Institut für Theoretische Informatik
Lehrstuhl Prof. Dr. D. Wagner

Übungsblatt 3

Vorlesung Theoretische Grundlagen der Informatik im WS 17/18

Ausgabe 21. November 2017
Abgabe 5. Dezember 2017, 11:00 Uhr (im Kasten im UG von Gebäude 50.34)

Aufgabe 1

(1 + 1 + 1 + 1 + 1 = 5 Punkte)

Betrachte das Post'sche Korrespondenzproblem (kurz: PKP) aus der Vorlesung mit Eingabe $K = ((x_1, y_1), \ldots, (x_n, y_n))$, $x_i, y_i \in \Sigma^*$ für $i = 1, \ldots, n$.

(a) Zeigen Sie dass PKP entscheidbar ist, wenn $|x_i| = |y_i|$ für $i = 1, \ldots, n$.

(b) Zeigen Sie dass PKP entscheidbar ist, wenn $|\Sigma| = 1$.

(c) Zeigen Sie dass $K = ((10, 101), (011, 11), (101, 011))$ eine Nein-Instanz des PKP ist.

(d) Zeigen Sie dass $K = ((001, 0), (01, 011), (01, 101), (10, 001))$ eine Ja-Instanz des PKP ist.

(e) Zeigen Sie dass PKP nicht entscheidbar ist, wenn $|\Sigma| = 2$.
(Benutzen Sie dass PKP nicht entscheidbar ist für allgemeine Σ.)

Lösung:

(a) Wir behaupten, dass K genau dann lösbar ist, wenn ein $i \in \{1, \ldots, n\}$ existiert mit $x_i = y_i$. Dann folgt die Entscheidbarkeit, weil die letztere Eigenschaft einfach algorithmisch getestet werden kann.

Erstens: Wenn es ein $i \in \{1, \ldots, n\}$ gibt mit $x_i = y_i$, dann ist K natürlich eine Ja-Instanz mit Lösung i. Zweitens: Wenn K eine Ja-Instanz mit Lösung $i_1, i_2, \ldots, i_t$ ist, dann gilt $x_{i_1} x_{i_2} \cdots x_{i_t} = y_{i_1} y_{i_2} \cdots y_{i_t}$ und weil $|x_{i_1}| = |y_{i_1}|$ insbesondere $x_{i_1} = y_{i_1}$.

(b) Wir behaupten, dass K genau dann lösbar ist, wenn es $i, j \in \{1, \ldots, n\}$ gibt mit $|x_i| > |y_i|$ und $|x_j| < |y_j|$, oder es ein $i \in \{1, \ldots, n\}$ gibt mit $|x_i| = |y_i|$. Dann folgt wieder die Entscheidbarkeit, weil auch diese Eigenschaft einfach algorithmisch getestet werden kann.

Erstens: Angenommen es gibt $i, j \in \{1, \ldots, n\}$ mit $|x_i| > |y_i|$ und $|x_j| < |y_j|$. Sei $|x_i| - |y_i| = a > 0$ und $|y_j| - |x_j| = b > 0$. Dann ist K eine Ja-Instanz mit Lösung $i^b j^a$, denn

$$x_i^b x_j^a = \sigma^{b(|y_i|+a)} \sigma^{a(|y_j|-b)} = \sigma^{b|y_i|+a|y_j|} = y_i^b y_j^a,$$

wobei $\Sigma = \{\sigma\}$. Wenn es ein $i \in \{1, \ldots, n\}$ gibt mit $|x_i| = |y_i|$, dann wird K natürlich gelöst durch die Folge i.

Zweitens: Wenn K nicht die geforderten Eigenschaften hat, dann gilt entweder für alle $i \in \{1, \ldots, n\}$ $|x_i| > |y_i|$ oder für alle $i \in \{1, \ldots, n\}$ $|x_i| < |y_i|$. In beiden Fällen ist K offensichtlich nicht lösbar.

(c) Seien die Elemente aus der Instanz K wie folgt bezeichnet: $A = (10, 101)$, $B = (011, 11)$, $C = (101, 011)$. Da Längen der einzelnen Wörter in A, B und C implizieren, dass in jeder Lösung von K mit a A's, b B's und c C's gelten muss

$$2a + 3b + 3c = 3a + 2b + 3c.$$

Denmach müsste also $a = b$ gelten. Betrachten wir die Anzahl der Nullen der einzelnen Wörter in A, B und C, so sehen wir dass

$$a + b + c = a + c$$

und demnach $b = 0$ gelten müsste. Also haben wir auch $a = 0$ und eine Lösung kann nur aus C's bestehen. Aber C kann weder das erste noch letzte Element einer Lösung sein, und es folgt, dass K eine Nein-Instanz ist.

(d) Seien die Elemente aus der Instanz K wie folgt bezeichnet: $A = (001, 0)$, $B = (01, 011)$, $C = (01, 101)$, $D = (10, 001)$. Offensichtlich kann keine Lösung mit C oder D beginnen. Wenn das erste Element ein A ist, kann das zweite nur wieder A sein. Allerdings gibt es nach AA keine Fortsetzung. Wir folgern, dass jede Lösung mit B beginnt, wobei die einzig mögliche Fortsetzung dann D ist.

