Prof. Dr. Jasmin Blanchette
Jannis Limperg
Lydia Kondylidou

Ludwig-Maximilians-Universität München
Institut für Informatik
Abgabe bis 5. Juli 2023, 10:00 Uhr

Lösungsvorschlag zur Übung 11 zur Vorlesung

Formale Sprachen und Komplexität

FSK11-1 ***μ-Rekursion*** (2 Punkte)

Nehmen Sie für diese Aufgabe an, dass Addition, Subtraktion, Multiplikation, Division und absolute Differenz $\mathit{absdiff}(x_1, x_2) = |x_1 - x_2|$ primitiv rekursiv sind. Beachten Sie, dass wir in der Definition von *absdiff* x_1 und x_2 als ganze Zahlen subtrahieren, d.h. $\mathit{absdiff}(3,4) = \mathit{absdiff}(4,3) = 1$.

a) Berechnen Sie μg für folgende Funktionen g.

- $g\colon \mathbb{N} \to \mathbb{N}$, $g(x) = 2x + 10$

LÖSUNGSVORSCHLAG:
Die Funktion g hat keine Nullstellen, also ist μg undefiniert:

$$\mu g\colon \mathbb{N}$$
$$\mu g = \text{undefiniert}$$

- $g\colon \mathbb{N} \to \mathbb{N}$, $g(x) = (x-3)^2 - 4$

LÖSUNGSVORSCHLAG:
Die Funktion g hat zwei Nullstellen 1 und 5, von denen wir die kleinere nehmen:

$$\mu g\colon \mathbb{N}$$
$$\mu g = 1$$

- $g\colon \mathbb{N}^2 \to \mathbb{N}$, $g(x_1, x_2) = x_1 + x_2$

LÖSUNGSVORSCHLAG:
Die Funktion g nimmt den Wert 0 an g.d.w. $x_1 = x_2 = 0$. Für $x_2 = 0$ ist μg also 0; für alle anderen Werte von x_2 ist es undefiniert.

$$\mu g\colon \mathbb{N} \to \mathbb{N}$$
$$(\mu g)(x_2) = \begin{cases} 0 & \text{falls } x_2 = 0 \\ \text{undefiniert} & \text{sonst} \end{cases}$$

b) Zeigen Sie, dass die Exponentiationsfunktion

$$exp\colon \mathbb{N}^2 \to \mathbb{N}, \quad exp(x_1, x_2) = x_1^{x_2}$$

primitiv rekursiv ist.

LÖSUNGSVORSCHLAG:

Exponentiation ist iterierte Multiplikation. Wir definieren zunächst hilfsweise eine Exponentiationsfunktion exp' mit vertauschten Argumenten (d.h. $exp'(x_1, x_2) = x_2^{x_1}$):

$$exp'(x_1, x_2) = \begin{cases} 1 & \text{falls } x_1 = 0 \\ x_2 \cdot exp'(x_1 - 1, x_2) & \text{sonst} \end{cases}$$

(Mit dieser Definition ist $0^0 = 1$. Eine Definition, bei der $0^0 = 0$ ist, wäre auch akzeptabel.) Um zu sehen, dass diese Definition tatsächlich primitiv rekursiv ist, definiere

$$\begin{aligned} g(x_2) &= 0 \\ h(r, s, x_2) &= mult(\pi_3^3(r, s, x_2), \pi_1^3(r, s, x_2)) \end{aligned}$$

Wenn die Multiplikationsfunktion *mult* primitiv rekursiv ist, sind die Funktionen g und h offensichtlich ebenfalls primitiv rekursiv und wir können exp' schreiben als

$$exp'(x_1, x_2) = \begin{cases} g(x_2) & \text{falls } x_1 = 0 \\ h(exp'(x_1 - 1, x_2), x_1 - 1, x_2) & \text{sonst} \end{cases}$$

Die gewünschte Exponentiationsfunktion ist dann

$$exp(x_1, x_2) = exp'(x_2, x_1) = exp'(\pi_2^2(x_1, x_2), \pi_1^2(x_1, x_2))$$

Korrekturhinweis: Es ist auch erlaubt, die Rekursion auf einem anderen Argument als dem ersten durchzuführen (wie in der Vorlesung teilweise geschehen). Damit kann man sich die separate Definition von exp' sparen. Ebenso ist es erlaubt, eine Funktion auf beliebige Kombinationen der Eingabeparameter anzuwenden, ohne das mit den Projektionsfunktionen π_i^j zu rechtfertigen.

