Grundbegriffe der Informatik

Kapitel 16a: Unentscheidbarkeit

Mattias Ulbrich

KIT · Institut für Theoretische Informatik

Wintersemester 2023/2024

Aufzählen der Teilmengen natürlicher Zahlen

- Aufzählen aller endlichen Mengen von natürlichen Zahlen
  - $e : \mathbb{N}_0 \to \{s \in \mathcal{P}(\mathbb{N}_0) \mid s \text{ ist endlich}\}$
  - Interpretiere Binärstellen einer Zahl n für die Menge $e(n)$
  - $e(n) = \{m \in \mathbb{N}_0 \mid \mathrm{Repr}_2(n)(m) = 1\}$
  - z.B.: $e(46) = e(101110_2) = \{1, 2, 3, 5\}$
  - e ist surjektiv.

- Aufzählen aller Mengen von natürlichen Zahlen ist unmöglich.
  - Es gibt keine surjektive Funktion $f : \mathbb{N}_0 \to \mathcal{P}(\mathbb{N}_0)$.
  - Beweis folgt nun.
  - Beweistechnik: Cantor-Diagonalisierung

Cantor-Diagonalisierung

- **Beweis durch Widerspruch**!
- Annahme: Es gibt eine *surjektive* Funktion $f : \mathbb{N}_0 \to \mathcal{P}(\mathbb{N}_0)$
- $f(0), f(1), f(2), \ldots$ zählt alle Teilmengen von $\mathbb{N}_0$ auf.

Cantor-Diagonalisierung

- **Beweis durch Widerspruch**!
- Annahme: Es gibt eine *surjektive* Funktion $f : \mathbb{N}_0 \to \mathcal{P}(\mathbb{N}_0)$
- $f(0), f(1), f(2), \ldots$ zählt alle Teilmengen von $\mathbb{N}_0$ auf.
- Wir können in einer Matrix festhalten, ob $i \in f(j)$ enthalten ist:

	0	1	2	…	i	…
0	$0 \in f(0)$	$1 \in f(0)$	$2 \in f(0)$	…		
1	$0 \in f(1)$	$1 \in f(1)$	$2 \in f(1)$	…		
2	$0 \in f(2)$	$1 \in f(2)$	$2 \in f(2)$	…		
⋮	⋮	⋮	⋮	⋱		
j					$i \in f(j)$	
⋮						

Cantor-Diagonalisierung

- **Beweis durch Widerspruch**!
- Annahme: Es gibt eine *surjektive* Funktion $f : \mathbb{N}_0 \to \mathcal{P}(\mathbb{N}_0)$
- $f(0), f(1), f(2), \ldots$ zählt alle Teilmengen von $\mathbb{N}_0$ auf.
- Wir können in einer Matrix festhalten, ob $i \in f(j)$ enthalten ist:

	0	1	2	…	i	…
0	$0 \in f(0)$	$1 \in f(0)$	$2 \in f(0)$	…		
1	$0 \in f(1)$	$1 \in f(1)$	$2 \in f(1)$	…		
2	$0 \in f(2)$	$1 \in f(2)$	$2 \in f(2)$	…		
⋮	⋮	⋮	⋮	⋱		
j					$i \in f(j)$	
⋮						

- Diagonalelemente dieser Matrix sind die $i \in f(i)$.

Cantor-Diagonalisierung

- **Beweis durch Widerspruch**!
- Annahme: Es gibt eine *surjektive* Funktion $f : \mathbb{N}_0 \to \mathcal{P}(\mathbb{N}_0)$
- $f(0), f(1), f(2), \ldots$ zählt alle Teilmengen von $\mathbb{N}_0$ auf.
- Wir können in einer Matrix festhalten, ob $i \in f(j)$ enthalten ist:
- Diagonalelemente dieser Matrix sind die $i \in f(i)$.
- Betrachte nun die Menge $S = \{k \in \mathbb{N}_0 \mid k \notin f(k)\}$.
- $S \subseteq \mathbb{N}_0$ also gibt es ein $s \in \mathbb{N}_0$ mit $f(s) = S$

Cantor-Diagonalisierung II

- Betrachte die Menge $S = \{k \in \mathbb{N}_0 \mid k \notin f(k)\}$.
- $S \subseteq \mathbb{N}_0$ also gibt es ein $s \in \mathbb{N}_0$ mit $f(s) = S$
- Beispiel für die Matrix:

	0	1	2	3	...
0	**w**	**f**	**w**	**f**	...
1	**f**	**f**	**f**	**f**	...
2	**w**	**w**	**f**	**f**	...
3	**w**	**w**	**f**	**w**	...
⋮	⋮	⋮	⋮	⋮	⋱
s	**f**	**w**	**w**	**f**	...

