Prof. Dr. Jasmin Blanchette
Elisabeth Lempa
Luca Maio

Ludwig-Maximilians-Universität München
Institut für Informatik
Besprechung 30.06.2025 bis 07.07.2025
Abgabe bis 14.07.2025, 10:00 Uhr

Lösungsvorschlag zur Übung 9 zur Vorlesung

Formale Sprachen und Komplexität

FSK9-1 *Halteprobleme*

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.

Um die Äquivalenz $z \in H_0 \iff f(z) \in L_1$ zu zeigen, beweisen wir die beiden Implikationen $z \in H_0 \Rightarrow f(z) \in L_1$ und $z \notin H_0 \Rightarrow f(z) \notin L_1$.

$$z \in H_0 \Rightarrow f(z) \in L_1\text{:}$$

$$\begin{array}{lll} & z \in H_0 & \\ \Rightarrow & z \text{ ist ein valides Encoding für } M \text{ und } M \text{ hält mit Eingabe } \varepsilon & \\ \Rightarrow & f(z) = w_{M'} \text{ und } M' \text{ hält mit jeder Eingabe (insbesondere 01)} & \\ \Rightarrow & f(z) \in L_1 & \end{array}$$

$$z \notin H_0 \Rightarrow f(z) \notin L_1\text{:}$$

$$\begin{array}{lll} & z \notin H_0 & \\ \text{Fall 1:} & z \text{ ein valides Encoding für } M & \\ \Rightarrow & M \text{ hält nicht mit Eingabe } \varepsilon & \\ \Rightarrow & f(z) = w_{M'} \text{ und } M' \text{ hält auf keiner Eingabe (insbesondere nicht 01)} & \\ \Rightarrow & f(z) \notin L_1 & \\ \text{Fall 2:} & z \text{ kein valides Encoding} & \\ \Rightarrow & f(z) = z \text{ ist kein valides Encoding} & \\ \Rightarrow & f(z) \notin L_1 & \end{array}$$

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

b) 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, indem 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.

FSK9-2 ***Entscheidbarkeit***

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.

a) 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.

b) 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.

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

LÖSUNGSVORSCHLAG:
Wahr. Algorithmus: Für $i = 0, 1, \dots$ 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.

FSK9-3 ***μ-Rekursion***

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.

Zur Erinnerung: Wir unterscheiden hier die (modifizierte) Subtraktion und die absoluten Differenz *absdiff*. (Wir schreiben in der folgenden Aufgabe zur besseren Unterscheidung $\dot{-}$ für die modifizierte Subtraktion.) Bei der (modifizierten) Subtraktion $x_1 \dot{-} x_2$ wird im Fall $x_2 > x_1$ auf 0 abgebildet, d.h. $3 \dot{-} 4 = 0$. Bei der absoluten Differenz werden dagegen x_1 und x_2 als ganze Zahlen subtrahiert, d.h. $|3 - 4| = 1$.

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:

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

- $g\colon \mathbb{N} \to \mathbb{N}$, $g(x) = (|x/2 - 4|)^2 \dot{-} 2$.

  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 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

$$sqrt = \mu sqrt'$$

FSK9-4 ***Primitiv rekursive Prädikate***

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.

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

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:

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

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.