c) Zeigen Sie, dass die Logarithmusfunktion

$$log\colon \mathbb{N}^2 \to \mathbb{N}, \quad log(x_1, x_2) = \log_{x_1} x_2$$

μ-rekursiv ist. Beachten Sie, dass der Logarithmus für natürliche Zahlen nicht überall definiert ist. Es gilt:

$$log(x_1, x_2) = n \iff x_1^n = x_2$$

LÖSUNGSVORSCHLAG:

Für *log* gilt:

$$log(x_1, x_2) = n \iff x_1^n = x_2 \iff |x_1^n - x_2| = 0$$

Wir definieren also die primitiv rekursiv Funktion log':

$$\begin{aligned} log'(n, x_1, x_2) &= absdiff(exp(x_1, n), x_2) \\ &= absdiff(exp(\pi_2^3(n, x_1, x_2), \pi_1^3(n, x_1, x_2)), \pi_3^3(n, x_1, x_2)) \end{aligned}$$

Die Logarithmusfunktion ist dann

$$log = \mu log'$$

FSK11-2 ***Entscheidbarkeit und Reduktionen*** (2 Punkte)

a) Betrachten Sie die Sprache

$$L_1 = \{z \in \{0,1\}^* \mid \text{TM } M_z \text{ hält für Eingabe } 01\}$$

Welche der folgenden Aussagen ist korrekt? Beweisen Sie Ihre Antwort.

- L_1 ist entscheidbar.
- L_1 ist semi-entscheidbar, aber nicht entscheidbar.
- L_1 ist weder entscheidbar noch semi-entscheidbar.

Um zu zeigen, dass L_1 (semi-)entscheidbar ist, beschreiben Sie kurz die Funktionsweise einer deterministischen Turingmaschine, die L_1 (semi-)entscheidet. Um zu zeigen, dass L_1 nicht (semi-)entscheidbar ist, reduzieren Sie ein geeignetes Problem auf L_1.

LÖSUNGSVORSCHLAG:

L_1 ist semi-entscheidbar: für eine Eingabe z simulieren wir die Maschine M_z mit Eingabe 01. Wenn die Simulation hält, geben wir 1 aus.

L_1 ist nicht entscheidbar. Wir beweisen das durch Reduktion von H_0 auf L_1.

Um $H_0 \leq L_1$ zu zeigen, müssen wir eine totale, berechenbare Funktion $f\colon \{0,1\}^* \to \{0,1\}^*$ angeben, sodass für alle z gilt: $z \in H_0 \iff f(z) \in L_1$. Wir wählen

$$f(z) = \begin{cases} \langle M' \rangle & \text{falls } z = \langle M \rangle \\ z & \text{sonst} \end{cases}$$

Dabei ist

- M' eine Turingmaschine, die zunächst ihre Eingabe löscht und dann M ausführt.
- $\langle M \rangle$ und $\langle M' \rangle$ die Binärkodierung der jeweiligen Turingmaschine.

Die Funktion f ist offensichtlich total. Sie ist auch berechenbar, denn:

- Wir können Turingmaschinen binär kodieren und dekodieren (und erkennen, ob ein binäres Wort eine Turingmaschine kodiert).
- Wir können M' aus M konstruieren.

Für M' gilt:

- Wenn M mit Eingabe ε hält, dann hält M' mit jeder beliebigen Eingabe und also insbesondere mit Eingabe 01.
- M' hält mit einer beliebigen Eingabe, und also insbesondere mit Eingabe 01, nur dann, wenn M mit Eingabe ε hält.

Somit gilt:

$$\begin{array}{ll} & z \in H_0 \\ \text{g.d.w} & z = \langle M \rangle \text{ und } M \text{ hält mit Eingabe } \varepsilon \\ \text{g.d.w} & f(z) = \langle M' \rangle \text{ und } M' \text{ hält mit Eingabe 01} \\ \text{g.d.w.} & f(z) \in L_1 \end{array}$$

Somit ist $H_0 \leq L_1$ und da H_0 unentscheidbar ist, ist auch L_1 unentscheidbar.

b) Prüfen Sie, ob die folgenden Behauptungen wahr oder falsch sind. Begründen Sie Ihre Antworten wie folgt: Wenn eine Sprache L (semi-)entscheidbar ist, beschreiben Sie die Funktionsweise einer deterministischen Turingmaschine, die die charakteristische Funktion χ_L bzw. χ'_L berechnet. Wenn L nicht (semi-)entscheidbar

ist, leiten Sie einen Widerspruch ab.

i) Wenn A und B entscheidbare Sprachen sind, dann ist $A \cap B$ entscheidbar.