Cantor-Diagonalisierung II

- Betrachte die Menge $S = \{k \in \mathbb{N}_0 \mid k \notin f(k)\}$.
- $S \subseteq \mathbb{N}_0$ also gibt es ein $s \in \mathbb{N}_0$ mit $f(s) = S$
- Beispiel für die Matrix:

	0	1	2	3	...
0	**w**	**f**	**w**	**f**	...
1	**f**	**f**	**f**	**f**	...
2	**w**	**w**	**f**	**f**	...
3	**w**	**w**	**f**	**w**	...
⋮	⋮	⋮	⋮	⋮	⋱
s	**f**	**w**	**w**	**f**	...

- Widerspruch:
  $f(s)$ unterscheidet sich von jeder Zeile $f(k)$ wenigstens an der Stelle k.

Cantor-Diagonalisierung II

- Betrachte die Menge $S = \{k \in \mathbb{N}_0 \mid k \notin f(k)\}$.
- $S \subseteq \mathbb{N}_0$ also gibt es ein $s \in \mathbb{N}_0$ mit $f(s) = S$
- Beispiel für die Matrix:

	0	1	2	3	...
0	**w**	**f**	**w**	**f**	...
1	**f**	**f**	**f**	**f**	...
2	**w**	**w**	**f**	**f**	...
3	**w**	**w**	**f**	**w**	...
⋮	⋮	⋮	⋮	⋮	⋱
s	**f**	**w**	**w**	**f**	...

- Widerspruch:
  $f(s)$ unterscheidet sich von jeder Zeile $f(k)$ wenigstens an der Stelle k.
- **Es kann also keine surjektive Funktion $f : \mathbb{N}_0 \to \mathcal{P}(\mathbb{N}_0)$ geben.**

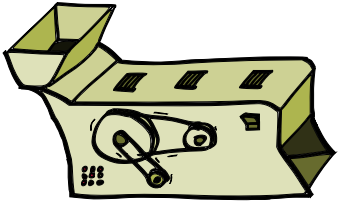
Cantor-Diagonalisierung III

Diagonalisierung:

Um den Widerspruch zu getroffenen Annahmen zu zeigen,
baue ein «Gebilde», das den Diagonalelementen einer Aufzählung widerspricht.

- Ein Widerspruchsargument.
- Für jede Menge M gilt dieses Argument hier, also:
- Die Potenzmenge 2^M ist mächtiger als die Menge M selbst.
- Georg Cantor (1845-1918)
- ... kommt gleich nochmal in leicht anderer Form

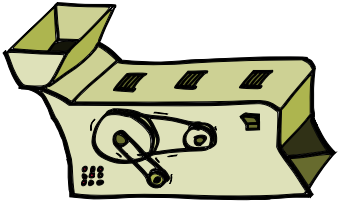
Wir betrachen «Rechenmaschinen»



Was sind solche Maschinen?

- **Turing**-Maschinen
- **Church**: λ-Kalkül
- **Gödel**: allg. rekursive Funktionen

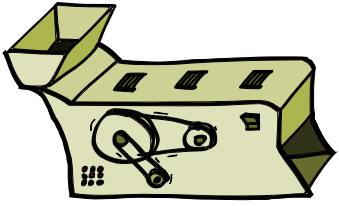
Wir betrachen «Rechenmaschinen»



Was sind solche Maschinen?

- **Turing**-Maschinen (1936)
- **Church**: λ-Kalkül (1936)
- **Gödel**: allg. rekursive Funktionen (1933)

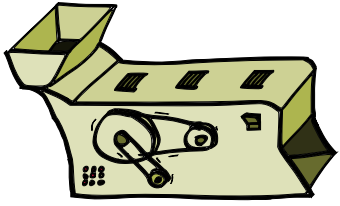
Wir betrachen «Rechenmaschinen»



Was sind solche Maschinen?

