II. Entscheidbarkeit

1. Entscheidbarkeit und Semi-Entscheidbarkeit

Vorbemerkungen:

a) Berechenbare Funktionen bisher: Funktionen auf natürlichen Zahlen.
Ausnahme Turing-Berechenbarkeit: auch schon definiert für Funktionen f: $\Sigma^* \to \Sigma^*$.

Lässt sich für andere Berechenbarkeitsbegriffe über Codierungen in natürliche Zahlen analog einführen:
seien code: $\Sigma^* \to \mathbb{N}$ und decode: $\mathbb{N} \to \Sigma^*$ totale Turing-berechenbare Funktionen, so dass decode(code(w)) = w.
(solche Codierungs- und Decodierungsfunktionen lassen sich immer finden, z.B. kann man jedes Wort über dem Alphabet $\Sigma = \{a_1, ..., a_k\}$ als Zahl zur Basis k+1 lesen).

Wir sagen f: $\Sigma^* \to \Sigma^*$ ist WHILE berechenbar, wenn es WHILE berechenbare Funktion g gibt mit f(w) = w' gdw decode(g(code(w)) = w'.
Ähnlich für andere Berechenbarkeitsbegriffe.

b) Sei M TM zur Berechnung von f. Aus M kann eine TM M' konstruiert werden, die testet, ob M nach s Schritten stoppt.
Grundidee: weiteres Band j, initialisiert mit s. Nach jedem Konfigurationsübergang von M wird Band j um 1 dekrementiert.

M stoppt bevor Band j = 0	=> stoppe mit Ausgabe 0
Band j = 0 bevor M stoppt	=> stoppe mit Ausgabe 0
M stoppt und gleichzeitig Band j = 0	=> stoppe mit Ausgabe 1

Def. : Eine Menge $A \subseteq \Sigma^*$ heißt entscheidbar, gdw. die charakteristische Funktion von A

$$\chi_A: \Sigma^* \to \{0,1\}$$

$$\text{mit} \quad \chi_A(w) = \begin{cases} 1 & \text{falls } w \in A \\ 0 & \text{falls } w \notin A \end{cases}$$

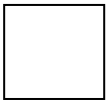
berechenbar ist.

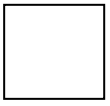
Eine Menge $A \subseteq \Sigma^*$ heißt semi-entscheidbar, gdw. die "halbe" charakteristische Funktion von A

$$\chi'_A: \Sigma^* \to \{0,1\}$$

$$\text{mit} \quad \chi'_A(w) = \begin{cases} 1 & \text{falls } w \in A \\ \text{undefiniert} & \text{falls } w \notin A \end{cases}$$

berechenbar ist.

entscheidbar: w ->  → ja / → nein terminiert immer

semi-entscheidbar: w ->  → ja / → undefiniert

Satz: A ist entscheidbar gdw $\overline{A}$ (das Komplement von A) entscheidbar ist.

Beweis: um aus einer TM M, die A entscheidet, eine TM zu erzeugen, die das Komplement von A entscheidet, sind nur die Ausgabewerte 1 und 0 zu vertauschen. Dazu ist M nur mit einer entsprechenden TM für das Vertauschen dieser Werte hintereinander zu schalten. Entsprechendes gilt für die umgekehrte Richtung.

Satz: A ist entscheidbar gdw A und $\overline{A}$ (das Komplement von A) semi-entscheidbar sind.

Beweis: "=>" offensichtlich, TM geht bei 0 bzw. 1 in Endlosschleife (im Fall des Komplements 0 in 1 umwandeln).
"<=" TM M_1 berechnet χ'_A, M_2 berechnet $\chi'_{\overline{A}}$. Konstruiere TM M, die A entscheidet, auf folgende Weise: M führt abwechselnd einen Schritt von M_1 und einen von M_2 aus. Wenn M_1 mit 1 terminiert, gibt M 1 aus. Wenn M_2 mit 1 terminiert, gibt M 0 aus.

Def.: $A \subseteq \Sigma^*$ heißt rekursiv aufzählbar gdw $A = \emptyset$ oder es gibt eine totale berechenbare Funktion $f: \mathbb{N} \rightarrow \Sigma^*$, so dass $A = \{f(0), f(1), f(2), ... \}$.
(Anmerkung: $f(i) = f(j)$ für $i \neq j$ zulässig).

Satz: Eine Sprache ist rekursiv aufzählbar gdw sie semi-entscheidbar ist.

Beweis: "=>" Sei f Funktion, die A aufzählt. Die TM, die χ'_A berechnet, lässt sich so beschreiben:

```
INPUT(x)
FOR n := 0,1,2, 3, … DO
        IF f(n) = x THEN OUTPUT(1) END
END
```

Anmerkung: wir verwenden hier eine Programmiersprachennotation für Turingmaschinen. Gemeint ist die Mehrband-TM M, die gestartet mit x auf dem Eingabeband und 0 auf dem Band n die TM M' zur Berechnung von f(n) ausführt (Eingabeband von M' ist Band n). Falls das Ergebnis der Berechnung von M' identisch mit x ist, schreibt M 1 auf das Ausgabeband, ansonsten wird n inkrementiert usw.

