eine neue hinter dem Ende des Wortes schreiben
  - Bei binärer Codierung:  $M$  muß nur die letzte Ziffer löschen

**Komplexere arithmetische Operationen benötigen  
Programmiertechniken für Turingmaschinen**

# AKZEPTIEREN ODER BERECHNEN?

- **Jede Funktion ist als Menge beschreibbar**

- **graph**( $f$ ) =  $\{(x, y) \mid f(x) = y\}$

Akzeptierende Maschinen erkennen Graphen berechenbarer Funktionen

**$f$  berechenbar  $\Leftrightarrow$  graph( $f$ ) semi-entscheidbar**  $\mapsto$  nächste Folie

- **Jede Menge ist als Funktion beschreibbar**

- $\chi_L(w) = \begin{cases} 1 & \text{falls } w \in L, \\ 0 & \text{sonst} \end{cases}$  **Charakteristische Funktion** der Sprache  $L$

- $\psi_L(w) = \begin{cases} 1 & \text{falls } w \in L, \\ \perp & \text{sonst} \end{cases}$  **Partiell-charakteristische Funktion** von  $L$

Charakteristische Funktionen erkannter Sprachen sind berechenbar

**$L$  semi-entscheidbar  $\Leftrightarrow \psi_L$  berechenbar**

**$L$  entscheidbar  $\Leftrightarrow \chi_L$  berechenbar**  $\mapsto$  nächste Folie

## Simuliere Abarbeitung der jeweils anderen Maschine

- **$f$  berechenbar  $\Leftrightarrow \text{graph}(f)$  semi-entscheidbar**

$\Rightarrow$  : Bei Eingabe  $(w, v)$  “teste” ob  $f(w) = v$  ergibt

$\Leftarrow$  : Bei Eingabe  $w$  suche das “erste” Wort  $v$  mit  $(w, v) \in \text{graph}(f)$

Suche muß Werte für  $w, v$  und Rechenzeitgrenze simultan durchlaufen !!

Maschinen müssen nicht bei jeder Eingabe anhalten

- **$L$  semi-entscheidbar  $\Leftrightarrow \psi_L$  berechenbar**

$\Rightarrow$  : Bei Eingabe  $w$  “teste” ob  $w$  akzeptiert wird und gebe ggf. 1 aus

$\Leftarrow$  : Bei Eingabe  $w$  “teste” ob  $\psi_L(w) = 1$  ergibt

Maschinen müssen nicht bei jeder Eingabe anhalten

- **$L$  entscheidbar  $\Leftrightarrow \chi_L$  berechenbar**

– Wie oben, aber beide Maschinen müssen bei jeder Eingabe anhalten