D. Hilbert

"Wir müssen wissen, und wir werden wissen."

- David Hilbert (1862-1943)
- ... Eine Diophantische Gleichung [...] sei vorgelegt: man soll ein *Verfahren* angeben, nach welchem sich mittelst einer endlichen Anzahl von Operationen entscheiden läßt, ob die Gleichung in ganzen rationalen Zahlen lösbar ist ...

D. Hilbert: Mathematische Probleme - Vortrag, gehalten auf dem internationalen Mathematiker-Kongreß zu Paris 1900. In: Nachrichten von der Königl. Gesellschaft der Wissenschaften zu Göttingen. Mathematisch-Physikalische Klasse, Heft 3, 1900, 253 – 297.

K. Gödel

"Either mathematics is too big for the human mind or the human mind is more than a machine. "

- Kurt Gödel (1906 - 1978)
- ... Wir haben also einen Satz vor uns, der seine eigene Unbeweisbarkeit behauptet ...

K. Gödel: Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. in: Monatshefte für Mathematik und Physik 38, 1931, 173 – 198.

A. Church

"I tried reading Hilbert. Only his papers published in mathematical periodicals were available at the time. Anybody who has tried those knows they are very hard reading. "

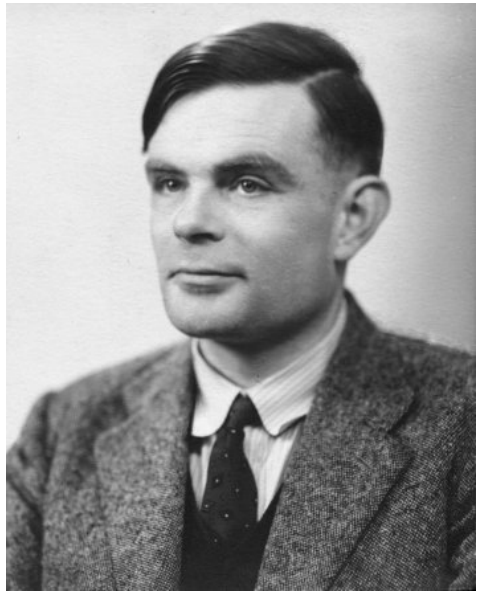
- Alonzo Church (1903 - 1995)
- Entwicklung des Lambda-Kalküls

A. Church: An unsolvable Problem of Elementary Number Theory, American Journal of Mathematics 58 (2), 1936, 345 – 363.

A. Turing

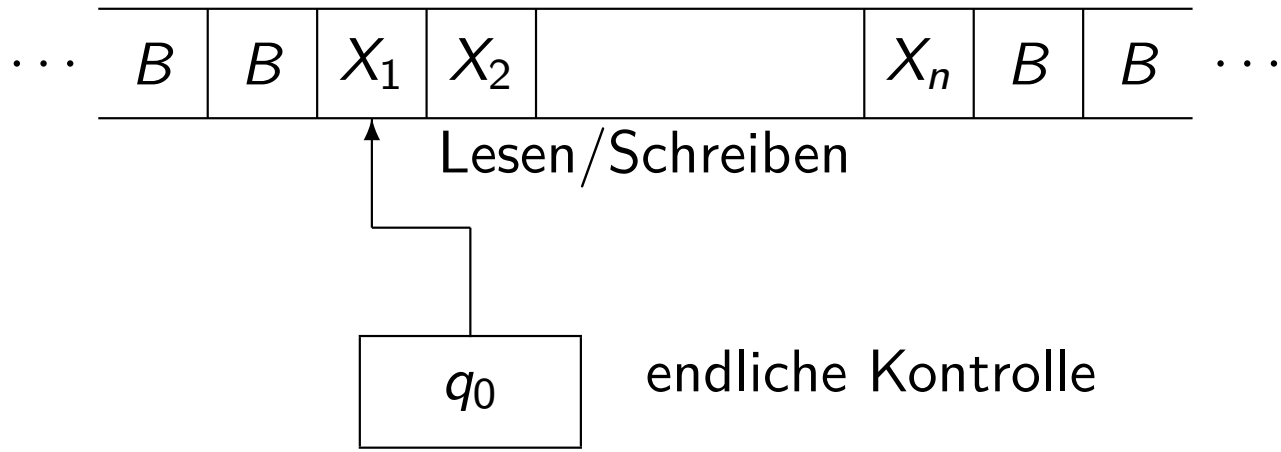
"A man provided with paper, pencil and rubber, and subject to strict discipline, is in effect a universal machine."

- Alan M. Turing (1912 - 1954)
- Turingmaschine als Modell für jede mögliche Berechnung



A. Turing: On Computable Numbers, with an application to the Entscheidungsproblem. Proc. Lond. Math. Soc. (2) 42, 1936, 230 –265.

Turingmaschinen



- Das *Band* enthält die Eingabe. Alle weiteren Zellen enthalten ein besonderes Blank-Symbol.
- Ein Lese-/Schreib-Kopf ist immer über einer der Bandzellen positioniert.
- Am Anfang befindet sich der Lese-/Schreib-Kopf über der am weitesten links stehenden Eingabezelle.

Turingmaschinen

Abhängig vom Zustand und dem gelesenen Bandsymbol führt die TM mit einer Bewegung Folgendes aus:

- Sie wechselt den Zustand
- Sie schreibt ein Bandsymbol in die gelesene Zelle
- Sie bewegt den Lese-/Schreib-Kopf nach links oder rechts, bzw. lässt ihn stehen

Formale Beschreibung einer Turingmaschine

Eine **Turingmaschine** ist ein Siebentupel

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

- Q endliche Menge von Zuständen,
- Σ endliche Menge der Eingabesymbole,
- Γ endliche Menge der Bandsymbole, $\Sigma \subset \Gamma$,
- $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R, S\}$ Übergangsfunktion,
- $q_0 \in Q$ Startzustand,
- $B \in \Gamma - \Sigma$ Blank-Symbol,
- $F \subseteq Q$ Menge von Endzuständen.

