

Theoretische Informatik II

Einheit 4

Allgemeine Berechenbarkeit



1. Turingmaschinen und Typ-0 Sprachen

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

THEORIE DER BERECHENBARKEIT

- **Welche Arten von Berechnung sind denkbar?**
 - Es gibt weit mehr Modelle als nur die Standard PC Architektur
 - Lisp Maschinen, Parallelrechner, Neuronale Netze, (Quantencomputer)
 - Sind manche Modelle leistungsstärker als andere?
- **Welche allgemeingültigen Zusammenhänge gibt es?**
 - Welche Grundeigenschaften muß jedes Berechnungsmodell erfüllen?
 - Wie kann man Lösungen schematisch wiederverwenden?
Abschlußigenschaften und Problemtransformation (“Reduktion”)
- **Wo liegen die Grenzen für Computer?**
 - Gibt es Funktionen, die prinzipiell nicht berechenbar sind?
 - Eigenschaften, die unentscheidbar sind?
 - Sprachen, deren Elemente nicht vollständig aufgezählt werden können?

Mit welchen Techniken kann man dies beweisen?

WARUM BRAUCHT MAN BERECHENBARKEITSMODELLE?

Welche der folgenden Funktionen ist berechenbar? Warum?

$$f(x) = \begin{cases} 1 & \text{wenn ein Anfangssegment der Dezimalentwicklung von } \pi \\ & \text{(unter Ignorierung des Punktes) identisch mit } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

$$g(x) = \begin{cases} 1 & \text{wenn ein beliebiges Teilsegment der Dezimalentwicklung} \\ & \text{von } \pi \text{ identisch mit der Zahl } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

$$h(x) = \begin{cases} 1 & \text{wenn in der Dezimalentwicklung von } \pi \text{ mindestens} \\ & x \text{ aufeinanderfolgende Neunen vorkommen,} \\ 0 & \text{sonst} \end{cases}$$

$$\pi = 3.1415952653589789845199165029043797403573989868\dots$$

Keine Aussage möglich ohne Präzisierung des Begriffs

ES GIBT VIELE MODELLE FÜR BERECHENBARKEIT ... LANGE BEVOR ES DIE ERSTEN COMPUTER GAB

- **Turingmaschine*** (Rechnen mit Papier und Bleistift)
- **Nichtdeterministische Turingmaschine*** (Parallelismus/Quantenrechner)
- **μ -rekursive Funktionen*** (Mathematisches Rechnen)
- **λ -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