"<=" $A \neq \emptyset$ sei semi-entscheidbar mittels M, a festes Element aus A. Gesucht ist eine totale berechenbare Funktion f mit Wertebereich A. Folgende (in Programmiersprachennotation beschriebene) TM berechnet ein solches f:

INPUT(n);
x:= e_c(n);
y:= f_c(n);
w := decode(x);
IF M gestartet mit Eingabe w stoppt nach y Schritten THEN OUTPUT(w) ELSE OUTPUT(a)

n ist die Codierung eines Paares natürlicher Zahlen (x,y), x ist Codierung eines Wortes aus Σ^*, decode(x) liefert dasjenige Wort aus Σ^*, das als x codiert wird. Der Algorithmus ist total, gibt nur Worte aus A aus. Noch zu zeigen: jedes Element aus A wird ausgegeben. Sei w beliebiges Wort aus A, z der Code von w. Sei n = c(z,s), wobei s die Zahl der Schritte ist, die M braucht, um bei Eingabe von w mit Ausgabe 1 zu terminieren. Nach Konstruktion liefert der Algorithmus w bei Eingabe n.

Satz: Folgende Aussagen sind äquivalent:

1) A ist rekursiv aufzählbar.
2) A ist semi-entscheidbar.
3) A ist vom Typ 0.
4) A = T(M) für eine Turingmaschine M.
5) χ'_A ist berechenbar.
6) A ist der Definitionsbereich einer berechenbaren Funktion.
7) A ist der Wertebereich einer berechenbaren Funktion.

Beweis:
1 <=> 2 soeben bewiesen; 3 <=> 4 im letzten Semester (Vorlesung Automaten und Sprachen) gezeigt; 2 <=> 5 Definition.
4 => 5 akzeptierende TM M wird so modifiziert, dass Ausgabe 1 produziert wird.
5 => 4 TM für χ'_A ist gleichzeitig auch akzeptierende TM für A.
5 => 6 Definitionsbereich von χ'_A ist gerade A.
6 => 5 wenn A Definitionsbereich von f ist, dann kann χ'_A berechnet werden, indem die Ausgabe von f mit 1 überschrieben wird.
1 => 7 Definition.
7 => 1 Konstruiere Aufzählungsfunktion h auf Basis von g mit Wertebereich A so:

h(n) = a, falls g bei Eingabe e(n) nach f(n) Schritten a liefert
a' sonst, wobei a' beliebiges festes Element von A ist.

(Falls g nicht $\mathbb{N}$ als Definitionsbereich hat, muss der Definitionsbereich von g erst noch in die natürlichen Zahlen codiert werden).

Bemerkung: der Begriff „höchstens abzählbar" (= endlich oder abzählbar unendlich) kann wie rekursive Aufzählbarkeit definiert werden (Existenz einer surjektiven Funktion von $\mathbb{N}$ in die betreffende Menge[1]), aber f muss nicht berechenbar sein.
Teilmengen höchstens abzählbarer Mengen sind höchstens abzählbar: sei $A' \subseteq A$, f Abzählung von A, a beliebiges Element aus A'. Dann ist

g(n) = f(n) falls f(n) $\in$ A'
a sonst

Abzählung von A. Aber: g nicht notwendiger Weise berechenbar, selbst wenn f berechenbar.

[1] Abzählbar unendlich heißt gleichmächtig mit $\mathbb{N}$, d.h. es gibt bijektive Funktion von $\mathbb{N}$ in die betreffende Menge. Falls A unendlich, so kann aus einer surjektiven Funktion f: $\mathbb{N} \to A$ immer eine bijektive g: $\mathbb{N} \to A$ konstruiert werden: g(n) = f(m), wobei $m \geq n$ die kleinste Zahl ist, so dass $f(m) \notin \{g(0), g(1), \ldots, g(n-1)\}$.

2. Nicht-entscheidbare Probleme: das Halteproblem

Codierung von Turingmaschinen als Wort über {0,1}.
Sei $\Gamma = \{a_0, ..., a_k\}$, $Z = \{z_0, ..., z_n\}$, z_0 Startzustand, z_n Endzustand (jede TM kann in eine äquivalente mit genau 1 Endzustand überführt werden).

