**Benotete Hausaufgaben zur Vorlesung** **WiSe 2023/2024**
**Berechenbarkeit und Komplexität** **TU Berlin**
**Weller/Froese/Kellerhals/Kunz/Peters** **18.12.2023**

# Hausaufgabenblatt

**Hinweise:**

- Die Hausaufgabe kann ab dem **17.01.2024, 12:00 Uhr** bis zum **19.01.2024, 23:59 Uhr** auf ISIS hochgeladen werden.
- Die Hausaufgabe sollte möglichst in Dreiergruppen bearbeitet werden. Bitte tragen Sie sich in ISIS bis zum **17.01.2024, 11:00 Uhr** in der Gruppenwahl ein. Die Hausaufgabe kann nur von eingetragenen Gruppen abgegeben werden.
- Bitte verwenden Sie die LaTeX-Vorlage auf ISIS für Ihre Abgabe.
- Plagiate werden nicht toleriert und werden scharf geahndet.
- Es können bis zu **25 Portfoliopunkte** erreicht werden.
- Alle Antworten sind zu begründen. Antworten ohne Begründung erhalten **0 Punkte**. Einzige Einschränkungen:
  - Um zu zeigen, dass eine Funktion (Sprache) von einer Turing-Maschine berechnet (akzeptiert) werden kann, reicht es aus, das Verhalten der Maschine algorithmisch zu beschreiben. Das Gleiche gilt für WHILE- und GOTO-Programme.
  - Sätze, die in der Vorlesung oder Modulkonferenz *bewiesen* wurden (auch skizzenhaft) dürfen verwendet werden, aber unbewiesene Mitteilungen und Lösungen zu Tutoriumsaufgaben dürfen nicht verwendet werden (bzw. Beweis muss erbracht werden).
  - Sie können die Existenz einer universellen Turing-Maschine (eine Maschine, die bei Eingabe $w\#x$ die Maschine $M_w$ auf Eingabe $x$ simuliert) annehmen.
  - Sie können verwenden, dass das allgemeine Halteproblem $H$ (Definition siehe unten) semi-entscheidbar ist.
- Wir behalten uns vor, pro Aufgabe mit $x$ erreichbaren Punkten nicht mehr als $x/2$ Seiten zu lesen.

**Erinnerungen:**

- Alle in den Aufgaben vorkommenden Turing-Maschinen sind deterministisch.
- $\Sigma$ ist ein beliebiges, endliches Alphabet. Das Symbol # ist ein Trennzeichen.
- Für jede Sprache $L \subseteq \Sigma^*$ ist $\overline{L} := \Sigma^* \setminus L$ ihr Komplement.
- Das allgemeine Halteproblem ist $H := \{w\#x \mid w, x \in \{0,1\}^* \text{ und } M_w \text{ hält bei Eingabe } x\}$.
- Das spezielle Halteproblem ist $K := \{w \in \{0,1\}^* \mid w\#w \in H\}$.
- Das Halteproblem auf leerem Band ist $H_0 := \{w \in \{0,1\}^* \mid w\# \in H\}$.
- Eine Turing-Maschine heißt „Rechtsdrall-Turing-Maschine“ falls alle definierten Übergänge den Kopf nach rechts bewegen, also in der Form $\delta(z_i, x) = (z_j, y, \mathtt{R})$ sind.

**Aufgabe 1.** *(Semi-)Entscheidbarkeit von Sprachen* 10 P.

Zeigen oder widerlegen Sie für jede der folgenden Sprachen jeweils Semi-Entscheidbarkeit und Entscheidbarkeit.

- $A \coloneqq \{w \in \{0,1\}^* \mid \text{es gibt eine Eingabe } x \text{ auf der } M_w \text{ die Ausgabe 0 produziert}\}$
- $B \coloneqq K \cap \{w \in \{0,1\}^* \mid M_w \text{ ist eine Rechtsdrall-Turing-Maschine}\}$
- $C \coloneqq \{w\#q \mid w, q \in \{0,1\}^* \text{ und } T(M_w) \subseteq T(M_q)\}$

Lösung:

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

  Nicht-entscheidbar wegen des Satzes von Rice auf $\mathcal{S} = \{f \mid \text{es gibt ein } x \text{ mit } f(x) = 0\}$ (offensichtlich eine nicht-triviale Menge).

  Semi-entscheidbar, weil die folgende TM genau A akzeptiert.