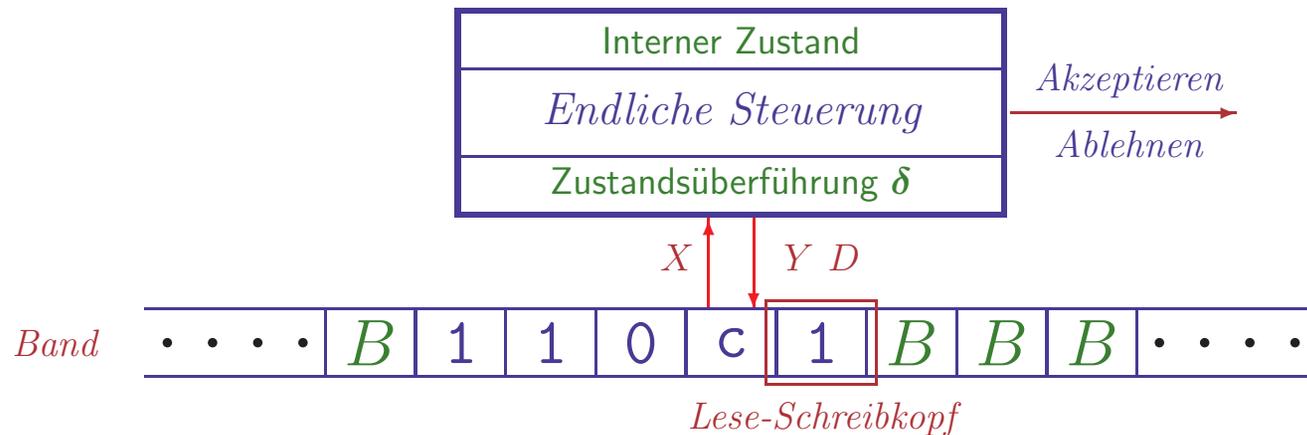
Einheit 4.2

Turing-Berechenbarkeit



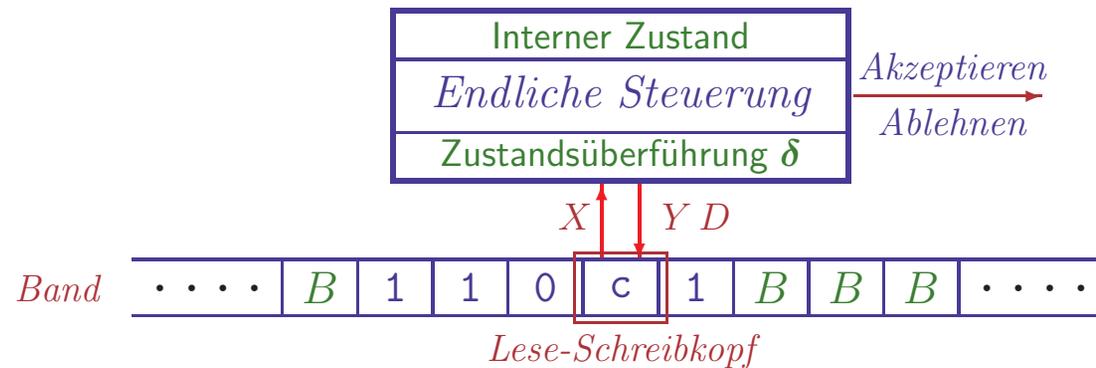
1. Rückblick: Turingmaschinen und Sprachen
2. Turing-berechenbare Funktionen
3. Zusammenhang: Berechnen vs. Akzeptieren

RÜCKBLICK: TURINGMASCHINEN INTUITIV



- **Endlicher Automat + lineares Band**
 - Endliche Steuerung liest Eingabesymbole
 - Gleichzeitig wird Bandsymbol unter **Lese-Schreibkopf** gelesen
- **Vereinfachung: keine separate Eingabe**
 - Eingabewort steht zu Anfang bereits auf dem 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 PRÄZISIERT



Eine **Turingmaschine** ist ein 7-Tupel

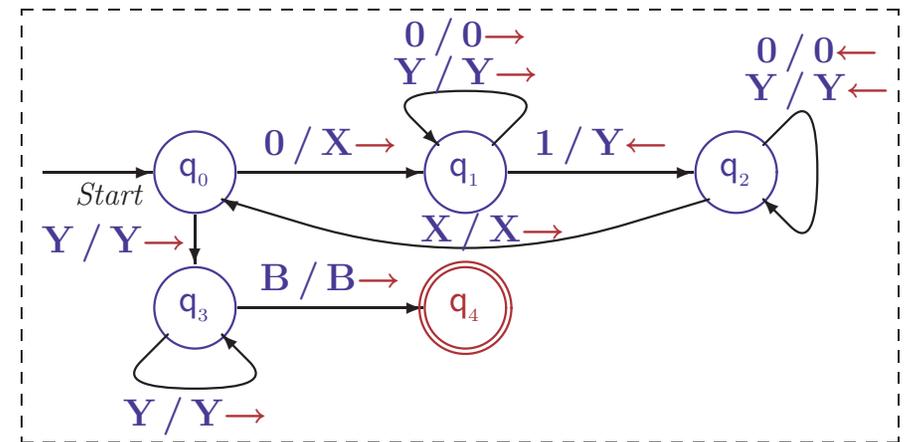
$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ mit

- Q nichtleere endliche **Zustandsmenge**
- Σ 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**

