Lösungsblatt 6

Aufgabe 1

Besprechung von Aufgabe 4.3 auf Ergänzungsblatt 5.

Lösung

Wähle $f\colon \{0,1\}^* \to \{0,1\}^*$, wobei $f(w)$ der Gödelindex einer Turingmaschine ist, die wie folgt arbeitet:

- Teste, ob das Eingabewort die Form $a^n b^n$ hat und falls nicht, gehe in eine Endlosschleife.
- Teste mit Dove-Tailing, ob M_w ein Wort akzeptiert, das mindestens Länge n hat und falls ja, gehe in einen Endzustand.

Dann ist f total und berechenbar und für alle $w \in \{0,1\}^*$ gilt

$$\begin{aligned} w \in \textsc{Fin} &\implies |T(M_w)| < \infty \\ &\implies \text{es gibt ein längstes Wort in } T(M_w) \\ &\implies T(M_{f(w)}) = \{a^n b^n \mid n \leq N\} \text{ für ein } N \in \mathbb{N} \\ &\implies T(M_{f(w)}) \text{ endlich} \\ &\implies T(M_{f(w)}) \text{ regulär} \\ &\implies f(w) \in \textsc{Reg} \end{aligned}$$

und

$$\begin{aligned} w \notin \textsc{Fin} &\implies |T(M_w)| = \infty \\ &\implies \text{für jedes } n \in \mathbb{N} \text{ gibt es ein } u \in T(M_w) \text{ mit } |u| \geq n \\ &\implies T(M_{f(w)}) = \{a^n b^n \mid n \in \mathbb{N}\} \\ &\implies T(M_{f(w)}) \text{ nicht regulär} \\ &\implies f(w) \notin \textsc{Reg}, \end{aligned}$$

also $w \in \textsc{Fin} \iff f(w) \in \textsc{Reg}$.

Aufgabe 2

Welche der folgenden Probleme sind semi- und welche co-semi-entscheidbar?

1. Problem A
   Eingabe: Eine DTM M
   Frage: Berechnet M die Ackermannfunktion a?

2. Problem B
   Eingabe: Eine DTM M
   Frage: Besitzt M mindestens 5 Zustände?

3. Problem C
   Eingabe: Eine DTM M.
   Frage: Besucht M, wenn auf ε gestartet, alle Zustände?

Lösung

Es gilt:

$$A = \{w \mid M_w \text{ berechnet die Ackermannfunktion } a\},$$
$$B = \{w \mid M_w \text{ besitzt mindestens 5 Zustände}\},$$
$$C = \{w \mid M_w \text{ besucht, wenn auf } \varepsilon \text{ gestartet, alle Zustände}\}.$$

1. Falscher Ansatz

   A ist co-semi-, aber nicht semi-entscheidbar.

   Wir zeigen, dass A co-semi-entscheidbar ist durch Angabe eines Algorithmus, das mit Dove-Tailing $\chi'_{\overline{A}}$ berechnet:

Eingabe: DTM M

```
for n = 0, 1, 2, ...:
    simuliere d_0(n) Berechnungsschritte von M auf (d_1(n), d_2(n))
    if M hält mit Ausgabe ungleich a(d_1(n), d_2(n)):
        return 1
```

   Erinnerung: Mit $d_i \colon \mathbb{N} \to \mathbb{N}$, $n \mapsto e(f^i(n))$ ist $\mathbb{N} \to \mathbb{N}^{k+1}$, $n \mapsto (d_0(n), \ldots, d_k(n))$ für jedes $k \in \mathbb{N}$ eine Bijektion von $\mathbb{N}$ auf $\mathbb{N}^{k+1}$.

   Wir zeigen mit dem Satz von Rice, dass A nicht semi-entscheidbar ist. Es gilt $A = \mathcal{C}(\mathcal{S})$ für $\mathcal{S} = \{a\}$. Wegen $\emptyset \subsetneq \{a\} \subsetneq \mathcal{R}$ ist $\mathcal{S}$ nichttrivial und A nach dem Satz vom Rice unentscheidbar. Da L co-semi-entscheidbar ist, kann L nicht semi-entscheidbar sein.

   Dieser Ansatz ist falsch! Was wurde übersehen?

   Der obige Algorithmus terminert nicht, falls M die Funktion Ω berechnet, obwohl er in diesem Fall 1 zurückgeben sollte.

   Richtiger Ansatz

   A ist weder semi- noch co-semi-entscheidbar.

