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

# Übungsblatt 3

Vorlesung Theoretische Grundlagen der Informatik im WS 19/20

**Ausgabe:** 19. November 2019
**Abgabe:** 3. Dezember 2019, 11:00 Uhr (im Kasten im UG von Gebäude 50.34)

## Aufgabe 1

(4 + 2 = 6 Punkte)

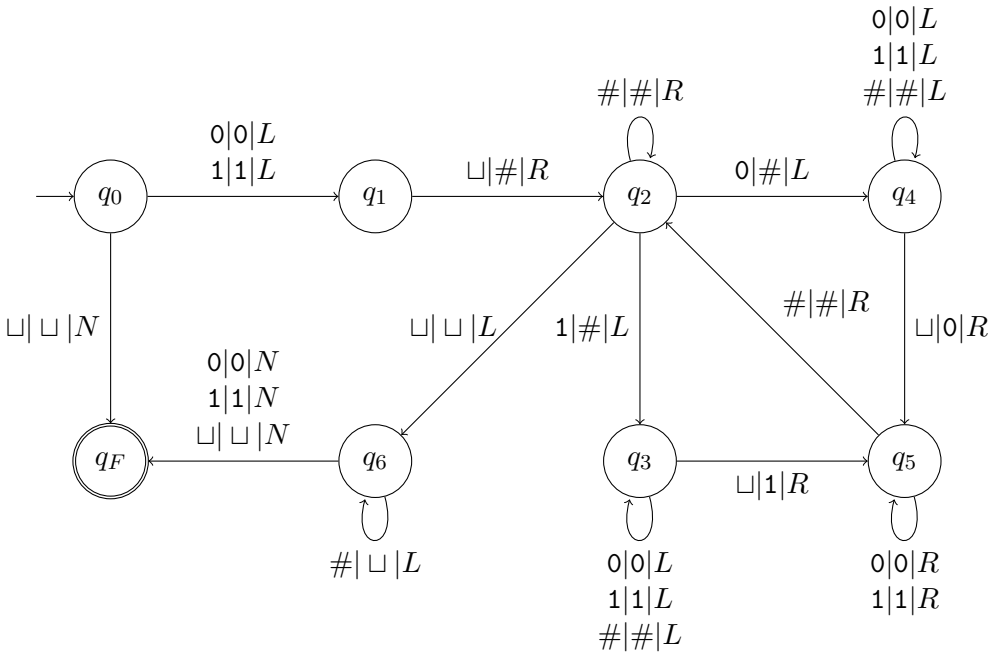
Gegeben sei das Alphabet $\Sigma = \{\mathtt{0}, \mathtt{1}\}$.

(a) Geben Sie das Bandalphabet sowie den Zustandsgraphen einer deterministischen Turing-Maschine an, die für eine Eingabe $w \in \Sigma^*$ das Spiegelwort $w^R$ berechnet. Die Eingabe `11010` wird also z.B. zu `01011` transformiert.

(b) Geben Sie die Konfigurationsfolge an, die bei Eingabe des Wortes `10` durchlaufen wird.

**Lösung:**

(a) $\Gamma = \{\mathtt{0}, \mathtt{1}, \sqcup, \#\}$




(b) $(q_0)\mathtt{10} \to (q_1)\sqcup\mathtt{10} \to \#(q_2)\mathtt{10} \to (q_3)\#\#\mathtt{0} \to (q_3)\sqcup\#\#\mathtt{0} \to \mathtt{1}(q_5)\#\#\mathtt{0} \to \mathtt{1}\#(q_2)\#\mathtt{0} \to \mathtt{1}\#\#(q_2)\mathtt{0} \to \mathtt{1}\#(q_4)\#\# \to \mathtt{1}(q_4)\#\#\# \to (q_4)\mathtt{1}\#\#\# \to (q_4)\sqcup\mathtt{1}\#\#\# \to \mathtt{0}(q_5)\mathtt{1}\#\#\# \to \mathtt{01}(q_5)\#\#\# \to \mathtt{01}\#(q_2)\#\# \to \mathtt{01}\#\#(q_2)\# \to \mathtt{01}\#\#\#(q_2)\sqcup \to \mathtt{01}\#\#(q_6)\# \to \mathtt{01}\#(q_6)\# \to \mathtt{01}(q_6)\# \to \mathtt{0}(q_6)\mathtt{1} \to \mathtt{0}(q_F)\mathtt{1}$

## 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 $M' \subseteq M$, sodass die Summe von $M'$ und die Summe von $M \setminus M'$ gleich sind?