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 $q \xrightarrow{X/YD} p$
- Σ und Γ implizit durch Diagramm bestimmt, Leersymbol heißt B



● Übergangstabellen

- Funktionstabelle für δ
 - — heißt “ δ nicht definiert”
- Pfeil \rightarrow kennzeichnet q_0
- Stern $*$ kennzeichnet F
- Σ , Γ und B implizit bestimmt

$Q \setminus \Gamma$	0	1	X	Y	B
$\rightarrow q_0$	(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_3, Y, R)	(q_4, B, R)
$* q_4$	—	—	—	—	—

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

RÜCKBLICK: ARBEITSWEISE VON TURINGMASCHINEN

● Erweiterter Begriff der Konfiguration

- Zustand q , Inhalt des Bandes und Kopfposition
- Formal dargestellt als Tripel $K = (u, q, v) \in \Gamma^* \times Q \times \Gamma^+$
 - u, v : String links/rechts vom Kopf
- Nur der bereits ‘besuchten’ 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

- $(\epsilon, q, Xv) \vdash (\epsilon, p, BYv)$, 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 $K \vdash^* K_2$

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

● Semi-entscheidbare Sprache

- Sprache, die von einer Turingmaschine M akzeptiert wird
- Alternative Bezeichnung: **(rekursiv) aufzählbare Sprache**
- Äquivalent zu Typ-0 Sprachen

● Entscheidbare Sprache

- Sprache, die von einer Turingmaschine M akzeptiert wird, die bei jeder Eingabe terminiert
- Alternative Bezeichnung: **rekursive Sprache**

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

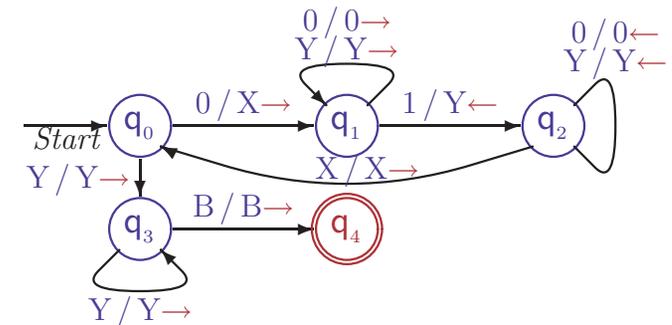
- **Komplexität der Turingmaschine für $\{0^n 1^n \mid n \geq 1\}$**

- Zeitaufwand für Schleife q_0, q_1, q_2, q_3 : $2n$

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

- Platzbedarf nicht größer als die Eingabe

- Lineare Speicherplatzkomplexität



DIE BERECHNETE FUNKTION EINER TURINGMASCHINE

- **Turingmaschinen berechnen Funktionen auf Σ^***

- Eingabe der Funktion wird aufs Band geschrieben
- Bandinhalt wird durch Abarbeitung des Programms verändert
- Wenn Maschine anhält, kann Bandinhalt ausgegeben werden

Die ursprünglich vorgesehene Verwendung von Turingmaschinen

- **Beschreibung mittels Konfigurationen**

- Anfangskonfiguration: $\alpha(w) := (\epsilon, q_0, w)$
- Rechenzeit $t_M(w) := \max\{j \mid \alpha(w) \vdash^j (u, q, Xv) \wedge \delta(q, X) \text{ undefiniert}\}$
Rechenzeit ist undefiniert falls dieses Maximum nicht existiert (d.h. M hält nicht)
- Ausgabefunktion: $\omega(u, q, v) := v|_{\Sigma}$ (längster Präfix von v der zu Σ^* gehört)
Ausgabe beginnt unter dem Kopf bis ein Symbol nicht aus Σ erreicht wird
- Berechnete Funktion: $f_M(w) := \omega(\kappa)$, wobei $\alpha(w) \vdash^{t_M(w)} \kappa$
 $f_M: \Sigma^* \rightarrow \Sigma^*$ ist wohldefiniert, da κ durch $\alpha(w) \vdash^{t_M(w)} \kappa$ eindeutig bestimmt

