Einführung in die Berechenbarkeitstheorie

Timo Kötzing

4. November 2013

1 Einleitung

Dieses Dokument bietet eine kurze Einführung in die Grundlagen der Berechenbarkeitstheorie (für eine Ausführliche Behandlung des Materials, siehe zum Beispiel [Rog67]). Dazu führen wir ein Berechnungsmodell ein, welches im wesentlichen formalisierter Pseudocode ist (die sogenannte WHILE-Sprache); dieses Modell definiert was „berechenbar“ bedeutet. Hieraus leiten wir wichtige Eigenschaften der berechenbaren Funktionen (insbesondere Abschlusseigenschaften) und Theoreme (Universalität, KRT, SMN) ab.

Die Einührung der Berechenbarkeit erfolgt in 5 Treffen, mit einem zusätzlichen Treffen für einen Test. folgende Auflistung gibt einen Überblick über Daten.

14.10.	Überblick und die WHILE-Sprache
16.10.	Beispiele: WHILE-Programme
21.10.	Codierung von Programmen, Universalität, s-m-n
23.10.	KRT, das Halteproblem
28.10.	Beispiele: Semantische Mengen, $\leq_1$
30.10.	Test

2 Die WHILE-Sprache

In diesem Abschnitt definieren wir ein Modell dafür, was "berechenbar" bedeutet. Im Wesentlichen ist dies eine "Programmiersprache" (wir verwenden die sogenannte WHILE-Sprache) und definieren Berechnungen in diesem Maschinenmodell; dann definieren wir, dass eine Funktion f *berechenbar* ist genau dann, wenn es ein Programm (in der WHILE-Sprache) gibt, welches f implementiert (also dasselbe Eingabe/Ausgabe-Verhalten aufweist). Equivalent könnte man zum Beispiel auch Turing Maschienen für ein solches Modell nehmen.

Wir benutzen die folgenden syntaktischen Komponenten für die WHILE-Sprache.

- Variablen: x_0 x_1 ...
- Konstanten: 0 1 ...
- Trennsymbole: ; :=
- Operatoren: + -
- Schlüsselwörter: WHILE DO END

Wir definieren die Menge aller WHILE-Programme rekursiv wie folgt. Jede Aussage der Form

$$\begin{aligned} x_i &:= x_j + c \\ x_i &:= x_j - c \end{aligned}$$

ist ein WHILE-Program ($c \in \mathbb{N}$ ist eine Konstante und x_i, x_j sind Variablen). Diese Aussagen repräsentieren Addition und Subtraktion. Wenn P_0 und P_1 WHILE-Programme sind, dann ist

$$P_0; P_1$$

auch ein WHILE-Programm ("erst P_0 ausführen, dann P_1"). Schließlich, falls P ein WHILE-Programm ist, dann ist auch

$$\text{WHILE } x_i \neq 0 \text{ DO } P \text{ END}$$

ein WHILE-Programm.

Die Semantik eines WHILE-Programms P ist wie folgt rekursiv definiert. Wenn eine k-stellige Funktion berechent werden soll, für Eingaben $n_1, \ldots, n_k \in \mathbb{N}$, so werden die Variablen $x_1, \ldots, x_k$ mit den Werten $n_1, \ldots, n_k$ initialisiert, alle anderen Variable mit 0. Falls P Addition oder Subtraktion ist, so wird diese Operation wie intuitiv erwartet ausgeführt, aber Werte < 0 werden zu 0 geändert. $P_0; P_1$ wird interpretiert indem erst P_0 und dann P_1 ausgeführt wird. Das WHILE-Programm $x_i \neq 0$ DO P END überprüft zuerst ob $x_i \neq 0$; falls dies zutrifft, so wird P ausgeführt und danach wird wieder $x_i \neq 0$ überprüft und so weiter bis $x_i = 0$. Der Ausgabewert eines WHILE-Programms auf n_1, $\ldots$, $n_k \in \mathbb{N}$ ist der Wert von x_0 nach der Ausführung von P. Eine k-stellige Funktion $f : \mathbb{N}^k \to \mathbb{N}$ heißt *berechenbar* falls es ein WHILE-Programm gibt welches f berechnet.

Als nächstes geben wir Beispiele für Makros (Kurzschreibweisen für Codeteile) und die Funktionen, die mithilfe dieser Makros berechnet werden können.