- **Turing**-Maschinen (1936)
- **Church**: λ-Kalkül (1936)
- **Gödel**: allg. rekursive Funktionen (1933)
- while-Programme über $\mathbb{N}_0$
- Java-Programme mit unbeschränktem Speicher
- fikt. MIMA mit unbeschränktem Adressraum

Wir betrachen «Rechenmaschinen»



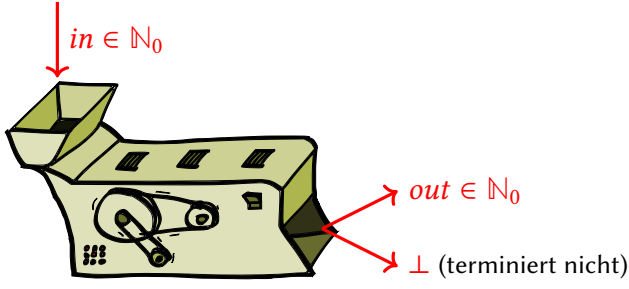
Alle Formalismen sind gleich ausdrucksstark:

man sagt sie sind turingvollständig.

Was sind solche Maschinen?

- **Turing**-Maschinen (1936)
- **Church**: λ-Kalkül (1936)
- **Gödel**: allg. rekursive Funktionen (1933)
- while-Programme über $\mathbb{N}_0$
- Java-Programme mit unbeschränktem Speicher
- fikt. MIMA mit unbeschränktem Adressraum

Berechnungsmodell Rechenmaschine



Eine Rechenmaschine R berechnet eine partielle Funktion $f_R : \mathbb{N}_0 \dashrightarrow \mathbb{N}_0$.

«Gödelisierung»

Karlsruher Institut für Technologie

Rechenmaschinen lassen sich als Zahl kodieren!

Historisch:

- Turingmaschine (Z, z_0, X, f, g, m)
- oBdA:
  - $Z = \{1, \ldots, |Z|\}$, $z_0 = 1$
  - $X = \{1, \ldots, |X|\}$

Gödel-Nummer
$\mathfrak{g} : TM \to \mathbb{N}_0$
oder auch
$\mathfrak{g} : TM \to Z_2^*$
(als Binärworte)

- Jede Regel $n \xrightarrow{x \mid yM} m$ der TM entspricht dann einem Tupel $(n, m, x, y, M+1) \in \mathbb{N}_0^5$
- Primzahlpotenzen $2^a \cdot 3^m \cdot 5^x \cdot 7^y \cdot 11^M \in \mathbb{N}_0$
- Eindeutige Primzahlzerlegung $\Rightarrow$ Rückrechnung möglich
- …

- **Insgesamt:** $\mathfrak{g}(T) \in \mathbb{N}_0$ Gödel-Nummer einer TM T

«Gödelisierung»

Karlsruher Institut für Technologie

Rechenmaschinen lassen sich als Zahl kodieren!

Historisch:

- Turingmaschine (Z, z_0, X, f, a, m)
- oBdA:
  - $Z = \{1, \ldots, |Z|\}$, $z_0 = 1$
  - $X = \{1, \ldots, |X|\}$
- Jede Regel $n \xrightarrow{x \mid yM} \ldots$ n Tupel $(n, m, x, y, M+1) \in \mathbb{N}_0^5$
- Primzahlpotenzen $2^a \cdot 3^m \cdot 5^x \cdot 7^y \cdot 11$
- Eindeutige Primzahlzerlegung $\Rightarrow$ Rückrechnung möglich
- …

Hauptbotschaft:
Es brauchte sehr geniale, aufwändige Modellierungstechniken, um TM als Zahlen zu kodieren.

Gödel-Nummer
$\mathfrak{g} : TM \to \mathbb{N}_0$
oder auch
$\mathfrak{g} : TM \to Z_2^*$
(als Binärworte)

- **Insgesamt:** $\mathfrak{g}(T) \in \mathbb{N}_0$ Gödel-Nummer einer TM T

«Gödelisierung»

Karlsruher Institut für Technologie

Rechenmaschinen lassen sich als Zahl kodieren!

Heute (85 Jahre später):

- Gegeben ein Java-Programm
- Quelltext lässt sich als Datei darstellen
- Die Bitfolge in Datei lässt sich als natürliche Zahl interpretieren.
- Interpretation als Binärwort ist offensichtlich

- Im Zeitalter der Digitalisierung ist (uns) klar,
  dass und wie Information (wie Text) als Zahlen kodiert werden kann.

