Lehrstuhl für Informatik 1
Prof. Dr. Berthold Vöcking
Melanie Winkler
Oliver Göbel
Sascha Geulen

WS 2011/12
10.01.2012

Präsenzübung
Berechenbarkeit und Komplexität

Musterlösung

Name: .....

Vorname: .....

Matrikelnummer: .....

Übungsgruppe:

☐	1	Mo 12:15 - 13:45	5056	Mi 13:30 - 15:00	5052	7	☐
☐	2	Mo 12:15 - 13:45	4017	Mi 13:30 - 15:00	4017	8	☐
☐	3	Mo 14:00 - 15:30	5056	Do 10:00 - 11:30	4017	9	☐
☐	4	Di 12:00 - 13:30	4017	Do 12:00 - 13:30	5056	10	☐
☐	5	Mi 12:00 - 13:30	4017	Do 16:00 - 17:30	4017	11	☐
☐	6	Mi 12:00 - 13:30	5054				

Hinweise zu dieser Musterlösung:

- Die vorliegende Musterlösung dient der Vorbereitung auf die Bachelorklausur bzw. der Vorbereitung auf die Wiederholung der Präsenzübung.
- Sie ist bewusst sehr ausführlich gehalten, um sowohl eine korrekte als auch vollständige und nachvollziehbare Lösung der einzelnen Aufgaben zu präsentieren.
- Bei der Bachelorklausur kommt es genau wie in der Präsenzübung darauf an in einer begrenzten Zeit eine sinnvolle Lösung zu den gegebenen Aufgabenstellungen zu formulieren.
- Die Lösung muss dabei nicht so ausführlich sein wie die in dieser Musterlösung vorgestellte; sie muss aber nachvollziehbar sein und die grundlegenden Schwierigkeiten der Aufgabenstellung korrekt lösen.
- Als grober Richtwert für die Bearbeitungszeit einer Aufgabe kann angenommen werden, dass für jeden Punkt, der bei einer korrekten Lösung der Aufgabe erzielt werden kann, eine Minute Bearbeitungszeit zur Verfügung steht.
- Zusammen mit dem Vorlesungsskript, den Vorlesungsfolien, den Übungsblättern und dem bereits veröffentlichten Fragenkatalog steht damit eine weitere Lernhilfe für die Bachelorklausur zur Verfügung.

(Name/Matrikelnummer)

Aufgabe 1. (Deterministische Turingmaschinen):

(a) Ergänzen Sie die fehlenden sechs Komponenten einer deterministischen Turingmaschine (TM). Erläutern Sie die Komponenten kurz, so wie dies bei der unten stehenden Komponente auch der Fall ist. **(3 Punkte)**

- (endliche) Zustandsmenge Q
- (endliches) Eingabealphabet Σ
- (endliches) Bandalphabet $\Gamma \supset \Sigma$
- Leerzeichen (Blank) $B \in \Gamma \setminus \Sigma$
- Anfangszustand $q_0 \in Q$
- Endzustand $\bar{q} \in Q$
- Zustandsüberführungsfunktion $\delta : (Q \setminus \{\bar{q}\} \times \Gamma) \to (Q \times \Gamma \times \{R, L, N\})$

(b) Kann die Länge einer Eingabe in der Zustandsmenge einer Turingmaschine gespeichert werden? Begründen Sie Ihre Antwort. **(2 Punkte)**

Nein, die Länge der Eingabe kann nicht in der Zustandsmenge gespeichert werden. Die Zustandsmenge ist gemäß Definition endlich und fester Größe, die Eingabe kann dagegen eine beliebige Länge haben.

(c) Was ist der wesentliche Unterschied zwischen einer (1-Band-)Turingmaschine mit k-Spuren und einer k-Band-Turingmaschine? **(2 Punkte)**

Eine (1-Band-)Turingmaschine mit k-Spuren verfügt über einen Lese/Schreib-Kopf. Eine k-Band-Turingmaschine hat dagegen k Lese/Schreib-Köpfe, die sich unabhängig voneinander bewegen.