Für jede δ-Regel: $\delta(z_i, a_j) = (z_{i'}, a_{j'}, y)$

zugeordnetes Wort: ##bin(i)#bin(j)#bin(i')#bin(j')#bin(m)
wobei m = 0 falls y = L
1 falls y = R
2 falls y = N

alle so erzeugten Wörter in beliebiger Reihenfolge ergeben Wort über {0,1,#}
daraus wird Wort über {0,1} erzeugt durch Codierung, etwa

0 -> 00
1 -> 01
-> 11

Nicht jedes Wort ist der Code einer Turingmaschine, deshalb:
Sei M^ beliebige, feste Turingmaschine. Für jedes $w \in \{0,1\}^*$ definieren wir:

M_w = M falls w Code der Turingmaschine M ist
M^ sonst

Def.: Das spezielle Halteproblem (Selbstanwendbarkeitsproblem) ist die Sprache

$$K = \{ w \in \{0,1\}^* \mid M_w \text{ hält bei Eingabe } w\}$$

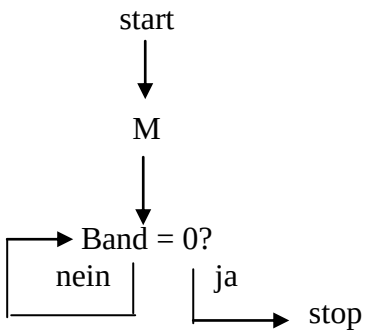
Satz: Das spezielle Halteproblem ist nicht entscheidbar.

Motivation des Beweises: Betrachte die folgende unendliche Tabelle, die angibt, ob M_{wi} bei Eingabe w_j anhält (1) oder nicht (0). w_0, w_1, w_2, … ist eine Aufzählung von $\{0,1\}^*$ (die Einträge in der folgenden Tabelle sind willkürlich gewählt). Die Einträge in der Diagonale geben an, ob eine TM bei Eingabe ihres eigenen Codes hält oder nicht:

input\TM	M_{w0}	M_{w1}	M_{w2}	M_{w3}	…
w_0	**0**	1	0	0	
w_1	1	**0**	1	1	
w_2	0	0	**1**	1	
w_3	1	1	0	**0**	
…					

Der Beweis konstruiert eine Turingmaschine M', die bei Eingabe w_i das dem Terminierungsverhalten von M_{wi} (bei derselben Eingabe) gegenteilige Verhalten aufweist. Da M' selber in der Aufzählung der Turingmaschinen vorkommen muss, müsste sich das Terminierungsverhalten von M' an mindestens einer Stelle von sich selbst unterscheiden, was nicht sein kann.

Bew.: Angenommen K ist entscheidbar. Dann gibt es eine Turingmaschine M, die χ_K berechnet. Aus dieser TM lässt sich folgende Turingmaschine M' konstruieren:



M' stoppt genau dann wenn M 0 berechnet, sonst geht M' in Endlosschleife.

Sei w' der Code von M'. Es gilt:

M' hält bei Eingabe w'	$\Leftrightarrow$	M berechnet 0 bei Eingabe w'
	$\Leftrightarrow$	$\chi_K(w') = 0$
	$\Leftrightarrow$	$w' \notin K$
	$\Leftrightarrow$	$M_{w'}$ hält nicht bei Eingabe w'
	$\Leftrightarrow$	M' hält nicht bei Eingabe w'

Widerspruch, damit muss die Annahme, K sei entscheidbar, falsch sein.

Unentscheidbarkeitsbeweise häufig durch Reduktion:

Def.: Eine Sprache $A \subseteq \Sigma^*$ heißt reduzierbar auf eine Sprache $B \subseteq \Gamma^*$ ($A \leq B$), genau dann wenn es eine totale berechenbare Funktion $f: \Sigma^* \rightarrow \Gamma^*$ gibt, so dass für alle $w \in \Sigma^*$:

$$w \in A \Leftrightarrow f(w) \in B.$$

Intuitiv: wenn A auf B reduzierbar ist, dann beinhaltet eine Lösung des Entscheidungsproblems von B eine Lösung von A. Es ist also mindestens so schwer, B zu entscheiden, wie A zu entscheiden.

Lemma: Falls $A \leq B$ und B (semi-)entscheidbar, so ist auch A (semi-)entscheidbar.
Beweis: Sowohl f wie χ_B sind berechenbar, damit auch ihre Komposition $\chi_B \circ f$. Es gilt

$$\chi_A = \chi_B \circ f$$

denn

$\chi_A(w) = 1 \Leftrightarrow w \in A \Leftrightarrow f(w) \in B \Leftrightarrow \chi_B(f(w)) = 1$

und

$\chi_A(w) = 0 \Leftrightarrow w \notin A \Leftrightarrow f(w) \in \Gamma^* \backslash B \Leftrightarrow \chi_B(f(w)) = 0$

Analog für χ'_A.

Oft Kontraposition des Lemmas verwendet: A unentscheidbar => B unentscheidbar.

Def.: Das (allgemeine) Halteproblem ist die Sprache

$$H = \{w\#x \mid w, x \in \{0,1\}^*, M_w \text{ hält bei Eingabe } x\}$$

Satz: Das Halteproblem ist nicht entscheidbar.
Beweis: $K \leq H$, denn für $f(w) = w\#w$ gilt: f ist total, offensichtlich berechenbar, und außerdem $w \in K \Leftrightarrow f(w) \in H$.