   Wir zeigen, dass A nicht semi-entscheidbar ist durch die Reduktion $\overline{H_0} \leq A$. Hierzu konstruieren wir zu einer gegebenen TM M eine TM M', die auf Eingabe w wie folgt arbeitet:

Eingabe: Wort w

```
if w codiert ein 2-Tupel (m, n) ∈ ℕ²:
    simuliere |w| Berechnungsschritte von M auf ε
    if M hält nicht innerhalb der Simulation:
        return a(m, n)
geh in eine Endlosschleife
```

Man kann sich leicht davon überzeugen, dass die entsprechende Reduktionsfunktion total und berechenbar ist. Zu zeigen ist noch, dass M genau dann nicht auf ε hält, wenn M' die Ackermannfunktion berechnet.

„$\Longrightarrow$“: Angenommen, M hält nicht auf ε. Dann gibt M' für jede Eingabe, die ein 2-Tupel $(m, n) \in \mathbb{N}^2$ codiert, den Wert $a(m, n)$ aus, d. h. M' berechnet die Ackermannfunktion.

„$\Longleftarrow$“: Angenommen es existiert eine Zahl N, sodass M nach N Schritten auf ε hält. Dann hält M' nur für Eingaben, die ein 2-Tupel codieren und die Länge kleiner als N haben. Da nur endlich viele solcher Eingaben existieren und a für unendlich viele Eingaben definiert ist, wird die Ackermannfunktion nicht von M' berechnet.

Dass A nicht co-semi-entscheidbar ist, kann durch die Reduktion $H_0 \leq A$ aus dem Beweis des Satzes von Rice (2. Fall, da $\Omega \notin \{a\} = \mathcal{S}$) gezeigt werden.

2. B ist entscheidbar.

   χ_B wird durch folgenden Algorithmus berechnet:

Eingabe: DTM $M = (Q, \Sigma, \Gamma, \delta, s, \square, F)$

```
if |Q| ≥ 5:
    return 1
else:
    return 0
```

3. C ist semi-, aber nicht co-semi-entscheidbar.

   Wir zeigen, dass C semi-entscheidbar ist durch Angabe eines Algorithmus für χ'_C:

Eingabe: DTM $M = (Q, \Sigma, \Gamma, \delta, s, \square, F)$

```
S := ∅
for n = 0, 1, 2, ...:
    simuliere n Schritte von M auf ε und speichere den erreichten Zustand in q
    S := S ∪ {q}
    if S = Q:
        return 1
```

Wir zeigen, dass C nicht co-semi-entscheidbar ist, indem wir $H_0 \leq C$ beweisen. Hierzu geben wir eine totale und berechenbare Funktion $f\colon \{0,1\}^* \to \{0,1\}^*$ an für die gilt: M_w hält genau dann auf leerem Band, wenn $M_{f(w)}$, gestartet auf ε, alle Zustände besucht.

O. B. d. A. habe die Turingmaschine M_w die Form

$$M_w = (\{q_0, \ldots, q_n\}, \Sigma, \{a_0, \ldots, a_m\}, \delta, q_0, a_0, F).$$

Wähle f so, dass

$$M_{f(w)} = (\{q_0, \ldots, q_{n+1}\}, \Sigma, \{a_0, \ldots, a_{m+1}\}, \delta', q_0, a_0, \{q_{n+1}\})$$

mit

$$\delta'(q_i, a_j) = \begin{cases} \delta(q_i, a_j), & \text{falls } q_i \notin F,\ i \neq n+1 \text{ und } j \neq m+1 \quad (1) \\ (q_0, a_{m+1}, N), & \text{falls } q_i \in F,\ i \neq n+1 \text{ und } j \neq m+1 \quad (2) \\ (q_{i+1}, a_j, N), & \text{falls } i \neq n+1 \text{ und } j = m+1 \quad (3) \\ (q_i, a_j, N) & \text{sonst} \quad (4) \end{cases}$$

gilt.

Erläuterungen zur Arbeitsweise von $M_{f(w)}$:

(1) Zu Beginn verhält sich M_w so lange wie $M_{f(w)}$, bis ein Endzustand von M_w erreicht wird.

(2) Erreicht $M_{f(w)}$ einen Endzustand von M_w, schreibt $M_{f(w)}$ zum ersten Mal das Zeichen a_{n+1} auf das Band und geht in den Zustand q_0 über. Ab dann gilt, dass das Lesen des Symbols a_{n+1} bedeutet, dass die Simulation von M_w beendet ist.

(3) Ab dann werden alle Zustände von $M_{f(w)}$ der Reihe nach besucht ohne den Lesekopf von dem Zeichen a_{n+1} zu bewegen.

