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

Übungsblatt 3

Vorlesung Theoretische Grundlagen der Informatik im WS 18/19

Ausgabe 20. November 2018
Abgabe 4. Dezember 2018, 11:00 Uhr (im Kasten im UG von Gebäude 50.34)

Aufgabe 1

(4 + 2 = 6 Punkte)

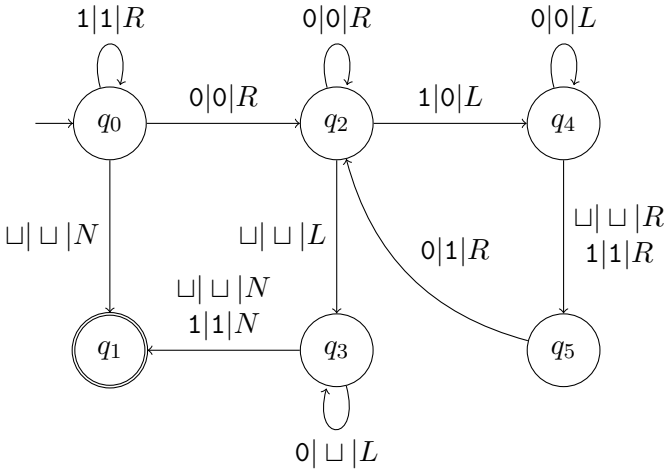
Gegeben sei das Alphabet $\Sigma = \{\texttt{0}, \texttt{1}\}$.

(a) Geben Sie den Zustandsgraphen einer deterministischen Turingmaschine an, die für eine Eingabe $w \in \Sigma^\star$ alle Vorkommen von `0` in w löscht. Die Eingabe `1101001` wird also z.B. zu `1111` transformiert.

(b) Geben Sie die Konfigurationsfolge an, die bei Eingabe des Wortes `01101` durchlaufen wird.

Lösung:

(a)



(b) $(q_0)\texttt{01101} \to \texttt{0}(q_2)\texttt{1101} \to (q_4)\texttt{00101} \to (q_4)\,\sqcup\,\texttt{00101} \to (q_5)\texttt{00101} \to \texttt{1}(q_2)\texttt{0101} \to \texttt{10}(q_2)\texttt{101} \to \texttt{1}(q_4)\texttt{0001} \to (q_4)\texttt{10001} \to \texttt{1}(q_5)\texttt{0001} \to \texttt{11}(q_2)\texttt{001} \to \texttt{110}(q_2)\texttt{01} \to \texttt{1100}(q_2)\texttt{1} \to \texttt{110}(q_4)\texttt{00} \to \texttt{11}(q_4)\texttt{000} \to \texttt{1}(q_4)\texttt{1000} \to \texttt{11}(q_5)\texttt{000} \to \texttt{111}(q_2)\texttt{00} \to \texttt{1110}(q_2)\texttt{0} \to \texttt{11100}(q_2) \to \texttt{1110}(q_3)\texttt{0} \to \texttt{111}(q_3)\texttt{0} \to \texttt{11}(q_3)\texttt{1} \to \texttt{11}(q_1)\texttt{1}$

Aufgabe 2

(1 + 1 + 1 = 3 Punkte)

Betrachten Sie die folgenden Entscheidungsprobleme:

(a) Gegeben eine natürliche Zahl n, ist n *keine* Primzahl?

(b) Gegeben eine Menge $M = \{m_1, \ldots, m_k\}$ von natürlichen Zahlen, gibt es eine Teilmenge $M' \subseteq M$, deren Summe genau 100 ist?

(c) Gegeben ein ungerichteter Graph $G = (V, E)$, gibt es einen Pfad, der jeden Knoten in V genau einmal besucht?

Geben Sie für jedes dieser Probleme eine Ja-Instanz und eine Nein-Instanz an. Beschreiben Sie außerdem, wie eine NTM arbeitet, die das Problem in polynomieller Zeit löst. Geben Sie dazu insbesondere an, wie die NTM den Lösungsvorschlag des Orakelmoduls interpretiert und was der deterministische Teil der NTM tun muss, um den Lösungsvorschlag zu überprüfen.

