Verständnis Berechenbarkeitstheorie

Beantworten Sie kurz, präzise und mit Begründung folgende Fragen: (Die Begründungen müssen keine formellen mathematischen Beweise sein)

(a) Warum genügt es, sich auf Funktionen zu beschränken, die natürliche Zahlen auf natürliche Zahlen abbilden, wenn man untersuchen will, was heutige Computer im Prinzip berechnen können?

> Jede WHILE-berechenbare Funktion ist turing-berechenbar. Jede turing-berechenbare Funktion ist WHILE-berechenbar. Jede nichtdeterministische Turing-Maschine kann durch eine deterministische Turing-Maschine simuliert werden.

(b) Was besagt die Church-Turing-These? Könnte man sie beweisen oder widerlegen?

> Die Church-Turing-These besagt, dass die Klasse der turing-berechenbaren Funktionen mit der Klasse der intuitiv berechenbaren Funktionen übereinstimmt. Die These kann nicht bewiesen oder widerlegt werden, weil es sich bei dem Begriff *„intuitiv berechenbare Funktion"* um keinen mathematisch exakt definitierten Begriff handelt. Würde man ihn genau definierten, würde ein konkretes Berechnungsmodell festlegt werden, was der Kernaussage dieses Begriffes widersprechen würde.

(c) Für reelle Zahlen, wie z. B. π, lässt sich die Dezimaldarstellung durch entsprechende Programme beliebig genau approximieren. Gilt das für alle reellen Zahlen, d. h. lässt sich für jede reelle Zahl die Dezimaldarstellung mit entsprechenden Programmen beliebig genau approximieren?

> Ja mit einer Berechnungsvorschrift, ja solange der speicherplatz reicht, z. B. mittels Intervallschachtelung.

(d) Was ist für die Berechnungskraft der wesentliche Unterschied zwischen While-Berechenbarkeit und Loop-Berechenbarkeit.

> Alle LOOP-Programme sind mathematisch betrachtet totale Funktionen, d. h. sie terminieren immer. WHILE-Programme hingegen partiellen Funktionen, die nicht für alle Eingabekombinationen terminieren.

(e) Die Ackermannfunktion ist ein Beispiel einer totalen Funktion, die While-berechenbar, aber nicht Loop-berechenbar ist. Sie verallgemeinert die Idee, dass Multiplikation die wiederholte Addition ist, Exponentiation die wiederholte Multiplikation, Hyperexponentiation die wiederholte Exponentiation usw. Die Stufe dieser hyper-hyper ... Exponentiation ist ein Parameter der Ackermannfunktion. Generieren Sie aus dieser Idee ein Argument, das illustriert, warum die Ackermannfunktion nicht Loop-berechenbar ist.

Jedes LOOP-Programm benötigt zur Berechnung der Funktion $x \uparrow^n y$ mindestens $n + 2$ Schleifen.

Gäbe es ein LOOP-Programm, das $ack(n, m)$ tatsächlich für beliebige Werte von n berechnet, so müsste dieses — im Widerspruch zum endlichen Aufbau eines Loop-Programms — unendlich viele Schleifenkonstrukte enthalten.

Hyperexponentiation

- $x \uparrow^{-1} y = x + y$
- $x \uparrow^0 y = x \cdot y$
- $x \uparrow^1 y = x^y$
- $x \uparrow^2 y = x^{y^y}$

(f) Geben Sie ein Beispiel einer Menge an, die abzählbar, aber nicht rekursiv aufzählbar ist, und begründen Sie es.

- Die Menge der Primzahlen. Die Primzahl sind unendlich groß und könnten abgezählt werden. Bei Primzahlen gibt es keine Berechnungsvorschrift, die z. b. die 7. Primzahl berechnet.
- Die Menge der Turningmaschine
- Diagonalsprache
- Das Komplement des Halteproblems. Die dem Halteproblem zugrundeliegende Lösungsmenge ist rekursiv aufzählbar. Wäre das Komplement des Halteproblems auch rekursiv aufzählbar, dann wäre das Halteproblem entscheidbar, was es aber nicht ist. [a]

[a] https://www.youtube.com/watch?v=om4ZTOeQuD0

(g) Wie ist der Zusammenhang zwischen rekursiv aufzählbar und semi-entscheidbar?

Die beiden Begriffe sind äquivalent. [a]

[a] https://www.youtube.com/watch?v=om4ZTOeQuD0