Übergänge

$$(q, X; p, Y, D) \in \delta$$

M liest im Zustand q auf dem Band Symbol X

- wechselt nun davon abhängig in den Zustand p,
- überschreibt Symbol X durch Symbol Y,
- bewegt den Lese-/Schreibkopf in die durch D beschriebene Richtung ($D \in \{L, R, S\}$).

Deterministische Turingmaschine (DTM)

Eine Turingmaschine M heißt **deterministisch**, wenn für alle $(q, X) \in Q \times \Sigma \times \Gamma$ höchstens ein Element $(q, X; p, Y, D) \in \delta$ existiert, $D \in \{L, R, S\}$; wir schreiben dann auch $\delta(q, X) = (p, Y, D)$.

Konfiguration

Am Anfang befindet sich nur das Eingabewort auf dem Band, welches nach links und rechts mit unendlich vielen Blank-Symbolen umgeben ist. Die TM ist im Startzustand und der Lese-/Schreibkopf befindet sich über dem am weitesten links stehenden Eingabesymbol.

Mit jeder Bewegung ändert die TM ihre Konfiguration.

Darstellung einer Konfiguration:

$$X_1 X_2 ... X_{i-1} q X_i X_{i+1} ... X_n$$

wobei

- q den Zustand der TM bezeichnet,
- der Lese-/Schreibkopf das $i-$te Symbol von links liest,
- $X_1 X_2 ... X_n$ den Abschnitt des Bandes zwischen den am weitesten links und am weitesten rechts stehenden Zeichen beschreibt, die keine Blank-Symbole (Leerzeichen) sind.

Konfiguration

$\Rightarrow$ ... Übergang zwischen Konfigurationen in einem Schritt[1]
$\Rightarrow^*$ ... Übergang zwischen Konfigurationen in beliebig vielen Schritten

Erfolgt die nächste Bewegung nach

- links, d.h. $\delta(q, X_i) = (p, Y, L)$:

$$X_1 X_2 ... X_{i-1} \mathbf{q} \mathbf{X_i} X_{i+1} ... X_n \Rightarrow X_1 X_2 ... X_{i-2} \mathbf{p} X_{i-1} \mathbf{Y} X_{i+1} ... X_n$$

- rechts, d.h. $\delta(q, X_i) = (p, Y, R)$:

$$X_1 X_2 ... X_{i-1} \mathbf{q} \mathbf{X_i} X_{i+1} ... X_n \Rightarrow X_1 X_2 ... X_{i-1} \mathbf{Y} \mathbf{p} X_{i+1} ... X_n$$

Bleibt der Kopf stehen:

- d.h., $\delta(q, X_i) = (p, Y, S)$:

$$X_1 X_2 ... X_{i-1} \mathbf{q} \mathbf{X_i} X_{i+1} ... X_n \Rightarrow X_1 X_2 ... X_{i-1} \mathbf{p} \mathbf{Y} X_{i+1} ... X_n$$

[1] in [HMU] wird $\Rightarrow$ mit $\vdash$ bezeichnet.

(Deterministische) Turingmaschinen: akzeptierte Sprache

Akzeptierte Sprache

Die von der (deterministischen) Turingmaschine M **akzeptierte Sprache** $L(M)$ besteht aus genau all jenen Wörtern, bei deren Anaylse M einen Endzustand erreicht:

$$L(M) = \{w \in \Sigma^* \mid q_0 w \Rightarrow^* \alpha p \beta, \quad p \in F, \quad \alpha, \beta \in \Gamma^*\}$$

Rekursiv aufzählbare Sprachen

Eine formale Sprache heißt **rekursiv aufzählbar**[2] wenn sie von einer (deterministischen) Turingmaschine akzeptiert wird.

[2] engl: recursively enumerable, auch bezeichnet als: Turing-erkennbar (Turing-recognizable), semi-entscheidbar (semi-decidable)

Beispiel

TM für $\{0^n 1^n \mid n \geq 1\}$

$M = (\{q_i \mid 0 \leq i \leq 4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$
wobei δ gegeben durch:

	Symbol				
Zustand	0	1	X	Y	B
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					

Beispiel ctd.

akzeptierende Berechnung von 0011

	Symbol				
Zustand	0	1	X	Y	B
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					

$$
\begin{array}{lllllllll}
q_0 0011 & \Rightarrow & X q_1 011 & \Rightarrow & X 0 q_1 11 & \Rightarrow & X q_2 0 Y 1 & \Rightarrow \\
q_2 X 0 Y 1 & \Rightarrow & X q_0 0 Y 1 & \Rightarrow & X X q_1 Y 1 & \Rightarrow & X X Y q_1 1 & \Rightarrow \\
X X q_2 Y Y & \Rightarrow & X q_2 X Y Y & \Rightarrow & X X q_0 Y Y & \Rightarrow & X X Y q_3 Y & \Rightarrow \\
X X Y Y q_3 B & \Rightarrow & X X Y Y B q_4 B & & & & & \\
\end{array}
$$

Beispiel ctd.

abweisende Berechnung von 0010

	Symbol				
Zustand	0	1	X	Y	B
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					

$$q_0 0010 \Rightarrow X q_1 010 \Rightarrow X 0 q_1 10 \Rightarrow X q_2 0 Y 0 \Rightarrow$$
$$q_2 X 0 Y 0 \Rightarrow X q_0 0 Y 0 \Rightarrow X X q_1 Y 0 \Rightarrow X X Y q_1 0 \Rightarrow$$
$$X X Y 0 q_1 B$$

(Deterministische) Turingmaschinen: Akzeptieren durch Anhalten

Wir können stets annehmen, dass eine Turingmaschine anhält, wenn sie akzeptiert.