(Name/Matrikelnummer)

Aufgabe 1. (Deterministische Turingmaschinen – Fortsetzung):

(d) Beschreiben Sie, was eine Gödelnummerierung ist und welche Eigenschaften sie hat. **(2 Punkte)**

Eine Gödelnummer ist die Darstellung einer Turingmaschine als Bitstring. Sie ist eine injektive und präfixfreie Abbildung von der Menge aller Turingmaschinen in die Menge $\{0,1\}^*$.

(e) Welche Mächtigkeiten haben die Menge aller Sprachen über dem Alphabet $\{0,1\}$ sowie die Menge $A = \{x \in \{0,1\}^* \mid x \text{ ist Gödelnummer einer TM } M\}$ und was folgt aus diesen beiden Eigenschaften für die Existenz nicht-rekursiver Probleme? **(2 Punkte)**

Die Menge aller Sprachen über dem Alphabet $\{0,1\}$ ist überabzählbar. Die Menge der Gödelnummern ist dagegen abzählbar. Folglich gibt es nicht-rekursive Sprachen.

(f) Was besagt die Church-Turing-These? **(1 Punkt)**

Die Klasse der TM-berechenbaren Funktionen stimmt mit der Klasse der „intuitiv berechenbaren“ Funktionen überein.

(Name/Matrikelnummer)

Aufgabe 1. (Deterministische Turingmaschinen – Fortsetzung (2)):

(g) Beschreiben Sie die Eingabe und das Akzeptanzverhalten der universellen Turingmaschine. Wie kann eine universelle Turingmaschine auf einer 3-Band-Turingmaschine implementiert werden? **(7 Punkte)**

Als Eingabe erhält die universelle Turingmaschine U die Gödelnummer einer Turingmaschine M und ein Wort $w \in \{0,1\}^*$, d.h. einen Bitstring der Form $\langle M \rangle w$. U simuliert nun M auf der Eingabe w und akzeptiert genau dann, wenn M auf Eingabe w akzeptiert.

Wir implementieren nun U auf einer 3-Band-Turingmaschine. Zunächst überprüft U, ob die Eingabe mit einer korrekten Gödelnummer beginnt. Falls nicht, wird ein Fehler ausgegeben. Dann kopiert U die Gödelnummer $\langle M \rangle$ auf Band 2 und schreibt die Binärkodierung des Anfangszustands auf Band 3. Jetzt bereitet U Band 1 so vor, dass es nur das Wort w enthält. Der Kopf steht auf dem ersten Zeichen von w. Während der Simulation übernehmen die drei Bänder von U die folgenden Aufgaben:

- Band 1 von U simuliert das Band der Turingmaschine M.
- Band 2 von U enthält die Gödelnummer von M.
- Auf Band 3 speichert U den jeweils aktuellen Zustand von M.

Zur Simulation eines Schritts von M sucht U zu dem Zeichen an der Kopfposition auf Band 1 und dem Zustand auf Band 3 die Kodierung des entsprechenden Übergangs von M auf Band 2. Wie in der Zustandsüberführungsfunktion beschrieben,

- aktualisiert U die Inschrift auf Band 1,
- bewegt U den Kopf auf Band 1, und
- verändert U den auf Band 3 abgespeicherten Zustand von M.

Wenn die Simulation terminiert, übernimmt U das Aktzeptanzverhalten von M.

(Name/Matrikelnummer)

Aufgabe 2. (Nichtdeterministische Turingmaschinen):

(a) Worin besteht der Unterschied zwischen der Definition von nichtdeterministischer Turingmaschine (NTM) und deterministischer Turingmaschine (TM)? **(1 Punkt)**

Eine deterministische Turingmaschine besitzt eine Zustandsüberführungsfunktion; eine nichtdeterministische Turingmaschine hat eine Zustandsüberführungsrelation.

(b) Wie ist das Akzeptanzverhalten einer nichtdeterministischen Turingmaschine definiert? **(1 Punkt)**

Eine nichtdeterministische Turingmaschine M akzeptiert die Eingabe $x \in \Sigma^*$, falls es mindestens einen Rechenweg von M gibt, der in eine akzeptierende Endkonfiguration führt.

