Prof. Dr. Jasmin Blanchette
Jannis Limperg
Luca Maio

Ludwig-Maximilians-Universität München
Institut für Informatik
Abgabe bis 3. Juli 2024, 10:00 Uhr

Lösungsvorschlag zur Übung 10 zur Vorlesung

Formale Sprachen und Komplexität

FSK10-1 ***Satz von Rice*** (2 Punkte)

Sei $M = (Z, \{a, b\}, \Gamma, \delta, z_0, \square, E)$ eine deterministische Turingmaschine. Welche der folgenden Fragestellungen zu M 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) Die Funktion die von M berechnet wird hat mindestens 3 Fixpunkte. Ein Fixpunkt einer Funktion $f : D \to D$ ist ein $x \in D$, sodass $f(x) = x$.

LÖSUNGSVORSCHLAG: Das Problem ist unentscheidbar:

Sei

$$\mathcal{S} = \{f \in \mathcal{R} \mid \exists x, y, z.\ x \neq y \wedge x \neq z \wedge y \neq z \wedge f(x) = x \wedge f(y) = y \wedge f(z) = z\},$$

also die Menge aller berechenbarer Funktionen die mindestens drei Fixpunkte haben.

Dann liegt die Funktion $f(x) = x$ in $\mathcal{S}$, da sie unendlich viele Fixpunkte hat, aber die Funktion $f(x) = x^2$ nicht, da sie nur zwei Fixpunkte hat (0 und 1).

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

Mit dem Satz von Rice ist

$$C(S) = \{w_M \mid \text{Die von } M \text{ berechnete Funktion hat mindestens drei Fixpunkte}\}$$

unentscheidbar.

b) Totalitätsproblem: Es gibt ein Wort $w \notin L(M)$.

LÖSUNGSVORSCHLAG: Das Problem ist unentscheidbar:

Sei $\mathcal{S}$ die Menge aller Funktionen, die an mindestens einer Stelle nicht definiert sind.

Dann liegt die Funktion Ω die überall undefiniert ist in $\mathcal{S}$, aber die Funktion $f(x) = x$ nicht.

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

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

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

c) Hat M einen Müllzustand? Formal definieren wir einen Müllzustand hier als einen Zustand z, der kein Endzustand ist und für den gilt $\delta(z, a) \subseteq \{z\}$ für alle a.

LÖSUNGSVORSCHLAG: Das Problem ist entscheidbar. Prüfe für alle Zustände, die keine Endzustände sind, die Übergänge für alle (endlich viele) Eingabesymbole. Wenn keiner der Übergänge zu einem anderen Zustand führt beende und gib Wahr zurück, sonst gib nach Prüfen aller Nicht-Endzustände Falsch zurück.

d) Ist die Funktion die von M berechnet wird in $O(n^2)$?

LÖSUNGSVORSCHLAG: Das Problem ist unentscheidbar:

Sei $\mathcal{S} = \{f \in \mathcal{R} \mid f \in O(n^2)\}$.

Dann ist die Funktion $f(x) = x$ in $\mathcal{S}$, aber *exp*, die Exponentialfunktion, ist nicht in $\mathcal{S}$

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

Mit dem Satz von Rice ist

$$C(\mathcal{S}) = \{w_M \mid \text{Die von } M \text{ berechnete Funktion ist in } O(n^2)\}$$

unentscheidbar.

e) M terminiert auf jeder Eingabe nach zwischen 50 und 55 Schritten.

LÖSUNGSVORSCHLAG: Das Problem ist entscheidbar. Simuliere M für bis zu 55 Schritten auf allen Eingaben mit einer Länge kleiner oder gleich 55 und falls M vor 50 Schritten terminiert oder nach 55 Schritten nicht terminiert gib Falsch zurück, sonst gib nach Prüfen aller (endlich vieler) Optionen Wahr zurück.

FSK10-2 ***Entscheidbarkeit und Reduktionen*** (0 Punkte)

a) Betrachten Sie die Sprache

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

wobei w_M das Encoding für M ist.

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 entscheidbar (bzw. 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

$$f(z) = \begin{cases} w_{M'} & \text{falls } z \text{ ein valides Encoding für eine Turingmaschine } M \text{ ist.} \\ z & \text{sonst} \end{cases}$$

Dabei ist

- M' eine Turingmaschine, die zunächst ihre Eingabe löscht und dann M ausführt.

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:

	$z \in H_0$
g.d.w	z ein valides Encoding für M ist und M hält mit Eingabe ε
g.d.w	$f(z) = w_{M'}$ und M' hält mit Eingabe 01
g.d.w.	$f(z) \in L_1$

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. Das ist aber unentscheidbar, was wir durch Reduktion auf H_0 zeigen.

