Grenzen der Algorithmisierung

- Welche Probleme sind beweisbar nicht durch Algorithmen zu lösen?
  - mathematische Grundbegriffe
  - bestimme Anzahl der Probleme
  - bestimme Anzahl der Algorithmen
  - allgemeine Aussagen zu algorithmisch unlösbaren Problemen

Grenzen der Algorithmisierung
Mathematische Grundbegriffe

- **Menge**: Zusammenfassung von wohlunterscheidbaren Objekten zu einem ganzen
- **Elemente**: Objekt x Element einer Menge M: $x \in M$.

  Gehört x nicht zu M: $x \notin M$

- Menge mit endlich vielen Elemente $x_1, x_2, ..., x_n$: $M = \{x_1, x_2, ..., x_n\}$.
  - Beispiel: Spielkartenfarben: SF={Kreuz,Pik,Herz,Karo}
  - EQ: 10 Quadratzahlen: EQ={1,4,9,16,25,36,49,64,81,100}.
- leere Menge $\varnothing$.

Mathematische Grundbegriffe
Mengen

- Definition von Mengen über Eigenschaften:

  $\{f(x) \mid \text{Eigenschaften von } x\}$.

  - EQ=$\{x \mid x \text{ ist Quadratzahl}, 1 \leq x \leq 100\}$
  - EQ=$\{x^2 \mid 1 \leq x \leq 10\}$
- $M \subseteq M'$: M **Teilmenge** von M'
- $M' \supseteq M$: M' **Obermenge** von M
- $M = M' \Leftrightarrow (M \subseteq M' \text{ und } M' \subseteq M)$

Mathematische Grundbegriffe
Mengen

- Definition von Mengen über Eigenschaften:

  {f(x) | Eigenschaften von x}.

  - EQ={x | x ist Quadratzahl, $1 \leq x \leq 100$}
  - EQ={x^2 | $1 \leq x \leq 10$}

- M⊆M': M **Teilmenge**
- M'⊇M: M' **Obermeng**
- M=M' ⇔ (M⊆M' und M'⊆M)

Äquivalenzpfeil ⇔

... genau dann, wenn ...
... dann und nur dann ...

Mathematische Grundbegriffe
Teilmengen - Obermengen

- $M \subset M'$: M **echte Teilmenge** von M'
  $\neq$
- $M' \supset M$: M' **echte Obermenge** von M
  $\neq$

Mathematische Grundbegriffe
Mengenoperationen

- **Vereinigung**: $M \cup M' = \{x \mid x \in M \text{ oder } x \in M'\}$
- **Durchschnitt**: $M \cap M' = \{x \mid x \in M \text{ und } x \in M'\}$
- **Differenz**: $M \setminus M' = \{x \mid x \in M \text{ und } x \notin M'\}$
- M fest gewählt: $M \setminus M'$ **Komplement** von M'

Mathematische Grundbegriffe
Standardmengen

- IN: natürliche Zahlen incl. 0 (oft auch $\mathbb{N}_0$)
- $\mathbb{Z}$: ganze Zahlen
- $\mathbb{Q}$: rationale Zahlen
- IR: reelle Zahlen

Mathematische Grundbegriffe
kartesisches Produkt

- kartesisches Produkt $M_1 \times M_2$: Menge aller geordneten Paare

$$M_1 \times M_2 = \{(x,y) \mid x \in M_1 \text{ und } y \in M_2\}.$$

- Menge der geordneten n-Tupel:

$$M_1 \times M_2 \times \ldots \times M_n = \{(x_1, x_2, \ldots, x_n) \mid x_i \in M_i \text{ für } i = 1, \ldots, n\}.$$

- alle Mengen M_i gleich einer Menge M, so schreibt man M^n

Mathematische Grundbegriffe
Folgen

- M^* Menge der endlichen Folgen über M: Menge aller geordneten n-Tupel über M für alle $n \in \mathbb{N}$:

$$M^* = M^0 \cup M^1 \cup M^2 \cup M^3 \cup \ldots = \bigcup_{n \geq 0} M^n$$

unendliche Vereinigung

- M^0 und M^1:

$$M^1 = M,\ M^0 = \{()\}.$$

$M^0 \neq \emptyset$!

Mathematische Grundbegriffe
Wörter

- Schreibweise (ohne Komma und Klammern):

  $M^* = \{x_1 x_2 \ldots x_n \mid n \in \mathbb{N}, x_i \in M \text{ für } i = 1, \ldots, n\}$

- $w = x_1 x_2 \ldots x_n \in M^*$ **Wörter**
- **leeres Wort** $\epsilon \in M^*$
- $|w| = n$ **Länge** des Wortes, $|\epsilon| = 0$

Mathematische Grundbegriffe
Konkatenation - Monoid

- Operation $\cdot$ auf M^*: $u,v \in M^*$: $u \cdot v = uv$
- Assoziativgesetz: $(uv)w = u(vw)$ für alle $u,v,w \in M^*$
- neutrales Element: $u\epsilon = \epsilon u = u$ für alle $u \in M^*$
- $|uv| = |u| + |v|$
- $(M^*, \cdot)$ Monoid (Halbgruppe mit Einselement)

Mathematische Grundbegriffe
Potenzmenge

- Potenzmenge 2^M Menge aller Teilmengen von M: $2^M=\{K \mid K \subseteq M\}$
- $2^M \neq \emptyset$, da stets $M \in 2^M$
- Beispiele:
  - $2^{\emptyset}=\{\emptyset\}$
  - $2^{2^{\emptyset}}=2^{\{\emptyset\}}=\{\emptyset,\{\emptyset\}\}$
- M und {M} verschieden!