Betrachten wir nun das Ende einer möglichen Lösung, sehen wir, dass das letzte Element ein C sein muss. Die einzig mögliche Fortsetzung (vor dem C) ist ein A, davor muss wieder ein A kommen. Tatsächlich beobachten wir, dass sich so die letzten 64 Elemente einer jeden Lösung eindeutig bestimmen lassen – jedes Element ist die eindeutige Fortsetzung der ihm nachfolgenden bereits gesetzten Symbole. Nach 64 festgesetzten Elementen gibt es erstmal die Wahl vor dem nun vorne stehenden C ein D oder ein C zu setzen. Aus der Vorlesung wissen wir dass es eine Lösung mit 66 Elementen gibt, und tatsächlich erhält man eine Lösung durch die Wahl von D und davor B:

$$BDCD^2BAB(DC)^2D^2CD^2BAD^2BACDA^2CD^3(BA)^2A^2(CD)^2ABAD^2\ldots$$
$$\ldots BADA^2CD(AAC)^2ABADA^2C$$

Das entsprechende Wort ist:

$$0110011010010010110011001101001101001001101001001011000 10\ldots$$
$$\ldots 0101101010010010100100100101100110001010011010 0\ldots$$
$$\ldots 1001100010010110001001010010010100101001100010 0101$$

(e) Zu jeder Instanz $K = \{(x_1, y_1), \ldots, (x_k, y_k)\}$ des PKP mit Alphabet Σ konstruieren wir eine zugehörige Instanz K' mit Alphabet $\{0, 1\}$, so dass K genau dann eine Ja-Instanz ist, wenn K' eine Ja-Instanz ist. Die Idee ist, die Symbole in Σ binär zu kodieren. Zu diesem Zweck sei jedes Symbol $\alpha \in \Sigma$ einem unterschiedlichen Wort $w_\alpha \in \{0, 1\}^b$ zugeordnet, wobei $b \geq \log_2(|\Sigma|)$ beliebig aber fest ist.

Für jedes Paar (x_i, y_i) aus K mit $x_i = \alpha_1 \cdots \alpha_i$, $y_i = \beta_1 \cdots \beta_j$ sei (x'_i, y'_i) das Paar für das gilt $x'_i = w_{\alpha_1} \cdots w_{\alpha_i}$ und $y'_i = w_{\beta_1} \cdots w_{\beta_j}$. Sei nun $K' = \{(x'_i, y'_i) \in \{0, 1\}^* \times \{0, 1\}^* \mid (x_i, y_i) \in K\}$.

Es ist klar, dass jede Lösung $i_1, i_2, \ldots, i_m$ von K auch eine Lösung von K' ist. Für die andere Richtung brauchen wir eine entscheidende Beobachtung: Für jedes Element $(x'_i, y'_i) \in K'$ mit zugehörigem Element $(x_i, y_i) \in K$ gilt $|x'_i| = b|x_i|$ und $|y'_i| = b|y_i|$. Das heißt, dass in jeder Lösung $i'_1, i'_2, \ldots, i'_n$ von K' die binär kodierten Symbole von K ordentlich ausgerichtet auftreten. Jeder Block von b aufeinanderfolgenden Symbolen in der Wort $x'_{i'_1} x'_{i'_2} \cdots x'_{i'_n}$ entspricht einem kodierten Symbol von Σ. So kann man sehen, dass $i'_1, i'_2, \ldots, i'_n$ auch eine Lösung von K ist.

Aufgabe 2 (3 Punkte)

Sei $\mathcal{M}$ eine nichtdeterministische Turingmaschine mit Eingabealphabet Σ und ausgezeichneten Endzuständen q_J, q_N. Nehmen Sie an, dass für jede natürliche Zahl n eine Eingabe x der Länge n existiert, die von $\mathcal{M}$ akzeptiert wird. Betrachten Sie die folgenden Ideen, die worst-case Laufzeit (a.k.a. Zeitkomplexitätsfunktion) für $\mathcal{M}$ alternativ zu definieren:

$$A_{\mathcal{M}}(n) := \max\left(\left\{ m \;\middle|\; \begin{array}{l}\text{Es gibt eine Eingabe } x \in L_{\mathcal{M}} \text{ mit } |x| = n, \text{ so dass}\\ \mathcal{M} \text{ in höchstens } m \text{ Schritten}\\ \text{in den Zustand } q_J \text{ überführt werden kann.}\end{array} \right\}\right)$$

$$B_{\mathcal{M}}(n) := \min\left(\left\{ m \;\middle|\; \begin{array}{l}\text{Für alle Eingaben } x \in L_{\mathcal{M}} \text{ mit } |x| = n, \text{ gilt dass}\\ \mathcal{M} \text{ nicht in } m \text{ Schritten}\\ \text{in den Zustand } q_J \text{ überführt werden kann.}\end{array} \right\}\right) + 1$$