LÖSUNGSVORSCHLAG:
Wahr. Sei T_A eine DTM, die A entscheidet, und T_B eine DTM, die B entscheidet. Konstruiere eine DTM für $A \cap B$, die sich wie folgt verhält: Gegeben x, berechne $T_A(x)$. Falls $T_A(x) = 0$, lehne x ab, ansonsten berechne $T_B(x)$. Gilt $T_B(x) = 0$, lehne x ab, sonst akzeptiere x. Da T_A, T_B die jeweiligen Mengen entscheiden, terminieren beide DTM immer, womit auch die DTM zu $A \cap B$ stets mit dem korrekten Ergebnis terminiert. Somit ist $A \cap B$ entscheidbar.

ii) Wenn A und $A \cup B$ entscheidbar sind, dann ist B entscheidbar.

LÖSUNGSVORSCHLAG:
Falsch. Sei $A = \{0,1\}^*$ und $B = H_0 \subseteq \{0,1\}^*$. Dann sind A und $A \cup B = A$ entscheidbar, aber B nicht.

iii) Das Problem, ob $L(M) \neq \emptyset$ für eine gegebene Turingmaschine M gilt, ist semi-entscheidbar.

LÖSUNGSVORSCHLAG:
Wahr. Algorithmus: Für $i = 0, 1, \ldots$ schreibe nacheinander alle Wörter $w \in \Sigma^*$ mit $|w| \leq i$ auf das Band und simuliere M auf jedem w für i Schritte. Falls M in i Schritten akzeptiert, gib 1 aus. Ist $L(M) \neq \emptyset$, so testen wir M irgendwann für ausreichend viele Schritte auf Wörtern ausreichender Länge, um ein Wort aus $L(M)$ zu finden.

iv) Das Problem, ob $L(M) = \emptyset$ für eine gegebene Turingmaschine M gilt, ist semi-entscheidbar.

LÖSUNGSVORSCHLAG:
Falsch. Wäre dieses Problem semi-entscheidbar, so wäre zusammen mit der vorigen Teilaufgabe entscheidbar, ob $L(M) \neq \emptyset$ ist. Sei P die Menge der Turingmaschinen M (binär kodiert), für die $L(M) \neq \emptyset$ ist. Wir zeigen: $H_0 \leq P$ und somit wäre H_0 entscheidbar, wenn P entscheidbar wäre.
Um $H_0 \leq P$ zu zeigen, definieren wir die Funktion f, die die (binär kodierte) Turingmaschine M auf eine (binär kodierte) Turingmaschine M' abbildet. Dabei verhält sich M' wie folgt: Ist die Eingabe nicht das leere Wort, so akzeptiert M' nicht. Ist die Eingabe das leere Wort, so simuliert M' zunächst

M auf dem leeren Wort und akzeptiert dann (sofern M' gehalten hat). M' akzeptiert also höchstens das leere Wort, und das genau dann wenn M das leere Wort akzeptiert. Die Funktion ist offensichtlich total und berechenbar.
Somit gilt:

$$\begin{array}{ll} & M \in H_0 \\ \text{g.d.w.} & M \text{ hält auf } \varepsilon \\ \text{g.d.w.} & L(f(M)) \neq \emptyset \\ \text{g.d.w.} & f(M) \in P \end{array}$$

c) Zeigen Sie, dass das folgende Problem für jede Turingmaschine T und natürliche Zahl n entscheidbar ist.

„T hält auf jeder Eingabe nach höchstens n Schritten."

LÖSUNGSVORSCHLAG:

Für jede Eingabe w können wir prüfen, ob T auf w in n Schritten hält, indem wir T für n Schritte simulieren. Weiterhin kann T in n Schritten nur höchstens n Zeichen lesen. Deshalb gilt für Wörter w mit $|w| > n$: Wenn T nicht in n Schritten auf $w[0] \dots w[n]$ hält, dann hält T auch nicht in n Schritten auf w. Somit genügt es, nur Wörter mit Länge höchstens n zu testen. Da das endlich viele Wörter sind, ist das Problem entscheidbar.