(c) Wie ist die Laufzeit einer nichtdeterministischen Turingmaschine bei Eingabe x im Falle einer akzeptierenden Rechnung definiert? **(2 Punkte)**

Die Laufzeit einer nichtdeterministischen Turingmaschine M bei Eingabe x im Falle einer akzeptierenden Rechnung ist definiert als die Länge des kürzesten akzeptierenden Rechenweges von M auf x.

(Name/Matrikelnummer)

Aufgabe 2. (Nichtdeterministische Turingmaschinen – Fortsetzung):

(d) Beschreiben Sie eine Simulation einer nichtdeterministischen Turingmaschine mit polynomieller Laufzeit $p(n)$ durch eine deterministische k-Band-Turingmaschine. Geben Sie eine Laufzeitschranke für die von Ihnen beschriebene Turingmaschine an und begründen Sie die Schranke. **(6 Punkte)**

Sei M_{NTM} eine nichtdeterministische Turingmaschine mit polynomieller Laufzeit $p(n)$. Wir konstruieren nun eine deterministische 2-Band-Turingmaschine M_{DTM}, die das Verhalten von M_{NTM} simuliert. Dazu simuliert M_{DTM} jeden der möglichen Rechenwege der Turingmaschine M_{NTM} wie folgt:

(1) Zu Beginn der Simulation schreibt M_{DTM} auf Band 1 die Eingabe der Turingmaschine M_{NTM}. Band 1 ist im Folgenden das Arbeitsband. Band 2 bleibt zunächst leer. Es wird als Speicher der bereits betrachteten Rechenwege dienen.

(2) Falls M_{NTM} in der aktuellen Konfiguration akzeptiert, dann akzeptiert auch M_{DTM}. Andernfalls wählt M_{DTM} eine der möglichen Nachfolgekonfigurationen und speichert diese auf Band 2. Falls die Länge des Rechenweges zur gewählten Nachfolgekonfiguration die durch das Polynom $p(n)$ gegebene Laufzeitschranke überschreitet oder keine Nachfolgekonfiguration existiert, dann geht M_{DTM} zu Schritt (4).

(3) Nun führt M_{DTM} die Schritte (2) bis (4) für die gewählte Nachfolgekonfiguration aus.

(4) Falls die betrachtete Nachfolgekonfiguration nicht zu einer akzeptierenden Endkonfiguration führt, wählt M_{DTM} eine andere Nachfolgekonfiguration aus bis alle Nachfolgekonfigurationen durchprobiert wurden (dieses kann mit Hilfe von Band 2 festgestellt werden).

Sei d die Anzahl der Nachfolgekonfigurationen der Konfiguration mit den meisten Nachfolgekonfigurationen. Im worst case führt nur der letzte betrachte Rechenweg zu einer akzeptierenden Endkonfiguration. In diesem Fall werden maximal $d^{p(n)}$ viele Konfigurationen durchlaufen.

(Name/Matrikelnummer)

Aufgabe 3. (Komplexitätstheorie und Mächtigkeit von Programmiersprachen):

(a) Wie kann man die Zugehörigkeit eines Entscheidungsproblems A zur Komplexitätsklasse P zeigen? **(1 Punkt)**

Die Zugehörigkeit von A zur Komplexitätsklasse P kann durch Angabe eines Polynomialzeit-Algorithmus gezeigt werden.

(b) Definieren Sie die Klasse NP. **(2 Punkte)**

(1) Definition mit Hilfe von nichtdeterministischen Turingmaschinen:

NP ist die Klasse der Entscheidungsprobleme, die durch eine nichtdeterministische Turingmaschine erkannt werden, deren worst case Laufzeit polynomiell beschränkt ist.