$$C_{\mathcal{M}}(n) := \max\left(\left\{ m \;\middle|\; \begin{array}{l}\text{Für alle Eingaben } x \in \Sigma^* \text{ mit } |x| = n \text{ gilt, dass}\\ \mathcal{M} \text{ in höchstens } m \text{ Schritten}\\ \text{in den Zustand } q_J \text{ oder } q_N \text{ überführt werden kann.}\end{array} \right\}\right)$$

$$D_{\mathcal{M}}(n) := \min\left(\left\{ m \;\middle|\; \begin{array}{l}\text{Für alle Eingaben } x \in \Sigma^* \text{ mit } |x| = n \text{ gilt, dass}\\ \mathcal{M} \text{ in höchstens } m \text{ Schritten}\\ \text{in den Zustand } q_J \text{ oder } q_N \text{ überführt werden kann.}\end{array} \right\}\right)$$

$$E_{\mathcal{M}}(n) := \max\left(\left\{ m \;\middle|\; \begin{array}{l}\text{Es gibt eine Eingabe } x \in \Sigma^* \text{ mit } |x| = n, \text{ so dass}\\ \mathcal{M} \text{ in höchstens } m \text{ Schritten hält.}\end{array} \right\}\right)$$

$$F_{\mathcal{M}}(n) := \max\left(\left\{ m \;\middle|\; \begin{array}{l}\text{Es gibt eine Eingabe } x \in L_{\mathcal{M}} \text{ mit } |x| = n, \text{ so dass}\\ \mathcal{M} \text{ nicht in } m \text{ Schritten}\\ \text{in den Zustand } q_J \text{ überführt werden kann.}\end{array} \right\}\right) + 1$$

Welche dieser Funktionen $A_{\mathcal{M}}, B_{\mathcal{M}}, C_{\mathcal{M}}, D_{\mathcal{M}}, E_{\mathcal{M}}, F_{\mathcal{M}}$ sind identisch mit der Zeitkomplexitätsfunktion $T_{\mathcal{M}}$ von $\mathcal{M}$ aus der Vorlesung? Begründen Sie!

Lösung:
Zuerst wiederholen wir die Definition der Zeitkomplexitätsfunktion für $\mathcal{M}$ aus der Vorlesung:

$$T_{\mathcal{M}}(n) := \max\left(\left\{ m \;\middle|\; \begin{array}{l}\text{Es gibt ein } x \in L_{\mathcal{M}} \text{ mit } |x| = n, \text{ so dass}\\ \text{die Zeit, die } \mathcal{M} \text{ benötigt,}\\ \text{um } x \text{ zu akzeptieren, } m \text{ ist.}\end{array} \right\}\right)$$

Hierbei ist die Zeit die $\mathcal{M}$ benötigt, um $x \in L_{\mathcal{M}}$ zu akzeptieren, gerade die minimale Anzahl von Schritten, die $\mathcal{M}$ in den Zustand q_J überführt. (Da wir vorraussetzen, dass für jedes n ein $x \in L_{\mathcal{M}}$ mit $|x| = n$ gibt, brauchen wir die 1 in die obige Menge nicht zusätzlich hinzufügen.)

Bezeichne für $Z = A, B, C, D, E, F, T$ mit Z_n die Menge von natürlichen Zahlen aus der Aufgabenstellung und der obigen Definition, über die $Z_{\mathcal{M}}(n)$ das Maximum bzw. Minimum bildet. Zum Beispiel gilt also $A_{\mathcal{M}}(n) = \max(A_n)$, $B_{\mathcal{M}}(n) = \min(B_n) + 1$ und $T_{\mathcal{M}}(n) = \max(T_n)$.

Im Falle von A_n gilt: Falls $m \in A_n$, dann auch $m + 1 \in A_n$. Also ist $A_{\mathcal{M}}(n) = \max(A_n) = \infty$ für alle n und demnach im Allgemeinen $A_{\mathcal{M}}(n) \neq T_{\mathcal{M}}(n)$.

Im Falle von B_n gilt: Falls $m \in B_n$, dann auch $m-1 \in B_n$. Also ist $B_{\mathcal{M}}(n) = \min(B_n) + 1 = 2$ (oder 1 falls $0 \in \mathbb{Z}^+$) für alle n und demnach im Allgemeinen $B_{\mathcal{M}}(n) \neq T_{\mathcal{M}}(n)$.

Im Falle von C_n gilt: Es könnte Eingaben $x \in \Sigma^* \setminus L_{\mathcal{M}}$ mit $|x| = n$ geben, für die $\mathcal{M}$ nicht hält, also weder in den Zustand q_J noch q_N überführt werden kann. Dementsprechend könnte $C_n = \emptyset$ für ein n gelten und dann $C_{\mathcal{M}}(n) = \max(C_n) = -\infty \neq T_{\mathcal{M}}(n)$.

