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

Übungsblatt 4

Vorlesung Theoretische Grundlagen der Informatik im WS 18/19

Ausgabe 4. Dezember 2018
Abgabe 18. Dezember 2018, 11:00 Uhr (im Kasten im UG von Gebäude 50.34)

Aufgabe 1 (3 Punkte)

Die in der Vorlesung definierte nichtdeterministische Turingmaschine wird auch als RV-NTM bezeichnet (Raten/Verifizieren). Eine andere Möglichkeit ist es, NTMs analog zu NEAs zu definieren (vgl. Wegener und Übung):

Eine solche alternative nichtdeterministische Turingmaschine (A-NTM) ist definiert wie eine TM, nur ist die Übergangsfunktion δ durch eine zweistellige Relation $\subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{L, R, N\})$ ersetzt, die wir ebenfalls δ nennen.

Die Arbeitsweise einer A-NTM ist die folgende. Wenn die A-NTM im Zustand $q \in Q$ das Zeichen $a \in \Gamma$ liest, ist für jedes $((q, a), (q', a', d)) \in \delta$ der Rechenschritt möglich, den eine DTM für $\delta(q, a) = (q', a', d)$ durchführt. Falls kein Rechenschritt möglich ist, stoppt die A-NTM. Für eine feste Eingabe x können offensichtlich viele Rechenwege möglich sein.

Eine A-NTM $\mathcal{M}$ akzeptiert eine Eingabe x, falls es mindestens einen Rechenweg von $\mathcal{M}$ gibt, der in einen akzeptierenden Endzustand führt. Die von $\mathcal{M}$ erkannte Sprache $L = L(\mathcal{M})$ besteht aus allen Worten, die $\mathcal{M}$ akzeptiert.

Die Rechenzeit für eine Eingabe $x \in L(\mathcal{M})$ ist gleich der Anzahl der Rechenschritte auf einem kürzesten akzeptierenden Rechenweg. Die Zeitkomplexität $T_{\mathcal{M}}(n)$ ist das Maximum der Rechenzeiten für alle Eingaben aus $L(\mathcal{M})$ der Länge n, und 1 falls es keine solche Eingabe gibt.

Die Klasse ANP ist die Klasse aller Probleme, die von einer solchen A-NTM in polynomieller Zeit entschieden werden können. In der Übung wurde/wird[1] gezeigt, dass $\mathsf{ANP} \subseteq \mathsf{NP}$ gilt. Zeigen Sie: $\mathsf{NP} \subseteq \mathsf{ANP}$.

Lösung:
Wir müssen zeigen, dass für jede RV-NTM $\mathcal{M}$ eine äquivalente A-NTM $\mathcal{M}'$ existiert, deren Laufzeit beschränkt ist durch ein Polynom in der Laufzeit von $\mathcal{M}$. Zu diesem Zweck modellieren wir das Orakelmodul von $\mathcal{M}$ mit Hilfe der erweiterten Übergangsfunktion einer A-NTM. Sei $\# \in \Gamma$ das Trennzeichen von $\mathcal{M}$, welches vom Orakelmodul genutzt wird, um die Eingabe vom Orakelwort auf dem Eingabeband zu trennen.

- Sei zunächst $\mathcal{M}_{\text{det}}$ der deterministische Teil von $\mathcal{M}$, also alle Zustände, Übergänge, etc. die die endliche Kontrolle betreffen. Sei s der Startzustand, Q die Zustandsmenge und δ die Übergangsfunktion von $\mathcal{M}_{\text{det}}$.

[1] am 6.12.2018

- Erweitere Q durch Hinzufügen eines neuen Zustands O (für Orakel).
- Erweitere δ zu einer Relation wie folgt:
  - Für jedes $a \in \Sigma$ sei (zusätzlich zu $((s,a),\delta(s,a))$) das Paar $((s,a),(O,a,L))$ in δ.
    Dies erlaubt der A-NTM in den Zustand O zu gehen.
  - Sei $((O,\sqcup),(O,\sqcup,L))$ in δ.
    Dies erlaubt der A-NTM im Zustand O, beliebig weit nach links von der Eingabe auf dem Eingabeband zu gehen.
  - Für jedes $a \in \Gamma$ sei $((O,\sqcup),(O,a,R))$ in δ.
    Dies erlaubt der A-NTM im Zustand O, ein beliebiges Wort in $\Gamma^\star$ links von der Eingabe auf das Rechenband zu schreiben.
  - Sei $((O,\sqcup),(s,\#,R))$ in δ.
    Dies erlaubt der A-NTM im Zustand O, das Trennzeichen direkt vor die Eingabe zu schreiben und wieder in den Startzustand s von $\mathcal{M}_{\text{det}}$ zu gehen.
- Bezeichne $\mathcal{M}'$ die so erhaltene Erweiterung von $\mathcal{M}_{\text{det}}$.