Def.: Das Halteproblem bei leerem Band ist die Sprache

$$H_0 = \{w \in \{0,1\}^* \mid M_w \text{ hält bei leerer Eingabe}\}$$

Satz: Das Halteproblem bei leerem Band ist nicht entscheidbar.
Beweis: wir zeigen $H \leq H_0$. Jedem Wort w#x kann man eine Turingmaschine $M_{w\#x}$ zuordnen, die gestartet mit leerem Band zuerst x auf das Band schreibt, danach M_w ausführt. Man kann zeigen, dass die Funktion f, die w#x den Code von $M_{w\#x}$ zuordnet, total und berechenbar ist. Es gilt $w\#x \in H$ gdw. $f(w\#x) \in H_0$, d.h. wir haben H auf H_0 reduziert und H_0 ist unentscheidbar, da H unentscheidbar ist.

Der folgende wichtige Satz geht auf Henry Gordon Rice zurück. Er besagt, dass es keinen Algorithmus geben kann, der bei Eingabe (des Codes einer) Turingmaschine entscheidet, ob diese eine Funktion berechnet, die zu einer vorgegebenen, nicht-trivialen Klasse von berechenbaren Funktionen gehört (eine nicht-triviale Klasse enthält mindestens eine berechenbare Funktion, aber nicht alle).
Beispiele: konstante Funktionen; totale Funktionen; Funktionen, die bei bestimmter Eingabe k terminieren; Funktionen, bei denen für mindestens eine Eingabe den Wert k produziert; usw.

Satz (Rice, 1953): Sei R die Klasse aller Turing-berechenbaren Funktionen, $S \neq \emptyset$ echte Teilmenge von R. Dann ist

$$C(S) = \{w \in \{0,1\}^* \mid \text{die von } M_w \text{ berechnete Funktion liegt in } S\}$$

unentscheidbar.

Beweisskizze: falls die überall undefinierte Funktion in S liegt, kann man $\overline{H_0}$ auf C(S) reduzieren, falls sie nicht in S liegt H_0 auf C(S). (Bemerkung: da H_0 unentscheidbar, muss auch das Komplement unentscheidbar sein.)

Anmerkung 1: wenn $A \leq B$ und $B \leq A$, so sind A und B "gleich schwer". Es gibt eine unendliche Folge von immer schwereren Problemen.

Anmerkung 2: Man kann eine universelle Turingmaschine U konstruieren, die bei Eingabe von w#x die Berechnung von M_w bei Eingabe x durchführt.

Anmerkung 3: H ist rekursiv aufzählbar. Aus der universellen Turingmaschine U lässt sich folgendermaßen ein Aufzählungsverfahren konstruieren:
Sei w'#x' ein beliebiges, festes Wort so dass $M_{w'}$ bei Eingabe x' hält. Definiere

$$f(n) = \begin{cases} w\#x, & \text{falls } w = \text{bin}(d_0(n)),\ x = \text{bin}(d_1(n)) \text{ und} \\ & U \text{ hält bei Eingabe } w\#x \text{ nach } d_2(n) \text{ Schritten} \end{cases}$$

w'#x' sonst.

d_i sind hier die Decodierfunktionen, d.h. n wird als Codierung von 3 Zahlen aufgefasst, die erste ist der Code einer TM, die zweite die Eingabe, die dritte eine vorgegebene Schrittzahl.

Anmerkung 4: Aus der rekursiven Aufzählbarkeit von H folgt zusammen mit der Unentscheidbarkeit sofort, dass das Komplement von H nicht rek. aufzählbar sein kann.

3. Das Postsche Korrespondenzproblem (PCP)

Emil Leon Post, polnisch-amerikanischer Mathematiker und Logiker, gest. 1954

gegeben: endliche Folge von Wortpaaren $(x_1, y_1), \ldots, (x_k, y_k)$
alle Wörter aus Σ^+ für ein Alphabet Σ.
gefragt: gibt es Folge von Indizes $i_1, i_2, \ldots, i_n$, so dass
$x_{i1}\ldots x_{in} = y_{i1}\ldots y_{in}$

Beispiel: K = ((10,1), (110, 0110), (01, 001))
x_1 y_1 x_2 y_2 x_3 y_3

Lösung: (1,2,1,3): 10 110 10 01
1 0110 1 001

Nicht immer so einfach:

Beispiel: K = ((001,0), (01,011), (01,101), (10,001))

Kürzeste Lösung Folge von 66 Indizes!

PCP semi-entscheidbar: man kann alle möglichen Indexfolgen der Reihe nach erzeugen und testen (alle Indexfolgen der Länge1, 2, 3 usw.)
Aber es gilt der folgende Satz:

Satz: Das Postsche Korrespondenzproblem (PCP) ist unentscheidbar.
Beweis: siehe etwa Schöning

Es ist sogar unentscheidbar, wenn man sich auf $\Sigma = \{0,1\}$ beschränkt.

Beweis durch Reduktion des Halteproblems auf das PCP.

Anmerkungen:
Da $H \leq PCP$ und PCP semi-entscheidbar, muss auch H semi-entscheidbar sein. Das heißt, es gibt eine Turingmaschine, die bei Eingabe von w#x hält genau dann wenn M_w bei Eingabe x hält.