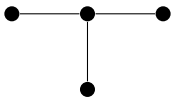
Lösung:

(a)
- Ja-Instanz: 4
- Nein-Instanz: 3
- Lösungsvorschlag: Zwei Faktoren $n_1, n_2 \in \mathbb{N} \setminus \{1\}$
- Arbeitsweise: Die NTM überprüft, ob $n_1 \cdot n_2 = n$ gilt.

(b)
- Ja-Instanz: $\{30, 70, 20\}$
- Nein-Instanz: $\{30, 60, 20\}$
- Lösungsvorschlag: Für jedes $m \in M$ wird angegeben, ob $m \in M'$. Das ist z.B. durch ein Binärwort w der Länge $|M|$ möglich, wobei w_i genau dann 1 ist, wenn $m \in M'$.
- Die NTM überprüft, ob der Lösungsvorschlag das korrekte Format hat. Wenn ja, addiert sie alle $m \in M'$ und überprüft, ob die Summe 100 ist.

(c)
- Ja-Instanz:

- Nein-Instanz:



- Lösungsvorschlag: Eine Permutation $(v'_1, \ldots, v'_k)$ der Knoten aus V.
- Die NTM überprüft, ob jeder Knoten genau einmal in der Permutation vorkommt und ob für $1 \leq i < k$ jeweils eine Kante zwischen v'_i und v'_{i+1} existiert.

Aufgabe 3 (3 Punkte)

Sei $\mathcal{M}$ eine nichtdeterministische Turingmaschine mit Eingabealphabet Σ. 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 Zeitkomplexitätsfunktion für $\mathcal{M}$ alternativ zu definieren:

(a)

$$A_{\mathcal{M}}(n) := \min\left(\left\{ m \;\middle|\; \begin{array}{l}\text{Es gibt ein } x \in L_{\mathcal{M}} \text{ mit } |x| = n,\\ \text{sodass alle akzeptierenden Berechnungen}\\ \text{höchstens } m \text{ Schritte brauchen.}\end{array} \right\}\right)$$

(b)

$$B_{\mathcal{M}}(n) := \max\left(\left\{ m \;\middle|\; \begin{array}{l}\text{Es gibt ein } x \in L_{\mathcal{M}} \text{ mit } |x| = n,\\ \text{sodass alle akzeptierenden Berechnungen}\\ \text{mindestens } m \text{ Schritte brauchen.}\end{array} \right\}\right)$$

(c)

$$C_{\mathcal{M}}(n) := \max\left(\left\{ m \;\middle|\; \begin{array}{l}\text{Für alle } x \in L_{\mathcal{M}} \text{ mit } |x| = n\\ \text{gibt es eine akzeptierende Berechnung,}\\ \text{die höchstens } m \text{ Schritte braucht.}\end{array} \right\}\right)$$

(d)

$$D_{\mathcal{M}}(n) := \min\left(\left\{ m \;\middle|\; \begin{array}{l}\text{Für alle } x \in L_{\mathcal{M}} \text{ mit } |x| = n\\ \text{gibt es eine akzeptierende Berechnung,}\\ \text{die höchstens } m \text{ Schritte braucht.}\end{array} \right\}\right)$$

(e)

$$E_{\mathcal{M}}(n) := \max\left(\left\{ m \;\middle|\; \begin{array}{l}\text{Es gibt ein } x \in L_{\mathcal{M}} \text{ mit } |x| = n,\\ \text{für das es eine akzeptierende Berechnung gibt,}\\ \text{die genau } m \text{ Schritte braucht.}\end{array} \right\}\right)$$