"$x := y$"

$x := y + 0$

Also ist $id : \mathbb{N} \to \mathbb{N}, a \mapsto a$ berechenbar mit dem Programm "$x_0 := x_1$".

"NULLIFY x"

WHILE $x \neq 0$ DO $x := x - 1$ END

Damit können wir $f_0 : \mathbb{N} \to \mathbb{N}, a \mapsto 0$ mit dem Programm "NULLIFY x_0" berechnen.

"$x := c$"

NULLIFY x;
$x := x + c$

Folglich, für jedes $c \in \mathbb{N}$, $f_c : \mathbb{N} \to \mathbb{N}, a \mapsto c$ ist berechenbar mit dem Programm "$x_0 := c$".

"LOOP x DO P END"

$t := x$;
WHILE $t \neq 0$ DO
$t := t - 1$;
P
END

"$x = y + z$"

$x := y$;
LOOP z DO
$x := x + 1$
END

Also ist $+ : \mathbb{N}^2 \to \mathbb{N}, (a, b) \mapsto a + b$ berechenbar mit dem Programm "$x_0 := x_1 + x_2$".

"IF $x = 0$ THEN P END"

$t := 1$;
LOOP x DO $t := 0$ END;
LOOP t DO P END

Aufgabe 1

Schreibe WHILE-Makros für die folgenden Ausdrücke.

(1) $x = y * z$
(2) $x = 2^y$
(3) $x = y^z$
(4) IF $x_i = 0$ THEN P_0 ELSE P_1 END

Aufgabe 2

Seien f, g, h berechenbar. Zeige, dass dann auch die folgenden Funktionen berechenbar sind.

(1) $f + g$ (genauer: $x \mapsto f(x) + g(x)$), $f - g$, $f \cdot g$, ...
(2) $f < g$ (genauer: $x \mapsto 1$, falls $f(x) < g(x)$ und 0 sonst), $f = g$, $f \wedge g$ (genauer: $x \mapsto 1$ falls $f(x) = 1$ und $g(x) = 1$, sonst 0), ...
(3) *Verkettung*: $f \circ g$.
(4) If-Then-Else; genauer: $x \mapsto \begin{cases} f(x), & \text{falls } h(x) > 0; \\ g(x), & \text{sonst.} \end{cases}$
(5) *Beschränkte Suche:* $x \mapsto \forall y < f(x) : g(x, y) > 0$, $x \mapsto \exists y < f(x) : g(x, y) > 0$.
(6) *Unbeschränkte Suche:* $x \mapsto \exists y : f(x, y) > 0$ (undefiniert, falls nicht existent).
(7) *Unbeschränkte Suche nach Minimum:* $x \mapsto \min_y f(x, y) > 0$ (undefiniert, falls nicht existent).

Aufgabe 3

(1) Welche Funktion wird durch
"WHILE $x_1 \neq 0$ DO $x_0 := x_0 + 0$ END" berechnet?
(2) Codierung von Paaren (Bijektion $\mathbb{N}^2 \to \mathbb{N}$): $\langle \cdot, \cdot \rangle : (a, b) \mapsto 2^a(2b - 1) - 1$. Zeige, dass diese Codierung berechenbar ist.
(3) Decodierungsfunktionen sind π_1, π_2 so dass, für alle x, $\langle \pi_1(x), \pi_2(x) \rangle = x$. Zeige, dass π_1 und π_2 berechenbar sind.
(4) *Induktion*: Seien f, g berechenbar. Zeige, dass

$$h : \mathbb{N}^2 \to \mathbb{N}, (a, b) \mapsto \begin{cases} f(b), & \text{falls } a = 0; \\ g(h(a - 1, b), a, b), & \text{sonst.} \end{cases}$$

berechenbar ist.

Für die folgende Aufgabe wollen wir die Abschlusseigenschaften der berechenbaren Funktionen (wie in den letzten Aufgaben gezeigt) nutzen.

Aufgabe 4

Zeige, dass die folgenden Funktionen berechenbar sind. Seien f, g berechenbare Funktionen.

(1) Fakultät: $x \mapsto x!$.

(2) Polynom: $x \mapsto 3x^3 + 2x^2 + x$.