(4) Da q_{n+1} Endzustand von $M_{f(w)}$ ist, ist die Definition von $\delta(q_i, a_j)$ für die restlichen Fälle irrelevant.

Dann ist f total und berechenbar und für alle $w \in \{0,1\}^*$ gilt

$$\begin{aligned} w \in H_0 &\implies M_w \text{ hält auf } \varepsilon \\ &\implies M_{f(w)} \text{ besucht, wenn auf } \varepsilon \text{ gestartet, alle Zustände} \\ &\implies f(w) \in C \end{aligned}$$

und

$$\begin{aligned} w \notin H_0 &\implies M_w \text{ hält nicht auf } \varepsilon \\ &\implies M_{f(w)} \text{ besucht, wenn auf } \varepsilon \text{ gestartet, nie den Zustand } q_{n+1} \\ &\implies f(w) \notin C, \end{aligned}$$

also $w \in H_0 \iff f(w) \in C$.

Aufgabe 3

Geben Sie für jede der folgenden Instanzen P des Postschen Korrespondenzproblems über $\Sigma = \{0, 1\}$ eine Lösung kürzester Länge an oder begründen Sie, warum diese keine Lösung besitzen.

1. $P = ((100, 1), (1, 01), (1, 010), (10, 001))$
2. $P = ((001, 00), (11, 110), (10, 101), (101, 1010))$
3. $P = ((001, 10), (01, 0100), (011, 11))$
4. $P = ((100, 11), (010, 101), (0, 01))$

Lösung

1. Lösung kürzester Länge: $(1, 4, 2)$.
2. Diese Instanz besitzt keine Lösung, weil in keinem Paar eine Komponente ein Suffix der anderen ist, d. h. eine Lösung kann mit keinem der Indizes enden.
3. Lösung kürzester Länge: $(2, 1, 3)$.
4. Diese Instanz besitzt keine Lösung, weil in jedem Paar die zweite Komponente mehr 1 enthält als die erste.

Aufgabe 4

In der Vorlesung wurde das modifizierte Postsche Korrespondenzproblem MPCP auf das Postsche Korrespondenzproblem PCP reduziert.

Zeigen Sie, dass auch PCP $\leq$ MPCP gilt.

Lösung

Sei w die Codierung einer PCP-Instanz

$$P = ((x_1, y_1), \ldots, (x_k, y_k)).$$

Wähle $f(w)$ als die Codierung der MPCP-Instanz

$$P' = (\overset{1}{(\#\#, \#)}, \overset{2}{(x_1, \#y_1)}, \ldots, \overset{k+1}{(x_k, \#y_k)}, \overset{k+2}{(x_1, y_1)}, \ldots, \overset{2k+1}{(x_k, y_k)}).$$

f ist trivialerweise berechenbar. Außerdem ist f total, denn wir fordern, analog zur Codierung von Turingmaschinen, dass jedes Wort w irgendeine PCP-Instanz codiert.

Wir zeigen nun beide Richtungen der Äquivalenz

$$w \in \text{PCP} \iff f(w) \in \text{MPCP}.$$

getrennt voneinander.

$\underline{w \in \text{PCP} \implies f(w) \in \text{MPCP}}$

Sei $w \in$ PCP. Dann gibt es eine Lösung $(i_1, \ldots, i_n)$ mit $n \geq 1$ und $i_1, \ldots, i_n \in [k]$ für P. Dann ist $(1, i_1 + 1, i_2 + k + 1, \ldots, i_n + k + 1)$ eine Lösung von P'. Also gilt $f(w) \in$ MPCP.

$\underline{f(w) \in \text{MPCP} \implies x \in \text{PCP}}$

Sei $f(w) \in \text{MPCP}$. Dann gibt es eine Lösung $(i_1, \ldots, i_n)$ mit $n \geq 1$, $i_1, \ldots, i_n \in [2k+1]$ und $i_1 = 1$ für P'. Man erkennt, dass nach jeder 1 in der Lösung ein Index aus $\{2, \ldots, k{+}1\}$ folgen muss. Durch das wiederholte Entfernen von Einsen und das Addieren von k auf die jeweils nächste Komponente kann man aus jeder Lösung für P' eine neue Lösung gewinnen, in der nur die erste Komponente eine 1 ist. Sei $(1, j_1, \ldots, j_m)$ eine solche Lösung. Dann ist $(j_1 + k, j_2, \ldots, j_m)$ eine Lösung zu P. Also gilt $w \in \text{PCP}$. □