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

Übungsblatt 4

Vorlesung Theoretische Grundlagen der Informatik im WS 19/20

Ausgabe: 3. Dezember 2019
Abgabe: 17. Dezember 2019, 11:00 Uhr (im Kasten im UG von Gebäude 50.34)

Aufgabe 1 (3 Punkte)

Die in der Vorlesung definierte nichtdeterministische Turing-Maschine 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 Turing-Maschine (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 polynomialer Zeit entschieden werden können. In der Übung wurde 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 $\mathcal{M}_{\text{det}}$.
- 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 genau dann zu akzeptieren, wenn $x \in L(\mathcal{M})$. Außerdem ist die Laufzeit von $\mathcal{M}'$ genau die Laufzeit von $\mathcal{M}$.

Aufgabe 2 (5 Punkte)

In der Vorlesung wurde (ohne Beweis) behauptet, dass eine universelle Turing-Maschine existiert, die als Eingabe eine Gödelnummer w und ein Wort v bekommt und dann die Ausführung der Turing-Maschine T_w mit der Eingabe v simuliert. In dieser Aufgabe sollen Sie konkret die Arbeitsweise einer solchen universellen Turing-Maschine T_U beschreiben. Ihre universelle Turing-Maschine darf mehrere Bänder verwenden. Der Einfachheit halber können Sie davon ausgehen, dass ein festes Bandalphabet Γ vorgegeben ist, dass sowohl T_U als auch alle von T_U simulierten Turing-Maschinen verwenden. Die Laufzeit Ihrer Simulation ist unerheblich.

Ihre Beschreibung von T_U sollte mindestens folgende Fragen beantworten:

- Wieviele Bänder hat T_U und wofür werden sie benutzt?
- Wie repräsentiert T_U die aktuelle Bandbeschriftung von T_w?
- Wie repräsentiert T_U den aktuellen Zustand der endlichen Kontrolle von T_w?
- Wie repräsentiert T_U die aktuelle Kopfposition von T_w?
- Wie identifiziert T_U den passenden Übergang von T_w für den aktuellen Zustand und die aktuelle Bandbeschriftung?
- Wie führt T_U diesen Übergang aus, d.h. wie aktualisiert sie Bandbeschriftung, Kopfposition und Zustand?

Lösung:
T_U benutzt 4 Bänder. Auf Band 1 steht die aktuelle Bandbeschriftung von T_w; der dazugehörige Kopf markiert die Kopfposition von T_w. Auf Band 2 steht die Gödelnummer w. Auf Band 3 steht unär kodiert der aktuelle Zustand von T_w, d.h. wenn T_w sich im Zustand q_i befindet, stehen auf Band 3 i Nullen. Auf Band 4 steht unär kodiert das aktuell von T_w gelesene Zeichen (also das

Zeichen, auf dem Kopf 1 aktuell steht), d.h. wenn T_w das Zeichen a_j liest, stehen auf Band 4 j Nullen.

Stehe also $\mathtt{0}^i$ auf Band 3 und $\mathtt{0}^j$ auf Band 4. Dann muss Kopf 2 in der Gödelnummer den Übergang $\delta(q_i, a_j)$ finden. Gehe dazu von links nach rechts die Gödelnummer auf Band 2 durch. Prüfe für jeden Übergang, ob er mit $\mathtt{0}^i\mathtt{1}\mathtt{0}^j$ anfängt. Dazu muss lediglich überprüft werden, ob die Anzahl der Nullen mit der Anzahl auf Band 3 bzw. 4 übereinstimmt. Falls kein Übergang die richtige Form hat, lehne ab.

Sei ansonsten ein Übergang $\mathtt{0}^i\mathtt{1}\mathtt{0}^j\mathtt{1}\mathtt{0}^r\mathtt{1}\mathtt{0}^s\mathtt{1}\mathtt{0}^t$ gefunden. Also geht T_w in Zustand q_r über, schreibt a_s auf das Band und bewegt den Kopf in Richtung d_t. Simuliere diesen Schritt mit T_U: Kopf 1 schreibt a_s auf Band 1 und bewegt sich in Richtung d_t. Auf Band 3 wird $\mathtt{0}^i$ durch $\mathtt{0}^r$ ersetzt; auf Band 4 wird $\mathtt{0}^j$ durch $\mathtt{0}^s$ ersetzt. Kopf 2 fährt zurück an den Anfang von w, um den Übergang für den nächsten Schritt zu suchen.