(c) Gegeben ein ungerichteter Graph $G = (V, E)$, ist $G$ *nicht* zusammenhängend?

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: $\{30, 50, 20\}$
- 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 alle $m \in M \setminus M'$ und überprüft, ob die Summen gleich sind.

(c)
- Ja-Instanz:

- Nein-Instanz:

- Lösungsvorschlag: Eine Bipartition von $V$ in zwei Mengen $V_1$ und $V_2$ mit $V_1 \cap V_2 = \emptyset$. Diese lässt sich z.B. durch ein Binärwort $w$ der Länge $|V|$ darstellen, wobei $w_i$ genau dann 1 ist, wenn $v_i \in V_2$.
- Die NTM überprüft, ob der Lösungsvorschlag das korrekte Format hat. Wenn ja, überprüft sie für jeden Knoten $v \in V_1$, ob eine seiner ausgehenden Kanten zu einem Knoten aus $V_2$ führt.

## Aufgabe 3 (3 + 2 + 3 = 8 Punkte)

Eine normale Turing-Maschine darf ihren Lese-/Schreibkopf bei jedem Übergang höchstens um ein Feld nach links oder rechts bewegen. In dieser Aufgabe betrachten wir *Sprung-Turing-Maschinen*

(STM), die bei der Kopfbewegung Felder überspringen dürfen. Zusätzlich zur endlichen Kontrolle hat eine STM einen Zähler für die *Sprungweite* $v \in \mathbb{N}$. Diese gibt an, wie viele Felder der Kopf bei einer Bewegung überspringt. Bei einem Übergang mit Kopfbewegung nach links (bzw. rechts) geht der Kopf also $v$ Felder nach links (bzw. rechts) statt nur eins.

Anfangs gilt $v = 1$, d.h. die STM verhält sich wie eine normale TM. Bei jedem Übergang hat die STM jedoch die Möglichkeit, die Sprungweite um 1 zu erhöhen oder zu verringern. Formal hat die Überführungsfunktion einer STM dann folgende Form:

$$\delta \colon Q \times \Gamma \to Q \times \Gamma \times \{R, L, N\} \times \{-, 0, +\}$$

Dabei steht „$-$“ für „Verringere Sprungweite um 1“, „$+$“ für „Erhöhe Sprungweite um 1“ und „0“ für „keine Änderung“. Wir fordern außerdem, dass die Sprungweite nie kleiner als 1 wird.

Wir betrachten nun das Alphabet $\Sigma = \{\mathtt{0}, \mathtt{1}\}$ und die Sprache $L = \{w \in \Sigma^* \mid \exists k \in \mathbb{N} : w_0 = w_k = w_{2k} = \mathtt{1}\}$. Zum Beispiel gilt: $\mathtt{1011001} \in L$ mit $k = 3$, aber $\mathtt{100101} \notin L$.

(a) Beschreiben Sie, wie man $L$ mit einer normalen TM entscheiden kann. Sie dürfen mehrere Spuren verwenden, aber nicht mehrere Bänder. Geben Sie die Laufzeit Ihrer TM im $\mathcal{O}$-Kalkül an.

(b) Zeigen Sie, dass eine STM $L$ in Linearzeit erkennen kann.

(c) Zeigen Sie, dass STMs und normale TMs gleich mächtig sind.

**Lösung:**

(a) Die Idee ist, in aufsteigender Reihenfolge für $k = 1, 2, 3, \ldots$ zu überprüfen, ob $w_0 = w_k = w_{2k} = \mathtt{1}$ gilt. Verwende dazu eine 2-Spur-TM mit der Eingabe auf der ersten Spur und drei Markierungssymbolen auf der zweiten Spur. Bei der Überprüfung für $k$ sollen die Markierungen auf den Feldern 0, $k$ und $2k$ stehen. Anfangs stehen sie also auf 0, 1 und 2. Wenn $k$ um 1 erhöht wird, wird die zweite Markierung um ein Feld nach rechts verschoben und die dritten Markierung um zwei Felder. Für jedes $k$ überprüft der Kopf, ob die Eingabesymbole auf den markierten Feldern alle $\mathtt{1}$ sind. Falls ja, wird akzeptiert. Ansonsten wird $k$ um 1 erhöht. Sobald die dritte Markierung über das Wortende hinausgeht, wird abgelehnt.