Universelle Rechenmaschinen

Historisch: Es gibt eine universelle Turingmaschine $U : \mathbb{N}_0 \times \mathbb{N}_0 \dashrightarrow \mathbb{N}_0$ mit

$$U(\mathfrak{g}(T), n) = T(N) \qquad \text{für alle TM } T \text{ und alle } n \in \mathbb{N}_0$$

$U(\mathfrak{g}(T), n) = T(N)$

- U kann jede andere TM emulieren.
- U erhält als Argumente:
  1. Gödel-Nummer der zu simulierende Maschine
  2. Eingabe für die Berechnung
- Falls erstes Argument keine Gödel-Nummer, dann darf U sich beliebig verhalten, z. B. nicht anhalten.

Universelle Rechenmaschinen

Historisch: Es gibt eine universelle Turingmaschine $U : \mathbb{N}_0 \times \mathbb{N}_0 \dashrightarrow \mathbb{N}_0$ mit

$$U(\mathfrak{g}(T), n) = T(N) \qquad \text{für alle TM } T \text{ und alle } n \in \mathbb{N}_0$$

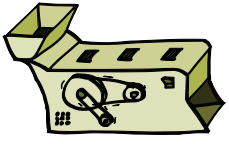
$U(\mathfrak{g}(T), n) = T(N)$

- U kann jede andere TM emulieren.
- U erhält als Argumente:
  1. Gödel-Nummer der zu simulierende Maschine
  2. Eingabe für die Berechnung
- Falls erstes Argument keine Gödel-Nummer, dann darf U sich beliebig verhalten, z. B. nicht anhalten.

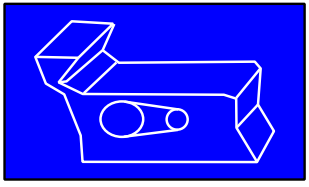
Heute (85 Jahre später):

- Es gibt ein universelles Javaprogramm U, das eine Zahl als Eingabe nimmt und diese Zahl als Javaprogramm T versteht und T ausführt.
- "In Java geschriebener Java-Interpreter"

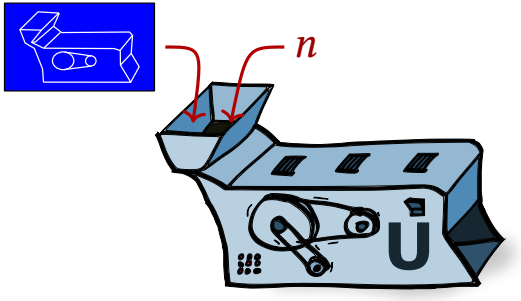
Grafisch



Maschine T



“Blaupause” Gödelisierung $\mathfrak{g}(T)$



Universelle Maschine U

Das Halteproblem ist unentscheidbar

- Das Halteproblem ist die formale Sprache

$$H = \{w \in Z_2^* \mid R \text{ eine Rechenmaschine, } w = \mathfrak{g}(R),\ U(w, w) \text{ hält.}\}$$

- **Satz**
  Das Halteproblem ist unentscheidbar,
  d. h. es gibt keine Turingmaschine, die H entscheidet.
- Es gibt Probleme, die man **NICHT** algorithmisch, also
  NICHT MIT EINEM KLASSISCHEN RECHNER lösen kann!
- ... und auch nicht mit einem *Quantencomputer*.

Es gibt Probleme, die

NICHT

algorithmisch

gelöst werden können!

Beweis durch Diagonalisierungsargument

- Zähle Rechenmaschinen $R_0, R_1, \ldots$ auf und Argumente $0, 1, 2, \ldots$

	0	1	2	…	i	…
R_0	$R_0(0)$ hält?	$R_0(1)$ hält?	$R_0(2)$ hält?	…		
R_1	$R_1(0)$ hält?	$R_1(1)$ hält?	$R_1(2)$ hält?	…		
R_2	$R_2(0)$ hält?	$R_2(1)$ hält?	$R_2(2)$ hält?	…		
⋮	⋮	⋮	⋮	⋱		
R_j					$R_j(i)$ hält?	
⋮						

Beweis durch Diagonalisierungsargument

- Zähle Rechenmaschinen $R_0, R_1, \ldots$ auf und Argumente $0, 1, 2, \ldots$

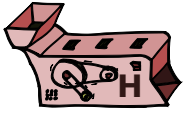
	0	1	2	…	i	…
R_0	$R_0(0)$ hält?	$R_0(1)$ hält?	$R_0(2)$ hält?	…		
R_1	$R_1(0)$ hält?	$R_1(1)$ hält?	$R_1(2)$ hält?	…		
R_2	$R_2(0)$ hält?	$R_2(1)$ hält?	$R_2(2)$ hält?	…		
⋮	⋮	⋮	⋮	⋱		
R_j					$R_j(i)$ hält?	
⋮						