Aufgabe 3 (3 Punkte)

Das Entscheidungsproblem ZWEIFACH-SAT ist wie folgt definiert: Gegeben ist eine Menge U von Variablen und eine Menge C von Klauseln (mit beliebig vielen Literalen) über U. Gefragt ist, ob es *mindestens zwei* erfüllende Wahrheitsbelegungen für C gibt. Zeigen Sie, dass ZWEIFACH-SAT NP-vollständig ist.

Achtung: ZWEIFACH-SAT ist nicht zu verwechseln mit 2SAT! Bei ZWEIFACH-SAT wird nach zwei erfüllenden Belegungen gefragt. Bei 2SAT ist die Klauselläge auf 2 *beschränkt.*

Lösung:
Eine NTM kann ZWEIFACH-SAT wie folgt entscheiden: Das Orakel gibt zwei Wahrheitsbelegungen vor. Der deterministische Teil prüft, ob die beiden Belegungen nicht identisch sind und ob sie jeweils C erfüllen. Analog zu SAT geht das in polynomialer Zeit, also gilt ZWEIFACH-SAT $\in$ NP.

Wir zeigen, dass ZWEIFACH-SAT NP-schwer ist, indem wir von SAT reduzieren. Sei $I = (U, C)$ eine SAT-Instanz. Wir führen eine neue Variable $x \notin U$ ein, die in keiner Klausel benutzt wird. Unsere konstruierte ZWEIFACH-SAT-Instanz ist also $I' = (U', C)$ mit $U' = U \cup \{x\}$. Offensichtlich lässt sich I' in linearer (also polynomialer) Zeit konstruieren. Es bleibt zu zeigen, dass I' genau dann eine Ja-Instanz von ZWEIFACH-SAT ist, wenn I eine Ja-Instanz von SAT ist.

Sei I eine Ja-Instanz von SAT. Dann gibt es mindestens eine Wahrheitsbelegung von U, die C erfüllt. Diese Belegung lässt sich auf zwei Arten zu einer Belegung von U' erweitern: x wird entweder auf `wahr` oder auf `falsch` abgebildet. In beiden Fällen sind nach wie vor alle Klauseln in C erfüllt. Also gibt es mindestens zwei erfüllende Belegungen für C'.

Sei I' eine Ja-Instanz von ZWEIFACH-SAT. Dann gibt es mindestens zwei Wahrheitsbelegungen von U', die C erfüllen. Wähle beliebig eine davon. Die entsprechende Belegung von U, bei der x einfach weggelassen wird, erfüllt C ebenfalls, da x in den Klauseln nicht vorkommt.

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

Aufgabe 4 (3 + 2 = 5 Punkte)

In dieser Aufgabe betrachten wir das Problem 2COLOR: Gegeben ist ein Graph $G = (V, E)$. Gefragt ist, ob es eine Färbung der Knoten mit zwei Farben gibt, sodass je zwei benachbarte Knoten nicht dieselbe Farbe haben.

(a) Es gilt 2COLOR $\in$ P. Geben Sie einen Polynomialzeitalgorithmus an, der entscheidet, ob ein gegebener Graph zweifärbbar ist.

(b) Gegeben ist folgender *falscher* Beweis für die NP-Schwere von 2COLOR:

> Reduziere von 3COLOR. Gegeben sei ein Graph $G = (V, E)$. Konstruiere daraus einen Graph G' mit einem zusätzlichen Knoten v, der zu allen anderen verbunden ist. Dies geht offensichtlich in polynomialer Zeit. Zeige: G besitzt genau dann eine Zweifärbung, wenn G' eine Dreifärbung besitzt.
> „⇒“: Betrachte eine gültige Zweifärbung von G. Weise v die dritte, noch unbenutzte Farbe zu. Das ergibt offensichtlich eine gültige Dreifärbung.
> „⇐“: Betrachte eine gültige Dreifärbung von G'. Da v zu allen anderen Knoten verbunden ist, kann die Farbe von v für keinen Knoten aus V benutzt worden sein. Also wurden für V nur zwei Farben benutzt und wir haben eine gültige Zweifärbung von G.

Wenn dieser Beweis richtig wäre, würde daraus P = NP folgen. Wo liegt der Fehler in dem Beweis?