(3) $x \mapsto \begin{cases} 1, & \text{falls } \forall y < f(x) + 15 : \exists z < 33 : f(y,z) > 109; \\ 0, & \text{sonst.} \end{cases}$

(4) $x \mapsto \begin{cases} 1, & \text{falls } \exists z : f(x,z) \cdot g(x,z) > 150; \\ \uparrow, & \text{sonst.} \end{cases}$

Aufgabe 5

Zeige die folgenden Aussagen.

(1) Für alle total berechenbaren Funktionen $f \in \mathcal{R}$ gibt es ein total berechenbares $g \in \mathcal{R}$, so dass für alle bis auf endlich viele x gilt

$$g(x) = \min(\text{range}(f)).$$

(2) Sei D eine endliche Menge. Zeige, dass

$$x \mapsto \begin{cases} 1, & \text{falls } x \in D; \\ 0, & \text{sonst;} \end{cases}$$

eine berechenbare Funktion.

(3) Sei D eine endliche Menge und f total berechenbar. Dann das

$$x \mapsto \begin{cases} 1, & \text{falls } \exists y \in D : f(x,y) = 1; \\ 0, & \text{sonst;} \end{cases}$$

eine berechenbare Funktion.

Wir definieren *Codierung von Tupeln* durch

$$\langle \cdot \rangle : \mathbb{N}^* \to \mathbb{N}, \vec{x} \mapsto \langle \text{len}(\vec{x}), \langle x_1, \langle x_2, \langle \ldots \langle x_{\text{len}(\vec{x})-1}, x_{\text{len}(\vec{x})} \rangle \ldots \rangle \rangle \rangle \rangle.$$

Es ist klar, dass decodierung der Länge und jedes Element des Tupels berechenbar ist. Dadurch können wir annehmen, dass alle Eingaben an eine Funktion immer als eine einzelne Zahl gegeben sind.

Wir definieren Codierung $\langle\cdot\rangle$ von Programmen P rekursiv wie folgt.

$$\begin{aligned} x_i := x_j + c &\mapsto \langle 0, i, j, c\rangle; \\ x_i := x_j - c &\mapsto \langle 1, i, j, c\rangle; \\ P_0; P_1 &\mapsto \langle 2, \langle P_0\rangle, \langle P_1\rangle\rangle; \\ \text{WHILE } x_i \neq 0 \text{ DO } P \text{ END} &\mapsto \langle 3, i, \langle P\rangle\rangle. \end{aligned}$$

Wir schreiben die *Funktion berechnet vom Programm mit Programm-Codezahl* p als

$$\varphi_p.$$

Falls p keine Programmzahl ist, so ist φ_p die üeberall undefinierte Funktion. Wir schreiben die Menge aller berechenbaren Funktionen als $\mathcal{P}$, die Menge aller totalen berechenbaren Funktionen als $\mathcal{R}$. Für eine Funktion f und x schreiben wir $f(x)\uparrow$, falls f *undefiniert* ist auf x; sonst schreiben wir $f(x)\downarrow$.

Ohne Beweis werden wir das folgende Theorem benutzen.

Theorem 2.1 (Universalität)**.** Es gibt u so dass, für alle p, x,

$$\varphi_u(p, x) = \varphi_p(x).$$

Mit anderen Worten: Die Funktion $(p, x) \mapsto \varphi_p(x)$ ist berechenbar.

Eine Menge $M \subseteq \mathbb{N}$ heißt berechenbar oder *entscheidbar*, falls ihre characteristische Funktion berechenbar ist. Falls die Funktion

$$x \mapsto \begin{cases} 0, & \text{falls } x \in M; \\ \uparrow, & \text{sonst;} \end{cases}$$

berechenbar ist, so heisst M *aufzählbar* (oder auch *semi-entscheidbar*).

Aufgabe 6

Zeige die folgenden Aussagen.
(1) $\{n \mid n \text{ gerade}\}$ ist berechenbar.
(2) $\{n \mid n \text{ Primzahl}\}$ ist berechenbar.
(3) Wenn M berechenbar ist, so ist auch $\overline{M}$ berechenbar.
(4) $\{\langle p, x\rangle \mid \varphi_p(x)\downarrow\}$ ist aufzählbar.

Wir können die Berechnungsschritte, die ein WHILE-Programm macht, zählen. Für eine Programm-Codezahl p und eine Eingabe x sei

$$\Phi_p(x)$$

die Anzahl der Schritte, die das zu p gehörige Programm auf Eingabe x bis zum Terminieren macht (undefiniert, falls das Programm nicht auf x hält). Ohne Beweis geben wir das folgende Theorem an.