  1. Sei $w$ die Eingabe.
  2. Zähle alle Wörter in $\Sigma^*$ auf, und simuliere $M_w$ *parallel* auf jeder Eingabe $x \in \Sigma^*$.

- $B$ ist entscheidbar. Das können wir wie folgt sehen.

  Sei $w \in \Sigma^*$. Wir können feststellen, ob $M_w$ eine RTM ist, indem wir all die Übergänge von $M_w$ überprüfen.

  Nun unter der Annahme, dass $M_w$ eine RTM ist, können wir feststellen, ob $M_w$ auf jeder Eingabe $x$ hält, d.h. ob $w\#x \in H$ (insbesondere ob $w \in K$). Innerhalb von $|x| + 1$ Schritten wird entweder $M_w$ halten, oder wird ihr Kopf in Schritt $|x| + 1$ über einem Blanksymbol sein. Sei $s$ die Anzahl der Zustände, die $M_w$ hat. Es ist leicht zu sehen, dass entweder in den nächsten $s$ Schritten wird $M_w$ halten, oder in einen früher-erreichten Zustand landen. Im letzteren Fall wird $M_w$ nie halten.

  Jetzt können wir eine TM für $B$ aufbauen.

  1. Sei $w$ die Eingabe und $s$ die Anzahl der Zustände in $M_w$.
  2. Überprüfe, ob all die Übergänge von $M_w$ den Kopf nach rechts bewegen. Wenn nicht, lehne $w$ ab.
  3. Simuliere $M_w$ auf $w$ für $s+|w|+1$ Schritte. Wenn $M_w$ innerhalb dieser Schritten hält, akzeptiere $w$. Sonst lehne $w$ ab.

- $C$ ist weder entscheidbar noch semi-entscheidbar. Wir zeigen dies mit zwei Reduktionen.

  Zunächst zeigen wir $H_0 \leq C$. Dazu sei $f\colon \Sigma^* \to \Sigma^*$ eine Reduktionsfunktion mit $x \mapsto w\#q$, wobei $M_w$ jede Eingabe akzeptiert (also ist $T(M_w) = \Sigma^*$). Klar gilt $T(M_w) \subseteq T(M_q)$ nur dann wenn $T(M_w) = T(M_q) = \Sigma^*$. Die Maschine $M_q$ soll auf jeder Eingabe die Maschine $M_x$ auf leerer Eingabe simuliert und dann akzeptiert wenn $M_x$ hält. Dann gilt:

  $$\begin{aligned} x \in H_0 &\iff M_x \text{ hält auf leerer Eingabe} \\ &\iff M_q \text{ akzeptiert jede Eingabe} \\ &\iff T(M_w) \subseteq T(M_q) \\ &\iff w\#x \in C. \end{aligned}$$

  Nun zeigen wir $\overline{H}_0 \leq C$. Dazu sei $g\colon \Sigma^* \to \Sigma^*$ eine Reduktionsfunktion mit $x \mapsto w\#q$, wobei $M_w$ jede Eingabe akzeptiert (also ist $T(M_w) = \Sigma^*$). Wie vorher gilt dann $T(M_w) \subseteq T(M_q)$ nur dann wenn $T(M_w) = T(M_q) = \Sigma^*$. Die Maschine $M_q$ soll auf Eingabe $y$ die Maschine $M_x$ für $n_y$ Schritte auf leerer Eingabe simulieren. Wenn in der Simulation die Maschine $M_x$ hält, dann verwirft $M_q$ die Eingabe. Wenn in der Simulation die Maschine $M_x$ nicht hält, dann akzeptiert $M_q$ die Eingabe.

Nun gilt:

$$\begin{aligned} x \in \overline{H_0} &\iff M_x \text{ hält } \textit{nicht} \text{ auf leerer Eingabe} \\ &\iff \text{es gibt kein } n \in \mathbb{N}, \text{ sodass } M_x \text{ auf leerer Eingabe innerhalb von } n \text{ Schritten hält} \\ &\iff \text{für jede Eingabe } y \text{ simuliert } M_q \text{ für } n_y \text{ Schritte und akzeptiert anschließend} \\ &\iff M_q \text{ akzeptiert jede Eingabe} \\ &\iff T(M_q) = \Sigma^* \\ &\iff T(M_w) \subseteq T(M_q) \\ &\iff w\#x \in C. \end{aligned}$$

**Aufgabe 2.** *Reduktionen* 8 P.

Sei $c_0 \colon \{0,1\}^* \to \{0,1\}^*$ die konstante 0-Funktion (d.h. $c_0(x) = 0$ für alle $x \in \{0,1\}^*$) und sei

$$L \coloneqq \{w \in \{0,1\}^* \mid M_w \text{ berechnet die Funktion } c_0\}.$$

1. Reduzieren Sie $H$ auf $L$.
2. Reduzieren Sie $\overline{H}$ auf $L$.
3. Zeigen oder widerlegen Sie, dass $L \leq H_0$ gilt.