Für die Überprüfung von $k$ braucht die TM $\mathcal{O}(k)$ viele Schritte, da sie von Feld 0 zu $2k$ und wieder zurück laufen muss. Da $k$ höchstens $\lfloor |w|/2 \rfloor$ sein kann, braucht sie insgesamt $\mathcal{O}(\sum_{k=0}^{\lfloor |w|/2 \rfloor} k) = \mathcal{O}(k^2)$ viele Schritte.

(b) Die STM arbeitet nach der gleichen Grundidee wie die normale TM, braucht aber keine Markierungen. Stattdessen setzt sie die Sprungweite immer auf das aktuell zu überprüfende $k$. Dann kann sie von Feld 0 aus mit einem Übergang Feld $k$ und mit einem weiteren Übergang Feld $2k$ erreichen, also in $\mathcal{O}(1)$ vielen Schritten überprüfen, ob $w_0 = w_k = w_{2k} = \mathtt{1}$ gilt. Daher braucht sie nur $\mathcal{O}(\sum_{k=0}^{\lfloor |w|/2 \rfloor} 1) = \mathcal{O}(k)$ viele Schritte.

(c) Eine STM kann trivialerweise eine normale TM simulieren, indem sie die Sprungweite nie verändert.

Wir simulieren eine STM mit einer 3-Spur-TM: Auf Spur 1 steht die Eingabe. Auf Spur 2 markieren wir drei Felder: die aktuelle Kopfposition der simulierten STM sowie die beiden Felder, auf die der Kopf im nächsten Schritt springen könnte. Wenn $v$ die aktuelle Sprungweite

ist, sind das also die Felder $v$ Schritte links und rechts von der aktuellen Kopfposition. Spur 3 verwenden wir, um die Markierungen zu verschieben, wenn die STM ihren Kopf bewegt.

Wir beschreiben o.B.d.A. eine Bewegung nach links: Zunächst kopieren wir die drei Markierungen von Spur 2 auf Spur 3. Nun verschieben wir alle drei Markierungen auf Spur 3 um je ein Feld nach links. Dies wiederholen wir solange, bis die mittlere Markierung auf Spur 3 auf demselben Feld steht wie die linke Markierung auf Spur 2. Dann haben wir alle drei Markierungen um genau $v$ Felder nach links bewegt. Wir löschen also die Markierungen auf Spur 2 und ersetzen sie durch die Markierungen auf Spur 3. Nun können wir den nächsten Schritt der STM simulieren.

Wenn die Sprungweite erhöht bzw. verringert wird, werden die äußeren Markierungen auf Spur 2 entsprechend um ein Feld nach außen bzw. innen verschoben.

## Aufgabe 4 (1 + 1 + 1 + 1 = 4 Punkte)

Zeigen oder widerlegen Sie:

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

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

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

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

**Lösung:**

(a) Falsch. Aus der Vorlesung wissen wir, dass das Halteproblem semi-entscheidbar ist. Angenommen, das Komplement des Halteproblems wäre ebenfalls semi-entscheidbar. Dann könnte man die beiden entsprechenden Turing-Maschinen „pseudoparallel“ laufen lassen, also abwechselnd jeweils einen Schritt der entsprechenden Turing-Maschine simulieren. Da die beiden Turing-Maschinen komplementäre Sprachen akzeptieren, hält mindestens eine von beiden nach endlicher Zeit. Damit wäre das Halteproblem entscheidbar. Widerspruch.

(b) Richtig. 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) Richtig. Seien $L_1$ und $L_2$ semi-entscheidbare Sprachen und $M_1$ und $M_2$ Turing-Maschinen, die $L_1$ bzw. $L_2$ akzeptieren. Konstruiere eine Turing-Maschine $M$, 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.

(d) Falsch. Sei $L_1 = \Sigma^*$ und $L_2 = \mathcal{H}$ (das Halteproblem). Beide Sprachen sind semi-entscheidbar. Es gilt $L_1 \setminus L_2 = \Sigma^* \cap \mathcal{H}^c = \mathcal{H}^c$. Aus (a) wissen wir aber, dass das Komplement des Halteproblems nicht semi-entscheidbar ist.