(f)

$$F_{\mathcal{M}}(n) := \max\left(\left\{ m \;\middle|\; \begin{array}{l}\text{Es gibt ein } x \in \Sigma^\star \text{ mit } |x| = n,\\ \text{sodass alle akzeptierenden Berechnungen}\\ \text{mindestens } m \text{ Schritte brauchen.}\end{array} \right\}\right)$$

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. Für ein $x \in L_{\mathcal{M}}$ bezeichnen wir die Anzahl der Schritte in der kürzesten akzeptierenden Berechnung mit $t(x)$. Damit gilt:

$$\begin{aligned} T_{\mathcal{M}}(n) &= \max\left(\left\{ m \;\middle|\; \begin{array}{l}\text{Es gibt ein } x \in L_{\mathcal{M}} \text{ mit } |x| = n,\\ \text{für das } t(x) = m \text{ gilt.}\end{array} \right\}\right)\\ &= \max\left(\left\{ t(x) \;\middle|\; x \in L_{\mathcal{M}} \text{ mit } |x| = n \right\}\right)\end{aligned}$$

Da wir voraussetzen, dass es 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 $A_{\mathcal{M}}(n) = \min(A_n)$, $B_{\mathcal{M}}(n) = \max(B_n)$ und $T_{\mathcal{M}}(n) = \max(T_n)$.

(a) Es gilt im Allgemeinen $A_{\mathcal{M}}(n) \neq T_{\mathcal{M}}(n)$. Betrachte z.B. eine NTM $\mathcal{M}$, bei der es für jedes Wort der Länge n genau zwei akzeptierende Berechnungen gibt, wobei eine n Schritte braucht und die andere $n+1$. Dann ist $A_{\mathcal{M}}(n) = n + 1$, aber $T_{\mathcal{M}}(n) = n$.

(b) Die Bedingung, dass alle akzeptierenden Berechnungen mindestens m Schritte brauchen, ist äquivalent dazu, dass die kürzeste akzeptierende Berechnung mindestens m Schritte braucht. Es gilt also:

$$
\begin{aligned}
B_{\mathcal{M}}(n) &= \max\left(\left\{m \;\middle|\; \begin{array}{l}\text{Es gibt ein } x \in L_{\mathcal{M}} \text{ mit } |x| = n, \\ \text{für das } t(x) \geq m \text{ gilt.}\end{array}\right\}\right) \\
&= \max\left(\left\{m \;\middle|\; \max\left(\left\{t(x) \;\middle|\; x \in L_{\mathcal{M}} \text{ mit } |x| = n\right\}\right) \geq m\right\}\right) \\
&= \max\left(\left\{m \;\middle|\; T_{\mathcal{M}}(n) \geq m\right\}\right) \\
&= T_{\mathcal{M}}(n)
\end{aligned}
$$

(c) Falls $m \in C_n$ gilt, dann auch $m + 1 \in C_n$. Also ist $C_{\mathcal{M}}(n) = \max(C_n) = \infty$ für alle n und demnach im Allgemeinen $C_{\mathcal{M}}(n) \neq A_{\mathcal{M}}(n)$.

(d) Die Bedingung, dass es eine akzeptierende Berechnung gibt, die höchstens m Schritte braucht, ist äquivalent dazu, dass die kürzeste akzeptierende Berechnung höchstens m Schritte braucht. Es gilt also:

$$
\begin{aligned}
D_{\mathcal{M}}(n) &= \min\left(\left\{m \;\middle|\; \begin{array}{l}\text{Für alle } x \in L_{\mathcal{M}} \text{ mit } |x| = n \\ \text{gilt } t(x) \leq m.\end{array}\right\}\right) \\
&= \min\left(\left\{m \;\middle|\; \max\left(\left\{t(x) \;\middle|\; x \in L_{\mathcal{M}} \text{ mit } |x| = n\right\}\right) \leq m\right\}\right) \\
&= \min\left(\left\{m \;\middle|\; T_{\mathcal{M}}(n) \leq m\right\}\right) \\
&= T_{\mathcal{M}}(n)
\end{aligned}
$$

(e) Es gilt im Allgemeinen $E_{\mathcal{M}}(n) \neq T_{\mathcal{M}}(n)$. Dies lässt sich mit demselben Beispiel wie für $A_{\mathcal{M}}(n)$ zeigen.