Lösung:

(a) Wähle einen beliebigen Startknoten s und weise ihm eine beliebige Farbe zu. Starte dann eine Tiefensuche von s aus. Weise dabei jedem erreichten Knoten die jeweils andere Farbe als seinem Elternknoten zu. Wenn ein bereits gefärbter Knoten erreicht wird, der dieselbe Farbe wie sein Elterknoten hat, ist G nicht zweifärbbar.

Sollten nach der Tiefensuche noch Knoten ungefärbt sein, ist der Graph nicht zusammenhängend. Führe dann eine weitere Tiefensuche von einem noch ungefärbten Knoten aus. Wiederhole dies solange, bis alle Knoten gefärbt sind.

(b) Die Reduktion wurde in die falsche Richtung durchgeführt. Es wurde 2COLOR $\propto$ 3COLOR gezeigt statt 3COLOR $\propto$ 2COLOR. Intuitiv bedeutet das, dass 3COLOR „mindestens so schwer“ ist wie 2COLOR – für die NP-Schwere von 2COLOR wäre aber genau die andere Richtung relevant.

Aufgabe 5

(2 + 1 (+ 2) = 3 (5) Punkte)

Für die Definition von NP-Schwere haben wir den Begriff der polynomialen Transformation verwendet. Ein Problem Π ist NP-schwer, wenn für jedes Problem $\Pi' \in$ NP eine polynomiale Transformation $\Pi' \propto \Pi$ existiert.

Allgemeiner ist eine *Transformation* eines Entscheidungsproblems Π_1 in ein Entscheidungsproblem Π_2 definiert als eine Funktion $f\colon D_{\Pi_1} \to D_{\Pi_2}$, sodass $f(I)$ genau dann eine Ja-Instanz von Π_2 ist, wenn I eine Ja-Instanz von Π_1 ist. Für die *polynomiale* Transformation haben wir zusätzlich gefordert, dass f von einer DTM in Polynomialzeit berechnet werden kann.

In dieser Aufgabe sollen Sie untersuchen, warum diese zusätzlichen Einschränkungen notwendig sind. Sei dazu Π_1 ein beliebiges entscheidbares Problem und Π_2 ein beliebiges *nichttriviales* Entscheidungsproblem, d.h. Π_2 besitzt sowohl Ja-Instanzen als auch Nein-Instanzen.

(a) Zeigen Sie: Es gibt eine Transformation von Π_1 zu Π_2, die von einer DTM (in beliebiger Zeit) berechnet werden kann.

(b) ~~Zeigen Sie: Wenn $\Pi_1 \in \mathsf{NP}$ gilt, dann gibt es eine Transformation von Π_1 zu Π_2, die von einer NTM in Polynomialzeit berechnet werden kann.~~

(c) Welche Konsequenzen hätte es jeweils für die Klasse der NP-schweren Probleme, wenn wir statt einer polynomialen Transformation nur

(i) eine Transformation, die von einer DTM berechenbar ist, oder

(ii) ~~eine Transformation, die von einer NTM in Polynomialzeit berechenbar ist,~~

fordern würden?

(d) *Bonusaufgabe:* In der Übung wurden die Komplexitätsklassen L und NL vorgestellt. Diese enthalten genau die Probleme, die sich in deterministisch bzw. nichtdeterministisch logarithmischem Platz lösen lassen.

Wir wollen nun analog zur NP-Schwere den Begriff der NL-Schwere einführen. Wir führen dazu eine neue Transformation $\leq_{\mathsf{NL}}$ ein, deren Eigenschaften noch zu bestimmen sind. Wir nennen ein Problem Π_1 NL-schwer, wenn für jedes Problem $\Pi' \in \mathsf{NL}$ eine solche Transformation $\Pi' \leq_{\mathsf{NL}} \Pi$ existiert.

In der Vorlesung wurde gezeigt: Wenn es ein NP-schweres Problem gibt, das in P liegt, dann gilt $\mathsf{P} = \mathsf{NP}$. Eine analoge Eigenschaft wollen wir auch für NL-Schwere haben: Wenn es ein NL-schweres Problem gibt, das in L liegt, dann gilt $\mathsf{L} = \mathsf{NL}$. Welche Eigenschaften muss die Transformation $\leq_{\mathsf{NL}}$ besitzen, damit dies gilt?