($\delta(q, X)$ kann für $q \in F$ als undefiniert bezeichet werden.)

Im Folgenden betrachten wir Turingmaschinen, die immer anhalten, wenn sie sich in einem akzeptierenden Zustand befinden.

Entscheidbare (Rekursive) Sprachen

Wird eine Turingmaschine M auf einem Input w gestartet, so gibt es folgende Möglichkeiten:

- M hält in einem Zustand $q \in F$ und akzeptiert somit die Eingabe
- M hält in einem Zustand $q \notin F$ und verwirft somit die Eingabe
- M gerät in eine Endlosschleife und hält nicht

Rekursive (Entscheidbare) Sprachen

Eine formale Sprache heißt **rekursiv** oder **entscheidbar**[3] wenn sie von einer (deterministischen) Turingmaschine akzeptiert wird, die **immer hält**. (Man sagt dann auch: die TM **entscheidet** die Sprache)

[3] engl: recursive, decidable, auch bezeichnet als: Turing-entscheidbar (Turing-decidable)

Varianten von Turingmaschinen

- ein Band, nur nach einer Seite unbeschränkt
- mehrere Bänder
- Bänder sind in mehrere Spuren unterteilt
- mehrere Lese-/Schreibköpfe auf den Bändern
- nichtdeterministische Turingmaschinen

Alle diese Varianten sind gleichmächtig in Bezug auf ihre Fähigkeit, Sprachen zu erkennen.

Beispiel

Eine nichtdeterministische Turingmaschine mit 50 sechs-dimensionalen Bändern ist nicht mächtiger als unser ursprüngliches Modell.

Nichtdeterministische Turingmaschinen

Eine **nichtdeterministische Turingmaschine** (NTM) unterscheidet sich zur determinisitischen Variante in der Übergangsfunktion δ:

$$\delta(q, X) = \{(q_1, Y_1, D_1), (q_2, Y_2, D_2), ..., (q_k, Y_k, D_k)\},$$

wobei k eine beliebige endliche ganze Zahl ist.

Eine NTM M akzeptiert eine Eingabe, wenn es eine Folge möglicher Bewegungen gibt, die von der Startkonfiguration zu einer Konfiguration mit einem akzeptierenden Zustand führt.[4]

Zu jeder nichtdeterministischen Turingmaschine M_N gibt es eine deterministische Turingmaschine M_D mit $L(M_N) = L(M_D)$.

[4] Die Existenz eventueller weiterer Auswahlmöglichkeiten ist dabei irrelevant.

(Nicht)deterministische Turingmaschinen

Turingmaschinen können auch Funktionen (auf nicht-negativen ganzen Zahlen) berechnen, bzw. Sprachen erzeugen:

Berechnung von Funktionen

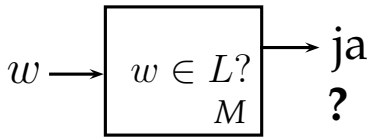
Eine (*deterministische*) Turingmaschine M kann auch zur **Berechnung von Funktionen** verwendet werden, d.h., M beginnt mit dem Input auf dem Band, der Output, also der Funktionswert zum Input, ist das Ergebnis der Berechnung, wenn M hält.

Erzeugung von Sprachen

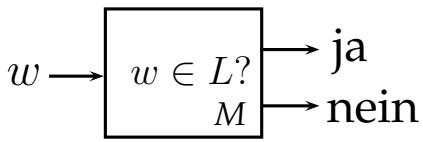
Eine (*nichtdeterministische*) Turingmaschine M kann auch zur **Erzeugung von Wörtern** verwendet werden, d.h., M beginnt mit dem leeren Band, das Ergebnis der Berechnung ist das Wort, das am Ende einer Berechnung, wenn M hält, auf dem Band steht.

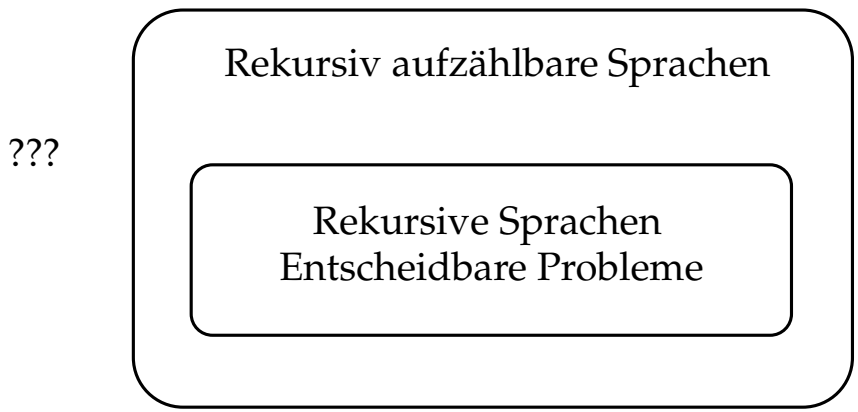
Problemlandschaft

Rekursiv aufzählbare Sprachen L:
Es gibt eine TM M die akzeptiert und hält wenn $w \in L$, sonst aber u.U. unendlich läuft.