Mathematische Grundbegriffe
Relationen

- zweistellige Relation R über einer Menge M: Teilmenge von M×M.
- Beispiel: Gleichheit auf M:

$$G \subseteq M \times M \text{ mit } G = \{(x,x) \mid x \in M\}$$

- Schreibe x=y, falls $(x,y) \in G$ ist

Mathematische Grundbegriffe
Funktionen

- *partielle* Funktion von M nach K: Teilmenge von M×K mit: zu jedem x∈M höchstens ein y∈K mit (x,y)∈f
- Statt f⊆M×K schreibe f: M→K
- f(x)=y *Funktionswert* für *Argument* x
- f: M→K **total**, wenn es zu jedem x∈M ein y∈K mit y=f(x) gibt
- anderenfalls f **partiell**

Mathematische Grundbegriffe
Funktionen

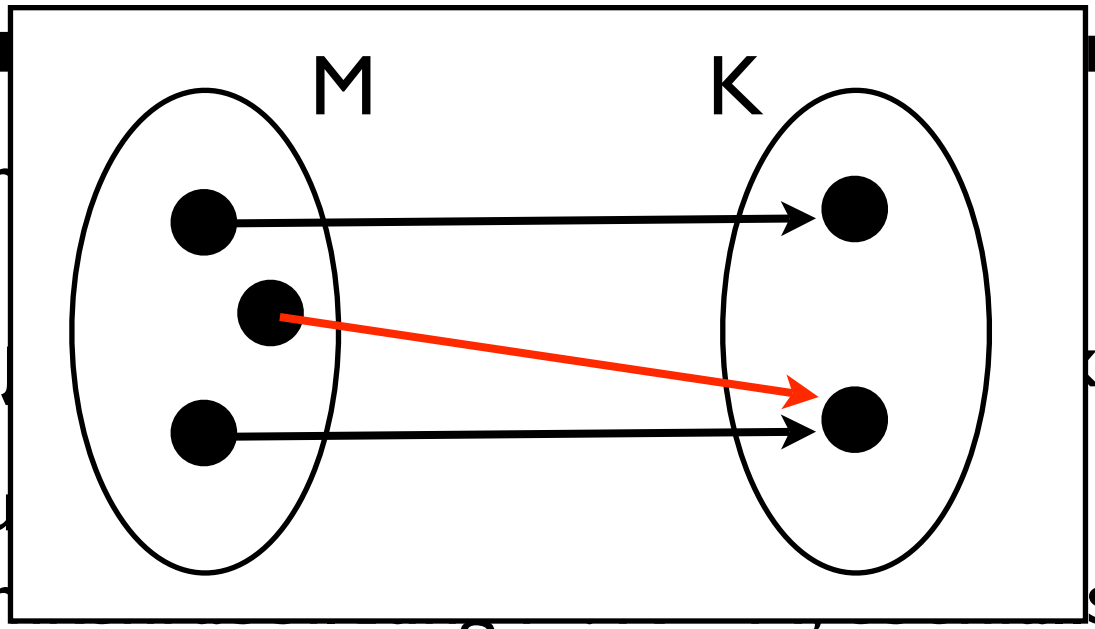
Eine totale Funktion $f: M \to K$ heißt

- **injektiv**, wenn aus $f(x_1)=f(x_2)$ stets $x_1=x_2$ folgt
- **surjektiv**, wenn es zu jedem $y \in K$ mindestens ein $x \in M$ mit $f(x)=y$ gibt
- **bijektiv**, wenn f injektiv und surjektiv ist
- Zu bijektiver Funktion $f: M \to K$ stets Umkehrabbildung $f^{-1}: K \to M$, ebenfalls bijektiv. Setze: $f^{-1}(y)=x \Leftrightarrow f(x)=y$.

Mathematische Grundbegriffe
Funktionen - injektiv

Eine totale Funktion $f: M \to K$ heißt

- **injektiv**, wenn aus $f(x_1)=f(x_2)$ stets $x_1=x_2$ folgt
- **su**
- **bi**
- Zu



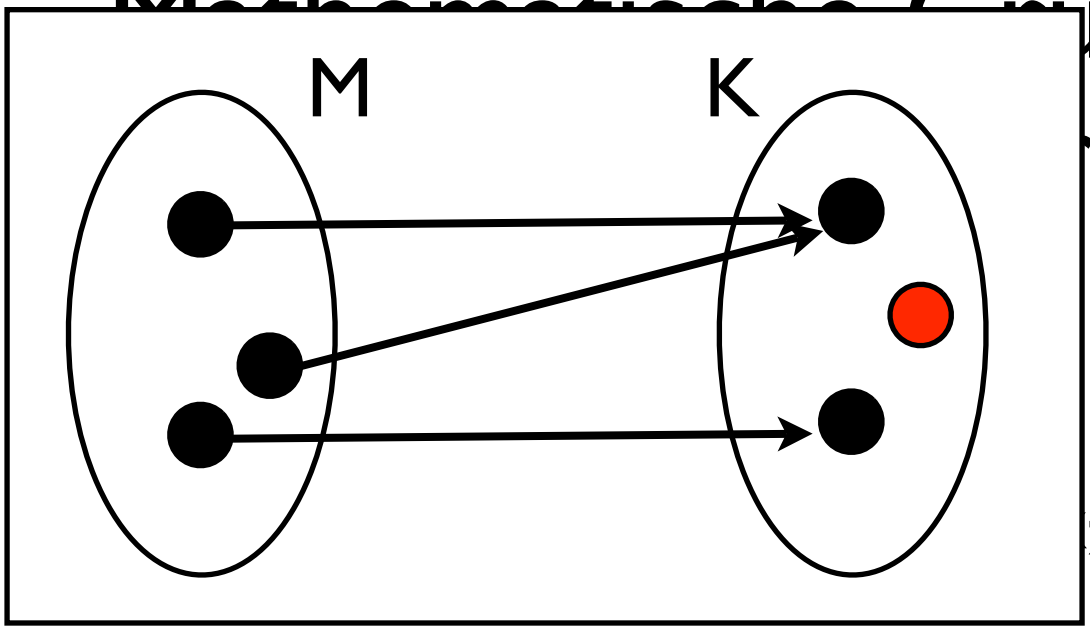
mindestens ein

tiv ist

Ur s bijektiv.

Setze: $f^{-1}(y)=x \Leftrightarrow f(x)=y$.