Es ist klar, dass $\mathcal{M}'$ für jede Eingabe $x \in \Sigma^\star$ die Möglichkeit hat, das Orakelmodul von $\mathcal{M}$ zu imitieren und so x zu akzeptieren, genau dann wenn $x \in L(\mathcal{M})$. Außerdem ist die Laufzeit von $\mathcal{M}'$ genau die Laufzeit von $\mathcal{M}$.

Aufgabe 2

(3 + 4 + 2 = 9 Punkte)

In der Vorlesung wurden Turingmaschinen mit einem eindimensionalen, in beide Richtungen unbeschränkten Band betrachtet. Der Index eines Feldes kann hier durch eine ganze Zahl $i \in \mathbb{Z}$ repräsentiert werden. Wir nennen diese Turingmaschinen daher im Folgenden $\mathbb{Z}$-TM. In dieser Aufgabe betrachten wir einige Turingmaschinen-Modelle mit anderen Ausprägungen des Bandes:

- $\mathbb{N}_0$-TM haben ein eindimensionales, in eine Richtung beschränktes Band, d.h. links von der Startposition 0 gibt es keine weiteren Felder.
- $\mathbb{Z}^2$-TM haben ein zweidimensionales, in jede Richtung unbeschränktes Band. Die Felder können hier durch zweistellige Koordinaten adressiert werden, z.B. ist die Startposition des Kopfes $(0,0)$.
- $\mathbb{N}_0^2$-TM haben ein zweidimensionales, in beiden Dimensionen einseitig beschränktes Band, d.h. Felder mit negativen Koordinaten existieren nicht.

Gemäß der Church'schen These sollten diese drei Modelle genauso mächtig wie $\mathbb{Z}$-TMs sein. Dies sollen Sie in dieser Aufgabe zeigen, indem Sie die entsprechenden (nichttrivialen) Simulationen angeben.

(a) Beschreiben Sie, wie eine $\mathbb{Z}$-TM von einer $\mathbb{N}_0$-TM simuliert werden kann.

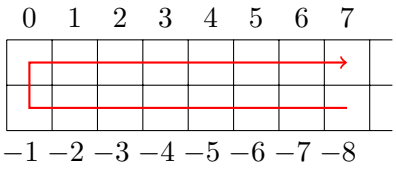
(b) Beschreiben Sie, wie eine $\mathbb{N}_0^2$-TM von einer $\mathbb{Z}$-TM simuliert werden kann.

(c) Beschreiben Sie, wie eine $\mathbb{Z}^2$-TM von einer $\mathbb{Z}$-TM simuliert werden kann.

Sie dürfen dabei die in der Übung bewiesene Tatsache verwenden, dass k-Band-TM genauso mächtig sind wie 1-Band-TM. Ihre simulierende TM darf also eine beliebige (aber konstante!) Anzahl von Bändern verwenden.

Lösung:

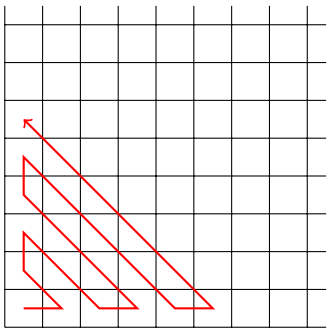
(a) Wir verwenden eine $\mathbb{N}_0$-TM mit zwei Spuren. Dabei wird das ursprüngliche Band gedanklich an Feld 0 „umgeklappt“, d.h. Feld i von Spur 1 speichert Feld i des ursprünglichen Bandes und Feld i von Spur 2 speichert Feld $-(i+1)$ des ursprünglichen Bandes. Das unbeschränkte Band wird also wie folgt in das beschränkte eingebettet:



Die Simulation läuft dann wie folgt ab:

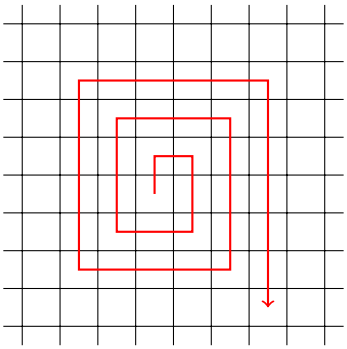
- Der Kopf ist jeweils nur auf einer der beiden Spuren aktiv und ignoriert die andere. Wenn Spur 1 aktiv ist, verhält sich die $\mathbb{N}_0$-TM genau wie die ursprüngliche $\mathbb{Z}$-TM. Wenn Spur 2 aktiv ist, werden die Kopfbewegungen umgedreht.
- Wenn sich der Kopf auf Feld 0 befindet und sich eigentlich nach links bewegen müsste, bleibt er stattdessen auf demselben Feld und wechselt auf die andere Spur.

(b) Wir benötigen eine Einbettung des eindimensionalen Bands in die zweidimensionale Ebene, sodass jedes Feld des zweidimensionalen Bands einen endlichen Index auf dem eindimensionalen Band zugewiesen bekommen. Das ist z.B. mit der folgenden Schlangenlinie möglich:



Felder, die auf dem 2D-Band benachbart sind, können nun auf dem 1D-Band weit entfernt voneinander liegen. Um einen Übergang der $\mathbb{N}_0^2$-TM zu simulieren, muss die $\mathbb{Z}$-TM sich den Index der aktuellen Kopfposition merken (z.B. auf einem zusätzlichen Band), daraus die neue Kopfposition berechnen und dann den Kopf dorthin bewegen.

(c) Die grundsätzliche Funktionweise ist identisch zu Aufgabenteil (b), allerdings funktioniert hier die Einbettung in Schlangenlinienform nicht mehr. Stattdessen ist z.B. eine Spiralform möglich:



Aufgabe 3 (3 + 2 + 2 = 7 Punkte)

(a) Geben Sie eine polynomielle Transformation von 3COLOR zu SAT an. Geben Sie dazu eine Vorschrift an, wie ein Graph $G = (V, E)$ in eine Menge von Klauseln überführt werden kann, die genau dann erfüllbar ist, wenn G dreifärbbar ist. Beschreiben Sie die Bedeutung der Variablen und Klauseln. Zeigen Sie, dass die Anzahl der Variablen und Klauseln sowie die Länge der Klauseln polynomiell in der Eingabegröße der 3COLOR-Instanz ist.

(b) Geben Sie eine *lineare* Transformation von SUBSET SUM zu KNAPSACK an.

(c) Gegeben seien zwei Probleme A und B und eine polynomielle Transformation von A nach B. Es sind keine weiteren Eigenschaften bekannt. Welche der folgenden Aussagen folgen *allein* aus der Existenz der polynomiellen Transformation? Begründen Sie!

(i) B ist NP-vollständig.

(ii) Wenn A NP-schwer ist, dann ist B NP-schwer.

(iii) Wenn B NP-vollständig ist, dann ist A NP-schwer.

(iv) Wenn A NP-schwer ist, dann ist B NP-vollständig.

Lösung:

(a) Die Menge der erlaubten Farben sei $C = \{r, g, b\}$. Wir definieren für jede Kombination aus Knoten $v \in V$ und Farbe $c \in C$ die Variable $x_{v,c}$, die genau dann wahr ist, wenn v die Farbe c hat. Es müssen nun drei Bedingungen erfüllt werden:

- Jeder Knoten hat mindestens eine Farbe. Dies erreichen wir, indem wir für jeden Knoten v die Klausel $(x_{v,r} \vee x_{v,g} \vee x_{v,b})$ hinzufügen.
- Jeder Knoten hat höchstens eine Farbe. Dazu fügen wir für jeden Knoten v die folgenden drei Klauseln ein: $(\overline{x_{v,r}} \vee \overline{x_{v,g}}) \wedge (\overline{x_{v,r}} \vee \overline{x_{v,b}}) \wedge (\overline{x_{v,g}} \vee \overline{x_{v,b}})$.
- Je zwei adjazente Knoten haben verschiedene Farben. Dazu fügen wir für jede Kanten (u, v) die folgenden drei Klauseln ein: $(\overline{x_{u,r}} \vee \overline{x_{v,r}}) \wedge (\overline{x_{u,g}} \vee \overline{x_{v,g}}) \wedge (\overline{x_{u,b}} \vee \overline{x_{v,b}})$.