Rekursive Sprachen (**Entscheidbare Probleme**) L:
Es gibt eine TM M mit $L(M) = L$ die immer hält. (d.h., es gibt einen **Algorithmus**, der bei Eingabe von w "ja" antwortet, wenn $w \in L$ und "nein" antwortet, wenn $w \notin L$)



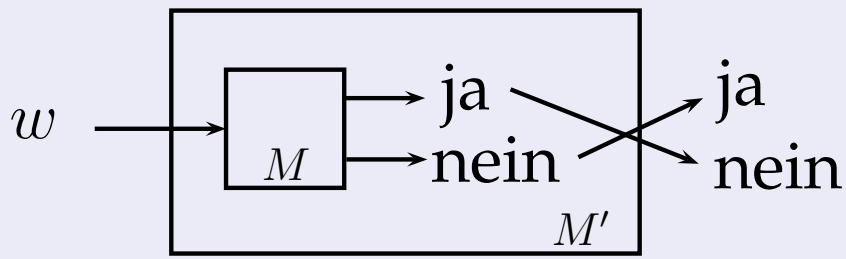


Rekursive (entscheidbare) Sprachen und ihr Komplement

Das Komplement einer rekursiven Sprache ist rekursiv.

Beweis

Sei L rekursiv und M eine TM mit $L(M) = L$, die immer hält. Wir konstruieren M' aus M so, dass M' stoppt ohne zu akzeptieren, wenn M auf einer Eingabe w in einen Endzustand übergeht. Hält M ohne zu akzeptieren, so geht M' in einen Endzustand über. Da eines dieser beiden Ereignisse eintritt, ist $\overline{L} = L(M')$ (das Komplement von L) eine rekursive Sprache.



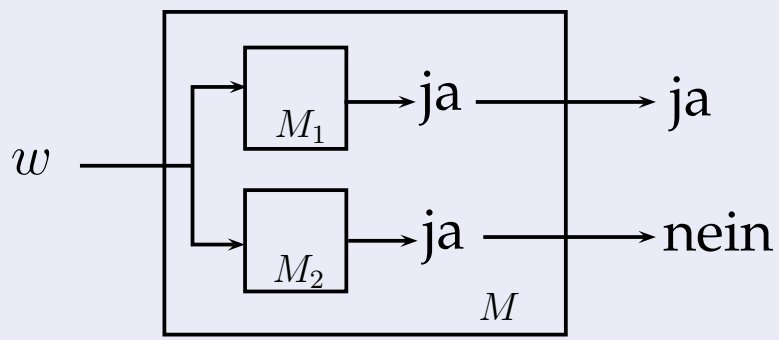
□

Rekursive (entscheidbare) Sprachen und ihr Komplement

Ist eine Sprache L und auch ihr Komplement $\overline{L}$ rekursiv aufzählbar, so ist L (wie auch $\overline{L}$) rekursiv.

Beweis

M_1 und M_2 akzeptieren L bzw. $\overline{L}$. Daraus konstruieren wir M so, dass sie w akzeptiert, wenn w von M_1 akzeptiert wird, und verwirft, wenn w von M_2 akzeptiert wird. Da w entweder in L oder in $\overline{L}$ ist, wird genau eine der beiden Turingmaschinen M_1, M_2 akzeptieren. Also wird M immer entweder "ja" oder "nein" ausgeben, aber nie beides. Somit ist L rekursiv.



□

Rekursive und rekursiv aufzählbare Sprachen

Konsequenz: Sei L eine Sprache und $\overline{L}$ ihr Komplement.
Dann gilt eine der folgenden Aussagen:

1. sowohl L als auch $\overline{L}$ ist rekursiv.
2. sowohl L als auch $\overline{L}$ ist nicht rekursiv aufzählbar.
3. entweder L oder $\overline{L}$ ist rekursiv aufzählbar, aber nicht rekursiv, und die jeweils andere Sprache ist nicht rekursiv aufzählbar.

Codierung von Turingmaschinen

Turingmaschinen können als (binäre) Strings codiert werden, beispielsweise so:

Sei $M = (\{q_1, q_2, ..., q_r\}, \{0, 1\}, \{0, 1, B, X_4, ..., X_s\}, \delta, q_1, B, \{q_2\})$

Codierung von $\delta(q_i, X_j) = (q_k, X_l, D_m)$:

$$0^i 1 0^j 1 0^k 1 0^l 1 0^m$$

Codierung von M:

$$C_1 11 C_2 11 ... 11 C_{n-1} 11 C_n$$

wobei C_i der Code einer Transition ist.

Nicht rekursiv aufzählbare Sprachen

Es gibt Sprachen über $\Sigma = \{0, 1\}$, die nicht rekursiv aufzählbar sind.

Beweis

- Zu jeder rekursiv aufzählbaren Sprache L über $\Sigma = \{0, 1\}$ (d.h. $L \in \mathcal{P}(\Sigma^*)$) gibt es eine Turingmaschine, die sie akzeptiert.
- Jede Turingmaschine M kann als String über $\Sigma = \{0, 1\}$ (d.h. $M \in \Sigma^*$) dargestellt werden.
- Die Menge aller Sprachen $\mathcal{P}(\Sigma^*)$ ist überabzählbar (unendlich), die Menge aller Wörter Σ^* hingegen abzählbar (unendlich).
- Es gibt also überabzählbar viele Sprachen, von denen jedoch nur abzählbar viele von einer Turingmaschine akzeptiert werden.

□

Church-Turing-These

Gibt es aber vielleicht andere Modelle von Generierungs- oder Analysemechanismen, durch die mehr formale Sprachen als durch Turingmaschinen beschrieben werden können?

Nein!