Im Falle von D_n gilt $D_n = C_n$. Entsprechend gilt für $D_n = \emptyset$, dass $D_{\mathcal{M}}(n) = \min(D_n) = \infty \neq T_{\mathcal{M}}(n)$.

Im Falle von E_n gilt: Es könnte Eingaben $x \in \Sigma^* \setminus L_{\mathcal{M}}$ mit $|x| = n$ geben, für die $\mathcal{M}$ mehr als $T_{\mathcal{M}}(n)$ Schritte benötigt, um in q_N überführt zu werden. Also eine spezielle Nein-Instanz der Größe n braucht länger als alle Ja-Instanzen der Größe n. In einem solchen Fall gilt $E_{\mathcal{M}}(n) = \max(E_n) > T_{\mathcal{M}}(n)$.

Im Falle von F_n gilt: Es könnte sein, dass für jede Eingabe $x \in L_{\mathcal{M}}$ es nur genau ein Orakel gibt für das $\mathcal{M}$ Eingabe x akzeptiert. Dann gilt es für jedes feste n, dass $|\mathbb{Z}^+ - F_n| < \infty$ und damit $F_{\mathcal{M}}(n) = \max(F_n) + 1 = \infty \neq T_{\mathcal{M}}(n)$.

Falls F_n so interpretiert wird, dass $\mathcal{M}$ nicht in m oder weniger Schritten in q_J überführt werden kann, so gilt $F_{\mathcal{M}}(n) = T_{\mathcal{M}}(n)$ für alle n. Zunächst ist klar, dass (mit dieser Interpretation) gilt, dass $F_n = \{1, \ldots, m'\}$ für $m' = F_{\mathcal{M}}(n) - 1$. Sei nun $x' \in L_{\mathcal{M}}$ die Eingabe mit $|x'| = n$ für die $\mathcal{M}$ am meisten Zeit benötigt, um sie zu akzeptieren. $\mathcal{M}$ benötigt also $T_{\mathcal{M}}(n)$ Zeit für x' und daher gilt $T_{\mathcal{M}}(n) - 1 \in F_n$. Außerdem benötigt $\mathcal{M}$ für alle Eingaben $x \in L_{\mathcal{M}}$ mit $|x| = n$ höchstens $T_{\mathcal{M}}(n)$ Zeit um x zu akzeptieren und daher gilt $T_{\mathcal{M}}(n) \notin F_n$. Es folgt $F_{\mathcal{M}}(n) = \max(F_n) + 1 = T_{\mathcal{M}}(n) - 1 + 1 = T_{\mathcal{M}}(n)$ für alle n.

Aufgabe 3

(1 + 1 + 1 + 1 = 4 Punkte)

Zeigen Sie:

(a) Das Komplement des Halteproblems ist nicht semi-entscheidbar.

(b) Das Komplement der Diagonalsprache ist semi-entscheidbar.

(c) Seien L_1 und L_2 semi-entscheidbare Sprachen. Dann ist auch $L_1 \cap L_2$ semi-entscheidbar.

(d) Seien L_1 und L_2 semi-entscheidbare Sprachen. Dann ist auch $L_1 \cup L_2$ semi-entscheidbar.

Lösung:

(a) Das Halteproblem ist semi-entscheidbar. Angenommen, das Komplement des Halteproblems wäre ebenfalls semi-entscheidbar. Dann könnte man die beiden entsprechenden Turingmaschinen „pseudoparallel“ laufen lassen. Da die beiden Turingmaschinen komplementäre Sprachen akzeptieren hält mindestens eine von beiden nach endlicher Zeit. Damit wäre das Halteproblem entscheidbar. Widerspruch.

(b) Das Komplement der Diagonalsprache ist $L_d^c = \{w_i \mid M_i \text{ akzeptiert } w_i\}$. Ist $w_i \in L_d^c$ kann dies durch Simulation von M_i auf der Eingabe w_i in endlicher Zeit festgestellt werden. Damit ist L_d^c semi-entscheidbar.

(c) Seien L_1 und L_2 semi-entscheidbare Sprachen und M_1 und M_2 Turingmaschinen, die L_1 bzw. L_2 akzeptieren. Konstruiere eine Turingmaschine M, die $L_1 \cap L_2$ akzeptiert. Dazu wird bei Eingabe von w zunächst M_1 mit Eingabe w simuliert. Falls $w \in L_1$ wird M_1 nach endlicher Zeit akzeptieren. Dann wird M_2 mit Eingabe w simuliert. Falls $w \in L_2$ wird M_2 nach endlicher Zeit akzeptieren. Damit akzeptiert M die Eingabe w nach endlicher Zeit, insofern $w \in L_1 \cap L_2$ gilt.