(2) Definition mit Hilfe von Zertifikaten und Polynomialzeit-Verifizierern:

Eine Sprache $L \subseteq \Sigma^*$ ist genau dann in NP, wenn es einen Polynomialzeit-Algorithmus V und ein Polynom p mit der folgenden Eigenschaft gibt:

$$x \in L \Leftrightarrow \exists y \in \{0,1\}^*, |y| \leq p(|x|) : V \text{ akzeptiert } y\#x \ .$$

(c) Definieren Sie, wann ein Entscheidungsproblem A NP-vollständig ist. **(1 Punkt)**

A ist NP-vollständig, falls gilt

(1) $A \in \mathsf{NP}$ und

(2) A ist NP-hart.

(Name/Matrikelnummer)

Aufgabe 3. (Komplexitätstheorie und Mächtigkeit von Programmiersprachen – Fortsetzung):

(d) Es sei **(6 Punkte)**

$$L_{\text{PALINDROM}} = \{x \in \{0,1\}^* \mid x = w_1 w_2 \dots w_n = w_n w_{n-1} w_{n-2} \dots w_2 w_1 \text{ mit } w_i \in \{0,1\}\}$$

die Sprache aller Palindrome über dem Alphabet $\{0,1\}$. Geben Sie eine nichtdeterministische 1-Band-Turingmaschine an, die das Komplement der Sprache $L_{\text{PALINDROM}}$ in Zeit $\mathcal{O}(n \log n)$ erkennt.

Eine nichtdeterministische Turingmaschine M_L, die das Komplement von $L_{\text{PALINDROM}}$ erkennt, kann wie folgt konstruiert werden:

(1) M_L läuft von links nach rechts über die Eingabe x und zählt bis zu einer Position i in einer zweiten Spur die Anzahl der gelesenen Zeichen. Zusätzlich speichert M_L das gelesene Zeichen w_i im aktuellen Zustand. Die Position i wird nichtdeterministisch bestimmt, indem die Zustandsüberführungsrelation an jeder Bandposition (innerhalb von x) die Wahl erlaubt, entweder weiterzulaufen oder in die folgende Phase überzugehen.

(2) Nun läuft M_L ans Ende der Eingabe.

(3) Daraufhin läuft M_L von rechts nach links über die Eingabe und zählt bei jedem Schritt den Zähler um Eins runter. Sobald der Zähler Null ist, liest M_L das aktuelle Zeichen ein und vergleicht dieses mit w_i.

(4) Falls die Zeichen unterschiedlich sind, dann ist x kein Palindrom und somit akzeptiert M_L.

Um die geforderte Laufzeit von $O(n \log n)$ zu erreichen, ist es notwendig, dass der Zähler auf der zweiten Spur effizient implementiert wird. Dieses können wir erreichen, indem wir den Zähler beim Durchlaufen der Eingabe jeweils mitschleppen, d.h. in jedem Schritt um eins nach rechts verschieben. Da der Zähler höchstens Länge $O(\log n)$ hat, können wir die Schritte (1) und (3) also in Zeit $O(n \log n)$ realisieren. Schritt (2) benötigt ebenfalls Zeit $O(n \log n)$ (über die Eingabe laufen und dabei den Zähler mitschleppen) und Schritt (4) kann in Zeit $O(1)$ durchgeführt werden. Damit ist die Gesamtlaufzeit von M_L beschränkt durch $O(n \log n)$.

Abschließend zeigen wir die Korrektheit von M_L: Falls x kein Palindrom ist, dann gibt es mindestens eine Position i, so dass $w_i \neq w_{n-i}$ ist. In diesem Fall wird in Schritt (4) akzeptiert. Falls die Eingabe hingegen ein Palindrom ist, dann gibt es auch keinen akzeptierenden Rechenweg.

(Name/Matrikelnummer)

Aufgabe 3. (Komplexitätstheorie und Mächtigkeit von Programmiersprachen – Fortsetzung (2))

(e) Was ist die Aussage des Satzes von Cook und Levin? **(1 Punkt)**

SAT ist NP-vollständig.

(f) Beschreiben Sie die drei syntaktischen Regeln zur Bildung von WHILE-Programmen. **(3 Punkte)**

- $x_i := x_j + c$
  mit $c \in \{-1, 0, 1\}$ ist ein WHILE-Programm.
- $P_1; P_2$
  ist ein WHILE-Programm, falls P_1 und P_2 WHILE-Programme sind.
- WHILE $x_i \neq 0$ DO P END
  ist ein WHILE-Programm, falls P ein WHILE-Programm ist.