Sei P die Menge der Turingmaschinen M (binär kodiert), für die $L(M) \neq \emptyset$ ist. Wir zeigen: $H_0 \leq P$ und somit ist P unentscheidbar.

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 deterministische Turingmaschine M und natürliche Zahl n entscheidbar ist.

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

LÖSUNGSVORSCHLAG:
Für jede Eingabe w können wir prüfen, ob M auf w in n Schritten hält, in-

dem wir M für n Schritte simulieren. Weiterhin kann M in n Schritten nur höchstens n Zeichen lesen. Deshalb gilt für Wörter w mit $|w| > n$: Wenn M nicht in n Schritten auf $w[0] \dots w[n]$ hält, dann hält M 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.

FSK10-3 ***μ-Rekursion*** (2 Punkte)

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

a) Berechnen Sie μg für folgende Funktionen g.

- $g\colon \mathbb{N} \to \mathbb{N}$, $g(x) = 3x^2 + 5x + 3$

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

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

- $g\colon \mathbb{N} \to \mathbb{N}$, $g(x) = (x/2 - 4)^2 - 2$, wobei wir $(x/2 - 4)^2$ ganzzahlig berechnen analog zu *absdiff*.

  LÖSUNGSVORSCHLAG:
  Die Funktion g hat Nullstellen zwischen 6 und 11, von denen wir die kleinste nehmen:

  $$\mu g\colon \mathbb{N}$$
  $$\mu g = 6$$

- $g\colon \mathbb{N}^2 \to \mathbb{N}$, $g(x_1, x_2) = (x_1 + 1) \cdot x_2$

  LÖSUNGSVORSCHLAG:
  Die Funktion g nimmt den Wert 0 an g.d.w. $x_1 + 1 = 0$ oder $x_2 = 0$. Für $x_2 = 0$ ist μg also 0, da der Wert von x_1 hier beliebig ist und wir somit den kleinsten Wert, also 0, wählen; für alle anderen Werte von x_2 ist es

undefiniert, da $x_1 \geq 0$ und damit nie gelten kann, dass $x_1 + 1 = 0$.

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

b) Zeigen Sie, dass die Fakultätsfunktion

$$fac \colon \mathbb{N} \to \mathbb{N}, \quad fac(x_1) = x_1!$$

primitiv rekursiv ist.

LÖSUNGSVORSCHLAG:
Die n-te Fakultät ist die Multiplikation $1 \cdot 2 \cdot \cdots \cdot (n-1) \cdot n = n!$

$fac(x_1) = x_1!$:

$$fac(x_1) = \begin{cases} 1 & \text{falls } x_1 = 0 \\ x_1 \cdot fac(x_1 - 1) & \text{sonst} \end{cases}$$

Um zu sehen, dass diese Definition tatsächlich primitiv rekursiv ist, definiere

$$g() = 1$$
$$h(r,s) = mult(succ(\pi_2^2(r,s)), \pi_1^2(r,s))$$

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

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

c) Zeigen Sie, dass die Quadratwurzelfunktion

$$sqrt \colon \mathbb{N} \to \mathbb{N}, \quad sqrt(x_1) = \sqrt{x_1}$$

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

$$sqrt(x_1) = n \iff n^2 = x_1$$

LÖSUNGSVORSCHLAG:
Für *sqrt* gilt:

$$sqrt(x_1) = n \iff n^2 = x_1 \iff |n^2 - x_1| = 0$$

Wir definieren also die primitiv rekursiv Funktion $sqrt'$:

$$\begin{aligned} sqrt'(n, x_1) &= absdiff(exp(n, 2), x_1) \\ &= absdiff(exp(\pi_1^2(n, x_1), g()), \pi_2^2(n, x_1)) \end{aligned}$$

wobei

$$g() = 2$$

Die Quadratwurzelfunktion ist dann

$$sqrt = \mu sqrt'$$

FSK10-4 ***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.

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

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.

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

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

$$\mathit{ifthenelse}(x_1, x_2, x_3) = \mathit{ifnotzero}(h_1(x_1, x_2, x_3), h_2(x_1, x_2, x_3), h_3(x_1, x_2, x_3))$$

wobei gilt:

$$\begin{aligned} h_1(x_1, x_2, x_3) &= p(\pi^3_1(x_1, x_2, x_3)) \\ h_2(x_1, x_2, x_3) &= \pi^3_2(x_1, x_2, x_3) \\ h_3(x_1, x_2, x_3) &= \pi^3_3(x_1, x_2, x_3) \end{aligned}$$

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.