(d) Seien L_1 und L_2 semi-entscheidbare Sprachen und M_1 und M_2 Turingmaschinen, die L_1 bzw. L_2 akzeptieren. Konstruiere eine Turingmaschine M, die $L_1 \cup L_2$ akzeptiert. Dazu werden bei Eingabe von w die Maschinen M_1 und M_2 „pseudoparallel“ jeweils mit Eingabe w simuliert. Das funktioniert, indem abwechselnd jeweils ein Schritt der beiden Simulationen ausgeführt wird. Akzeptiert eine der beiden Simulationen die Eingabe akzeptiert auch M. Gilt $w \in L_1$ oder $w \in L_2$ terminiert mindestens eine der beiden Simulationen nach endlicher Zeit, so dass auch M nach endlicher Zeit akzeptiert. Damit akzeptiert M die Sprache $L_1 \cup L_2$.

Hier ist es wichtig, dass die Simulationen nicht einfach hintereinander ausgeführt werden. Dann wäre es nämlich möglich, dass die erste Simulation nicht terminiert, wodurch die zweite Simulation nie feststellen kann, dass die Eingabe zur gesuchten Sprache gehört.

Aufgabe 4

(2 + 3 = 5 Punkte)

Eine Turingmaschine M zählt eine Sprache L auf, wenn M niemals stoppt und eine Liste $w_1, w_2, \ldots$ genau der Wörter aus L ausgibt. Dabei ignoriert M die Eingabe und die Wörter der ausgegebenen Liste sind eindeutig voneinander getrennt. Die Wörter aus L dürfen in beliebiger Reihenfolge in der Liste vorkommen. Eine Sprache L ist aufzählbar falls eine Turingmaschine existiert, die L aufzählt.

(a) Zeigen Sie, dass L in aufsteigender Reihenfolge aufzählbar ist, wenn L entscheidbar ist.

(b) Zeigen Sie, dass L aufzählbar ist, wenn L semi-entscheidbar ist. Erklären Sie auch, wieso die Worte in beliebiger Reihenfolge vorkommen können.

Lösung:

(a) Sei L eine aufzählbare Sprache zusammen mit einer aufzählenden Turingmaschine M. Konstruiere eine Turingmaschine M', die L semi-entscheidet. Sei w eine Eingabe für M'. Simuliere die Turingmaschine M. Gibt M das Wort w aus, stoppt M' und akzeptiert w. Andernfalls wird M weiter simuliert. Falls $w \in L$ ist wird M' das Wort w nach endlicher Zeit akzeptieren. Damit ist L eine semi-entscheidbare Sprache.

Sei umgekehrt L eine semi-entscheidbare Sprache mit einer semi-entscheidenden Turingmaschine M. Konstruiere eine Turingmaschine M', die die Sprache L aufzählt. Dazu simuliert M' die Turingmaschine M „pseudoparallel“ für alle Wörter $w_1, w_2, \ldots, \ldots$ in aufsteigender Reihenfolge. Das funktioniert folgendermaßen. In Schritt 1 simuliert M' einen Schritt von M auf der Eingabe w_1. In Schritt 2 simuliert M' einen (weiteren) Schritt von M auf den Eingaben w_1, w_2. In Schritt 3 simuliert M' einen (weiteren) Schritt von M auf den Eingaben w_1, w_2, w_3, und so weiter. Sobald M' eine Eingabe w akzeptiert schreibt M' das Wort w auf das Ausgabeband. Da L semi-entscheidbar ist wird jedes Wort $w \in L$ irgendwann auf das Ausgabeband geschrieben. Also zählt M' die Sprache L auf. Betrachte zwei Wörter $w' > w$. Die Laufzeit von M für die Eingaben w' und w kann sehr unterschiedlich sein, weshalb die Reihenfolge von w' und w in der Ausgabe a priori nicht bekannt ist.

Man bemerke, dass es hier wichtig ist, die Simulationen „pseudoparallel“ ablaufen zu lassen, damit eine nicht-terminierende Simulation andere, terminierende Simulationen nicht aufhält.

(b) Sei L eine aufzählbare Sprache zusammen mit einer aufzählenden Turingmaschine M. Konstruiere eine Turingmaschine M', die entscheidet, ob das Wort w zu L gehört. Dazu simuliert M' die Turingmaschine M solange, bis das erste Wort $w' \geq w$ aufgezählt wird. Da M die Sprache L aufzählt geschieht dies nach endlicher Zeit. Gilt $w' = w$ folgt $w \in L$, andernfalls gilt $w \notin L$. Die Turingmaschine M' entscheidet also L.

Sei umgekehrt L eine entscheidbare Sprache mit einer entscheidenden Turingmaschine M. Konstruiere eine Turingmaschine M', die die Sprache L in aufsteigender Reihenfolge aufzählt. Dazu schreibt M' in aufsteigender Reihenfolge alle Worte in Σ^* auf das Arbeitsband. Für jedes Wort w wird dann M mit w als Eingabe simuliert. Da L entscheidbar ist, stoppt M in jedem Fall. Wird w von M akzeptiert, schreibe w auf das Ausgabeband.

Aufgabe 5

(2 + 2 = 4 Punkte)

In der Übung[1] wurde Cantors zweites Diagonalargument vorgestellt. Es wurde bewiesen, dass die Teilmenge $(0, 1)$ der reellen Zahlen überabzählbar ist. In dieser Aufgabe geht es darum, das Argument in angepasster Form zu verwenden, um ganz einfach zu zeigen, dass nicht alle Sprachen entscheidbar sind.