Auch alle anderen seither entwickelten Modelle zur Formalisierung des Begriffs Algorithmus erwiesen sich als nicht mächtiger als Turingmaschinen. Folgende These wird daher allgemein akzeptiert:

Church-Turing-These

Gibt es ein endlich beschreibbares Verfahren zur exakten Spezifizierung einer formalen Sprache L, so gibt es eine Turingmaschine, die L akzeptiert.

(Un)Entscheidbarkeit

Si te mentiri dicis idque verum dicis, mentiris an verum dicis?

Cicero
Academicorum priorum, Lib. II, Kap. 29, Par. 95

(Wenn du sagst, dass du lügst, und das die Wahrheit ist, lügst du dann oder sprichst du die Wahrheit?)

Universelle Turingmaschinen

Es gibt Turingmaschinen, welche das Verhalten beliebiger Maschinen auf beliebigen Eingabewörtern simulieren können:

Eine **Universelle Turingmaschine** U nimmt als Input

- den Code einer Turingmaschine M
- ein Eingabewort w

und akzeptiert (M, w) genau dann, wenn M die Eingabe w akzeptiert.

Turingmaschinen können also auf ihren eigenen Code angesetzt werden!

Das Halteproblem

Im Folgenden betrachten wir Turingmaschinen, die genau dann akzeptieren, wenn sie halten.

Gibt es auch eine Turingmaschine, die auf jeder beliebigen Eingabe hält, und somit einen Algorithmus darstellt, der für beliebige Turingmaschinen und beliebige Wörter über dem Alphabet $\{0, 1\}$ die Frage entscheidet, ob die Turingmaschine M auf dem Eingabewort w hält (und damit akzeptiert) oder nicht?

Halteproblem: Gibt es einen Algorithmus (eine Turingmaschine), der (die), gegeben den Code einer beliebigen Turingmaschine M und den Code eines Eingabewortes w, immer entscheiden kann, ob M auf w hält, bzw. ob M die Eingabe w akzeptiert?

Die Antwort darauf ist **nein**.

Das Halteproblem ist rekursiv aufzählbar

$L_u = \{(M, w) \mid M \text{ ist eine TM die } w \text{ akzeptiert } \}$

L_u ist **rekursiv aufzählbar**, aber **nicht entscheidbar**.

Beweis Teil 1: L_u ist rekursiv aufzählbar

Wir konstruieren eine universelle Turingmaschine U wie folgt: Mit Eingabe (M, w) simuliert U die TM M auf w. Wenn M die Eingabe akzeptiert, so akzeptiert auch U, akzeptiert M nicht, so akzeptiert auch U nicht:

$$U \text{ akzeptiert } (M, w) \iff M \text{ akzeptiert } w \iff (M, w) \in L_u$$

Folglich akzeptiert U die Sprache L_u. □

Das Halteproblem ist unentscheidbar

Beweis Teil 2: L_u ist **nicht entscheidbar**

Beweis indirekt. Angenommen, es gibt eine Maschine H, die L_u *entscheidet*:

$$H((M,w)) = \begin{cases} \textbf{ja} & \text{wenn } w \text{ von } M \text{ akzeptiert wird} \\ \textbf{nein} & \text{wenn } w \text{ von } M \text{ nicht akzeptiert wird} \end{cases}$$

Dann gibt es aber auch eine Maschine D, die genau das Gegenteil von H macht.
D simuliert zunächst H mit der Eingabe $(M,(M))$ und gibt dann **ja** aus, wenn M nicht akzeptiert, bzw. **nein**, wenn M akzeptiert:

$$D((M)) = \begin{cases} \textbf{ja} & \text{wenn } (M) \text{ von } M \text{ nicht akzeptiert wird} \\ \textbf{nein} & \text{wenn } (M) \text{ von } M \text{ akzeptiert wird} \end{cases}$$

Offensichtlich gilt: wenn H existiert, so existiert auch D.

Das Halteproblem ist unentscheidbar - ctd.

Setzt man nun D auf ihren eigenen Code an, so ergibt sich folgendes Verhalten:

$$D((D)) = \begin{cases} \textbf{ja} & \text{wenn } (D) \text{ von } D \text{ nicht akzeptiert wird} \\ \textbf{nein} & \text{wenn } (D) \text{ von } D \text{ akzeptiert wird} \end{cases}$$

Was auch immer D macht, sie muss genau das Gegenteil machen. Das ist offensichtlich ein Widerspruch! Es kann also weder D noch H geben, demnach ist L_u nicht entscheidbar. $\square$

Diagonalargument

- H akzeptiert $(M, w) \iff M$ akzeptiert w
- D weist (M) zurück $\iff M$ akzeptiert (M)
- D weist (D) zurück $\iff D$ akzeptiert (D)

	(M_1)	(M_2)	(M_3)	$\cdots$	(D)	$\cdots$
M_1	**ja**	*nein*	*ja*	$\cdots$	*ja*	$\cdots$
M_2	*ja*	**ja**	*nein*	$\cdots$	*ja*	$\cdots$
M_3	*nein*	*nein*	**nein**	$\cdots$	*nein*	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$		
D	**nein**	**nein**	**ja**	$\cdots$	**?**	
$\vdots$	$\vdots$	$\vdots$	$\vdots$			$\ddots$

Ein nicht rekursiv aufzählbares Problem

$\overline{L_u} = \{(M, w) \mid M \text{ ist eine TM die } w \text{ nicht akzeptiert } \}$

$\overline{L_u}$ ist **nicht rekursiv aufzählbar**.