4. Unentscheidbare Grammatikprobleme

Satz: Seien G_1 und G_2 kontextfreie Grammatiken. Folgende Probleme sind unentscheidbar:

1. $L(G_1) \cap L(G_2) = \varnothing$?
2. $|L(G_1) \cap L(G_2)| = \infty$?
3. $L(G_1) \cap L(G_2)$ kontextfrei?
4. $L(G_1) \subseteq L(G_2)$?
5. $L(G_1) = L(G_2)$?

Beweisskizze:
(1) Wir reduzieren das PCP auf das Schnittproblem:
Sei $K = ((x_1, y_1), ..., (x_k, y_k))$ eine Instanz des Korrespondenzproblems über {0,1}.

Wir definieren zu K die beiden Grammatiken G_{K1} und G_{K2} (Terminalalphabet {0,1,$, a_1, ..., a_k}), so dass K eine Lösung hat gdw. der Schnitt der von G_{K1} und G_{k2} erzeugten Sprachen nicht leer ist:

G_{K1}: S -> A$B
A -> a_1Ax_1 | … | a_kAx_k
A -> a_1x_1 | … | a_kx_k
B -> $\tilde{y}_1Ba_1$ | …| $\tilde{y}_kBa_k$
B -> $\tilde{y}_1a_1$ | …| $\tilde{y}_ka_k$

$w^{\sim}$ ist die Umkehrung von w.
erzeugte Sprache: $L_1 = \{a_{in}...a_{i1}x_{i1}...x_{in}\$\ \tilde{y}_{jm}...\ \tilde{y}_{j1}a_{j1}...a_{jm} \mid n, m \geq 1\}$

also: beliebige umgedrehte Folge von Indizes, Folge von entsprechenden x_is, $, beliebige umgedrehte Folge von gespiegelten y_js, entsprechende Folge von Indizes.

G_{K2}: S -> a_1Sa_1 | … | a_kSa_k | T
T -> 0T0 | 1T1 | $

erzeugte Sprache: $L_2 = \{uv\$v^{\sim}u^{\sim} \mid u \in \{a_1, ..., a_k\}^*, v \in \{0,1\}^*\}$

Ein Wort $a_{in}...a_{i1}x_{i1}...x_{in}\$\ \tilde{y}_{jm}...\ \tilde{y}_{j1}a_{j1}...a_{jm}$ wird aus <u>beiden</u> Grammatiken erzeugt (liegt also im Schnitt der erzeugten Sprachen), genau dann wenn gilt:

1. das Wort $x_{i1}...x_{in}$ wird durch die Indizes $i_1, ...,i_n$ erzeugt (wegen G_{K1}),
2. das Wort $y_{j1}...y_{jm}$ wird durch die Indizes $j_1, ...,j_m$ erzeugt (wegen G_{K1}),
3. $x_{i1}...x_{in} = y_{j1}...y_{jm}$ (wegen G_{K2}) und
4. $a_{i1}...a_{in} = a_{j1}...a_{jm}$ (wegen G_{K2}).

Damit besitzt K die Lösung $i_1 ... i_n$ genau dann wenn $a_{in}...a_{i1}x_{i1}...x_{in}\$\ \tilde{y}_{in}...\ \tilde{y}_{i1}a_{i1}...a_{in}$ im Schnitt der von G_{K1} und G_{K2} erzeugten Sprachen liegt.

Da die Funktion, die einer Instanz K des PCP die beiden Grammatiken G_{K1} und G_{K2} zuordnet, total und berechenbar ist, ist das PCP auf das Schnittproblem reduziert, das somit unentscheidbar ist.

Beispiel: K = ((10,1), (11, 011)) Lösung: 1,2

G_{K1}: S -> A$B
A -> a_1A10 | a_2A11
A -> $a_1$10 | $a_2$11
B -> 1Ba_1 | 110Ba_2
B -> 1a_1 | 110a_2

aus G_{K1} herleitbar: z. B. $a_1$10$1101$a_1a_2$ oder $a_2a_1$1011$110$a_2$, aber auch:

$$a_2a_1\text{1011\$1101}a_1a_2$$

dieses Wort ist auch aus G_{K2} herleitbar. Das bedeutet, dass 1,2 eine Lösung von K ist.

(2) Wenn K eine Lösung hat, dann auch unendlich viele (Indexfolge kann beliebig oft wiederholt werden, im obigen Beispiel: auch 1,2,1,2 und 1,2,1,2,1,2 etc. Lösungen). Also hat K Lösung gdw. mit obigen Grammatiken $|L(G_{K1}) \cap L(G_{K2})| = \infty$. Damit ist K auch auf Problem 2. reduziert.

(3) Falls K keine Lösung besitzt, ist $L_1 \cap L_2$ leer und damit kontextfrei. Falls K Lösungen besitzt, ist $L_1 \cap L_2$ unendlich, und man kann mit dem Pumping Lemma zeigen, dass der Schnitt nicht kontextfrei ist. Wäre also Kontextfreiheit des Schnitts entscheidbar, so auch das PCP.