(a) Sei A eine nichtendliche, abzählbare Menge. Zeigen Sie, dass die Potenzmenge 2^A überabzählbar ist. Gehen Sie dabei analog zur Übung vor, d.h. geben Sie eine Funktion f und eine geeignete beispielhafte Tabelle an, und konstruieren Sie ein Element $x \in 2^A \setminus \text{Bild}(f)$.

(b) Nutzen Sie Teilaufgabe (a), um zu zeigen, dass nicht alle Sprachen entscheidbar sind.

Lösung:

(a) Angenommen, 2^A wäre abzählbar mit Bijektion $f : \mathbb{N} \to 2^A$. Bezeichne $1, 2, \ldots$ die Elemente von A. Interpretiere $f(i)$ als charakteristische Funktion, d.h. es ist $f(i)_j = 1$ falls $j \in f(i)$ und sonst $f(i)_j = 0$. Konstruiere die charakteristische Funktion des Elements x folgendermaßen. Setze $x_i = 0$ falls $f(i)_i = 1$ und $x_i = 1$ falls $f(i)_i = 0$. Damit gilt $x \neq f(i)$ für jedes i und damit $x \notin \text{Bild}(f)$. Dies ist einer Widerspruch dazu, dass f eine Bijektion ist. Damit ist 2^A überabzählbar.

	1	2	3	4	5	...	
$f(1) =$	1	0	0	0	1	...	$f(1) = \{1, 5, \ldots\}$
$f(2) =$	0	1	1	1	1	...	$f(2) = \{2, 3, 4, 5, \ldots\}$
$f(3) =$	0	1	0	0	0	...	$f(3) = \{2, \ldots\}$
$f(4) =$	0	0	1	0	0	...	$f(4) = \{3, \ldots\}$
$f(5) =$	1	0	1	1	1	...	$f(5) = \{1, 3, 4, 5, \ldots\}$
$x =$	0	0	1	1	0	...	$x = \{3, 4, \ldots\}$

(b) Sei Σ ein festes endliches Alphabet. Dann ist Σ^* eine abzählbare Menge. Die Menge aller Sprachen ist die Potenzmenge 2^{Σ^*} von Σ^*. Man erinnere sich, dass die Menge aller Turingmaschinen abzählbar ist, etwa mit Bijektion $\langle\cdot\rangle$ der Gödelnummerkodierung. Angenommen,

[1] zweite Übung, Folie 15

alle Sprachen wären entscheidbar, dann gäbe es eine Bijektion f zwischen Turingmaschinen und Sprachen. Dann wäre die geeignete Verknüpfung von f und $\langle\cdot\rangle$ eine Bijektion zwischen den natürlichen Zahlen und der Menge alle Sprachen. Dies ist ein Widerspruch zu (a). Also können nicht alle Sprachen entscheidbar sein.

Aufgabe 6

(2 + 2 = 4 Punkte)

In der Vorlesung hatten wir gesehen, dass die endliche Automaten *genau* die regulären Sprachen akzeptieren können. Bei Turingmaschinen ist die Situation leider nicht so einfach. Neben den entscheidbaren Sprachen gibt es noch die semi-entscheidbaren und die nicht-entscheidbaren Sprachen. Wäre es nicht schön, wenn wir ein Berechnungsmodell hätten, das *genau* die entscheidbaren Sprachen akzeptiert?

Der ebenso optimistische wie unüberlegte Superwissenschaftler Doktor Meta ist schon dabei, eine solche Maschine, die sogenannte *Metamagiemaschine* (M^3), zu konstruieren – reichlich Gelder von seriösen Investoren hat er bereits eingestrichen[2]. Sie sind dazu angestellt, die letzten Details der M^3 auszuarbeiten. Die M^3 soll folgenden Anforderungen genügen:

- Jede entscheidbare Sprache wird von einer M^3 akzeptiert, und jede M^3 akzeptiert eine entscheidbare Sprache.
- Jede M^3 stoppt bei jeder Eingabe nach endlicher Zeit.
- Zu jeder M^3 M lässt sich eine endliche Beschreibung $\langle M\rangle$ finden.
- Die M^3 ist ein effektives Berechnungsmodell. Das heißt, die Church-Turing-These besagt, dass Turingmaschinen mindestens so mächtig sind wie die M^3. Anders gesagt: es gibt eine Turingmaschine, die bei Eingabe einer kodierten M^3 $\langle M\rangle$ und einem Wort w die Berechnung von M mit Eingabe w simuliert.

So eine Maschine kann es leider nicht geben. Beweisen Sie das in folgenden zwei Schritten. Geben Sie immer explizit an, wie Sie die oben geforderten Eigenschaften nutzen.

(a) Zeigen Sie, dass die Sprache $L = \{\langle M\rangle \mid M$ ist eine M^3, die $\langle M\rangle$ ablehnt$\}$ entscheidbar ist.

(b) Zeigen Sie, dass die M^3 nicht existieren kann.