FSK11-3 ***Primitiv rekursive Prädikate*** (0 Punkte)

Primitiv rekursive Funktionen, die auf $\{0, 1\}$ abbilden, können auch als Prädikate aufgefasst werden. Wir betrachten in dieser Aufgabe nur einstellige Prädikate p, wobei $p(x) = 0$ bedeutet, dass x die Eigenschaft p nicht besitzt, und $p(x) = 1$ bedeutet, dass x die Eigenschaft p besitzt.

a) Zeigen Sie, dass die Funktion *iszero* ein primitiv rekursives Prädikat ist.

$$\mathit{iszero}(x_1) = \begin{cases} 0 & \text{für } x_1 > 0 \\ 1 & \text{für } x_1 = 0 \end{cases}$$

LÖSUNGSVORSCHLAG:

Dass *iszero* auf {0,1} abbildet, ist an der Fallunterscheidung bereits zu sehen. Daher müssen wir nur noch zeigen, dass *iszero* primitiv rekursiv ist. Das kann man tun, indem man *iszero* wie folgt im primitiv rekursiven Schema angibt:

$$iszero(x_1) = \begin{cases} 1 & \text{für } x_1 = 0 \\ 0 & \text{sonst} \end{cases}$$

Dies entspricht dem Schema, denn es ist gleich

$$iszero(x_1) = \begin{cases} g() & \text{für } x_1 = 0 \\ h(iszero(x_1 - 1), x_1 - 1) & \text{sonst} \end{cases}$$

mit $g() = 1$ und $h(y_1, y_2) = 0$ (konstante Funktionen)

b) Zeigen Sie, dass die Funktion *even* ein primitiv rekursives Prädikat ist.

$$even(x_1) = \begin{cases} 1 & \text{für } x_1 \text{ gerade} \\ 0 & \text{für } x_1 \text{ ungerade} \end{cases}$$

LÖSUNGSVORSCHLAG:

Auch hier genügt es, *even* im primitiv rekursiven Schema anzugeben:

$$even(x_1) = \begin{cases} 1 & \text{für } x_1 = 0 \\ iszero(even(x_1 - 1)) & \text{sonst} \end{cases}$$

Die verwendeten Funktionen sind $g() = 1$ (konstante Funktion) und $h(y_1, y_2) = iszero(\pi_1^2(y_1, y_2))$ (Komposition).

Diese sind ebenfalls wieder in das primitiv rekursive Schema einzusetzen.

c) Zeigen Sie, dass die Funktion *ifnotzero* primitiv rekursiv ist.

$$ifnotzero(x_1, x_2, x_3) = \begin{cases} x_2 & \text{falls } x_1 > 0 \\ x_3 & \text{sonst} \end{cases}$$

LÖSUNGSVORSCHLAG:
Mit $g(y_1, y_2) = \pi_2^2(y_1, y_2)$, $h(y_1, y_2, y_3, y_4) = \pi_3^4(y_1, y_2, y_3, y_4)$ im primitiv rekursiven Schema erhält man

$$ifnotzero(x_1, x_2, x_3) = \begin{cases} g(x_2, x_3) & \text{falls } x_1 = 0 \\ h(ifnotzero(x_1 - 1), x_1 - 1, x_2, x_3) & \text{sonst} \end{cases}$$

d) Zeigen Sie, dass die Funktion *ifthenelse* primitiv rekursiv ist, angenommen dass $p(x)$ ein primitiv rekursives Prädikat ist.

$$ifthenelse(x_1, x_2, x_3) = \begin{cases} x_2 & \text{falls } p(x_1) \\ x_3 & \text{sonst} \end{cases}$$

LÖSUNGSVORSCHLAG: $ifthenelse(x_1, x_2, x_3) = ifnotzero(p(x_1), x_2, x_3)$
Dies entspricht dem primitiv rekursiven Schema, denn es ist gleich

$$ifthenelse(x_1, x_2, x_3) = ifnotzero(h_1(x_1, x_2, x_3), h_2(x_1, x_2, x_3), h_3(x_1, x_2, x_3))$$

wobei gilt:

$$\begin{aligned} h_1(x_1, x_2, x_3) &= p(\pi_1^3(x_1, x_2, x_3)) \\ h_2(x_1, x_2, x_3) &= \pi_2^3(x_1, x_2, x_3) \\ h_3(x_1, x_2, x_3) &= \pi_3^3(x_1, x_2, x_3) \end{aligned}$$

e) Zeigen Sie, dass, gegeben zwei primitiv rekursive Prädikate p und q, auch die Konjunktion $p \wedge q$ primitiv rekursiv ist.

LÖSUNGSVORSCHLAG:
Mit Multiplikation oder *ifnotzero*.

Wir zeigen es hier sogar für beliebig-stellige Prädikate:

$$(p \wedge q)(x_1, \ldots, x_k) = \mathit{ifnotzero}(p(x_1, \ldots, x_k), q(x_1, \ldots, x_k), z(x_1, \ldots, x_k))$$

Dabei ist $z(x_1, \ldots, x_k) = 0$ eine konstante Funktion.