Beweis

Angenommen, $\overline{L_u}$ ist rekursiv aufzählbar. Da L_u rekursiv aufzählbar ist, müssten demnach beide entscheidbar sein, was aber nicht der Fall ist!
$\overline{L_u}$ kann nicht rekursiv aufzählbar sein. □

Berechenbarkeit

Eine Funktion $f : \Sigma^* \longrightarrow \Sigma^*$ heißt **partiell berechenbar**, wenn es eine Turingmaschine M_f gibt, für die mit $x \in \Sigma^*$ gilt:

- Ist $f(x)$ definiert, so hält M_f, gestartet mit x, und $f(x)$ steht dann auf dem Band.
- Ist $f(x)$ nicht definiert, so hält M_f, gestartet mit x, nicht.

Ist f total, d. h., für alle $x \in \Sigma^*$ definiert, und partiell berechenbar, so heißt f (total) **berechenbar**.[5]

[5] engl: computable

Reduktion

Reduktion ist eine Technik, die Unentscheidbarkeit von Problemen durch Rückführung auf andere Probleme nachzuweisen, deren Unentscheidbarkeit bereits bekannt ist.

Reduktion

Eine Sprache $A \subseteq \Sigma^*$ heißt **reduzierbar** auf die Sprache $B \subseteq \Gamma^*$, (bezeichnet mit $A \leq B$), wenn es eine (total) berechenbare Funktion $f : \Sigma^* \longrightarrow \Gamma^*$ so gibt, dass

$$\text{für alle } w \in \Sigma^*\text{: } w \in A \text{ genau dann, wenn } f(w) \in B.$$

In diesem Fall nennt man f eine **Reduktion** von A auf B.

(Anmerkung: die Relation $\leq$ ist transitiv und reflexiv)

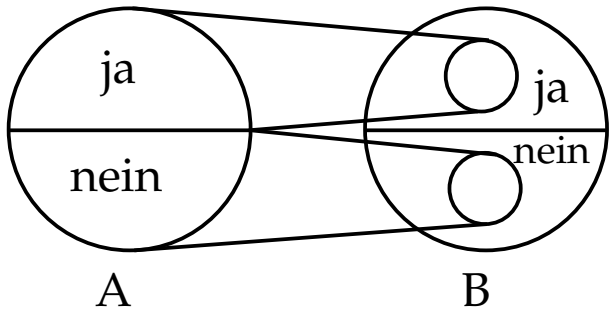
Reduktionen und (Nicht-)Entscheidbarkeit

Seien A und B Sprachen mit $A \leq B$. Dann gilt:

- Ist A nicht entscheidbar, so ist auch B nicht entscheidbar.
- Ist A nicht rekursiv aufzählbar, so ist auch B nicht rekursiv aufzählbar.
- Ist B entscheidbar, so ist auch A entscheidbar.
- Ist B rekursiv aufzählbar, so ist auch A rekursiv aufzählbar.

Reduktion als Beweistechnik

Reduktionen überführen positive in positive Instanzen und negative in negative Instanzen:



Ist die Unentscheidbarkeit von A bekannt, so genügt für den Beweis der Unentscheidbarkeit von B die Angabe einer **Reduktion von** A **auf** B.

Ist die Entscheidbarkeit von B bekannt, so genügt für den Beweis der Entscheidbarkeit von A die Angabe einer **Reduktion von** A **auf** B.

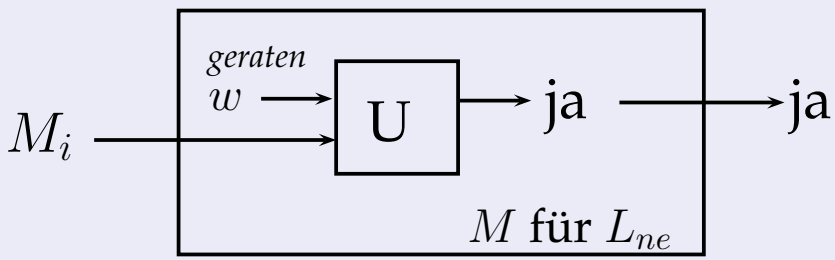
Weitere Unentscheidbare Probleme

$L_e = \{M \mid L(M) = \{\} \quad \}$
$L_{ne} = \{M \mid L(M) \neq \{\} \quad \}$

$L_{ne} = \{M \mid L(M) \neq \{\} \quad \}$ ist rekursiv aufzählbar.

Beweis (Idee)

Nichtdeterministische Turingmaschine, die L_{ne} akzeptiert:



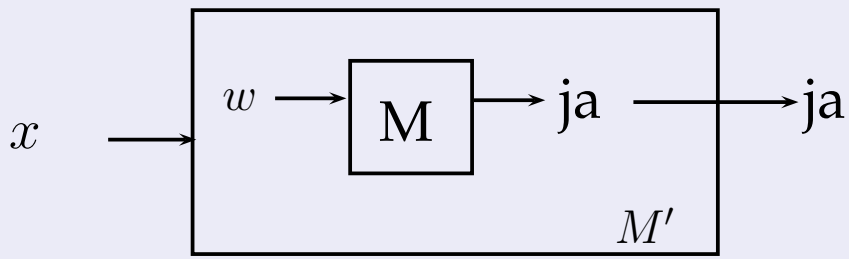
Weitere Unentscheidbare Probleme

$L_{ne} = \{M \mid L(M) \neq \{\} \quad \}$ ist nicht entscheidbar.