Lösung:

(a) Eine M^3 kann gemäß der Forderungen endlich beschrieben werden. Ferner kann eine M^3 durch eine Turingmaschine simuliert werden. Es existiert also eine Turingmaschine T, die bei Eingabe einer Kodierung $\langle M\rangle$ einer M^3 M diese auf der Eingabe $\langle M\rangle$ simuliert. Da M wie gefordert in jedem Fall nach endlicher Zeit stoppt, stoppt auch T nach endlicher Zeit. Durch Übernehmen des Akzeptanzverhaltens von M entscheidet dann T die Sprache L. Damit ist L entscheidbar.

[2] `https://medium.com/the-mission/d7908013f8c9`

(b) Aus (a) ist bekannt, dass L eine entscheidbare Sprache ist. Es wurde gefordert, dass jede entscheidbare Sprache von einer M^3 akzeptiert wird. Sei M also eine M^3, die L akzeptiert. Es wurde ebenso gefordert, dass jede M^3 endlich beschrieben werden kann. Betrachte also das Wort $\langle M \rangle$. Ist $\langle M \rangle \in L$, so muss M die Eingabe $\langle M \rangle$ akzeptieren. Nach Definition von L bedeutet $\langle M \rangle in L$ aber genau, dass M die Eingabe $\langle M \rangle$ ablehnt. Dies ist ein Widerspruch zu der Forderung, dass jede entscheidbare Sprache von einer M^3 erkannt wird.

Aufgabe 7

(2+1+3+2 = 8 Punkte)

Ziel dieser Aufgabe ist es, den Satz von Rice zu beweisen. Dazu gehen Sie nach folgenden Schritten vor.

(a) Betrachten Sie das modifizierte Halteproblem $\mathcal{H}_\varepsilon = \{w \in \{0,1\}^* \mid T_w \text{ hält bei Eingabe } \varepsilon\}$ und zeigen Sie, dass $\mathcal{H}_\varepsilon$ nicht entscheidbar ist.

(b) Betrachten Sie die berechenbare Funktion f_∞ für die gilt dass $f_\infty(v)$ undefiniert ist für jedes $v \in \{0,1\}^*$, und zeigen Sie dass o.B.d.A. $f_\infty \notin S$.

(c) Betrachten Sie für jede berechenbare Funktion f eine Turingmaschine $\mathcal{M}_f$ die f realisiert. Zeigen Sie dass eine Turingmaschine $\mathcal{M}^*$ berechnet[3] werden kann, die als Eingabe zwei (als Gödelnummer kodierte) Turingmaschinen $\mathcal{M}_1$, $\mathcal{M}_2$, sowie ein Wort $v \in \{0,1\}^*$ erhält und $\mathcal{M}_1$ mit Eingabe ε, gefolgt von $\mathcal{M}_2$ mit Eingabe v simuliert.

(d) Reduzieren Sie schliesslich das modifizierte Halteproblem $\mathcal{H}_\varepsilon$ auf die Sprache $L(S)$ aus dem Satz von Rice. Zu diesem Zweck konstruieren Sie eine Turingmaschine die $\mathcal{H}_\varepsilon$ entscheidet aus einer hypothetischen Turingmaschine $M_{L(S)}$ die S entscheidet. Benutzen Sie dabei die Turingmaschine $\mathcal{M}^*$ mit Eingabe $\mathcal{M}_{f_\infty}$, $\mathcal{M}_{f^*}$, $v \in \{0,1\}^*$ für ein beliebig aber festes $f^* \in S$.

Lösung:

(a) Angenommen $\mathcal{H}_\varepsilon$ sei entscheidbar, entschieden durch eine Turing-Maschine $\mathcal{M}_\varepsilon$. Dann konstruiere eine Turing-Maschine $\mathcal{M}_H$ für das allgemeine Halteproblem $\mathcal{H}$ wie folgt:

- Bei Eingabe $(\langle \mathcal{M} \rangle, w)$ berechne die Gödelnummer einer Turing-Maschine $\tilde{\mathcal{M}}$ die bei leerer Eingabe $\mathcal{M}$ mit Eingabe w simuliert.
- Benutze $\mathcal{M}_\varepsilon$ mit Eingabe $\langle \tilde{M} \rangle$ und akzeptiere genau dann, wenn $\mathcal{M}_\varepsilon$ akzeptiert.

Es ist nun klar, dass

$$\mathcal{M}_H \text{ akzeptiert } (\langle \mathcal{M} \rangle, w) \Leftrightarrow \mathcal{M}_\varepsilon \text{ akzeptiert } \langle \tilde{M} \rangle \Leftrightarrow \tilde{\mathcal{M}} \text{ hält bei } \varepsilon \Leftrightarrow \mathcal{M} \text{ hält bei } w.$$

Somit würde $\mathcal{M}_H$ die Sprache $\mathcal{H}$ entscheiden – ein Widerspruch.