Theorem 2.2 (Zeitschranken)**.** Das Prädikat

$$\Phi_p(x) \leq t$$

ist entscheidbar (also gibt es eine totale Funktion welche 1 ist genau dann, wenn das Prädikat hält).

Aufgabe 7

Zeige die folgenden Aussagen.

(1) Die Abbildung

$$x \mapsto \begin{cases} 1, & \text{falls } \exists y : \varphi_x(y) = 15; \\ \uparrow, & \text{sonst.} \end{cases}$$

ist berechenbar.

(2) Die Menge

$$\{p \in \mathbb{N} \mid \mathrm{dom}(\varphi_p) \neq \emptyset\}$$

ist aufzählbar.

(3) Wenn M und $\overline{M}$ aufählbar sind, so ist M berechenbar.

3 SMN und KRT

Das nächste wichtige Theorem ist s-m-n, manchmal auch Paramter Theorem genannt.

Theorem 3.1 (S-m-n Theorem)**.** Es gibt eine *injektive* total berechenbare Funktion $s \in \mathcal{R}$ so dass, für alle p, c, x,

$$\varphi_{s(p,c)}(x) = \varphi_p(c, x).$$

Proof. Intuitiv wollen wir c im Programm p hineinschreiben ("hard-coding"). Im Wesentlichen wollen wir also s wie folgt definieren.

$$P, c \mapsto \begin{array}{ll} x_2 & := x_1; \\ x_1 & := x_0 + c; \\ P & \end{array}$$

Allerdings wollen wir keine Programme, sondern *Programm-Codezahlen* abbilden. Deshalb definieren wir

$$s : \mathbb{N}^2 \to \mathbb{N}, (p, c) \mapsto \langle 2, \langle 0, 2, 1, 0\rangle, \langle 2, \langle 0, 1, 0, c\rangle, p\rangle\rangle.$$

Es ist klar, dass s injektiv ist und unseren Ansprüchen genügt. □

Aufgabe 8

Nicht-konstruktives s-m-n: Zeige: Für jede berechenbare Funktion f gibt es eine injektive, total berechenbare Funktion s, so dass

$$\forall x : \varphi_{s(c)}(x) = f(c, x).$$

Als direkte Konsequenz erhalten wir das *Padding Theorem*.

Theorem 3.2 (Padding Theorem)**.** Es gibt eine *injektive* total berechenbare Funktion $\text{pad} \in \mathcal{R}$ so dass, für alle p, c, x,

$$\varphi_{\text{pad}(p,c)}(x) = \varphi_p(x).$$

Proof. Sei f total berechenbar so dass

$$\forall p, c, x : f(\langle p, c\rangle, x) = \varphi_p(x).$$

f ist berechenbar dank Universalität. Mit nicht-konstruktivem s-m-n gibt es nun eine injektive, total berechenbare Funktion pad so, dass

$$\forall p, x : \varphi_{\text{pad}(p,c)}(x) = f(\langle p, c\rangle, x) = \varphi_p(x).$$

□

Sei A eine Menge so dass, für alle x, y mit $\varphi_x = \varphi_y$ und $x \in A$ auch gilt $y \in A$ heißt *semantisch*.

Aufgabe 9

Sei A eine nicht-leere semantische Menge. Zeige, dass A unendlich ist.

Als nächstes zeigen wir Kleene's Recursion Theorem (KRT). Intuitiv gibt uns KRT eine Möglichkeit berechenbare Funktionen in Abhängigkeit von einem Programm *für sich selbst* zu definieren.