(4) (5) Man kann zeigen: L_1 und L_2 sind sogar deterministisch kontextfrei. Damit ist das Komplement kontextfrei und es gibt effektiv konstruierbare Grammatiken G'_{K1} und G'_{K2} für die jeweils komplementären Sprachen. Es gilt:

$$\begin{aligned} L_1 \cap L_2 \text{ leer} &\Leftrightarrow L_1 \subseteq L(G'_{K2}) \\ &\Leftrightarrow L_1 \cup L(G'_{K2}) = L(G'_{K2}) \\ &\Leftrightarrow L(G_3) = L(G'_{K2}) \end{aligned}$$

G_3 ist hier die Grammatik, die die Vereinigung $L_1 \cup L(G'_{K2})$ erzeugt. Wären Inklusion oder Gleichheit kontextfreier Sprachen entscheidbar, so könnte man demnach Leerheit des Schnitts entscheiden, im Widerspruch zu 1.

Anmerkung: hieraus folgt sofort die Unentscheidbarkeit des Äquivalenzproblems für alle Formalismen, in die kontextfreie Grammatiken übersetzbar sind: Kellerautomaten, LBAs, kontextsensitive Grammatiken, Turingmaschinen, While-, Goto-Programme.

Wir nennen die kontextfreie Grammatik G deterministisch kontextfrei, wenn L(G) deterministisch kontextfrei ist (L(G) wird von det. Kellerautomat akzeptiert). Mit obigem Beweis ist implizit auch bewiesen:

Satz: Seien G_1 und G_2 deterministisch kontextfreie Grammatiken. Folgende Probleme sind unentscheidbar:

1. $L(G_1) \cap L(G_2) = \emptyset$?
2. $|L(G_1) \cap L(G_2)| = \infty$?
3. $L(G_1) \cap L(G_2)$ kontextfrei?
4. $L(G_1) \subseteq L(G_2)$?

Äquivalenz deterministisch kontextfreier Sprachen dagegen ist entscheidbar (Senizergues 1997)! Der obige Unentscheidbarkeits-Beweis schlägt fehl, da die Vereinigung deterministisch kontextfreier Sprachen nicht notwendigerweise det. kontextfrei ist.

Weitere unentscheidbare Probleme kontextfreier Grammatiken (ohne Beweis):

Satz: Sei G kontextfreie Grammatik. Folgende Probleme sind unentscheidbar:

1. Ist G mehrdeutig?
2. Ist das Komplement der von G erzeugten Sprache kontextfrei?
3. Ist L(G) regulär?
4. Ist L(G) deterministisch kontextfrei?

Satz: Das Leerheitsproblem und das Endlichkeitsproblem für Typ 1 Sprachen sind nicht entscheidbar.

5. Der Gödelsche Unvollständigkeitssatz

Gödel zur Zeit seiner Dissertation

Gödel mit Einstein

Kurt Gödel (*1906 in Brünn, gehörte dem berühmten Wiener Kreis an, †1978 in Princeton), Mathematiker und Logiker, oft als der bedeutendste Logiker des 20. Jahrhunderts angesehen.

Beschäftigte sich mit Grenzen der Beweisbarkeit. Hauptresultat: jedes konsistente Beweissystem für die Arithmetik ist unvollständig: es gibt wahre Sätze der Arithmetik, die man in dem System nicht beweisen kann.

Erhebliche Bedeutung für die Grundlagen der Mathematik/Informatik. Widerlegt das Hilbertsche Programm (David Hilbert, 1862-1943, Göttingen), durch das die gesamte Mathematik auf die Arithmetik zurückgeführt und auf ein axiomatisches System gestellt werden sollte, aus dem alle mathematischen Sätze streng ableitbar sind.

a) Arithmetische Formeln

sind aufgebaut wie prädikatenlogische Formeln mit Gleichheit (= ist das einzige verwendete Prädikatensymbol), mit der Einschränkung, dass Terme induktiv wie folgt definiert sind:

1. jede natürliche Zahl n (repräsentiert durch Symbole 0, 1, 2, …) ist ein Term,
2. jede Variable ist ein Term,
3. wenn t_1 und t_2 Terme sind, so sind $(t_1 + t_2)$ und $(t_1 * t_2)$ Terme.

Die Symbole + und * haben eine feste Interpretation, nämlich Addition und Multiplikation auf den natürlichen Zahlen. Freie und gebundene Variablen sind wie üblich definiert (die letzteren liegen im Bereich eines Quantors). Eine Belegung ϕ ordnet jeder Variablen eine natürliche Zahl als Wert zu. Falls t Term ist, dann ist $\phi(t)$ die Zahl, die sich ergibt, wenn jede Variable x in t durch $\phi(x)$ ersetzt und dann „ausgerechnet“ wird.