BERECHNUNG MIT TURING-MASCHINEN AM BEISPIEL

- $M_1 = (\{q_0, q_1, q_2\}, \{1\}, \{1, B\}, \delta_1, q_0, B, \{q_2\})$ mit

δ_1	1	B
$\rightarrow q_0$	($q_0, 1, R$)	($q_1, 1, L$)
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: $(\epsilon, q_0, 1^n) \vdash (1, q_0, 1^{n-1}) \vdash^{n-1} (1^n, q_0, B)$
 $\vdash (1^{n-1}, q_1, 11) \vdash^n (\epsilon, q_1, B1^{n+1}) \vdash (\epsilon, q_2, 1^{n+1})$
- Terminierung: $\min\{j \mid \alpha(w) \vdash^j (u, q, Xv) \wedge \delta(q, X) \text{ undefiniert}\} = 2n + 2$
- Ergebnis: $(\epsilon, q_0, 1^n) \vdash^{2n+2} (\epsilon, q_2, 1^{n+1})$
- Ausgabefunktion: $\omega(\epsilon, q_2, 1^{n+1}) = 1^{n+1}$

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

BEISPIELE FÜR TURING-MASCHINEN

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

mit

δ_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

δ_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

δ_4	1	B
→ 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

δ_5	0	1	B
→ 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**

RÜCKBLICK: PROGRAMMIERTECHNIKEN FÜR TURINGMASCHINEN

- **Datenregister speichern Werte aus Menge Δ**
 - Simulation durch erweiterte Zustandsmenge $Q' := Q \times \Delta^k$
- **Mehrspur-Maschinen mit k Datenspuren**
 - Simulation durch erweitertes Bandalphabet $\Sigma' := \Sigma^k$
- **Mehrband-Maschinen**
 - Simulation von k unabhängigen Bändern mit $2k+1$ Spuren
 - Je eine Spur für Kopfposition eines Bandes + Endmarker für Suche
- **Unterprogramme**
 - Simulation wie bei Unterprogrammen in Assemblersprachen
- **Beschränkte Modelle für Beweise**
 - Halbseitig unendliches Band kann beidseitiges Band simulieren
 - Jedes Alphabet kann über Bandalphabet $\Gamma = \{1, B\}$ codiert werden
 - 2 Stacks können jede Konfiguration einer Turingmaschine simulieren

Genauso leistungsfähig wie konventionelle Computer

AKZEPTIEREN ODER BERECHNEN?

- **Jede Funktion ist als Menge beschreibbar**

- **graph(f)** = $\{(x, y) \mid f(x) = y\}$
- Akzeptierende Maschinen erkennen Graphen berechenbarer Funktionen

Satz: **f berechenbar \Leftrightarrow graph(f) semi-entscheidbar**

- **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

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

L entscheidbar $\Leftrightarrow \chi_L$ berechenbar

AKZEPTIEREN VS. BERECHNEN: BEWEISIDEEN

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

TURINGMASCHINEN IM RÜCKBLICK

- **Allgemeinstes Maschinenmodell**

- Deterministischer endlicher Automat mit unendlichem Speicherband
- “Beliebiger” Zugriff auf Speicherzellen
- Erkennung von Wörtern durch Endzustand
- Berechnen von Werten durch Ausgabe nach Terminierung
- Beide Modelle sind gleich mächtig

- **Verhaltensanalyse durch Konfigurationsübergänge**

- Konfigurationen beschreiben Zustand, Bandinhalt & Kopfposition

- **Äquivalent zu realen Computern**

- Register, mehrere Bänder, Unterprogramme, etc. simulierbar

- **Äquivalent zu Typ-0 Grammatiken**

- Umwandlung von Konfigurationsübergängen in Regeln und umgekehrt

- **Nichtdeterministische Variante ist gleich stark**

- Simulationsaufwand durch deterministische Maschine ist exponentiell