Theorem 3.3 (Kleene's Recursion Theorem (KRT))**.** Es gibt eine injektive total berechenbare Funktion e so dass, für alle p, x,

$$\varphi_{e(p)}(x) = \varphi_p(e(p), x).$$

Proof. Sei eine berechenbare Funktion s wie im s-m-n Theorem gegeben. Sei d so, dass

$$\forall a, b, c : \varphi_d(a, b, c) = \varphi_b(s(a, \langle a, b\rangle), c).$$

Definiere $e \in \mathcal{R}$ so, dass, für alle p, $e(p) = s(d, \langle d, p\rangle)$. Wir haben

$$\begin{aligned} \forall p, x : \varphi_{e(p)}(x) &= \varphi_{s(d,\langle d,p\rangle)}(x) \\ &= \varphi_d(d, p, x) \\ &= \varphi_p(s(d, \langle d, p\rangle), x) \\ &= \varphi_p(e(p), x). \end{aligned}$$

Ausßerdem ist e injektiv. □

Aufgabe 10

Nicht-konstruktives KRT: Zeige: Für jede berechenbare Funktion f gibt es ein Programm e, so dass

$$\forall x : \varphi_e(x) = f(e, x).$$

Als nächstes wenden wir uns *unberechenbaren* Mengen zu. Das *Halteproblem* ist definiert als

$$\mathrm{HP} = \{\langle p, x\rangle \in \mathbb{N} \mid \varphi_p(x)\downarrow\}.$$

Wir zeigen jetzt mittels KRT die berühmte *Unentscheidbarkeit des Halteproblems.*

Theorem 3.4 ([Rog67])**.** Das Halteproblem ist unentscheidbar.

Proof. Wir beweisen diesen Satz über einen Widerspruch. Nehmen wir also an, das Halteproblem ist entscheidbar. Dann gibt es also eine berechenbare Funktion g so, dass für alle p, x,

$$\begin{aligned} \varphi_p(x)\downarrow &\Rightarrow g(p, x) = 1; \\ \varphi_p(x)\uparrow &\Rightarrow g(p, x) = 0. \end{aligned}$$

Definere f berechenbar so, dass

$$\forall e, x : f(e, x) = \begin{cases} 0, & \text{falls } g(e, x) = 0; \\ \uparrow, & \text{sonst.} \end{cases}$$

Mit nicht-konstruktivem KRT gibt es e, so dass

$$\forall x : \varphi_e(x) = f(e, x).$$

Wir haben nun

$$\varphi_e(0)\downarrow \Leftrightarrow g(e, x) = 0 \Leftrightarrow \varphi_e(0)\uparrow,$$

ein Widerspruch. □

Aufgabe 11

Zeige, dass die folgenden Mengen unentscheidbar sind.
(1) ε-Halteproblem: $\{p \in \mathbb{N} \mid \varphi_p(0)\downarrow\}$.
(2) Diagonales Halteproblem: $K = \{p \in \mathbb{N} \mid \varphi_p(p)\downarrow\}$.
(3) Totalität: $\{p \in \mathbb{N} \mid \varphi_p \text{ ist total}\}$.
(4) Test auf Lehrheit: $\{p \in \mathbb{N} \mid \text{dom}(\varphi_p) \neq \emptyset\}$.

Wir erinnern uns: Eine Menge A heißt *semantisch*, falls für alle x, y mit $\varphi_x = \varphi_y$ und $x \in A$ auch gilt $y \in A$. Alle in der letzten Aufgabe definierten Mengen sind Beispiele für semantische Mengen.

Wir verallgemeinern die Unentscheidbarkeit des Halteproblems zum Satz von Rice im folgenden Theorem.

Theorem 3.5 (Satz von Rice)**.** Sei A eine semantische Menge, welche nicht aus $\{\emptyset, \mathbb{N}\}$ ist. Dann ist A unentscheidbar.

Proof. Wir beweisen diesen Satz über einen Widerspruch. Nehmen wir also an, das A entscheidbar ist. Dann gibt es also eine berechenbare Funktion g so, dass für alle x,

$$\begin{aligned} x \in A &\Rightarrow g(x) = 1; \\ x \notin A &\Rightarrow g(x) = 0. \end{aligned}$$

Sei $p_0 \in A$ und $p_1 \notin A$. Definere f berechenbar so, dass

$$\forall e, x : f(e, x) = \begin{cases} \varphi_{p_0}(x), & \text{falls } g(e) = 0; \\ \varphi_{p_1}(x), & \text{sonst.} \end{cases}$$

Mit nicht-konstruktivem KRT gibt es e, so dass

$$\forall x : \varphi_e(x) = f(e, x).$$

Wir haben nun

$$\begin{aligned} e \in A &\Rightarrow g(e) = 1 \\ &\Rightarrow \varphi_e = \varphi_{p_1} \\ &\Rightarrow e \notin A. \end{aligned}$$