(g) Was ist der wichtigste Unterschied (bezogen auf die Mächtigkeit der Programmiersprachen) zwischen den in WHILE- und LOOP-Programmen verwendeten Schleifenarten? Begründen Sie Ihre Antwort. **(2 Punkte)**

Innerhalb einer LOOP-Schleife darf die Schleifenvariable nicht vorkommen. Damit kann sie auch nicht geändert werden. Folglich terminiert jede LOOP-Schleife nach einer endlichen Anzahl von Durchläufen und damit terminiert auch jedes LOOP-Programm.

In WHILE-Programmen ist dies nicht der Fall: Hierbei darf die Schleifenvariable sowohl gelesen als auch geändert werden.

(h) Ist das Entscheidungsproblem, ob ein gegebenes LOOP-Programm zu einer Eingabe x die Ausgabe y berechnet, rekursiv? Begründen Sie Ihre Antwort. **(2 Punkte)**

Ja. LOOP-Programme terminieren nach einer endlichen Anzahl von Schritten. Dadurch kann am Ende der Berechnung die Ausgabe des LOOP-Programm bei Eingabe x mit dem Wert y verglichen werden.

(i) Ist das Entscheidungsproblem, ob ein gegebenes WHILE-Programm zu einer Eingabe x die Ausgabe y berechnet, rekursiv? Begründen Sie Ihre Antwort. **(2 Punkte)**

Nein. WHILE-Programme sind Turing-mächtig. Gemäß des Satzes von Rice kann damit keine Aussage über die von einem WHILE-Programm berechnete Funktion gemacht werden.

(Name/Matrikelnummer)

Aufgabe 4. (Unterprogrammtechnik):

(a) Geben Sie eine Ihnen aus der Vorlesung bekannte Sprache an, die nicht rekursiv, aber rekursiv aufzählbar ist und definieren Sie diese. **(2 Punkte)**

Das spezielle Halteproblem: $H_\epsilon = \{\langle M \rangle \mid M \text{ hält auf Eingabe } \epsilon\}$

(b) Geben Sie eine Ihnen aus der Vorlesung bekannte Sprache an, die nicht rekursiv und nicht rekursiv aufzählbar ist und definieren Sie diese. **(2 Punkte)**

Das allgemeine Halteproblem: $H_{\text{all}} = \{\langle M \rangle \mid M \text{ hält auf jeder Eingabe}\}$

(c) Geben Sie eine Ihnen aus der Vorlesung bekannte Sprache an, die rekursiv ist und definieren Sie diese. **(2 Punkte)**

Die Turingmaschine M stoppt nach höchstens 17 Schritten:
$H_{17} = \{\langle M \rangle \mid \text{Auf jeder Eingabe stoppt } M \text{ nach } \leq 17 \text{ Schritten}\}$

(Name/Matrikelnummer)

Aufgabe 4. (Unterprogrammtechnik – Fortsetzung):

(d) Definieren Sie das spezielle Halteproblem. **(1 Punkt)**

$H_\epsilon = \{\langle M \rangle \mid M \text{ hält auf Eingabe } \epsilon\}$

(e) Zeigen Sie durch Unterprogrammtechnik, dass die Sprache **(10 Punkte)**

$$L_{\text{SELF}} = \{\langle M \rangle \mid M \text{ hält auf der Eingabe } \langle M \rangle\}$$

unentscheidbar ist. Beweisen Sie insbesondere die Korrektheit Ihrer Konstruktion.

Zum Widerspruch nehmen wir an, es gibt eine Turingmaschine M_{SELF}, die die Sprache L_{SELF} entscheidet. Gemäß der Definition rekursiver Sprachen hält M_{SELF} auf jeder Eingabe und akzeptiert eine Eingabe x genau dann, wenn $x \in L_{\text{SELF}}$. Wir konstruieren nun eine Turingmaschine M_{H_ϵ}, die das spezielle Halteproblem H_ϵ entscheidet und M_{SELF} als Unterprogramm verwendet.