Beweis

(Indirekt, durch Reduktion von L_u auf L_{ne} ($L_u \leq L_{ne}$))
Angenommen, es gibt eine Turingmaschine M, sodass $L(M) = L_{ne}$.
Dann konstruieren wir eine Maschine M', die beliebige Eingaben genau dann akzeptiert, wenn M die Eingabe w akzeptiert:

$$w \in L(M) \Leftrightarrow L(M') \neq \{\}$$



Nachdem L_u nicht entscheidbar ist, kann auch L_{ne} nicht entscheidbar sein. □

Eigenschaften von Sprachen

Eine **Eigenschaft** von Sprachen ist eine Teilmenge P von rekursiv aufzählbaren Sprachen (über einem geeigneten Alphabet).

Beispiele für Eigenschaften

$P_1 = \{L \mid L \text{ ist entscheidbar } \}$
$P_2 = \{L \mid L \text{ ist endlich } \}$
$P_3 = \{L \mid L = L^*\}$
$P_4 = \{\{\varepsilon\}\}$
$P_5 = \{\}$
$P_6 = \{L \mid L \text{ ist rekursiv aufzählbar } \}$

$\{a, b\}^*$ hat die Eigenschaften P_1, P_3, P_6, aber nicht P_2, P_4, P_5

Der Satz von Rice[6]

Eine Eigenschaft ist **trivial**, wenn sie entweder leer ist (d.h., sie kommt keiner Sprache zu), oder aus allen rekursiv aufzählbaren Sprachen besteht. Andernfalls ist sie nicht trivial.

Triviale Eigenschaften

$P_5 = \{\}$
$P_6 = \{L \mid L \text{ ist rekursiv aufzählbar } \}$
sind triviale Eigenschaften.

Satz von Rice

Jede nicht triviale Eigenschaft der rekursiv aufzählbaren Sprachen ist unentscheidbar.

[6] Henry Gordon Rice (* 1920)

Anmerkung

Wir können eine Menge von Sprachen nicht als die Sprachen selbst erkennen. Stattdessen müssen wir die Turingmaschinen erkennen, die diese Sprache akzeptieren.

Ist P eine Eigenschaft der rekursiv aufzählbaren Sprachen, dann besteht die Sprache L_P aus der Menge der Codes für Turingmaschinen M_i, sodass $L(M_i)$ eine Sprache aus P ist (d.h. $L_P = \{M_i \mid L(M_i) \in P\}$).

Wenn wir über die Entscheidbarkeit einer Eigenschaft P sprechen, so meinen wir die Entscheidbarkeit der Sprache L_P.

Der Satz von Rice: Anwendungen

Einige Beispiele von Eigenschaften, für die der Satz von Rice unmittelbar die Unentscheidbarkeit impliziert.

- Es ist nicht entscheidbar, ob eine Turingmaschine mehr als sieben Wörter akzeptiert.
  (Die Eigenschaft P ist also definiert durch $P = \{L \mid |L| \geq 7\}$.
  Um den Satz von Rice anwenden zu können, müssen wir noch überprüfen, ob die Eigenschaft P nicht-trivial ist. Wir müssen also mindestens eine rekursiv aufzählbare Sprache finden, die die Eigenschaft erfüllt, und eine, die die Eigenschaft nicht erfüllt: z.B. $L_1 = \{a, a^2, a^3, \ldots, a^7\} \in P$ und $L_2 = \{\} \notin P$.)
- Es ist nicht entscheidbar, ob eine Turingmaschine nur endlich viele Wörter akzeptiert (d. h. ob die von einer Turingmaschine akzeptierte Sprache endlich ist).
- Es ist nicht entscheidbar, ob die von einer Turingmaschine akzeptierte Sprache regulär ist.

Nicht Anwendbarkeit des Satzes von Rice

Es gibt aber durchaus Eigenschaften für Turingmaschinen, die entscheidbar sind. Für diese Eigenschaften ist dann aber eine der Voraussetzungen des Satzes von Rice nicht erfüllt. (Entweder handelt es sich nicht um eine Eigenschaft der akzeptierten Sprache der Turingmaschine oder die Eigenschaft ist trivial)

- Es ist entscheidbar, ob eine Turingmaschine mindestens 7 Zustände besitzt.
  (Diese Eigenschaft ist keine Eigenschaft der akzeptierten Sprache, sondern eine Eigenschaft der Turingmaschine selbst.)
- Es ist entscheidbar, ob die von einer Turingmaschine akzeptierte Sprache rekursiv aufzählbar ist.
  (Diese Eigenschaft ist trivial, sie ist für jede rekursiv aufzählbare Sprache per Definition erfüllt.)

Das Postsche Korrespondenz Problem[7]

Das Postsche Korrespondenz Problem (PCP)

Gegeben: Alphabet A, $k \in \mathbb{N}$, sowie die Folge von Wortpaaren $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$ mit $x_i, y_i \in A^+$ für $1 \leq i \leq k$.

Frage: Gibt es eine Folge von Indizes $i_1, i_2, \ldots, i_n$ mit $i_j \in \{1, 2, \ldots, k\}$ für $1 \leq j \leq n, n \in \mathbb{N}$ so dass $x_{i_1} x_{i_2} \ldots x_{i_n} = y_{i_1} y_{i_2} \ldots y_{i_n}$ gilt?

(Post 1946)

Das Das Postsche Korrespondenzproblem ist nicht entscheidbar.

Das Das Postsche Korrespondenzproblem ist rekursiv aufzählbar.