Mathematische Grundbegriffe



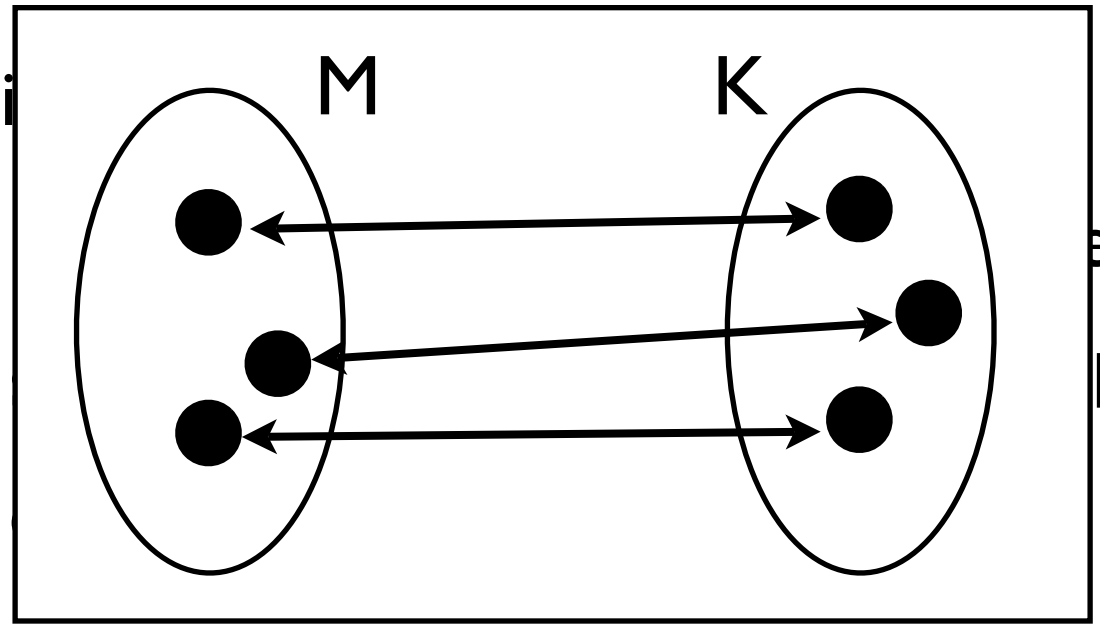
jektiv

eißt

$_2$) stets $x_1=x_2$ folgt

- **surjektiv**, wenn es zu jedem $y \in K$ mindestens ein $x \in M$ mit $f(x)=y$ gibt
- **bijektiv**, wenn f injektiv und surjektiv ist
- Zu bijektiver Funktion $f: M \to K$ stets Umkehrabbildung $f^{-1}: K \to M$, ebenfalls bijektiv. Setze: $f^{-1}(y)=x \Leftrightarrow f(x)=y$.

Mathematische Grundbegriffe
Funktionen - bijektiv



- Ei
- ets $x_1=x_2$ folgt
- K mindestens

- **bijektiv**, wenn f injektiv und surjektiv ist
- Zu bijektiver Funktion f: M→K stets Umkehrabbildung f^{-1}: K→M, ebenfalls bijektiv. Setze: $f^{-1}(y)=x \Leftrightarrow f(x)=y$.

Mathematische Grundbegriffe
Mächtigkeit

- f: M→K bijektiv, dann M und K *gleichmächtig*
- **Mächtigkeit**: Anzahl der Elemente von M
- endliche Mengen:

  $|M|=n \Leftrightarrow$ bijektive Funktion $f: M \to \{0,1,2,...,n-1\}$

- M und K **isomorph**, wenn es f: M→K bijektiv gibt
- M **abzählbar** unendlich ⇔ M gleichmächtig zu IN; später $\mathbb{Z}$ und $\mathbb{Q}$ abzählbar, IR nicht.

Mathematische Grundbegriffe
Mengen und Funktionen

- $K^M = \{f \mid f: M \rightarrow K, f \text{ total}\}$
- K selbst als $K^{\{\emptyset\}}$ auffassen. Betrachte dazu:
  - $g: K \rightarrow K^{\{\emptyset\}}$ mit $g(k)=f_k$, wobei

    für $f_k: \{\emptyset\} \rightarrow K$ gilt: $f_k(\emptyset)=k$.

  - g ist bijektiv => Zusammenhang zwischen Funktionen und Mengen, speziell Datenmengen (s. später)

Mathematische Grundbegriffe
Definitions- und Wertebereich

- Sei $f: M \to K$ partiell:
  - $D(f)=\{x \in M \mid f(x) \text{ existiert}\}$

    Definitionsbereich, **Domain**, **Quellbereich**
  - $R(f)=\{y \in K \mid \text{Es ex. } x \in M \text{ mit } f(x)=y\}$

    Wertebereich, **Range**, **Zielbereich**
- $M \to K$ **Funktionalität** von f oder – falls M und K (Daten-)Typen sind – **Typ** von f
- f total $\Leftrightarrow$ $D(f)=M$; f surjektiv $\Leftrightarrow$ $R(f)=K$

Mathematische Grundbegriffe
Vervollständigung

- Idee: Erweitere partielle Funktion $f: A \to B$, so daß sie total wird
- Setze $A^{\perp} := A \cup \{\perp\}$, $B^{\perp} := B \cup \{\perp\}$;

  Vervollständigungen von A und B
- erweitere $f: A \to B$ zu $f^{\perp}: A^{\perp} \to B^{\perp}$ mit

$$f^{\perp}(a) = \begin{cases} f(a), \text{ falls } f(a) \text{ definiert,} \\ \perp, \text{ sonst.} \end{cases}$$

- $\perp$=Pseudoausgabe eines Algorithmus, der für die gewählte Eingabe nicht terminiert

Mathematische Grundbegriffe
Vervollständigung bei Produkten

- Wahlfreiheiten bei $A=A_1\times A_2\times\ldots\times A_n$: was ist $f(a_1,\ldots,a_n)$, wenn einige $a_i=\perp$ und einige $a_i\neq\perp$
- **strikte** Fortsetzung:
  $f^{\perp}: A_1\times A_2\times\ldots\times A_n\to B$ mit

$$f^{\perp}(a_1,\ldots,a_n)=\begin{cases} f(a_1,\ldots,a_n), \text{ falls } a_i\neq\perp \text{ für } 1\leq i\leq n, \\ \perp, \text{ falls } a_i=\perp \text{ für ein } i\in\{1,\ldots,n\}. \end{cases}$$