(f) Für Eingaben $x \in L_{\mathcal{M}}^c$ mit $|x| = n$ gibt es keine akzeptierenden Berechnungen, also gilt für jedes m, dass alle akzeptierenden Berechnungen mindestens m Schritte brauchen. Wenn also $L_{\mathcal{M}}^c$ nicht leer ist, ist $F_{\mathcal{M}}(n) = \max(F_n) = \infty$ für alle n und demnach im Allgemeinen $F_{\mathcal{M}}(n) \neq T_{\mathcal{M}}(n)$.

Aufgabe 4

(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, also abwechselnd jeweils einen Schritt der entsprechenden Turingmaschine simulieren. 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 $w \in L_2$, so akzeptiert mindestens eine der beiden Simulationen nach endlicher Zeit, sodass 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 5

(2 + 3 = 5 Punkte)

Eine Turingmaschine M zählt eine unendliche 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. Für die Reihenfolge der Wörter in der Liste muss gelten, dass jedes Wort aus L nach endlich vielen Schritten ausgegeben wird. Eine unendliche Sprache L ist aufzählbar, falls eine Turingmaschine existiert, die L aufzählt.

(a) Zeigen Sie, dass L genau dann entscheidbar ist, wenn L in kanonischer Reihenfolge[1] aufzählbar ist.

(b) Zeigen Sie, dass L genau dann semi-entscheidbar ist, wenn L aufzählbar ist. Erklären Sie auch, warum sich hier im Gegensatz zu Aufgabenteil (a) nicht fordern lässt, dass die Reihenfolge der Aufzählung kanonisch ist.

Lösung:

(a) Sei L eine aufzählbare Sprache zusammen mit einer Turingmaschine M, die L in kanonischer Reihenfolge aufzählt. Konstruiere eine Turingmaschine M', die für jedes w entscheidet, ob es 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 $w' = w$, so 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' mit separatem Arbeits- und Ausgabeband, die die Sprache L in kanonischer Reihenfolge aufzählt. Dazu schreibt M' in kanonischer Reihenfolge alle

[1] Siehe Vorlesung 6, Folie 19. Die kanonische Reihenfolge sortiert Wörter nach aufsteigender Länge und Wörter der gleichen Länge lexikographisch.

Wörter in $\Sigma^\star$ 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.

(b) 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 solange, bis w ausgegeben wird, und akzeptiere dann. 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 $w_1, w_2, \ldots$ in kanonischer 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. Man bemerke, dass es hier wichtig ist, die Simulationen „pseudoparallel“ ablaufen zu lassen, damit eine nicht-terminierende Simulation andere, terminierende Simulationen nicht aufhält.

Betrachte zwei Wörter $w' > w$. Die Laufzeit von M für die Eingaben w' und w kann sehr unterschiedlich sein, weshalb es vorkommen kann, dass w' vor w ausgegeben wird. Daher ist die Reihenfolge der Ausgabe nicht unbedingt kanonisch.

Aufgabe 6 (2 Punkte)

Zeigen Sie, dass die Sprache $\mathcal{H}_{\text{all}} = \{w \mid T_w \text{ hält auf allen Eingaben}\}$ nicht entscheidbar ist. Benutzen Sie dafür nicht den Satz von Rice!

Lösung:
Angenommen, es gibt eine Turingmaschine M_{all}, die $\mathcal{H}_{\text{all}}$ entscheidet. Wir konstruieren nun eine Turingmaschine M_{halt}, die das Halteproblem entscheidet. Da das Halteproblem unentscheidbar ist, ergibt sich ein Widerspruch.