(b) Sei f_∞ die berechenbare Funktion mit $f_\infty(v)$ undefiniert für jedes $v \in \{0,1\}^*$. Diese partielle Funktion ist berechenbar da man leicht eine Turing-Maschine konstruieren kann, die niemals hält. Wir wollen zeigen, dass $f_\infty \notin S$. Andernfalls, also wenn $f_\infty \in S$, dann führen wir den Beweis mit $\mathcal{R} \setminus S$, wobei $\mathcal{R}$ die Menge aller berechenbarer Funktionen bezeichne. Dies würde zeigen, dass die Sprache $L(\mathcal{R} \setminus S)$ nicht entscheidbar ist. Da $L(S) = L(\mathcal{R} \setminus S)^c$ folgt die Unentscheidbarkeit von $L(S)$ aus der Abgeschlossenheit von entscheidbaren Sprachen unter Komplementbildung.

[3] D.h. es gibt eine Turingmaschine die (unabhängig von der Eingabe) M^* (als Gödelnummer) ausgibt.

(c) Wir wissen, dass eine universelle Turing-Maschine bei Eingabe $(\langle \mathcal{M} \rangle, w)$ die Turing-Maschine $\mathcal{M}$ bei Eingabe w simuliert. Sei U eine solche universelle Turing-Maschine. Wir konstruieren nun die folgende Turing-Maschine $\mathcal{M}^*$, die als Eingabe ein Tripel $(\langle \mathcal{M}_1 \rangle, \langle \mathcal{M}_2 \rangle, v)$ bekommt und folgendes tut:

- Prüfe, ob die ersten zwei Einträge der Eingabe zwei Gödelnummern sind.
- Simuliere U mit Eingabe $(\langle \mathcal{M}_1 \rangle, \varepsilon)$ auf einer für diesen Zweck reservierten Spur.
- Simuliere U mit Eingabe $(\langle \mathcal{M}_2 \rangle, v)$ auf einer für diesen Zweck reservierten Spur.

Da die Zustände, Übergänge, etc. von $\mathcal{M}^*$ nur von U abhängen, und nicht etwa von $\mathcal{M}_1$, $\mathcal{M}_2$ oder v, können wir die Gödelnummer von $\mathcal{M}^*$ vorher berechnen und durch eine andere Turing-Maschine ausgeben lassen.

(d) Angenommen $\mathcal{M}_{L(S)}$ sei eine Turing-Maschine die $L(S)$ entscheidet. Sei f^* eine beliebige Funktion in S. Wir konstruieren daraus eine Turing-Maschine $\mathcal{M}_\varepsilon$ die $\mathcal{H}_\varepsilon$ entscheidet wie folgt:

- Prüfe, ob die Eingabe $\langle \mathcal{M} \rangle$ eine Gödelnummer ist.
- Berechne die Gödelnummer $\langle \mathcal{M}^* \rangle$ der Turing-Maschine $\mathcal{M}^*$ aus Teil (c) mit $\mathcal{M}_1 = \mathcal{M}$ und $\mathcal{M}_2 = \mathcal{M}_{f^*}$.
- Starte $\mathcal{M}_{L(S)}$ mit der Eingabe $\langle \mathcal{M}^* \rangle$ und akzeptiere genau dann, wenn $\mathcal{M}_{L(S)}$ akzeptiert.

Nun bleibt zu prüfen, dass $\mathcal{M}_\varepsilon$ tatsächlich das spezielle Halteproblem $\mathcal{H}_\varepsilon$ entscheidet. Bei Eingabe von $\langle \mathcal{M} \rangle$ gibt es zwei Fälle zu unterscheiden:

$$\begin{aligned}
\langle \mathcal{M} \rangle \in \mathcal{H}_\varepsilon &\Rightarrow \mathcal{M} \text{ hält bei Eingabe } \varepsilon \\
&\Rightarrow \mathcal{M}^* \text{ berechnet } f^* \\
&\Rightarrow \text{da } f^* \in S, \text{ gilt } \langle \mathcal{M}^* \rangle \in L(S) \\
&\Rightarrow \mathcal{M}_{L(S)} \text{ akzeptiert bei Eingabe } \langle \mathcal{M}^* \rangle \\
&\Rightarrow \mathcal{M}_\varepsilon \text{ akzeptiert die Eingabe } \langle \mathcal{M} \rangle
\end{aligned}$$

$$\begin{aligned}
\langle \mathcal{M} \rangle \notin \mathcal{H}_\varepsilon &\Rightarrow \mathcal{M} \text{ hält nicht bei Eingabe } \varepsilon \\
&\Rightarrow \mathcal{M}^* \text{ berechnet } f_\infty \\
&\Rightarrow \text{da } f_\infty \notin S, \text{ gilt } \langle \mathcal{M}^* \rangle \notin L(S) \\
&\Rightarrow \mathcal{M}_{L(S)} \text{ verwirft bei Eingabe } \langle \mathcal{M}^* \rangle \\
&\Rightarrow \mathcal{M}_\varepsilon \text{ verwirft die Eingabe } \langle \mathcal{M} \rangle
\end{aligned}$$