Andererseits haben wir aber auch

$$\begin{aligned} e \notin A &\Rightarrow g(e) = 0 \\ &\Rightarrow \varphi_e = \varphi_{p_0} \\ &\Rightarrow e \in A, \end{aligned}$$

ein Widerspruch. □

4 Beispiele

Für zwei Mengen A und B schreiben wir $A \leq_1 B$ falls es eine injektive total berechenbare Funktion f gibt, so dass, für alle e, [$e \in A$ genau dann wenn $f(e) \in B$]. Dies induziert eine prä-Ordnung auf der Potenzmenge von $\mathbb{N}$; intuitiv gesehen sind Mengen höher in der Ordnung schwerer zu berechnen. Für $A, B \subseteq \mathbb{N}$ schreiben wir $A \equiv_1 B$ falls $A \leq_1 B$ und $B \leq_1 A$.

Wir definieren die folgenden Mengen.

$$\begin{aligned} A_1 &= \{e \in \mathbb{N} \mid \varphi_e(0)\downarrow\}; \\ A_2 &= \{e \in \mathbb{N} \mid \varphi_e \text{ ist total}\}; \\ A_3 &= \{e \in \mathbb{N} \mid \text{dom}(\varphi_e) \text{ ist unenedlich}\}; \\ A_4 &= \{e \in \mathbb{N} \mid \text{dom}(\varphi_e) \neq \emptyset\}; \\ A_5 &= \{e \in \mathbb{N} \mid \text{range}(\varphi_e) \text{ ist unendlich}\}. \end{aligned}$$

Theorem 4.1. Es gilt

$$A_1 \equiv_1 A_4 \leq_1 A_2 \equiv_1 A_3 \equiv_1 A_5.$$

Proof. Wir zeigen zuerst $A_1 \leq_1 A_4$. Mit nicht-konstruktivem s-m-n und Zeitschranken gibt es eine injektive total berechenbare Funktion f so, dass

$$\forall e, x : \varphi_{f(e)}(x) = \varphi_e(0).$$

Wir zeigen, dass für alle e, $e \in A_1$ genau dann wenn $f(e) \in A_4$.

Nehmen wir zuerst $e \in A_1$ an. Dann haben wir $\varphi_e(0)\downarrow$; daher gilt $\varphi_{f(e)}(0)\downarrow$, wodurch $0 \in \text{dom}(\varphi_{f(e)})$, welches $f(e) \in A_4$ impliziert, wie gewünscht.

Nehmen wir nun $f(e) \in A_4$ an. Wir haben $\exists x : \varphi_{f(e)}(x)\downarrow$; wähle ein solches x. Nun gilt $\varphi_{f(e)}(x) = \varphi_e(0)$; das zeigt $\varphi_e(0)\downarrow$, also $e \in A_1$ wie gewünscht.

Als zweites zeigen wir $A_4 \leq_1 A_1$. Mit nicht-konstruktivem s-m-n und Universalität gibt es eine injektive total berechenbare Funktion f so, dass

$$\forall e, x : \varphi_{f(e)}(x) = \mu t . \exists a \leq t : \Phi_e(a) \leq t.$$

Nehme an, dass $e \in A_4$, und sei $x \in \text{dom}(\varphi_e)$. Sei außerdem $t = \max(x, \Phi_e(x))$. Dieses t ist ein Beispiel t so, dass $\exists a \leq t : \Phi_e(a) \leq t$; das zeigt $\varphi_{f(e)}(0)\downarrow$, und, somit, $f(e) \in A_1$ wie gewünscht.

Sei nun $f(e) \in A_1$. Daher haben wir $\varphi_{f(e)}(0)\downarrow$. Dann gibt es also ein t so, dass $\exists a \leq t : \Phi_e(a) \leq t$; wähle ein solches t und sei a so, dass $\Phi_e(a) \leq t$. Dann haben wir $a \in \text{dom}(\varphi_e)$, was $e \in A_4$ wie gewünscht gibt. □

Aufgabe 12

Zeige $A_1 \leq_1 A_2$ und $A_2 \equiv_1 A_3$.