Lösung:

1. $H \leq L$: Wir definieren die Reduktionsfunktion $f \colon \{0,1,\#\}^* \to \{0,1\}^*$ wie folgt. Für $s = w\#x$ mit $w, x \in \{0,1\}^*$ sei $f(s) \coloneqq \langle M^s \rangle$, wobei $M^s$ eine Turing-Maschine ist, die das Folgende tut:
   - Überschreibe das Eingabewort mit $x$.
   - Verfahre wie $M_w$ auf dem Eingabewort $x$. Falls $M_w$ hält, so gib den Funktionswert 0 aus.

   Für alle anderen $s \in \{0,1,\#\}^*$ sei $f(s) \coloneqq \epsilon \notin L$. Die Funktion $f$ ist offensichtlich total und berechenbar (Kodieren von TM).

   Korrektheit: Sei $s = w\#x$ (für alle anderen $s$ ist Korrektheit klar).

   Falls $s \in H$, so hält $M_w$ auf $x$. Also gibt die TM $M^s$ für alle Eingaben eine 0 aus. Somit gilt $f(s) \in L$.

   Falls $f(s) \in L$, so berechnet $M^s$ die Funktion $c_0$. Per Konstruktion muss dann $M_w$ auf $x$ gehalten haben. Also gilt $s \in H$.

2. $\overline{H} \leq L$: Wir definieren die Reduktionsfunktion $f \colon \{0,1,\#\}^* \to \{0,1\}^*$ wie folgt. Für alle $s \in \{0,1,\#\}^*$, die nicht die Form $s = w\#x$ mit $w, x \in \{0,1\}^*$ haben, sei $f(s) \coloneqq \langle M^0 \rangle \in L$, wobei $M^0$ eine feste TM ist, die $c_0$ berechnet. Eine solche TM existiert.

   Für $s = w\#x$ sei $f(s) \coloneqq \langle M^s \rangle$, wobei $M^s$ eine Turing-Maschine ist, die bei Eingabe $a \in \{0,1\}^*$ das Folgende tut:
   - Simuliere $M_w$ auf Eingabe $x$ für $n \coloneqq \lfloor a \rfloor_2$ Schritte.
   - Falls $M_w$ innerhalb dieser $n$ Schritte hält, gib 1 aus. Sonst gib 0 aus.

   Korrektheit: Sei $s = w\#x$ (für alle anderen $s$ ist Korrektheit klar).

   Falls $s \in \overline{H}$, so hält $M_w$ nicht auf $x$. Also gibt die TM $M^s$ für alle Eingaben eine 0 aus. Somit gilt $f(s) \in L$.

   Falls $f(s) \in L$, so berechnet $M^s$ die Funktion $c_0$. Per Konstruktion gilt also, dass $M_w$ auf Eingabe $x$ für kein $n \in \mathbb{N}$ nach $n$ Schritten hält. Also gilt $s \in \overline{H}$.

3. $L \leq H_0$ gilt nicht. Da $H$ semi-entscheidbar aber unentscheidbar ist, ist $\overline{H}$ nicht semi-entscheidbar (Satz VL). Wenn nun $L \leq H_0$, dann hätten wir mit 2.

$$\overline{H} \leq L \leq H_0 \leq H$$

(die Reduktion $H_0 \leq H$ ist trivial gegeben durch $f(w) \to w\#$). Somit wäre $\overline{H}$ semi-entscheidbar. Widerspruch.

---

**Aufgabe 3.** *Postsches Korrespondenzproblem* 7 P.

1. Welche der drei folgenden Wörter sind in der Sprache PCP mit Alphabet $\{a, b\}$ enthalten?

$$I_1 = \langle((aa, ab), (aaa, ab))\rangle \qquad I_2 = \langle((aaab, aa), (b, abb))\rangle \qquad I_3 = \langle((aa, a), (a, aaa))\rangle$$

Lösung:
$I_1$ ist nicht in PCP, da man mit keinem der beiden Tupel beginnen kann.
$I_2$ hat die Lösung $i_1 = 1, i_2 = 2$.
$I_3$ hat die Lösung $i_1 = i_2 = 1$, $i_3 = 2$.

