Berechenbarkeit und die Diagonalisierungsmethode

Die Diagonalisierungsmethode.

Barberparadoxon (unechnische Illustration).

Verordnung des Bürgermeisters von Russelsdorf:

Alle Männer müssen sich entweder selbst rasieren oder von Herrn Meyer rasiert werden.

Jeder Mann, der sich selbst rasiert, darf nicht von Herrn Meyer rasiert werden.

Jeder Mann, der sich von Herrn Meyer rasieren lässt, darf sich nicht selbst rasieren

Algorithmen kann man durch ihre Quellcodes angeben.
Quellcodes sind Texte ≃ files ≃ Strings ≃ Bitstrings.

HALTEPROBLEM: Man entscheide für einen gegebenen Algorithmus A und eine Eingabe x, ob A auf x terminiert.
(A als Quellcode gegeben).

Bsp. easy_code.cpp

```
string x;
cin >> x;
if (x == "stopp")
    return 0;
while (true) {
    cout << "Ich bin noch nicht fertig";}
```

easy_code_cpp = " string x; ..... "

x = " stopp "

HALTEPROBLEM ⟶ Rückgabe: true

(Easy Code terminiert auf "stopp")

easy_code_cpp = " string x; ..... "

x = "Hallo hallo!"

HALTEPROBLEM ⟶ Rückgabe: false

(der Code terminiert nicht auf "Hallo hallo".

Def. Ein Entscheidungsproblem heißt (algorithmisch) unentscheidbar, wenn es keinen Algorithmus existiert, der dieses Problem entscheidet.

Theorem. HALTEPROBLEM ist nicht entscheidbar.

Beweis: Angenommen, das HALTEPROBLEM wäre entscheidbar. Dann gäbe es einen Algorithmus, der das HALTEPROBLEM löst. Wir betrachten einen Quellcode davon:

```
bool haelt ( string Algo, string Eing )
   /* Rückgabe true, wenn der code in Algo
      auf Eing terminiert. Rückgabe false
      sonst
   */
{  // hypothetische Umsetzung
}
```

Wir erstellen darauf basierend den Code:

Barbier.cpp:

```
bool haelt (String Algo, String x){
    // hier die Umsetzung
}
string Algo;
cin >> Algo; // Quellcode aus der Standard-
                Eingabe, als String aufgefangen

if (haelt(Algo, Algo))
        while (true) {
            cout << "Ich bin noch nicht fertig!" << endl;
        }
else {
    cout << "Ich bin schon fertig!" << endl; }
```

Barbier.cpp —Compiler→ Barbier
(ausführbare Datei)

cat Barbier.cpp	Barbier

Barbier bekommt den Inhalt von Barbier.cpp als Eingabe.

Terminiert Barbier auf dem Inhalt von Barbier.cpp ?

Barbier_cpp = " Inhalt von Barbier.cpp "

Barbier terminiert auf dem Inhalt von

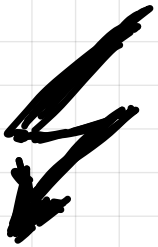
Barbier.cpp $\Longleftrightarrow$

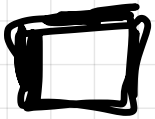
halte(Barbier_cpp, Barbier_cpp) = falsch

$\Longleftrightarrow$ (Nach unserer Voraussetzung an die Programmfunktion halte)

Das Algorithmus mit dem Quellcode aus Barbier_cpp auf der Eingabe Barbier_cpp terminiert nicht

$\Longleftrightarrow$ Barbier terminiert nicht auf dem Inhalt von Barbier.cpp





Bemerkung. Diagonalisierung ist damit auch nützlich in der theoretischen Informatik.

Zusammenhang mit mathematischen Grundlagen:

Sei THM das Entscheidungsproblem, bei dem man für eine gegebene mathematische Aussage A (als String bzw. Bitstring gegeben) entscheiden soll, ob das eine wahre oder eine falsche Aussage ist.

Theorem. THM ist unentscheidbar.

Beweisidee: Angenommen, man hätte einen Algorithmus THM_SOLVER:

THM_SOLVER(Aussage) ⟹ Nicht (Aussage richtig)
(als String) ⟹ Falsch (Aussage falsch).

Sei Algo Quellcode eines Algorithmus (beliebig)
Sei x beliebige Eingabe (String).

↳ Aussage = "Der Algorithmus mit dem Quellcode Algo terminiert nach der Eingabe x".

Dann sagt uns TM_SOLVE(Aussage),
ob Algo auf x terminiert.

Auf diese Weise haben wir das HALTEPROBLEM
entschieden. $\lightning$ zu dem vorigen Theorem über
Unentscheidbarkeit des HALTEPROBLEMs. □

Bemerkung: Basiert paradoxon zeigt auch, dass
die naive Menge scheitert (im Allgemeinen!

$$y := \{ x : x \notin x \}$$

Naive Beschreibung
eines "dings" y.

$$y \in y \iff y \notin y.$$

Das zeigt, dass so ein
y nicht existieren kann.