Def.: Eine arithmetische Formel F' ist in WA, der Menge der wahren arithmetischen Formeln, gdw. F' geschlossen ist (keine freien Variablen) und eine der folgenden Bedingungen gilt:

1. $F' = (t_1 = t_2)$ und t_1 wird zu derselben natürlichen Zahl ausgewertet wie t_2.
2. $F' = \neg F$ und F ist nicht wahr.
3. $F' = (F \wedge G)$ und F wahr und G wahr.
4. $F' = (F \vee G)$ und F wahr oder G wahr.
5. $F' = \exists x\, F$ und es gibt ein n, so dass F(x/n) wahr (Substitution von x durch n in F).
6. $F' = \forall x\, F$ und für alle n ist F(x/n) wahr.

Beispiele:
$2 + 3 = 5$, $\forall x\, \forall y\, (x + y) = (y + x)$, $\forall x\, \exists y\, \exists z\, (x = (y + z))$ sind wahre arithmetische Formeln.

Hinweis: für jede geschlossene arithmetische Formel F gilt: entweder $F \in WA$ oder $\neg F \in WA$.

Def.: Die Funktion $f: \mathbb{N}^k \to \mathbb{N}$ ist arithmetisch repräsentierbar, falls es eine arithmetische Formel $F(x_1, ..., x_k, y)$ gibt, so dass $f(n_1, ..., n_k) = m$ gdw. $F(n_1, ..., n_k, m)$ wahr ist.

Beispiel: Sei $(a < b)$ Abkürzung für: $\exists\, z\, (a + z + 1 = b)$

DIV: $\exists\, r\, ((r < x_2) \wedge (x_1 = y * x_2 + r))$
MOD: $\exists\, k\, ((y < x_2) \wedge (x_1 = k * x_2 + y))$

Satz: Jede WHILE-berechenbare Funktion ist arithmetisch repräsentierbar.

Satz: Die Menge WA der wahren arithmetischen Formeln ist nicht rekursiv aufzählbar.

Beweis: Für jede geschlossene Formel F gilt: entweder F ist wahr oder $\neg F$ ist wahr. Wäre die Menge WA rekursiv aufzählbar, so wäre WA auch entscheidbar: bei Eingabe F zähle WA auf, $WA = (F_0, F_1, ...)$, bis entweder F oder $\neg F$ in der Aufzählung vorkommt. Im ersten Fall gib 1 aus, im zweiten 0.

Wir zeigen: WA ist nicht entscheidbar, und damit nicht rekursiv aufzählbar.

Sei A rekursiv aufzählbare, aber nicht entscheidbare Sprache (etwa H, PCP, ...). Die halbe charakteristische Funktion von A ist WHILE-berechenbar und damit arithmetisch repräsentierbar durch eine Formel F(x,y). Es gilt:

$$\begin{aligned} n \in A &\Leftrightarrow \chi'_A(n) = 1 \\ &\Leftrightarrow F(n,1) \text{ ist wahr} \\ &\Leftrightarrow F(n,1) \in WA \end{aligned}$$

Da die Funktion, die jedem n die Formel F(n,1) zuordnet, total und berechenbar ist, wird dadurch A auf WA reduziert. Da A nach Voraussetzung nicht entscheidbar ist, kann es auch WA nicht sein. Damit ist WA auch nicht rekursiv aufzählbar.

Hinweis: wir sind im Beweis davon ausgegangen, dass A schon in codierter Form als Menge natürlicher Zahlen vorliegt. Dass so etwas immer möglich ist, haben wir am Anfang von Teil II diskutiert.

(PS: der Beweis funktioniert nicht für allgemeine Prädikatenlogik, da dort nicht gilt: F allgemeingültig oder ¬F allgemeingültig.)

b) Beweissysteme

Ein Beweissystem (Kalkül) besteht üblicher Weise aus einer Formelsprache, einer Menge von Axiomen aus dieser Sprache und einer Menge von Ableitungsregeln (etwa modus ponens: aus p und $p \to q$ leite q ab). Ein formaler Beweis ist dann eine Folge von Formeln, wobei jede Formel in der Folge entweder ein Axiom ist (ohne Herleitung gültig) oder sich aus vorhergehenden Formeln durch Anwendung einer Regel erzeugen lässt. Die bewiesene Formel ist dann die letzte in der Folge.

Ein Beweissystem ist immer ein System *für* eine bestimmte Teilmenge A von Formeln (Beispiel: Beweissystem für die aussagenlogischen Tautologien). Es heißt *korrekt*, wenn es nur Beweise für Formeln in A gibt, *vollständig*, wenn alle Formeln in A bewiesen werden können.

Beispiel: Kalkül für aussagenlogische Tautologien (Lukasiewicz). Nur $\neg$ und $\to$ verwendet, $p \vee q$ kann durch $\neg p \to q$ ausgedrückt werden, $p \wedge q$ durch $\neg(p \to \neg q)$.
Axiome: alle Instanzen der folgenden Ausdrücke
(für p, q und r können beliebige aussagenlogische Formeln substituiert werden):

$p \to (q \to p)$
$(p \to (q \to r)) \to ((p \to q)\ (p \to r))$
$(\neg p \to \neg q) \to (q \to p)$

Inferenzregel: Modus ponens. Korrekt und vollständig für aussagenlogische Tautologien.