---

2. Zeigen oder widerlegen Sie die Entscheidbarkeit folgender Sprache:

$$\begin{aligned} P^* \coloneqq \{\langle((x_1, y_1), \ldots, (x_k, y_k))\rangle \mid\ & k \geq 1, x_i, y_i \in \{0,1\}^* \text{ für alle } i \in \{1, \ldots, k\}, \\ & \text{wobei } x_i \text{ und } y_i \text{ keine zwei 1'en hintereinander enthalten,} \\ & \text{und es existieren } n \geq 1 \text{ und } i_1, \ldots, i_n \in \{1, \ldots, k\}, \\ & \text{sodass } x_{i_1} \cdot \ldots \cdot x_{i_n} = y_{i_1} \cdot \ldots \cdot y_{i_n}\} \end{aligned}$$

Lösung:
$P^*$ ist unentscheidbar. Beweis mittels Reduktion PCP $\leq P^*$. Sei $\Sigma = \{a_1, \ldots, a_m\}$ ein endliches Alphabet. Wir definieren

$$h\colon \Sigma \to \{0,1\}^*, \quad h(a_i) \coloneqq 10^i$$

und

$$g\colon \Sigma^* \to \{0,1\}^*, \quad g(w_1 w_2 \ldots w_\ell) \coloneqq h(w_1)h(w_2)\cdots h(w_\ell).$$

Die Funktionen $h$ und $g$ sind offensichtlich total und berechenbar. Außerdem ist $h$ injektiv, da jeder Buchstabe mit einer anderen Anzahl 0'en kodiert wird. Daher ist auch $g$ injektiv.

Wir definieren die Reduktionsfunktion $f\colon \{0,1\}^* \to \{0,1\}^*$ wie folgt: Für alle $x \in \{0,1\}^*$, die keine korrekt kodierte PCP Instanz darstellen, setzen wir $f(x) \coloneqq \epsilon \notin P^*$. Für $x = \langle((x_1, y_1), \ldots, (x_k, y_k))\rangle$ definieren wir $f(x) \coloneqq \langle((g(x_1), g(y_1)), \ldots, (g(x_k), g(y_k)))\rangle$.

Korrektheit: Sei $x = \langle((x_1, y_1), \ldots, (x_k, y_k))\rangle$ (für ungültige Kodierungen $x$ gilt $x \notin$ PCP und $f(x) \notin P^*$).

Falls $x \in$ PCP, so gibt es $i_1, \ldots, i_n \in \{1, \ldots, k\}$ mit $x_{i_1} \cdot \ldots \cdot x_{i_n} = y_{i_1} \cdot \ldots \cdot y_{i_n}$. Zu zeigen: $f(x) \in P^*$. Zunächst stellen wir fest, dass $g(x_j) \in \{0,1\}^*$ und $g(y_j) \in \{0,1\}^*$ für alle $j \in \{1, \ldots, k\}$ per Konstruktion keine zwei 1'en hintereinander enthalten. Außerdem gilt per Konstruktion, dass

$$g(x_{i_1}) \cdot \ldots \cdot g(x_{i_n}) = g(x_{i_1} \cdot \ldots \cdot x_{i_n}) = g(y_{i_1} \cdot \ldots \cdot y_{i_n}) = g(y_{i_1}) \cdot \ldots \cdot g(y_{i_n})$$

und somit $f(x) \in P^*$.

Falls $f(x) \in P^*$, so gibt es $i_1, \ldots, i_n \in \{1, \ldots, k\}$ mit $g(x_{i_1}) \cdot \ldots \cdot g(x_{i_n}) = g(y_{i_1}) \cdot \ldots \cdot g(y_{i_n})$. Da $g$ injektiv ist, gilt

$x_{i_1} \cdot \ldots \cdot x_{i_n} = g^{-1}(g(x_{i_1} \cdot \ldots \cdot x_{i_n})) = g^{-1}(g(x_{i_1}) \cdot \ldots \cdot g(x_{i_n})) = g^{-1}(g(y_{i_1}) \cdot \ldots \cdot g(y_{i_n})) = g^{-1}(g(x_{i_1} \cdot \ldots \cdot y_{i_n}) = y_{i_1} \cdot \ldots \cdot y_{i_n}$.

Somit gilt $x \in$ PCP.

---