- Rest der Vorlesung: alle Wertemengen vervollständigt

nicht berechenbare Funktionen
Idee

Idee

- Algorithmen arbeiten auf Texten
- Algorithmen sind selbst Texte
- Algorithmen können auf Algorithmen angewendet werden
- Algorithmen können auf sich selbst angewendet werden
- Konstruiere auf dieser Idee Widersprüche

nicht berechenbare Funktionen
Alphabet

- Algorithmen sind Wörter über einem endlichen Alphabet

$$X=\{a,...,z,A,...,Z,0,...9,;,:=,...,-,+,\diamondsuit\}$$

- Leerzeichen $\diamondsuit$
- jeder Algorithmus ist ein $\alpha \in X^*$
- im folgenden Zeichen stets in Hochkomma
- Ein- und Ausgaben von Algorithmen ebenfalls Zeichenfolgen aus X^*
- $\alpha \in X^*$ berechnet eine Funktion (Bez.: f_α): $f_\alpha: X^* \rightarrow X^*$.

nicht berechenbare Funktionen
berechenbar

Definition

Eine Funktion $f: X^* \to X^*$ heißt **berechenbar**, wenn es einen Algorithmus $\alpha \in X^*$ gibt mit $f = f_\alpha$.

nicht berechenbare Funktionen
Vorgehen

1. Bestimme Anzahl der Algorithmen
2. Bestimme Anzahl der Funktionen
3. Vergleiche: mehr Funktionen als Algorithmen
4. Es ex. mind. eine Funktion, zu der es keinen Algorithmus gibt

- Problem: beide Mengen unendlich; welche ist größer
- Hilfsmittel: Abzählbarkeit

nicht berechenbare Funktionen
Abzählbarkeit

Definition

Eine unendliche Menge M heißt **abzählbar unendlich** (kurz: **abzählbar**), wenn es eine bijektive Abbildung f: IN→M gibt (oder – was gleichwertig ist – wenn es eine bijektive Abbildung g: M→IN gibt). f nennt man **Abzählung**. Ist M unendlich, aber nicht abzählbar, so heißt M **überabzählbar**.

Vereinbarung: endliche Mengen sind abzählbar.

nicht berechenbare Funktionen
Abzählbarkeit

Hilfssatz

Jede unendliche Teilmenge einer abzählbaren Menge ist abzählbar. Jede Obermenge einer überabzählbaren Menge ist überabzählbar.

Beweis (Teil 1)

Sei M abzählbare Menge, $M' \subseteq M$ unendliche Teilmenge. Sei

$f: \mathbb{N} \to M$ eine Abzählung von M. Seien $k_0, k_1, k_2, k_3, \ldots \in \mathbb{N}$

diejenigen natürlichen Zahlen, für die $f(k_i) \in M'$ gilt. Eine

Abzählung f' von M' erhält man dann durch die Definition: $f': \mathbb{N} \to M'$ mit $f'(i) = f(k_i)$ für alle $i \in \mathbb{N}$. ♦

nicht berechenbare Funktionen
Abzählbarkeit - Beispiel 1

Menge der ganzen Zahlen $\mathbb{Z}$ ist abzählbar

- Definiere bijektive Abbildung f: $\mathbb{N} \rightarrow \mathbb{Z}$:

$$f(i) = \begin{cases} i/2, \text{ falls } i \text{ gerade,} \\ {}^{1}/_{2}(-1-i), \text{ falls } i \text{ ungerade.} \end{cases}$$

- Numerierung

i	0	1	2	3	4	5	6	7	8	9	10	...
f(i)	0	-1	1	-2	2	-3	3	-4	4	-5	5	...

nicht berechenbare Funktionen
Abzählbarkeit - Beispiel 2

Menge der rationalen Zahlen $\mathbb{Q}$ ist abzählbar

- Zeige für $\mathbb{Q}^+$ und nutze Hilfssatz
- stelle $\mathbb{Q}^+$ als Brüche p/q dar und nutze Hilfssatz
- Numerierung

q \ p	0	1	2	3	4	5	6	...
1	0	2	5	9	14	20	...	
2	1	4	8	13	19	...		
3	3	7	12	18	...			
4	6	11	17	...				
5	10	16	...					
6	15	...						
...								

nicht berechenbare Funktionen
Abzählbarkeit - Beispiel 2

- Allgemein lautet die Aufzählung:

  f: "Menge der nichtnegativen Brüche" → IN mit

  $f(^{p}/_{q})=n$, wobei

  $n=((p+q)^2+p-q)/2$ für alle $p \geq 0, q \geq 1$

- negative Brüche hinzunehmen
- Brüche abzählbar => $\mathbb{Q}$ abzählbar wegen Hilfssatz

nicht berechenbare Funktionen
Abzählbarkeit der Algorithmen

Satz

Die Menge aller Algorithmen ist abzählbar.

Beweis

- Menge aller Algorithmen⊆Menge aller endlichen Texte X*
- Zähle X* ab
- nutze Hilfssatz

nicht berechenbare Funktionen
Abzählbarkeit der Algorithmen

- Ordne Elemente von X an:

$$X=\{a,...,z,A,...,Z,0,...9,;,:=,...,-,+,\Diamond\}$$

- Sei $p: X \rightarrow \{1,2,...,|X|\}$ totale Funktion mit

$$p('a')=1,\ p('z')=26,\ p('7')=60 \text{ usw.}$$

- Sei $w \in X^*$, $|w|=n \in \mathbb{N}$, d.h.

$$w=x_1x_2x_3...x_n,\ x_i \in X \text{ für alle } i \in \{1,...,n\}.$$

- Definiere $\varphi: X^* \rightarrow \mathbb{N}$ durch

$$\varphi(w)=\varphi(x_1x_2x_3...x_n)=\sum_{i=1}^{n} p(x_i)\cdot(|X|+1)^i$$

speziell $\varphi(\epsilon)=0$

nicht berechenbare Funktionen
Abzählbarkeit der Algorithmen

- Offenbar: φ total
- Behauptung: φ injektiv
- Hilfsbeobachtung:

$$\varphi(x_1x_2x_3...x_n) = \sum_{i=1}^{n} p(x_i)\cdot(|X|+1)^i < (|X|+1)^{n+1}$$