Wir abstrahieren hier so weit wie möglich von konkreten Kalkülen. Wir benötigen hier nur 2 minimale Anforderungen an Beweissysteme, nämlich dass man entscheiden kann, ob etwas tatsächlich ein Beweis ist, und dass man für jeden Beweis berechnen kann, was er beweist:

Def.: Ein Beweissystem für $A \subseteq \Gamma^*$ ist ein Paar (B, F), wobei gilt:

1. $B \subseteq \Sigma^*$ ist eine entscheidbare Menge, die Menge der Beweise.

2. F: B -> Γ* ist eine totale berechenbare Funktion, die jedem Beweis das durch ihn bewiesene Element aus Γ* zuordnet.

(B, F) heißt korrekt (für A), wenn für alle $b \in B$ gilt: $F(b) \in A$.
vollständig (für A), wenn für $a \in A$ ein $b \in B$ existiert mit $F(b) = a$.

Satz (Gödelscher Unvollständigkeitssatz):
Jedes Beweissystem für WA ist inkorrekt oder unvollständig.

Beweis: Angenommen (B, F) wäre vollständig und korrekt für WA. Dann könnte man WA aufzählen, indem man alle Beweise $b \in B$ (die wegen der Entscheidbarkeit von B auch aufzählbar sein müssen) der Reihe nach generiert und F(b) ausgibt. WA ist aber, wie bereits bewiesen, nicht rekursiv aufzählbar.

Äquivalente Formulierungen:
Jedes vollständige Beweissystem für WA ist inkorrekt
Jedes vollständige Beweissystem für WA ist inkonsistent (da vollständig und inkorrekt impliziert inkonsistent)
Jedes konsistente Beweissystem für WA ist unvollständig.

Das heißt also: nicht alles, was wahr ist, lässt sich in einem formalen System auch beweisen!!

Anhang: Gödel'scher Unvollständigkeitssatz - Skizze des Originalbeweises

Eine Zahl kann Verschiedenes repräsentieren (wenn $n \in \mathbb{N}$ ein $o \notin \mathbb{N}$ repräsentiert, so nennt man n auch Gödelnummer von o):

- die Zahl selbst (25)
- eine arithmetische Aussage (5 * 5 = 25)
- einen Beweis für eine arithmetische Aussage (Folge von Aussagen, abhängig von Beweissystem)
- Aussagen über Beweise (z.B. Beweis mit Codierung (= Gödelnummer) x beweist Aussage mit Gödelnummer y), usw.

Gödels Trick:

Konstruiere einen Satz, der besagt, dass er selbst nicht beweisbar ist.

Vorbemerkungen

- es ist entscheidbar, ob A ein Beweis für B ist (d.h. die char. Funktion $f: \mathbb{N} \times \mathbb{N} \rightarrow \{0,1\}$ mit f(a,b) = 1 gdw. a Gödelnummer von Beweis für Formel mit Gödelnummer b ist, 0 sonst, ist berechenbar).
- damit ist die Aussage „B ist (Gödelnummer eines) Beweis(es) für A" arithmetisierbar, d.h. wir können das Prädikat beweist(a,b) repräsentieren, wobei a Gödelnummer eines Beweises und b Gödelnummer der durch ihn bewiesenen Aussage ist.
- es ist entscheidbar, ob durch Substitution der freien Variablen in der Formel mit Gödelnummer x durch y die Formel mit Gödelnummer z entsteht (decodieren von x, Ersetzen der freien Variablen durch y, codieren, Vergleich mit z): das entsprechende Prädikat sub(x,y,z) ist damit arithmetisch repräsentierbar.
- selbstsub(z,y) ist Abkürzung für sub(z,z,y): Substitution der eigenen Gödelnummer für die freie Variable in der Formel mit Gödelnummer z ergibt die Formel mit der Gödelnummer y.

Betrachte nun die Aussage:

$$(U) \quad \neg\exists x,y\ [\text{beweist}(x,y) \wedge \text{selbstsub}(z,y)]$$

Sei g die Gödelnummer dieses Satzes. Wenn man in (U) für z g einsetzt, erhält man die Selbstsubstitution von g.

$$(G) \quad \neg\exists x,y\ [\text{beweist}(x,y) \wedge \text{selbstsub}(g,y)]$$

Der Satz besagt: es gibt keinen Beweis für y, wobei y die Selbstsubstitution von g ist.

Es gilt aber: (G) ist selber die Selbstsubstitution von g.
Also sagt (G): (G) ist nicht beweisbar.
oder: Ich bin nicht beweisbar.

Nun gilt: Entweder (G) ist wahr: dann ist dieser Satz (G) nicht beweisbar,
=> also gibt es einen wahren Satz, der nicht beweisbar ist.
Oder (G) ist falsch: dann ist der Satz beweisbar,
=> also gibt es einen falschen Satz, der beweisbar ist.

Ausführlicher etwa in: D. Hofstätter, Gödel, Escher, Bach: An Eternal Golden Braid, 1979