Unsere SAT-Instanz hat $3|V|$ Variablen. Alle Klauseln bestehen aus konstant vielen Literalen. Die erste Klauselgruppe besteht aus $|V|$ Klauseln, die zweite aus $3|V|$ und die dritte aus $3|E|$. Insgesamt liegt die Größe der SAT-Instanz also in $\mathcal{O}(|V| + |E|)$.

(b) Eine SUBSET SUM-Instanz besteht aus einer endlichen Menge M_s, einer Gewichtsfunktion $w_s\colon M_s \to \mathbb{N}_0$ und einem Zielgewicht $K \in \mathbb{N}_0$. Gesucht ist eine Teilmenge $M' \subseteq M_s$ mit Gesamtgewicht von genau K, also $\sum_{a\in M'} w_s(a) = K$.

Eine KNAPSACK-Instanz besteht aus einer endlichen Menge M_k, einer Gewichtsfunktion $w_k\colon M_k \to \mathbb{N}_0$, einer Kostenfunktion $c_k\colon M_k \to \mathbb{N}_0$, einem Höchstgewicht $W \in \mathbb{N}_0$ und Mindestkosten $C \in \mathbb{N}_0$. Gesucht ist eine Teilmenge $M' \subseteq M_k$ mit Gesamtgewicht von höchstens W und Gesamtkosten von mindestens C, also $\sum_{a\in M'} w_k(a) \leq W$ und $\sum_{a\in M'} c_k(a) \geq C$.

Wir wählen $M_k = M_s$, $w_k = w_s$, $c_k = w_s$, $W = K$ und $C = K$. Diese KNAPSACK-Instanz ist genau dann erfüllbar, wenn die urpsrüngliche SUBSET SUM-Instanz erfüllbar ist:

- Sei M' eine Lösung der SUBSET SUM-Instanz. Dann gilt $\sum_{a\in M'} w_k(a) = \sum_{a\in M'} w_s(a) = K = W$ und $\sum_{a\in M'} c_k(a) = \sum_{a\in M'} w_s(a) = K = C$, also löst M' auch die KNAPSACK-Instanz.
- Sei umgekehrt M' eine Lösung der KNAPSACK-Instanz. Dann gilt $\sum_{a\in M'} w_s(a) = \sum_{a\in M'} w_k(a) \leq W = K$ und $\sum_{a\in M'} w_s(a) = \sum_{a\in M'} c_k(a) \geq C = K$. Also muss $\sum_{a\in M'} w_s(a) = K$ gelten und somit löst M' die SUBSET SUM-Instanz.

Die Transformation ist offensichtlich linear, da nur Teile der Eingabe dupliziert werden müssen.

(c) (i) Falsch. Das ließe sich nur folgern, wenn A NP-schwer ist.

(ii) Richtig. Da A NP-schwer ist, existiert für alle Probleme in NP eine polynomielle Transformation auf A. Da polynomielle Transformationen transitiv sind, existiert für alle Problem in NP ebenfalls eine polynomielle Transformation auf B.

(iii) Falsch. Dazu müsste eine polynomielle Transformation von B auf A existieren.

(iv) Falsch. Es ist nicht bekannt, ob B in NP liegt.

Aufgabe 4 (3 Punkte)

Bezeichne mit 42-COLOR das Problem zu entscheiden, ob sich die Knoten eines Graphen mit höchstens 42 Farben färben lassen, so dass keine zwei adjazenten Knoten dieselbe Farbe erhalten. Zeigen Sie die NP-Vollständigkeit von 42-COLOR, indem sie von 3-COLOR reduzieren.

Lösung:
Um zu überprüfen, ob ein gefärbter Graph eine gültige 42-Färbung ist, reicht es erstens zu prüfen, dass insgesamt höchstens 42 Farben verwendet werden und zweitens für jede Kanten zu überprüfen, dass die Farben der beiden Endknoten unterschiedlich sind. Damit gilt 42-COLOR $\in$ NP.

Es bleibt zu zeigen, dass 42-COLOR NP-schwer ist. Sei $G = (V, E)$ eine Instanz von 3-COLOR. Konstruiere daraus wie folgt eine Instanz $G' = (V', E')$ von 42-COLOR. Setze $V' = V \dot\cup \{1, 2, \ldots, 39\}$ und

$$E' = E \cup \{1, 2, \ldots, 39\}^2 \cup \{\{i, v\} \mid \forall i \in \{1, 2, \ldots, 39\}, v \in V\}.$$

Der Graph G' entsteht also aus G, indem eine Clique der Größe 39 hinzugefügt wird und deren Knoten zu allen Knoten von G verbunden werden.