[7] Emil Post, amerikanischer Mathematiker, 1897-1954

PCP: Beispiel

Das Korrespondenzproblem
$A = \{0, 1\}, \quad k = 3, \quad K = ((1, 101), (10, 00), (011, 11))$, also

$$x_1 = 1 \qquad x_2 = 10 \quad x_3 = 011$$
$$y_1 = 101 \quad y_2 = 00 \quad y_3 = 11$$

besitzt die Lösung $(1, 3, 2, 3)$, denn es gilt

$$x_1 x_3 x_2 x_3 = 101110011 = 101110011 = y_1 y_3 y_2 y_3.$$

Anmerkung:
Von der Variante PCP(s) konnte Folgendes gezeigt werden:
(Gibt es eine Lösung zu einem Satz mit maximaler Größe s?)
für $s = 2$ entscheidbar (Ehrenfeucht, Karhumäki, Rozenberg 1982)
für $s >= 7$ unentscheidbar (Matiyasevich, Senizergues 1996)
für $s >= 4$ unentscheidbar (Neary, 2013)
für $s = 3$ noch ungeklärt

Das 10. Hilbertsche Problem[8]

10. Hilbertsche Problem

Gegeben: $n \in \mathbb{N}$, ein Polynom $p(x_1, \ldots, x_n)$ in n Unbekannten
Frage: Besitzt p ganzzahlige Nullstellen?

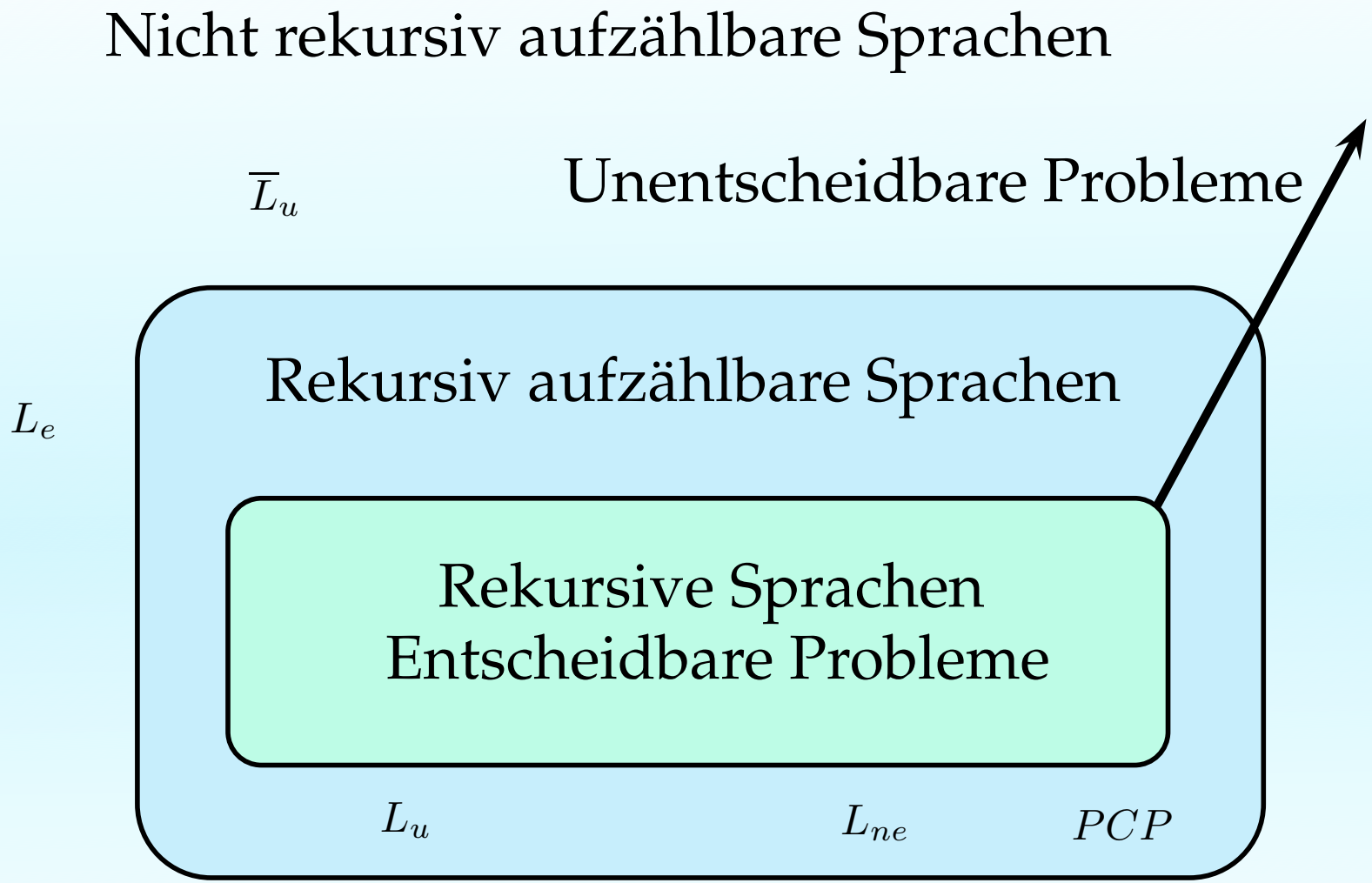
(Matijassewitsch, 1970)

Das 10. Hilbertsche Problem ist nicht entscheidbar.

[8] David Hilbert (1862-1943)

Beispiele für unentscheidbare Probleme in der Praxis

Folgende Probleme sind nur in Einzelfällen lösbar, nicht jedoch für beliebige Programme und Spezifikationen:

- Feststellen, ob eine bestimmte Codezeile in einem Programm jemals ausgeführt wird
- Feststellen, ob es eine Endlosschleife in einem Programm gibt
- Feststellen, ob 2 Programme die gleiche Wirkung (Semantik) haben
- Feststellen, ob ein Programm eine Spezifikation erfüllt
- Zu einer Spezifikation automatisch ein Programm generieren
- Feststellen, ob 2 Spezifikationen die gleiche Klasse von Programmen festlegen

Problemlandschaft