Anmerkung: Die Aufgabenstellungen von 5b und dem zweiten Teil von 5c sind fehlerhaft. Diese Aufgaben wurden dementsprechend gestrichen.

Lösung:

(a) Wähle willkürlich eine feste Ja-Instanz I_J und eine feste Nein-Instanz I_N von Π_2. Definiere dann die Transformation f wie folgt:

$$f(I) := \begin{cases} I_J & I \text{ ist Ja-Instanz von } \Pi_1 \\ I_N & I \text{ ist Nein-Instanz von } \Pi_1 \end{cases}$$

Offensichtlich ist $f(I)$ genau dann eine Ja-Instanz von Π_2, wenn I eine Ja-Instanz von Π_1 ist. Da Π_1 entscheidbar ist, kann eine DTM f berechnen, indem sie Π_1 entscheidet.

(b) ~~Sei f genauso wie in Aufgabenteil (a). Da $\Pi_1 \in \mathsf{NP}$ gilt, ist f von einer NTM in Polynomialzeit berechenbar.~~

Anmerkung: Um $f(I)$ zu berechnen, müsste die NTM in Polynomialzeit entscheiden, ob I eine Nein-Instanz ist. Das ist nur möglich, wenn $\Pi_1 \in$ co-$\mathcal{NP}$ gilt, was nicht vorausgesetzt war.

(c) Es wären alle nichttrivialen Entscheidungsprobleme NP-schwer.

(d) Die Transformation $\leq_{\mathsf{NL}}$ muss von einer DTM mit logarithmischem Platzbedarf berechenbar sein. Sei Π' ein NL-schweres Problem, das in L liegt, d.h. es gibt einen deterministischen Algorithmus mit logarithmischem Platzbedarf, der Π' löst. Dann lässt sich für jedes Problem $\Pi \in \mathsf{NL}$ ebenfalls ein Algorithmus mit logarithmischem Platzbedarf angeben: Da $\Pi \leq_{\mathsf{NL}}$

Π' gilt, lässt sich jede Π-Instanz deterministisch mit logarithmischem Platzbedarf in eine äquivalente Π'-Instanz transformieren. Auf diese kann dann der Algorithmus zum Lösen von Π' angewendet werden.

Aufgabe 6 (5 Punkte)

Das Entscheidungsproblem LERNGRUPPE ist wie folgt definiert: Gegeben sind eine Menge S von n Studenten, die sich auf die TGI-Klausur vorbereiten wollen, und eine Menge T von k klausurrelevanten Themen. Die Studenten wollen sich in 3 Lerngruppen aufteilen.

Zwar haben sich alle Studenten auf alle Themen vorbereitet, aber manche Studenten haben zu manchen Themen fehlerhaftes Wissen. Zum Beispiel sagt Alice: „Ein endlicher Automat ist genau dann NP-vollständig, wenn er einen Fehlerzustand hat." Bob sagt: „Der Äquivalenzklassenautomat entscheidet, ob P und NP äquivalent ist." Alice und Bob haben also fehlerhaftes Wissen zum Thema „endliche Automaten". Wenn sie zusammen in eine Lerngruppe eingeteilt würden, könnten sie die anderen Teilnehmer von ihrem falschen Wissen überzeugen. Deswegen darf es innerhalb jeder Lerngruppe zu jedem Thema nur einen Studenten mit fehlerhaftem Wissen geben.

Formal stellen wir das fehlerhafte Wissen als Abbildung $f\colon S \times T \to \{0,1\}$ dar. Dabei gilt $f(s,t) = 1$ genau dann, wenn Student s zum Thema t fehlerhaftes Wissen hat. Beim Entscheidungsproblem LERNGRUPPE sind also S und T sowie die Abbildung f gegeben. Gesucht ist eine Einteilung von S in drei disjunkte Teilmengen G_1, G_2, G_3, sodass für jedes G_i und jedes Thema t gilt:

$$\sum_{s \in G_i} f(s,t) \leq 1$$

Zeigen Sie, dass LERNGRUPPE NP-vollständig ist. Reduzieren Sie von 3COLOR.

Lösung:
LERNGRUPPE $\in$ NP: Verifiziere für jede Gruppe und jedes Thema, ob höchstens ein Student in der Gruppe fehlerhaftes Wissen zu dem Thema hat. Dies ist in Zeit $\mathcal{O}(nk)$ möglich, also polynomial.