- Seien $v,w \in X^*$
- $|w| \neq |v| \Rightarrow \varphi(w) \neq \varphi(v)$ wegen obiger Ungleichung
- Sei daher $|w|=|v|$, also $w=x_1x_2x_3...x_n$ und $v=y_1y_2y_3...y_n$. Weiterhin: $\varphi(w)=\varphi(v)$.

nicht berechenbare Funktionen
Abzählbarkeit der Algorithmen

- Dann:

$$\varphi(w) = \sum_{i=1}^{n} p(x_i)\cdot(|X|+1)^i = \sum_{i=1}^{n} p(y_i)\cdot(|X|+1)^i = \varphi(v)$$

- Folglich:

$$\sum_{i=1}^{n} (p(x_i)-p(y_i))\cdot(|X|+1)^i = 0$$

- Dies geht nur, wenn $p(x_i)-p(y_i)=0$ für alle i
- Dann ist $x_i=y_i$ und w=v.
- Folglich: $\varphi(w)=\varphi(v) \Rightarrow v=w$, und φ injektiv.

nicht berechenbare Funktionen
Abzählbarkeit der Algorithmen

- Betrachte nun $\varphi(X^*)$
- X^* unendlich, φ injektiv $\Rightarrow$ $\varphi(X^*)$ unendlich
- Nach Hilfssatz: jede unendliche Teilmenge von IN abzählbar unendlich, also $\varphi(X^*)$
- $\varphi(X^*)$ gleichmächtig zur Menge X^* $\Rightarrow$

  Abzählbarkeit von X^* $\Rightarrow$ Menge aller

  Algorithmen als unendliche Teilmenge von X^* abzählbar. ♦

nicht berechenbare Funktionen
Abzählbarkeit der Algorithmen

- Hintergrund: φ basiert auf der Idee des Stellenwertsystems der Form S=(|X|+1,X,p)
- |X|+1 die Basis des Systems
- X Menge der "Ziffernsymbole"
- p liefert das Gewicht jedes "Ziffernsymbols" aus X
- Dezimalsystem (10,{'0','1',...,'9'},p), p('i')=i, d.h., das Ziffernsymbol 'i' wird auf die Ziffer i abgebildet

nicht berechenbare Funktionen
Überabzählbarkeit der Funktionen

Satz

Die Menge aller Abbildungen $f: X^* \rightarrow X^*$ ist überabzählbar.

Beweis

Zeige Behauptung für die Menge aller totalen Funktionen $F:=\{f \mid f: X^* \rightarrow X^* \text{ und } f \text{ ist total}\}$ und nutze Hilfssatz.

Indirekte Beweisführung: Nimm Gegenteil an und zeige Widerspruch.

nicht berechenbare Funktionen
Überabzählbarkeit der Funktionen

Annahme: F ist abzählbar.

Dann gibt es Abzählung h: $\mathbb{N} \to F$, d.h., man kann die Abbildungen der Reihe nach durchnumerieren:

$f_0, f_1, f_2, ..., f_{1761}, ...$.

Satz C $\Rightarrow$ Menge aller Wörter $w \in X^*$ abzählbar:

$w_0, w_1, w_2, ...$ mit $w_i \in X^*$.

nicht berechenbare Funktionen
Überabzählbarkeit der Funktionen

Sei $a \in X$ beliebiges festgewähltes Zeichen.

Betrachte Funktion

$$g: X^* \rightarrow X^* \quad \text{mit}$$

$$g(w_i) = f_i(w_i) \cdot a \quad \text{für alle } w_i \in X^*$$

g offenbar total, d.h. $g \in F$.

Dann kommt g in der Abzählung f_j vor.

nicht berechenbare Funktionen

Überabzählbarkeit der Funktionen

Es muß also ein $k \in \mathbb{N}$ geben mit $g = f_k$ bzw.

allgemein

$$g(v) = f_k(v) \quad \text{für alle } v \in X^*.$$

Für das spezielle Wort w_k folgt dann einerseits

$$g(w_k) = f_k(w_k),$$

andererseits aus der Definition von g:

$$g(w_k) = f_k(w_k) \cdot a \neq f_k(w_k).$$

nicht berechenbare Funktionen

Überabzählbarkeit der Funktionen

Es muß also ein $k \in \mathbb{N}$ geben mit $g = f_k$ bzw.

allgemein

$$g(v) = f_k(v) \quad \text{für alle } v \in X^*.$$

Für das spezielle Wort w_k folgt dann einerseits

$$g(w_k) = f_k(w_k),$$

andererseits aus der Definition von g:

$$g(w_k) = f_k(w_k) \cdot a \neq f_k(w_k).$$

nicht berechenbare Funktionen

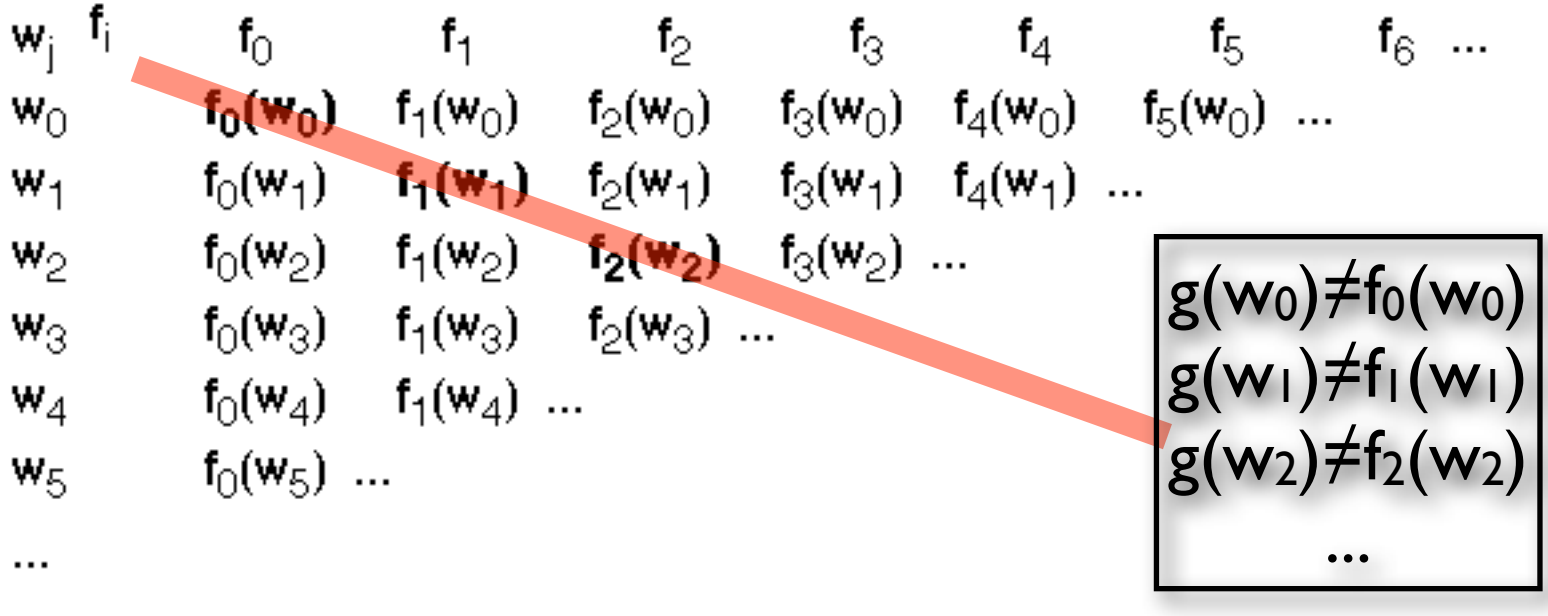
Überabzählbarkeit der Funktionen

- Widerspruch!
- Annahme war also falsch
- Behauptung also bewiesen, d.h. F ist nicht abzählbar. ♦

- Es gibt mehr Funktionen als Algorithmen.

nicht berechenbare Funktionen
Diagonalisierung

- Diagonalisierungsverfahren nach G. Cantor (19. Jhdt.): Menge der reellen Zahlen überabzählbar
- **Diagonalisierung**: Widerspruch zur Annahme bei den Elementen der Diagonalen



nicht berechenbare Funktionen
Nicht-Berechenbarkeit

Satz

Es gibt eine Funktion $f: X^* \rightarrow X^*$, die nicht durch einen Algorithmus berechnet werden kann.

Bem.:

- nur Existenzaussage => Beispiele später
- Ex. sogar überabzählbar viele nicht berechenbare Funktionen. Zahl der berechenbaren Funktionen homöopatisch.

nicht berechenbare Funktionen
Beispiele nicht-berechenbarer Funkt.

Idee

- Algorithmus $\alpha \in X^*$: $f_\alpha: X^* \rightarrow X^*$ Funktion, die α berechnet
- α kann für eine Eingabe definiert oder undefiniert sein
- z.B.: α Multiplikationsalgorithmus, dann

  $f_\alpha(7,'peter')$ undefiniert

  $(7,'peter') \notin D(f_\alpha)$

- Wende α auf seinen eigenen Text an:

  Berechne $f_\alpha(\alpha)$; gilt $\alpha \in D(f_\alpha)$?

nicht berechenbare Funktionen
Selbstanwendung

Beispiel

- Betrachte Algorithmus $\delta \in X^*$

  (Zeichenzähler):

  $$f_\delta: X^* \rightarrow X^* \quad \text{mit} \quad f_\delta(w)=|w|$$

- Definitionsbereich $D(f_\delta)=X^*$
- δ ist selbst Wort aus X^*, also δ als Eingabe:

  $$f_\delta(\delta)=|\delta|$$

nicht berechenbare Funktionen
Selbstanwendung

- Kann man maschinell entscheiden, ob ein Algorithmus auf seinem eigenen Text definiert ist oder nicht?
- *Gibt es einen Algorithmus* β*, der beliebige Texte (von Algorithmen)* $\alpha \in X^*$ *als Eingabe erhält, und 'ja' ausgibt, falls* $\alpha \in D(f_\alpha)$*, und 'nein', falls* $\alpha \notin D(f_\alpha)$*?*
- β soll also Funktion $S: X^* \to X^*$ berechnen:

$$S(\alpha) = \begin{cases} \text{'ja', falls } \alpha \text{ Algorithmus ist und } \alpha \in D(f_\alpha), \\ \text{'nein', sonst.} \end{cases}$$

- S ist totale Funktion.

nicht berechenbare Funktionen
Selbstanwendungsproblem

Satz

Es gibt keinen Algorithmus, der bei Eingabe beliebiger Algorithmen $\alpha \in X^*$ ausgibt, ob $\alpha \in D(f_\alpha)$

gilt oder nicht, d.h., obige Funktion S ist nicht berechenbar.

Beweis (indirekt durch Widerspruch)

Annahme: es ex. ein Algorithmus, der die Funktion S berechnet.

Methode: Widerspruchserzeugung durch *Diagonalisierung*.

nicht berechenbare Funktionen
Selbstanwendungsproblem - Beweis

- Sei $a \in X$ ein fest gewähltes Zeichen.
- S berechenbar $\Rightarrow$ totale Funktion $g: X^* \rightarrow X^*$ berechenbar:

$$g(\alpha) = \begin{cases} f_\alpha(\alpha) \cdot a, & \text{falls } S(\alpha) = \text{'ja'}, \\ \text{'nein'}, & \text{falls } S(\alpha) = \text{'nein'}. \end{cases}$$

- zu g paßt folgender Algorithmus γ:

γ: Lies beliebiges Wort $\alpha \in X^*$ ein;

berechne $S(\alpha)$ [nach Annahme möglich];
falls Ergebnis='ja', dann
    berechne $f_\alpha(\alpha)$ [dieser Wert existiert dann!]
    und gib den um das Zeichen 'a' verlängerten
    Ergebnistext aus;
anderenfalls gib 'nein' aus.

nicht berechenbare Funktionen
Selbstanwendungsproblem - Beweis

- Sei $a \in X$ ein fest gewähltes Zeichen.
- S berechenbar $\Rightarrow$ totale Funktion $g: X^* \to X^*$ berechenbar:

$$g(\alpha) = \begin{cases} f_\alpha(\alpha) \cdot a, \text{ falls } S(\alpha)=\text{'ja'}, \\ \text{'nein', falls } S(\alpha)=\text{'nein'}. \end{cases}$$

- zu g paßt folgender Algorithmus γ:

γ: Lies beliebiges Wort $\alpha \in X^*$ ein;

berechne $S(\alpha)$ [nach Annahme möglic
falls Ergebnis='ja', dann
berechne $f_\alpha(\alpha)$ [dieser Wert existiert dann!]
und gib den um das Zeichen 'a' verlängerten
Ergebnistext aus;
anderenfalls gib 'nein' aus.

also $g=f_\gamma$,
speziell
$g(\gamma)=f_\gamma(\gamma)$

nicht berechenbare Funktionen
Selbstanwendungsproblem - Beweis

- Andererseits folgt aus der Definition von g:

$$g(\gamma)=\begin{cases} f_\gamma(\gamma)\cdot a, \text{ falls } S(\gamma)=\text{'ja'}, \\ \text{'nein', falls } S(\gamma)=\text{'nein'}. \end{cases}$$

nicht berechenbare Funktionen

Selbstanwendungsproblem - Beweis

1. Möglichkeit
geht nicht, denn vorher
war ja schon
$g(\gamma)=f_\gamma(\gamma)\neq f_\gamma(\gamma)\cdot a$

- Andererseits folgt aus der Definition von g:

$$g(\gamma)=\begin{cases} f_\gamma(\gamma)\cdot a, \text{ falls } S(\gamma)=\text{'ja'}, \\ \text{'nein', falls } S(\gamma)=\text{'nein'}. \end{cases}$$

nicht berechenbare Funktionen

Selbstanwendungsproblem - Beweis

- Andererseits folgt aus

$$g(\gamma)=\begin{cases} f_\gamma(\gamma)\cdot a, \text{ falls } S(\gamma) \\ \text{'nein', falls } S(\gamma)=\text{'nein'}. \end{cases}$$

bleibt nur
2. Möglichkeit
$g(\gamma)=f_\gamma(\gamma)=S(\gamma)=$'nein'

nicht berechenbare Funktionen

Selbstanwendungsproblem - Beweis

- Andererseits folgt aus der Definition von g:

$$g(\gamma)=\begin{cases} f_\gamma(\gamma)\cdot a, \text{ falls } S(\gamma)=\text{'ja'}, \\ \text{'nein', falls } S(\gamma)=\text{'nein'}. \end{cases}$$

- $g(\gamma)=f_\gamma(\gamma)=S(\gamma)=$'nein' (2. Möglichkeit)
- $S(\gamma)=$'nein' $\Rightarrow \gamma \notin D(f_\gamma)$ nach Def. von S

nicht berechenbare Funktionen

Selbstanwendungsproblem - Beweis

- Andererseits folgt aus der Definition von g:

$$g(\gamma)=\begin{cases} f_\gamma(\gamma)\cdot a, \text{ falls } S(\gamma)=\text{'ja'}, \\ \text{'nein', falls } S(\gamma)=\text{'nein'}. \end{cases}$$

- $g(\gamma)=f_\gamma(\gamma)=S(\gamma)=$'nein' (2. Möglichkeit)
- $S(\gamma)=$'nein' $\Rightarrow \gamma \notin D(f_\gamma)$ nach Def. von S

$$S(\alpha)=\begin{cases} \text{'ja', falls } \alpha \text{ Algorithmus ist und } \alpha \in D(f_\alpha), \\ \text{'nein', sonst.} \end{cases}$$

nicht berechenbare Funktionen

Selbstanwendungsproblem - Beweis

- Andererseits folgt aus der Definition von g:

$$g(\gamma)=\begin{cases} f_\gamma(\gamma)\cdot a, \text{ falls } S(\gamma)=\text{'ja'}, \\ \text{'nein', falls } S(\gamma)=\text{'nein'}. \end{cases}$$

- $g(\gamma)=f_\gamma(\gamma)=S(\gamma)=$'nein' (2. Möglichkeit)
- $S(\gamma)=$'nein' $\Rightarrow \gamma \notin D(f_\gamma)$ nach Def. von S
- $\Rightarrow$ Algorithmus γ auf seinem eigenen Text nicht definiert.

nicht berechenbare Funktionen

Selbstanwendungsproblem - Beweis

- Andererseits folgt aus der Definition von g:

$$g(\gamma)=\begin{cases} f_\gamma(\gamma)\cdot a, \text{ falls } S(\gamma)=\text{'ja'}, \\ \text{'nein', falls } S(\gamma)=\text{'nein'}. \end{cases}$$

- $g(\gamma)=f_\gamma(\gamma)=S(\gamma)=$'nein' (2. Möglichkeit)
- $S(\gamma)=$'nein' $\Rightarrow \gamma \notin D(f_\gamma)$ nach Def. von S
- $\Rightarrow$ Algorithmus γ auf seinem eigenen Text nicht definiert.
- $\Rightarrow$ Widerspruch, denn $f_\gamma(\gamma)$ ist definiert:

$$f_\gamma(\gamma)=g(\gamma)=\text{'nein'}.$$

nicht berechenbare Funktionen
Selbstanwendungsproblem - Beweis

- Also war die Annahme falsch, d.h., S kann nicht mit einem Algorithmus berechnet werden. ♦

- S ist unsere erste nicht-berechenbare Funktion, das sog. **Selbstanwendungsproblem (SAP)**.

nicht berechenbare Funktionen
Selbstanwendungsproblem - Reduktion

- praktische Bedeutung?
- SAP ist das einfachste nicht-berechenbare (Basis-) Problem
- Idee des weiteren Vorgehens: **Reduktion** von unbekannten Problemen auf bekannte (fundamentale Idee der Informatik)
- Konkretes Vorgehen: Wenn man schon nicht algorithmisch entscheiden kann, ob ein beliebiger Algorithmus für seinen eigenen Text als Eingabe definiert ist oder nicht, dann kann man erst recht nicht entscheiden, ob ein beliebiger Algorithmus überhaupt für eine Eingabe definiert ist (**Halteproblem**)

nicht berechenbare Funktionen
Halteproblem

Satz

Es gibt keinen Algorithmus, der bei Eingabe beliebiger Algorithmen $\alpha \in X^*$ und beliebiger

Wörter $w \in X^*$ ausgibt, ob $w \in D(f_\alpha)$ gilt oder nicht.