Da *iszero* auf $\{0, 1\}$ wie die Negation wirkt, ist diese auch primitiv rekursiv. Da $\{\neg, \wedge\}$ funktional vollständig ist, können wir damit beliebige logische Verknüpfungen herstellen.

FSK11-4 ***Satz von Rice*** (0 Punkte)

Sei $T = (Z, \{a, b\}, \Gamma, \delta, z_0, \square, E)$ eine Turingmaschine. Welche der folgenden Fragestellungen zu T sind entscheidbar?

Zeigen Sie die Unentscheidbarkeit unentscheidbarer Fragestellungen mittels des Satzes von Rice. Bei entscheidbaren Fragestellungen erläutern Sie, wie die charakteristische Funktion berechnet wird.

LÖSUNGSVORSCHLAG:

Wiederholung:

Satz von Rice: Sei $\mathcal{R}$ die Klasse aller Turingberechenbaren Funktionen. Sei $\mathcal{S}$ eine beliebige Teilmenge, sodass $\emptyset \subset \mathcal{S} \subset \mathcal{R}$. Dann ist die Sprache

$$C(\mathcal{S}) = \{w \mid \text{die von } M_w \text{ berechnete Funktion liegt in } \mathcal{S}\}$$

unentscheidbar.

a) T hat mindestens $|\Gamma|!$ viele Zustände. (! bezeichnet die Fakultätsfunktion.)

LÖSUNGSVORSCHLAG: Das Problem ist entscheidbar, indem man aus der Turingmaschinenbeschreibung zunächst $|\Gamma|$ und $|Z|$ extrahiert, anschließend $|\Gamma|!$ berechnet und anschließend die Zahlen vergleicht. All das ist von einer Turingmaschine berechenbar.

b) Leerheitsproblem: Es gibt ein Wort $w \in L(T)$

LÖSUNGSVORSCHLAG: Das Problem ist unentscheidbar:
Sei $\mathcal{S} = \mathcal{R} \setminus \{\Omega\}$, also die Menge aller berechenbaren Funktionen, bis auf die überall undefinierte Funktion Ω.

$\mathcal{S}$ ist damit nicht leer und auch nicht die Menge aller berechenbaren Funktionen.

Dann ist $C(\mathcal{S}) = \{w_M \mid M \text{ akzeptiert mindestens für eine Eingabe}\}$

Mit dem Satz von Rice ist $C(\mathcal{S})$ unentscheidbar und damit ist das Leerheitsproblem (also ob eine Turingmaschine mindestens ein Wort akzeptiert) unentscheidbar.

c) Wenn $a \in L(T)$, dann auch $aa \in L(T)$

LÖSUNGSVORSCHLAG: Das Problem ist unentscheidbar:
Sei $\mathcal{S} = \{f \in \mathcal{R} \mid f(a) \text{ definiert } \Rightarrow f(aa) \text{ definiert}\}$. Dann sind totale (überall definierte) Funktionen wie $f\colon \{a,b\}^* \to \{a,b\}^*, w \mapsto a$ in $\mathcal{S}$ und die folgende (partielle) Funktion g ist nicht in $\mathcal{S}$:

$$g\colon \{a,b\}^* \to \{a,b\}^*, w \mapsto \begin{cases} a & \text{falls } w = a \\ \text{undefiniert} & \text{sonst} \end{cases}$$

Damit ist $\mathcal{S}$ nicht leer und auch nicht die Menge aller berechenbaren Funktionen.

Mit dem Satz von Rice ist $C(\mathcal{S}) = \{w_M \mid M \text{ akzeptiert } a \Rightarrow M \text{ akzeptiert } aa\}$ unentscheidbar.

d) Wenn es ein Wort $w \in L(T)$ gibt, dann gibt es auch ein Wort u, welches ein b enthält mit $u \in L(T)$

LÖSUNGSVORSCHLAG: Das Problem ist unentscheidbar:
Sei $\mathcal{S} = \{f \in \mathcal{R} \mid \exists w : f(w) \text{ definiert } \Rightarrow \exists u, u' \in \{a,b\}^* : f(ubu') \text{ definiert}\}$. Dann sind alle totalen Funktionen in $\mathcal{S}$ und die Funktion g aus c) ist wieder nicht in $\mathcal{S}$. Damit ist $\mathcal{S}$ nicht leer und auch nicht die Menge aller berechenbaren Funktionen.

Mit dem Satz von Rice ist $C(\mathcal{S}) = \{w_M \mid M \text{ akzeptiert eine Eingabe } \Rightarrow$ M akzeptiert eine Eingabe, die b enthält$\}$ unentscheidbar.