Die Laufzeit dieser Konstruktion ist linear (also polynomiell) in der Größe von G. Jede 3-Färbung von G induziert eine 42-Färbung von G', indem die Färbung der Knoten von G unverändert bleibt und jeder der zusätzlichen 39 Knoten aus G' eine neue, eigene Farbe erhält. Umgekehrt induziert jede 42-Färbung von G' auch eine 3-Färbung von G. Beobachte dazu, dass jeder der 39 neuen

Knoten eine eigene Farbe haben muss, da diese Knoten eine Clique bilden. Da jeder der 39 neuen Knoten auch zu allen Knoten von G verbunden ist, bleiben für die verbleibenden Knoten, die alle in V liegen, bleiben also noch drei Farben übrig. Da alle Kanten von G auch in G' enthalten sind, sind alle Paare von Knoten, die in G adjazent sind, auch in G' adjazent und haben demnach unterschiedliche Farben. Damit ist gezeigt, dass G genau dann 3-färbbar ist, wenn G' 42-färbbar ist.

Insgesamt haben wir gezeigt, dass 42-COLOR in NP liegt und NP-schwer ist, womit gezeigt ist, dass 42-COLOR NP-vollständig ist.

Aufgabe 5 (2 Punkte)

In Vorlesung und Übung wurde gezeigt, wie eine beliebige Turingmaschine $\mathcal{M}$ durch ein Wort $\langle \mathcal{M} \rangle \in \{0,1\}^\star$ kodiert werden kann. Umgekehrt haben wir (ohne genauere Erläuterungen) gefordert, dass jedes Wort aus $\{0,1\}^\star$ eine Turingmaschine kodieren möge. Anders als in der Vorlesung lassen wir nun die Vereinbarung, dass ein Wort, das keine Turingmaschine im Sinne der Gödelnummer beschreibt, eine Turingmaschine kodiert, die die Sprache $\emptyset$ akzeptiert, fallen. Stattdessen möchten wir (zur späteren Verwendung in Aufgabe 6) jede Turingmaschine durch unendlich viele Wörter aus $\{0,1\}^\star$ kodieren.

Beschreiben Sie eine Dekodierungsfunktion $\langle \cdot \rangle^{-1}$, die jedem Wort aus $\{0,1\}^\star$ eine Turingmaschine zuordnet und die die genannten Forderungen erfüllt.

Lösung:
Gibt es für ein Wort $w \in \{0,1\}^\star$ eine Zerlegung $w = \langle \mathcal{M} \rangle v$ mit $v \in \{0,1\}^\star$, dann definiere $\langle w \rangle^{-1} = \mathcal{M}$. Da die Kodierung $\langle \cdot \rangle$ präfixfrei ist, d.h., keine Kodierung einer Turingmaschine Präfix einer anderen Kodierung einer Turingmaschine ist, ist diese Vorschrift wohldefiniert. Für jedes verbleibende Wort w' vereinbaren wir wieder wie in der Vorlesung, dass w' eine Turingmaschine kodiert, die die Sprache $\emptyset$ akzeptiert.

Aufgabe 6 (3 + 2 + 4 = 9 Punkte)

Die Algorithmik beschäftigt sich intensiv mit dem Entwurf von Algorithmen mit möglichst geringer Laufzeit. Aber können z.B. innerhalb quadratischer Laufzeit wirklich mehr Sprachen entschieden werden als in linearer Zeit? Ziel dieser Aufgabe ist es, zu beweisen, dass dies tatsächlich so ist, dass also $\mathsf{DTIME}(n) \subsetneq \mathsf{DTIME}(n^2)$ gilt. Zur Erinnerung:

> $\mathsf{DTIME}(n^k)$ bezeichnet die Menge der Sprachen, die von einer deterministischen Turingmaschine in $\mathcal{O}(n^k)$ Laufzeit entschieden werden können.

Wir verwenden ein Diagonalargument, das ganz ähnlich ist zu dem Beweis, dass die Diagonalsprache unentscheidbar ist. Dazu definieren wir eine Sprache, die in quadratischer Laufzeit entschieden werden kann und zeigen dann, dass diese Sprache von keiner Turingmaschine in Linearzeit entschieden werden kann.