Konstruktion:

Zu einer Eingabe x prüft M_{H_ϵ} zunächst, ob $x = \langle M \rangle$ gilt und die Eingabe somit die richtige Form hat. Ist dies nicht der Fall, so verwirft M_{H_ϵ}. Andernfalls ist $x = \langle M \rangle$. Nun berechnet M_{H_ϵ} die Gödelnummer $\langle M^* \rangle$ einer Turingmaschine M^* mit den nachfolgenden Eigenschaften:

- M^* löscht zunächst ihre Eingabe.
- Dann simuliert M^* die Turingmaschine M.
- Sobald M hält, hält auch M^*.

Damit arbeitet M^* auf jeder Eingabe so, wie M auf ϵ und hält genau dann, wenn M hält. Nun ruft M_{H_ϵ} die Turingmaschine M_{SELF} mit Eingabe $\langle M^* \rangle$ auf und übernimmt ihr Akzeptanzverhalten.

Terminierung:

M_{H_ϵ} hält auf jeder Eingabe, da der Syntax-Check und die Konstruktion von M^* offensichtlich terminieren und M_{SELF} gemäß der Voraussetzung auf jeder Eingabe hält.

Korrektheit:

Falls x keine gültige Gödelnummer ist, so ist $x \notin H_\epsilon$ und M_{H_ϵ} verwirft die Eingabe x.

Sei nun $x = \langle M \rangle$. Es gilt:

$$\begin{aligned} x \in H_\epsilon &\Rightarrow M \text{ hält auf } \epsilon \\ &\Rightarrow M^* \text{ hält auf jeder Eingabe} \\ &\Rightarrow M^* \text{ hält auf der Eingabe } \langle M^* \rangle \\ &\Rightarrow \langle M^* \rangle \in L_{\text{SELF}} \\ &\Rightarrow M_{\text{SELF}} \text{ akzeptiert } \langle M^* \rangle \\ &\Rightarrow M_{H_\epsilon} \text{ akzeptiert } x \end{aligned}$$

$$\begin{aligned} x \notin H_\epsilon &\Rightarrow M \text{ hält nicht auf } \epsilon \\ &\Rightarrow M^* \text{ hält auf keiner Eingabe} \\ &\Rightarrow M^* \text{ hält nicht auf der Eingabe } \langle M^* \rangle \\ &\Rightarrow \langle M^* \rangle \notin L_{\text{SELF}} \\ &\Rightarrow M_{\text{SELF}} \text{ verwirft } \langle M^* \rangle \\ &\Rightarrow M_{H_\epsilon} \text{ verwirft } x \end{aligned}$$

Die Turingmaschine M_{H_ϵ} entscheidet damit H_ϵ. Dies ist aber ein Widerspruch zur Unentscheidbarkeit von H_ϵ. Die getroffene Annahme, dass die Turingmaschine M_{SELF} existiert, ist somit falsch und die Sprache L_{SELF} ist unentscheidbar.

(Name/Matrikelnummer)

Aufgabe 5. (Satz von Rice):

(a) Was besagt der Satz von Rice? **(2 Punkte)**

Sei $\mathcal{R}$ die Menge der von Turingmaschinen berechenbaren partiellen Funktionen und S eine Teilmenge von $\mathcal{R}$ mit $\emptyset \neq S \neq \mathcal{R}$. Dann ist die Sprache

$$L(S) = \{\langle M \rangle \mid M \text{ berechnet eine Funktion aus } S\}$$

nicht rekursiv.

In anderen Worten: Aussagen über die von einer Turingmaschine berechneten Funktionen sind nicht entscheidbar.

(b) Begründen Sie mit Hilfe des Satzes von Rice, dass die Sprache **(5 Punkte)**

$$L_{101} = \{\langle M \rangle \mid M \text{ akzeptiert alle Eingaben, die mit 101 beginnen}\}$$

nicht entscheidbar ist. Zeigen Sie dabei auch, dass die Bedingungen des Satzes von Rice erfüllt sind.