Reduziere 3COLOR auf LERNGRUPPE: Sei $G = (V, E)$ eine 3COLOR-Instanz. Wir konstruieren daraus eine LERNGRUPPE-Instanz wie folgt: $S = V$, $T = E$ und $f(s,t) = 1$ genau dann, wenn Kante t zu Knoten s inzident ist. Offensichtlich lässt sich diese Instanz in linearer (also polynomialer) Zeit konstruieren.

Sei G eine Ja-Instanz von 3COLOR. Dann existiert eine 3-Färbung $c\colon V \to \{R, G, B\}$, sodass keine zwei inzidenten Knoten dieselbe Farbe haben. Die 3-Färbung induziert eine Einteilung der Studenten in drei Gruppen, wo Studenten mit derselben Farbe zusammen gruppiert werden. Jedes Thema enspricht einer Kante (u, v). Also gibt es genau zwei Studenten, die über dieses Thema fehlerhaftes Wissen haben, nämlich diejenigen, die den indizenten Knoten u und v entsprechen. Da u und v in G benachbart sind, wurden ihnen verschiedene Farben zugewiesen, d.h. sie sind in verschiedenen Gruppen. Die Einteilung ist also gültig.

Betrachte umgekehrt eine Ja-Instanz von LERNGRUPPE. Dann existiert eine Einteilung von S in 3 Gruppen, sodass in jeder Gruppe zu jedem Thema höchstens ein Student fehlerhaftes Wissen hat. Die Einteilung induziert eine 3-Färbung von V, die jeder Gruppe eine Farbe zuweist. Betrachte zwei benachbarte Knoten u und v. Die Kante (u, v) entspricht einem Thema, zu dem u und v beide fehlerhaftes Wissen haben. Also wurden sie in verschiedene Gruppen eingeteilt und haben daher verschiedene Farbe. Die Färbung ist also gültig.

Aufgabe 7 (4 Punkte)

Die Komplexitätsklasse EXPTIME ist definiert als die Menge aller Sprachen, die von einer DTM in Exponentialzeit entschieden werden können, d.h. in Zeit $\mathcal{O}(2^{n^c})$ für eine Konstante c. Analog dazu ist NEXPTIME die Menge aller Sprachen, die von einer NTM in Exponentialzeit entschieden werden können.

Zeigen Sie: Wenn P = NP gilt, dann gilt auch EXPTIME = NEXPTIME.

Hinweis: Gegeben eine Sprache $L \in$ NEXPTIME, konstruieren Sie eine Sprache L', die genau dann in polynomialer Zeit entscheidbar ist, wenn L in exponentieller Zeit entscheidbar ist. Versuchen Sie, die Wörter aus L geeignet auf exponentielle Länge zu „strecken".

Lösung:
Offensichtlich gilt EXPTIME $\subseteq$ NEXPTIME. Es bleibt zu zeigen, dass unter der Annahme P = NP gilt: NEXPTIME $\subseteq$ EXPTIME.

Sei L eine Sprache aus NEXPTIME. Es gibt also eine NTM $\mathcal{M}$, die L in Zeit $\mathcal{O}(2^{n^c})$ für eine Konstante c entscheidet. Definiere die Sprache

$$L' = \{w0^{2^{|w|^c}} \mid w \in L\}$$

.

Diese Sprache kann von einer NTM $\mathcal{M}'$ wie folgt in Polynomialzeit erkannt werden: Überprüfe zunächst, ob die Eingabe die Form $w0^{2^{|w|^c}}$ hat. Überprüfe dann durch Simulation von $\mathcal{M}$ auf w, ob $w \in L$ gilt. Wir wissen, dass $\mathcal{M}$ Zeit $\mathcal{O}(2^{|w|^c})$ benötigt. Das ist linear in der Länge der Eingabe von $w0^{2^{|w|^c}}$. Also wird L' von $\mathcal{M}'$ in Polynomialzeit erkannt.

Wegen P = NP existiert auch eine DTM $\mathcal{M}''$, die L' in Polynomialzeit erkennt. Dann entscheidet folgende DTM L in Exponentialzeit: Gegeben eine Eingabe w, füge $0^{2^{|w|^c}}$ an das Ende der Eingabe an und simuliere dann $\mathcal{M}''$ auf $w0^{2^{|w|^c}}$. Also gilt $L \in$ EXPTIME.