- Die Eingabe von M_{halt} sind eine Turingmaschine M und ein Wort w. M_{halt} muss akzeptieren, falls M bei Eingabe von w hält, und sonst ablehnen.
- Konstruiere eine Turingmaschine X, die ihre Eingabe verwirft und M auf w simuliert.
- Simuliere M_{all} mit der Eingabe $\langle X \rangle$. Akzeptiere, falls M_{all} akzeptiert. Sonst lehne ab.

M_{all} entscheidet, ob X auf allen Eingaben hält. Das ist genau dann der Fall, wenn M auf w hält. Also entscheidet M_{halt} das Halteproblem.

Aufgabe 7 (1 + 2 + 1 = 4 Punkte)

In der Übung wurde das Äquivalenzproblem für Turingmaschinen vorgestellt:

$$L_{\text{äq}} = \{w\#v \in \{0,1,\#\}^\star \mid L(T_w) = L(T_v)\}$$

In dieser Aufgabe sollen Sie zeigen, dass weder $L_{\text{äq}}$ noch $L_{\text{äq}}^c$ semi-entscheidbar ist. Gehen Sie dazu wie folgt vor:

(a) Zeigen Sie, dass das Komplement der universellen Sprache L_u^c nicht semi-entscheidbar ist.

(b) Zeigen Sie, dass $L_{\text{äq}}$ nicht semi-entscheidbar ist. Nehmen Sie dazu an, es gäbe eine Turingmaschine $M_{\text{äq}}$, die $L_{\text{äq}}$ akzeptiert. Konstruieren Sie daraus eine Turingmaschine M_u^c, die L_u^c akzeptiert.

(c) Zeigen Sie mit der gleichen Herangehensweise wie in Aufgabenteil (b), dass $L_{\text{äq}}^c$ nicht semi-entscheidbar ist.

Lösung:

(a) In der Vorlesung wurde gezeigt, dass L_u nicht entscheidbar, aber semi-entscheidbar ist. Außerdem wurde in der Übung gezeigt, dass eine Sprache L genau dann entscheidbar ist, wenn L und L^c semi-entscheidbar sind. Wäre L_u^c also entscheidbar, wäre L_u entscheidbar, was zum Widerspruch führt.

(b) Wir konstruieren eine Turingmaschine M_u^c, die L_u^c akzeptiert, wie folgt:

- Die Eingabe von M_u^c sind eine Turingmaschine M und ein Wort w. M_u^c muss genau dann akzeptieren, falls w nicht von M akzeptiert wird.
- Konstruiere eine Turingmaschine M_1, die ihre Eingabe ignoriert und stattdessen M auf w simuliert.
- Konstruiere eine Turingmaschine M_2, die alle Eingaben ablehnt.
- Simuliere $M_{\text{äq}}$ auf der Eingabe $\langle M_1\rangle\#\langle M_2\rangle$. Falls $M_{\text{äq}}$ akzeptiert, akzeptiere. Sonst lehne ab.

Es gilt:

$$L(M_1) = \begin{cases} \{0,1\}^\star & \text{, falls } w \text{ von } M \text{ akzeptiert wird} \\ \emptyset & \text{sonst} \end{cases}$$

Außerdem gilt $L(M_2) = \emptyset$. Es gilt also $L(M_1) = L(M_2)$ genau dann, wenn w von M nicht akzeptiert wird. Genau in diesem Fall akzeptiert M_u^c, d.h. M_u^c akzeptiert L_u^c.

(c) Die Konstruktion ist gleich zu Aufgabenteil (b), außer dass M_2 nun eine Turingmaschine ist, die alle Eingaben akzeptiert, und für die Simulation $M_{\text{äq}}^c$ statt $M_{\text{äq}}$ verwendet wird. Nun gilt also $L(M_2) = \{0,1\}^\star$ und somit gilt $L(M_1) \neq L(M_2)$ genau dann, wenn w von M nicht akzeptiert wird. Genau in diesem Fall akzeptiert M_u^c, d.h. M_u^c akzeptiert L_u^c.