Die Sprache L_{101} ist äquivalent zur Menge

$$S = \{f_M \mid \forall w \in \{0,1\}^* : \text{falls } w = 101w' \text{ mit } w' \in \{0,1\}^*, \text{ dann } f_M(w) = 1\} \ .$$

S ist offensichtlich ungleich $\emptyset$, da die Funktion $f_M(w) = 1$ für alle $w \in \{0,1\}^*$, d.h. die Turingmaschine, die unabhängig von der Eingabe sofort akzeptiert, in S ist. Ebenfalls ist S ungleich $\mathcal{R}$, da die Funktion $f_M(w) = 0$ für alle $w \in \{0,1\}^*$, d.h. die Turingmaschine, die auf allen Eingaben sofort verwirft, offensichtlich nicht in S enthalten ist. Damit ist die Sprache

$$\begin{aligned} L(S) &= \{\langle M \rangle \mid M \text{ berechnet eine Funktion aus } S\} \\ &= \{\langle M \rangle \mid M \text{ akzeptiert alle Eingaben, die mit 101 beginnen}\} \end{aligned}$$

gemäß des Satzes von Rice nicht entscheidbar.

(Name/Matrikelnummer)

Aufgabe 6. (Beweis von Unentscheidbarkeit):

(a) Zeigen Sie nun mittels Reduktion, dass die oben definierte Sprache L_{101} nicht rekursiv ist. Beweisen Sie insbesondere die Korrektheit Ihrer Reduktion. **(7 Punkte)**

Wir zeigen die Unentscheidbarkeit von L_{101} durch eine Reduktion von H_ϵ. Dazu konstruieren wir eine berechenbare Funktion $f : \Sigma^* \to \Sigma^*$, so dass $x \in H_\epsilon \Leftrightarrow f(x) \in L_{101}$ gilt.

Sei x das Argument von f. Falls x keine gültige Gödelnummer ist, so setze $f(x) = x$. Andernfalls ist $x = \langle M \rangle$ einer Turingmaschine M. Wir konstruieren nun die Gödelnummer $\langle M^* \rangle$ einer Turingmaschine M^* und setzen $f(x) = \langle M^* \rangle$. M^* arbeitet dabei wie folgt: Zunächst löscht M^* die Eingabe. Dann simuliert M^* die Turingmaschine M auf Eingabe ϵ. Wenn M hält, dann akzeptiert M^*.

Berechenbarkeit:

Offensichtlich ist die Funktion f berechenbar.

Korrektheit:

Falls x keine gültige Gödelnummer ist, so ist $x \notin H_\epsilon$ und gemäß der Konstruktion gilt $f(x) \notin L_{101}$.

Sei nun $x = \langle M \rangle$. Es gilt:

$$
\begin{aligned}
x \in H_\epsilon \quad &\Rightarrow \quad M \text{ hält auf } \epsilon \\
&\Rightarrow \quad M^* \text{ akzeptiert jede Eingabe} \\
&\Rightarrow \quad M^* \text{ akzeptiert jede Eingabe, die mit 101 beginnt} \\
&\Rightarrow \quad f(x) = \langle M^* \rangle \in L_{101} \\
x \notin H_\epsilon \quad &\Rightarrow \quad M \text{ hält nicht auf } \epsilon \\
&\Rightarrow \quad M^* \text{ akzeptiert keine Eingabe} \\
&\Rightarrow \quad M^* \text{ akzeptiert keine Eingabe, die mit 101 beginnt} \\
&\Rightarrow \quad f(x) = \langle M^* \rangle \notin L_{101}
\end{aligned}
$$

Damit ist die Sprache L_{101} nicht rekursiv.