Aktueller Stand:

10 Axiome (ZFC Mengenlehre

ZF = Zermelo-Fraenkel

C = Auswahlaxiom).

"10 Spielregeln für Mengen".

Axiom 3 als Beispiel. Paarmengenaxiom

Für alle A und B gibt es eine Menge C,
die genau A und B als Elemente hat.

Axiom 5 als ein weiteres Beispiel: Unendlichkeitsaxiom.

$X \rightsquigarrow$ Rückgabemenge X.

$\{\emptyset,$

$\{\emptyset\},$

$\{\emptyset, \{\emptyset\}\},$

$\{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\},$

$\{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}, \{\emptyset, \emptyset\}, \{\emptyset, \{\emptyset\}\}\},$

$\vdots$

$\}$

Eine weitere Anwendung der Diagonalisierung:

$[0,1] := \{x \in \mathbb{R} : 0 \leq x \leq 1\}$

$\mathbb{N}$ die Menge der natürlichen Zahlen.

Thm. Es gibt keine surjektive Abbildung

$a : \mathbb{N} \to [0,1]$. Mit anderen Worten:

$[0,1]$ ist nicht abzählbar.

Beweis: Wir nehmen an, es gäbe so ein a.

$0 = 0,000000000\ldots\ldots$

$1 = 0,999999999\ldots\ldots$

$\frac{1}{2} = 0,500000000$

$\frac{1}{3} = 0,333333333$

$a(1) = 0,\mathbf{3}3333333333\ldots$

$a(2) = 0,1\mathbf{1}111111111\ldots$

$a(3) = 0,314\mathbf{1}59265\ldots$

$a(4) = 0,9718\mathbf{8}2818285\ldots$

$a(5) = 0,34833\mathbf{5}68345$

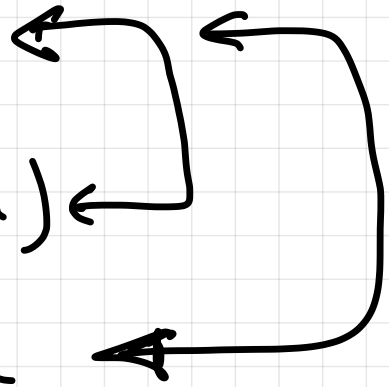
$\vdots$

$x = 0,78306\ldots\ldots\ldots$

$x \neq a(i)$, weil sie die i-te Nachkommastelle von x anders als die i-te Nachkommastelle von $a(i)$ fixieren.

$x \notin a(\mathbb{N}) \Rightarrow a$ nicht surjektiv. $\square$

1. Mathe Grundlagen (+Komplexe Zahlen)
2. Kombinatorik
3. Algo / Programmieren
4. Graphen (mit Algorithmen)
5. Boolesche Funktionen



(1) $A = B$: Axiom 1 legt fest, wann man zwei Mengen auf Gleichheit prüft.

(2) $A = \emptyset$: Axiom 2 legt fest, dass $\emptyset$ existiert

(3) $A = \{x, y\}$ Axiom 3 legt fest, dass für x, y die Menge $\{x, y\}$ existiert.

(4) $A = \bigcup \mathcal{F}$: Axiom 4 ermöglicht die Konstruktion einer Vereinigung

(5) Axiom 5 beschreibt, dass die folgende Menge existiert:

$$\{\emptyset, \emptyset', \emptyset'', \emptyset''', \emptyset'''', \emptyset''''', \dots\}$$

mit $X' := X \cup \{X\}$

(6) Axiom 6 ermöglicht die Konstruktion der Potenzmenge

$$2^X = \{y : y \subseteq X\}$$

7 Fundierungsaxiom: Verhindert die Möglichkeit von unendlichen Folgen

$$x_1 \ni x_2 \ni x_3 \ni x_3 \ni \dots \dots$$

z. B. man kann nicht $A \ni B \ni A \ni B \ni A \ni \dots$ haben.

8 Axiom 8 ermöglicht das Durchstreichen einer Menge durch eine Bedingung.

$$\{x \in A : P(x)\}$$

9

$$\{y : x \in A \wedge P(x,y)\}$$

10 Auswahlaxiom: ist $\mathcal{F}$ so, dass für alle $A \in \mathcal{F}$, die Bed. $A \neq \emptyset$ gilt, so gibt es

eine Abbildung $f : \mathcal{F} \to \bigcup \mathcal{F}$ mit $f(A) \in A$ für alle $A \in \mathcal{F}$.

In jeder Familie von nichtleeren Mengen
kann man in jeder Menge der Familie ein Element
auswählen (dieses Axiom wird entscheidend
in der Analysis benutzt).