Theorem 4.2. Es gilt

$$A_1 \not\leq_1 \overline{A_1} \text{ und } \overline{A_1} \not\leq_1 A_1.$$

Proof. Nehme an es gelte $A_1 \leq_1 \overline{A_1}$, vermittelt durch $s \in \mathcal{R}$. Dann gilt also, für alle e,

$$\varphi_e(0)\downarrow \Leftrightarrow \varphi_{s(e)}(0)\uparrow.$$

Mit KRT (Theorem 3.3) gibt es dann ein e so dass $\varphi_e = \varphi_{s(e)}$, ein Widerspruch.

Ebenso erhalten wir $\overline{A_1} \not\leq_1 A_1$. □

Theorem 4.3. Es gilt

$$A_2 \not\leq_1 A_1.$$

Proof. Nehme an es gäbe ein $s \in \mathcal{R}$ so dass, für alle e,

$$\varphi_e \text{ ist total } \Leftrightarrow \varphi_{s(e)}(0)\downarrow.$$

Mit KRT (Theorem 3.3), gibt es ein e so dass, für alle x,

$$\varphi_e(x) = \begin{cases} 0, & \text{falls } \neg(\Phi_{s(e)}(0) \leq x); \\ \uparrow, & \text{sonst.} \end{cases}$$

Es gilt nun

$$\begin{aligned} \varphi_e \text{ ist total} &\Rightarrow \forall x : \varphi_e(x)\downarrow \\ &\Rightarrow \forall x : \neg(\Phi_{s(e)}(0) \leq x) \\ &\Rightarrow \varphi_{s(e)}(0)\uparrow. \end{aligned}$$

Andererseits haben wir aber auch

$$\begin{aligned} \varphi_e \text{ ist nicht total} &\Rightarrow \exists x : \varphi_e(x)\uparrow \\ &\Rightarrow \exists x : \Phi_{s(e)}(0) \leq x \\ &\Rightarrow \varphi_{s(e)}(0)\downarrow. \end{aligned}$$

Dies ist ein Widerspruch zur Wahl von s. □

Aufgabe 13

Zeige die folgenden Aussagen mittels KRT.

(1) $A_3 \not\leq_1 A_4$.

(2) $A_2 \not\leq_1 \overline{A_1}$.

(3) $A_5 \not\leq_1 A_1$.

Literatur

[Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw Hill, New York, 1967. Reprinted by MIT Press, Cambridge, Massachusetts, 1987.

A Beweise ausgewählter Aufgaben

Lösung zu Aufgabe 2 (2)

Wir zeigen die folgende Aussage: Seien f, g berechenbar. Dann ist auch

$$x \mapsto \begin{cases} 1, & \text{falls } f(x) < g(x); \\ 0, & \text{sonst;} \end{cases}$$

berechenbar.

Beweis. Seien f und g als berechenbar gegeben. Dann gibt es also Programme P_0 und P_1 so dass P_0 f berechnet und P_1 berechnet g. Sei n mindestens 2 so, dass in P_0 und P_1 keine Variable mit Index $\geq n$ vorkommt. Das folgende Programm berechnet die gewünschte Funktion.

"$f < g$"

$x_n := x_1$;
P_0;
$x_{n+1} := x_0$;
$x_0 := 0$;
...
$x_{n-1} := 0$;
$x_1 := x_n$;
P_1;
$x_{n+2} := x_0 - x_{n+1}$;
IF $x_{n+2} = 0$ THEN
$x_0 := 1$
ELSE
$x_0 := 0$
END

□

Lösung zu Aufgabenblatt 1, Aufgabe 3

Wir zeigen die folgende Aussage: Seien f berechenbar und sei g eine Funktion so dass es eine endliche Menge D gibt so dass, für alle $x \in \mathbb{N} \setminus D$, $f(x) = g(x)$. Dann ist auch g berechenbar.

Beweis. Wir beweisen die Aussage mit Induktion über die Mächtigkeit von D und mit den Abschlusseigenschaften aus Aufgabe 2. Der Induktionsbeginn mit $\forall x : f(x) = g(x)$ ist trivial. Sei also jetzt g' eine Funktion die auf allen Eingaben bis auf einem z ist wie g, auf z ist sie wie f. Mit Induktionsannahme ist g' berechenbar. Sei c die Konstante $g(x)$ (undefiniert, falls $g(x)$ undefiniert). Mit Abschlusseigenschaften ist auch die folgende Funktion berechenbar.

$$x \mapsto \begin{cases} c, & \text{falls } x = z; \\ g'(x), & \text{sonst.} \end{cases}$$

Diese Funktion ist gleich g. □