(Name/Matrikelnummer)

Aufgabe 6. (Beweis von Unentscheidbarkeit – Fortsetzung):

(b) Es sei **(10 Punkte)**

$$L_z = \left\{ \langle M \rangle w \;\middle|\; \begin{array}{l} M \text{ ist eine Turingmaschine mit Zustandsmenge } Q \text{ und} \\ M \text{ besucht bei Eingabe } w \text{ nicht den Zustand } q_z \in Q \end{array} \right\} .$$

Zeigen Sie mit einem der in der Vorlesung vorgestellten Verfahren, dass L_z nicht rekursiv ist. Gehen Sie dabei insbesondere auf die Korrektheit Ihres Beweises ein.

Zum Widerspruch nehmen wir an, es gibt eine Turingmaschine M_z, die die Sprache L_z entscheidet. Gemäß der Definition rekursiver Sprachen hält M_z auf jeder Eingabe und akzeptiert eine Eingabe x genau dann, wenn $x \in L_z$. Wir konstruieren nun eine Turingmaschine $M_{\bar{H}}$, die das Komplement des Halteproblems, also $\bar{H}$, entscheidet und M_z als Unterprogramm verwendet.

Konstruktion:

Zu einer Eingabe x prüft $M_{\bar{H}}$ zunächst, ob $x = \langle M \rangle w$ gilt und die Eingabe somit aus einer gültigen Gödelnummer und einem Eingabewort besteht. Ist dies nicht der Fall, so akzeptiert $M_{\bar{H}}$. Andernfalls ist $x = \langle M \rangle w$. Nun berechnet $M_{\bar{H}}$ die Gödelnummer $\langle M^* \rangle$ einer Turingmaschine M^* mit den nachfolgenden Eigenschaften: M^* hat zusätzlich zu den Zuständen von M einen neuen Zustand q^*. Die Zustandsüberführungsfunktion von M^* entspricht der von M außer, dass

- alle Übergänge in den Zustand q_z durch äquivalente Übergänge in q^* ersetzt werden,
- alle Übergänge aus dem Zustand q_z durch äquivalente Übergänge aus q^* ersetzt werden,
- jede Transition nach $\bar{q}$ durch äquivalente Übergänge nach q_z ersetzt werden und
- alle Transitionen von q_z zu $\bar{q}$ führen (das Band und die Kopfposition bleiben stets unverändert).

Damit geht M^* bei Eingabe w genau dann in Zustand q_z, wenn M bei Eingabe w hält. Nun ruft $M_{\bar{H}}$ die Turingmaschine M_z mit Eingabe $\langle M^* \rangle w$ auf und übernimmt ihr Akzeptanzverhalten.

Terminierung:

$M_{\bar{H}}$ hält auf jeder Eingabe, da der Syntax-Check und die Konstruktion von M^* offensichtlich terminieren und M_z gemäß der Voraussetzung auf jeder Eingabe hält.

Korrektheit:

Falls x nicht mit einer gültigen Gödelnummer beginnt, so ist $x \in \bar{H}$ und $M_{\bar{H}}$ akzeptiert die Eingabe x.

Sei nun $x = \langle M \rangle w$. Es gilt:

$x \in \bar{H}$	$\Rightarrow$	M hält nicht auf w	$x \notin \bar{H}$	$\Rightarrow$	M hält auf w
	$\Rightarrow$	M^* hält nicht auf w		$\Rightarrow$	M^* hält auf w
	$\Rightarrow$	M^* besucht nicht den Zustand q_z		$\Rightarrow$	M^* besucht den Zustand q_z
	$\Rightarrow$	$\langle M^* \rangle w \in L_z$		$\Rightarrow$	$\langle M^* \rangle w \notin L_z$
	$\Rightarrow$	M_z akzeptiert $\langle M^* \rangle w$		$\Rightarrow$	M_z verwirft $\langle M^* \rangle w$
	$\Rightarrow$	$M_{\bar{H}}$ akzeptiert x		$\Rightarrow$	$M_{\bar{H}}$ verwirft x

Die Turingmaschine $M_{\bar{H}}$ entscheidet damit $\bar{H}$. Dies ist aber ein Widerspruch zur Unentscheidbarkeit von $\bar{H}$. Die getroffene Annahme, dass die Turingmaschine M_z existiert, ist somit falsch und die Sprache L_z ist unentscheidbar.