Betrachten Sie dazu die Turingmaschine D. Bei Eingabe von w läuft D für exakt $|w|^2$ Schritte. Insbesondere gilt also $L(D) \in \mathsf{DTIME}(n^2)$. Dabei simuliert D die Turingmaschine T_w auf Eingabe w. Sei m die Anzahl der Ausführungsschritte von T_w, die durch D in $|w|^2$ Schritten simuliert werden können. Um m Schritte einer Turingmaschine zu simulieren, braucht man im Allgemeinen

$\Theta(m \log m)$ Schritte. Wir gehen davon aus, dass D $cm \log m$ Schritte braucht (mit einer Konstante c), um m Schritte von T_w zu simulieren. Es gilt also $cm \log m = |w|^2$. Falls T_w innerhalb dieser m Schritte terminiert, so akzeptiert D die Eingabe w genau dann, wenn T_w die Eingabe w ablehnt. Ansonsten lehnt D die Eingabe w ab.

Wir möchten nun zeigen, dass es keine Turingmaschine gibt, die $L(D)$ in Linearzeit entscheidet. Um Aussagen über alle Turingmaschinen zu machen, betrachten wir ein beliebiges Wort w und interpretieren dieses als kodierte Turingmaschine T_w. Auch dieses Vorgehen kennen Sie schon vom Beweis der Unentscheidbarkeit der Diagonalsprache.

(a) Gehen Sie zunächst davon aus, dass die Simulation von T_w bei Eingabe von w stets innerhalb der geforderten Anzahl an Schritten terminiert. Zeigen Sie, dass dann $L(T_w) \neq L(D)$ gilt.

(b) Sei T_w eine Turingmaschine, die $L(D)$ in Linearzeit entscheidet. D simuliert m Schritte von T_w, wobei $cm \log m = |w|^2$ gilt. Erklären Sie, wieso T_w nicht notwendigerweise nach Ablauf der m Schritte terminiert.

(c) Sei w so gewählt, sodass T_w eine Turingmaschine ist, die $L(D)$ in Linearzeit entscheidet, deren Simulation durch D aber innerhalb der $|w|^2$ Ausführungsschritte von D nicht terminiert. Erinnern Sie sich an die Forderung aus Aufgabe 5, dass jede Turingmaschine durch unendlich viele Wörter kodiert sein muss. Zeigen Sie, dass T_w dann auch durch ein anderes Wort w' kodiert wird (d.h., $T_w = T_{w'}$), sodass die Simulation von $T_{w'}$ durch D innerhalb von $|w'|^2$ Schritten terminiert. Folgern Sie, dass $T_w = T_{w'}$ somit $L(D)$ nicht entscheiden kann.

Lösung:

(a) Verwende genau dasselbe Argument wie beim Beweis, dass die Diagonalsprache unentscheidbar ist. Nehme an, dass T_w eine Turingmaschine ist, die $L(D)$ in Linearzeit entscheidet. Dann simuliert D bei Eingabe von $w = \langle T_w \rangle$ die Turingmaschine T_w für $|w|^2$ Schritte. Gemäß der Aufgabenstellung ist davon auszugehen, dass die T_w innerhalb dieser Anzahl an Schritten terminiert. Dann akzeptiert aber D die Eingabe w genau dann, wenn T_w die Eingabe w ablehnt. Damit kann also T_w nicht dieselbe Sprache wie D entscheiden.

(b) T_w läuft in Linearzeit, also $c'|w|$ Schritte für eine weitere Konstante c'. Damit T_w rechtzeitig terminiert, muss also $cc'|w| \log(c'|w|) \leq |w|^2$ gelten. Je nach Wahl von $|w|$ in Relation zu c und c' ist dies nicht unbedingt der Fall, z.B. für $|w| < cc'$.

(c) Ist T_w eine Turingmaschine, die wie in Teil b) beschrieben nicht terminiert, so wählen wir ein Wort w' mit $\langle w' \rangle^{-1} = T_w$, das "lang genug" ist. Konkret muss $cc'|w'| \log(c'|w'|) \leq |w'|^2$ gelten, was gemäß dem Θ-Kalkül für alle w' ab einer gewissen Länge gelten muss. Da es nach Aufgabe 5 unendlich viele Kodierungen von T_w gibt, lässt sich eine geeignete Kodierung w' finden. Dann greift die Argumentation aus Teil a), d.h., $T_{w'} = T_w$ kann $L(D)$ nicht entscheiden.