## Aufgabe 5

(2 + 3 = 5 Punkte)

Eine Turing-Maschine $M$ 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 Turing-Maschine 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 Turing-Maschine $M$, die $L$ in kanonischer Reihenfolge aufzählt. Konstruiere eine Turing-Maschine $M'$, die für jedes $w$ entscheidet, ob es zu $L$ gehört. Dazu simuliert $M'$ die Turing-Maschine $M$ 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 Turing-Maschine $M'$ entscheidet also $L$.

Sei umgekehrt $L$ eine entscheidbare Sprache mit einer entscheidenden Turing-Maschine $M$. Konstruiere eine Turing-Maschine $M'$ mit separatem Arbeits- und Ausgabeband, die die Sprache $L$ in kanonischer Reihenfolge aufzählt. Dazu schreibt $M'$ in kanonischer Reihenfolge alle Wörter in $\Sigma^*$ 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 Turing-Maschine $M$. Konstruiere eine Turing-Maschine $M'$, die $L$ semi-entscheidet. Sei $w$ eine Eingabe für $M'$. Simuliere die Turing-Maschine $M$ 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 Turing-Maschine $M$. Konstruiere eine Turing-Maschine $M'$, die die Sprache $L$ aufzählt. Dazu simuliert $M'$ die Turing-Maschine $M$ „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

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

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 $L_{\text{write}} = \{wva \mid T_w \text{ schreibt bei Eingabe } v \text{ das Symbol } a \text{ auf das Band}\}$ nicht entscheidbar ist. Benutzen Sie dafür nicht den Satz von Rice!

**Lösung:**
Angenommen, es gibt eine Turing-Maschine $M_{\text{write}}$, die $L_{\text{write}}$ entscheidet. Wir konstruieren nun eine Turing-Maschine $M_u$, die die universelle Sprache entscheidet. Da die universelle Sprache unentscheidbar ist, ergibt sich ein Widerspruch.

- Die Eingabe von $M_u$ sind eine Turing-Maschine $M$ und ein Wort $w$. $M_u$ muss akzeptieren, falls $M$ die Eingabe $w$ akzeptiert, und sonst ablehnen.
- Sei $a$ ein neues Symbol, das nicht im Bandalphabet von $M$ vorkommt. $M_u$ konstruiert nun eine neue Turing-Maschine $X$, die $M$ auf ihrer Eingabe simuliert. Falls $M$ akzeptiert, schreibt $X$ das Symbol $a$ auf das Band und akzeptiert. Sonst lehnt $X$ ab.
- $M_u$ simuliert nun $M_{\text{write}}$ mit der Eingabe $\langle X \rangle$ und $w$. Akzeptiere, falls $M_{\text{write}}$ akzeptiert. Sonst lehne ab.

$M_{\text{write}}$ entscheidet, ob $X$ das Symbol $a$ auf das Band schreibt. Nach Konstruktion von $X$ ist das genau dann der Fall, wenn $w$ von $M$ akzeptiert wird. Also entscheidet $M_u$ die universelle Sprache.

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

In der Übung wurde das Äquivalenzproblem für Turing-Maschinen vorgestellt:

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

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 Turing-Maschine $M_{\text{äq}}$, die $L_{\text{äq}}$ akzeptiert. Konstruieren Sie daraus eine Turing-Maschine $M_u^c$, 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 Turing-Maschine $M_u^c$, die $L_u^c$ akzeptiert, wie folgt:

- Die Eingabe von $M_u^c$ sind eine Turing-Maschine $M$ und ein Wort $w$. $M_u^c$ muss genau dann akzeptieren, falls $w$ nicht von $M$ akzeptiert wird.
- Konstruiere eine Turing-Maschine $M_1$, die ihre Eingabe ignoriert und stattdessen $M$ auf $w$ simuliert.
- Konstruiere eine Turing-Maschine $M_2$, 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:

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

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 Turing-Maschine ist, die alle Eingaben akzeptiert, und für die Simulation $M_{\text{äq}}^c$ statt $M_{\text{äq}}$ verwendet wird. Nun gilt also $L(M_2) = \{0,1\}^*$ 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 $L_u^c$.