- Diagonalelemente: $R_i(i)$ hält?
- **Ziel:** Baue eine Rechenmaschine X, die den Diagonalelementen widerspricht:

$$X(i) \text{ hält} \iff R_i(i) \text{ hält nicht.}$$

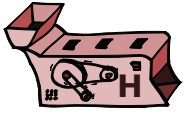
Beweis der Unentscheidbarkeit des Halteproblems

- **Annahme:** Es gibt eine Maschine H, die das Halteproblem entscheidet

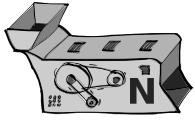


Beweis der Unentscheidbarkeit des Halteproblems

- **Annahme:** Es gibt eine Maschine H, die das Halteproblem entscheidet

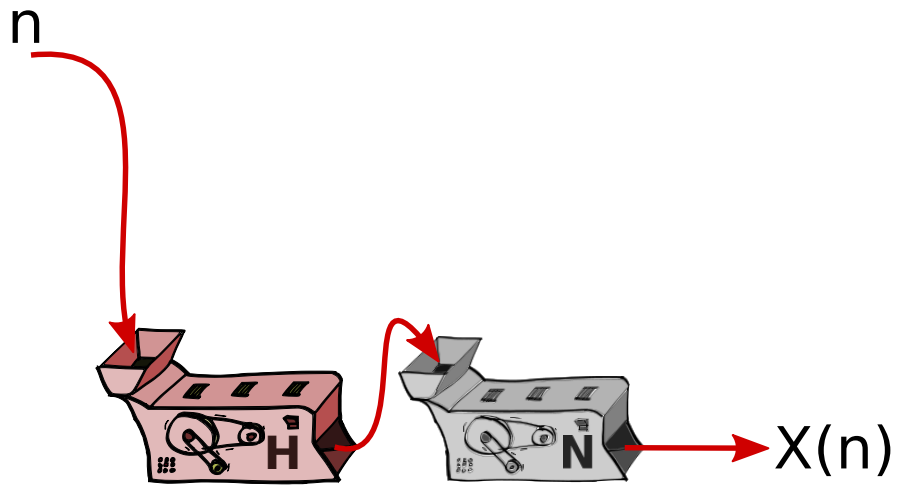


- **Klar:** Es gibt eine Maschine N, die genau für die Eingabe 0 hält.

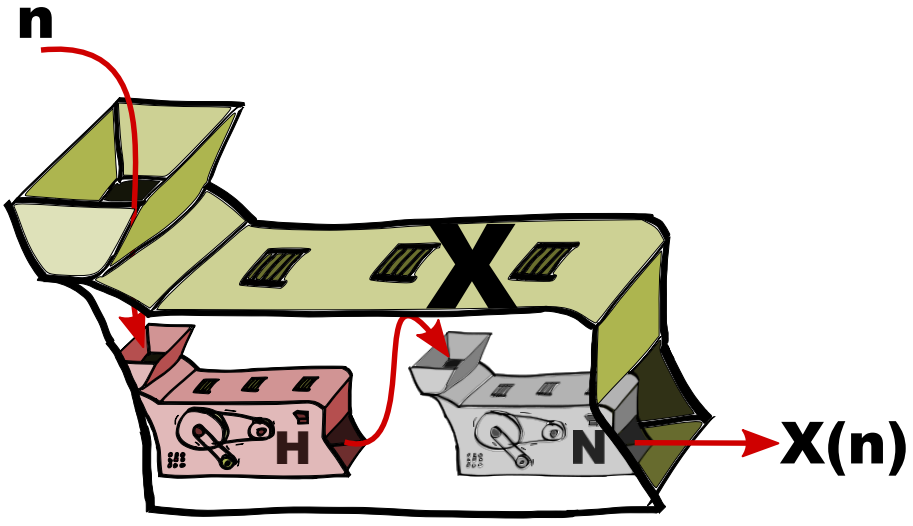


z. B. durch das While-Programm **while** $x \neq 0$ **do skip od**

Beweis der Unentscheidbarkeit des Halteproblems (1)



Beweis der Unentscheidbarkeit des Halteproblems (1)