Präziser: Die totale Funktion $h: X^* \times X^* \to X^*$ mit

$$h(\alpha,w) = \begin{cases} \text{'ja', falls } w \in D(f_\alpha), \\ \text{'nein', sonst,} \end{cases}$$

ist nicht berechenbar.

nicht berechenbare Funktionen
Halteproblem - Beweis

Beweis

- *Reduktion* auf das SAP: zeige, daß aus Lösbarkeit des Halteproblems Lösbarkeit des SAP folgt (Widerspruch zum Satz).
- *Annahme*: es gibt Algorithmus, der Halteproblem löst, also h berechnet.
- Definiere neue totale Funktion

$$g: X^* \rightarrow X^* \text{ mit } g(\alpha) = h(\alpha,\alpha).$$

- h berechenbar $\Rightarrow$ g berechenbar
- g beschreibt genau das SAP, denn $g(\alpha)$ = 'ja' genau dann, wenn $\alpha \in D(f_\alpha)$ ist.
- Widerspruch! Folglich ist Satz bewiesen. ♦

nicht berechenbare Funktionen
Hinweise

Bem.:

- Probleme nicht *algorithmisch* für *alle* Algorithmen $\alpha \in X^*$ (und ggf. $w \in X^*$) lösbar
- für einzelne Algorithmen α kann man und **muß man** sich spezielle Beweise einfallen lassen
- Halteproblem inhärent Teil jeder Softwareentwicklung

nicht berechenbare Funktionen
Hinweise

Bem.:

- Probleme nicht *algorithmisch* für *alle* Algorithmen $\alpha \in X^*$ (und ggf. $w \in X^*$) lösbar
- für einzelne Algori… **muß man** sich s… lassen
- Halteproblem inhärent Teil jeder Softwareentwicklung

dieser Aspekt erfordert daher immer Handarbeit und sichert Ihren Arbeitsplatz

nicht berechenbare Funktionen
Hinweise

Bem.:

- Probleme nicht *algorithmisch* für *alle* Algorithmen $\alpha \in X^*$ (und ggf. $w \in X^*$) lösbar
- für einzelne Algorithmen α kann man und **muß man** sich spezielle Beweise einfallen lassen
- Halteproblem inhärent Teil jeder Softwareentwicklung
- Aber: es kommt noch schlimmer/besser ...

nicht berechenbare Funktionen
Satz von Rice

Einem Algorithmus "anzusehen", ob die durch ihn berechnete Funktion eine bestimmte Eigenschaft besitzt, ist algorithmisch unlösbar.

Satz (umgangssprachlich, nicht exakt)

Sei E eine beliebige Eigenschaft, die von mindestens einer, aber nicht von allen berechenbaren Funktionen $f: X^* \rightarrow X^*$ erfüllt wird. Dann gibt es keinen Algorithmus, der für beliebige Algorithmen $\alpha \in X^*$

ausgibt, ob f_α die Eigenschaft E erfüllt oder nicht.

nicht berechenbare Funktionen
Churchsche These

- Bisherige Argumentation lief über Algorithmen versus Funktionen
- Lassen sich die Ergebnisse auf Maschinen übertragen?
- Können Maschinen vielleicht mehr als Algorithmen? Oder weniger?

nicht berechenbare Funktionen
Churchsche These

- Bisherige Argumentation lief über Algorithmen versus Funktionen
- Lassen sich die Ergebnisse auf Maschinen übertragen?
- Können Maschinen vielleicht mehr als Algorithmen? Oder weniger?
- Mehr??

nicht berechenbare Funktionen
Churchsche These

- Bisherige Argumentation lief über Algorithmen versus Funktionen
- Lassen sich die Ergebnisse auf Maschinen übertragen?
- Können Maschinen vielleicht mehr als Algorithmen? Oder weniger?
- Mehr??
    - Maschinen (Computer) arbeiten Programme ab; diese Abarbeitungen von Programmen sind mechanisch durchführbare Verfahren, also Algorithmen

nicht berechenbare Funktionen
Churchsche These

- weniger?
  - vorstellbar, daß es ein algorithmisches Verfahren gibt, das im intuitiven Sinne effektiv durchführbar ist, das aber von keinem Computer abgearbeitet werden kann
- Jahrzehntelange Untersuchungen führten zu keinem allgemein anerkannten Gegenbeispiel.
- Man glaubt daher an folgende These des amerikanischen Logikers A. Church (1936):

Jede im intuitiven Sinne berechenbare Funktion ist maschinell berechenbar und umgekehrt.

nicht berechenbare Funktionen
Churchsche These

- Anschaulich: jedes Problem, zu dem man ein algorithmisches Lösungsverfahren angeben kann, kann auch von einem Computer gelöst werden und umgekehrt.
- "Im intuitiven Sinne": Verfahren von einer genügend großen Zahl von Fachleuten als algorithmisch anerkannt.
- kein mathematischer Satz und nicht beweisbar, da der Begriff "im intuitiven Sinne" nicht präzisiert ist und auch nicht exakt präzisiert werden kann.
- Da Widerlegung der Churchschen These seit über 50 Jahren gescheitert, wird sie allgemein anerkannt. Mathematische Beweise, die sich auf sie berufen, werden akzeptiert.

nicht berechenbare Funktionen
Churchsche These

- Sätze zu Nicht-Berechenbarkeitsaussagen, bewiesen für Algorithmen, können auf Maschinen übertragen werden, da beide Modelle nach der Churchschen These äquivalent sind.
- s. Vorlesungen über Theoretische Informatik

nicht berechenbare Funktionen
Beispiel

- Ein Software-Haus entwickelt laufend Computerprogramme für die unterschiedlichsten Zwecke und möchte jedesmal überprüfen, ob das fertige Programm das leistet, was der Entwickler oder der Auftraggeber sich vorgestellt haben. Diese Überprüfung kann man nicht von einem Computer für alle Programme vornehmen lassen, da das Problem nicht maschinell lösbar ist (Handarbeit nötig).

- Beispiel zeigt: wesentliche Elemente bei der Programmentwicklung (Erstellung, Überprüfung usw.) nicht automatisch durchführbar.

- Jede einzelne Entwicklung eines Programms erfordert eigene Überlegungen, die vom Menschen durchzuführen sind. Gewisse allgemeine Methoden erleichtern in der Praxis die Programmerstellung wesentlich.