Beweis der Unentscheidbarkeit des Halteproblems (2)

$$H(n) = \begin{cases} 1 & \text{wenn } U(n,n) \text{ hält.} \\ 0 & \text{wenn } U(n,n) \text{ nicht hält.} \end{cases}$$

$$N(n) = \begin{cases} 0 & \text{wenn } n = 0 \\ \text{hält nicht} & \text{sonst} \end{cases}$$

$$X(n) = (N \circ H)(n) = \begin{cases} \text{hält nicht} & \text{wenn } U(x,x) \text{ hält.} \\ 0 & \text{wenn } U(x,x) \text{ nicht hält} \end{cases}$$

Beweis der Unentscheidbarkeit des Halteproblems (2)

$$H(n) = \begin{cases} 1 & \text{wenn } U(n,n) \text{ hält.} \\ 0 & \text{wenn } U(n,n) \text{ nicht hält.} \end{cases}$$

$$N(n) = \begin{cases} 0 & \text{wenn } n = 0 \\ \text{hält nicht} & \text{sonst} \end{cases}$$

$$X(n) = (N \circ H)(n) = \begin{cases} \text{hält nicht} & \text{wenn } U(x,x) \text{ hält.} \\ 0 & \text{wenn } U(x,x) \text{ nicht hält} \end{cases}$$

Entscheidende Frage: Was ist mit $X(\mathfrak{g}(X))$?

Beweis der Unentscheidbarkeit des Halteproblems (2)

$$H(n) = \begin{cases} 1 & \text{wenn } U(n,n) \text{ hält.} \\ 0 & \text{wenn } U(n,n) \text{ nicht hält.} \end{cases}$$

$$N(n) = \begin{cases} 0 & \text{wenn } n = 0 \\ \text{hält nicht} & \text{sonst} \end{cases}$$

$$X(n) = (N \circ H)(n) = \begin{cases} \text{hält nicht} & \text{wenn } U(x,x) \text{ hält.} \\ 0 & \text{wenn } U(x,x) \text{ nicht hält} \end{cases}$$

Entscheidende Frage: Was ist mit $X(\mathfrak{g}(X))$?

$$X(\mathfrak{g}(X)) = \begin{cases} \text{hält nicht} & \text{wenn } U(\mathfrak{g}(X), \mathfrak{g}(X)) \text{ hält} \\ 0 & \text{wenn } U(\mathfrak{g}(X), \mathfrak{g}(X)) \text{ nicht hält} \end{cases}$$

Beweis der Unentscheidbarkeit des Halteproblems (2)

$$H(n) = \begin{cases} 1 & \text{wenn } U(n,n) \text{ hält.} \\ 0 & \text{wenn } U(n,n) \text{ nicht hält.} \end{cases}$$

$$N(n) = \begin{cases} 0 & \text{wenn } n = 0 \\ \text{hält nicht} & \text{sonst} \end{cases}$$

$$X(n) = (N \circ H)(n) = \begin{cases} \text{hält nicht} & \text{wenn } U(x,x) \text{ hält.} \\ 0 & \text{wenn } U(x,x) \text{ nicht hält} \end{cases}$$

Entscheidende Frage: Was ist mit $X(\mathfrak{g}(X))$?

$$X(\mathfrak{g}(X)) = \begin{cases} \text{hält nicht} & \text{wenn } U(\mathfrak{g}(X), \mathfrak{g}(X)) \text{ hält} \\ 0 & \text{wenn } U(\mathfrak{g}(X), \mathfrak{g}(X)) \text{ nicht hält} \end{cases} = \begin{cases} \text{hält nicht} & X(\mathfrak{g}(X)) \text{ hält} \\ 0 & X(\mathfrak{g}(X)) \text{ hält nicht.} \end{cases}$$

$$\lightning \quad X(\mathfrak{g}(X)) \text{ hält} \iff X(\mathfrak{g}(X)) \text{ hält nicht.}$$

Annahme, dass H existiert, muss fallengelassen werden.

Weitere unentscheidbare Probleme

- Varianten des Halteproblems
  - Beispiel:
    Hält gegebene TM, wenn das Band zu Beginn völlig leer ist?
- Äquivalenzproblem:
  Liefern zwei TM für jede Eingabe die gleiche Ausgabe?
  - automatischer Vergleich mit „Musterlösungen“ unmöglich
- Wird ein bestimmter Zustand einer TM jemals gebraucht?
  - Erreichbarkeit von Codestücken unentscheidbar
- vieles vieles vieles vieles vieles vieles vieles mehr
- Beachte: statt Turingmaschine kann man immer
  Java-Programm einsetzen