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

Übungsblatt 4

Vorlesung Theoretische Grundlagen der Informatik im WS 17/18

Ausgabe 6. Dezember 2017
Abgabe 19. Dezember 2017, 11:00 Uhr (im Kasten im UG von Gebäude 50.34)

Aufgabe 1

(2 + 2 = 4 Punkte)

Betrachten Sie folgende Probleme[1]:

> Das Problem 2Color ist das Problem Color aus der Vorlesung mit festem Parameter $k = 2$.
>
> Bei dem Problem Max2Color ist ein Graph G und eine natürliche Zahl k gegeben. Es wird gefragt, ob es eine 2-Färbung der Knoten von G gibt, so dass mindestens k Kanten von G Endpunkte in beiden Farben haben.

(a) Zeigen Sie dass 2Color in P ist.

(b) Zeigen Sie dass Max2Color NP-vollständig ist.

Lösung:

(a) Wir zeigen 2Color $\propto$ 2Sat. Mit 2Sat $\in \mathsf{P}$ (aus der Übung) folgt dann, dass 2Color $\in \mathsf{P}$.

Sei also G ein Graph, d.h. eine Instanz des Problems 2Color. Bezeichne V die Knotenmenge und E die Kantenmenge von G. Wir definieren eine Instanz (U, C) von 2Sat mit Variablenmenge U und Klauselmenge C wie folgt.

- $U = V$
  Die Knoten von G entsprechen genau den Variablen.
- $C = \bigcup_{u_1 u_2 \in E} \{u_1 \vee u_2, \overline{u_1} \vee \overline{u_2}\}$
  Jede Kante entspricht zwei Klauseln in C, wobei eine Klausel die Veroderung der positiven Literale der Variablen der zugehörigen Knoten ist und die andere Klausel die Veroderung der negative Literale dieser Variablen.

Nun interpretieren wir Wahrheitsbelegungen der Variablen in U als 2-Färbungen der Knoten in V. Ein Knoten u sei rot, wenn die Variable u auf `wahr` gesetzt ist, und u ist blau, wenn Variable u auf `falsch` gesetzt ist. Sei $u_1 u_2$ eine Kante von G. Wenn beide Knoten rot sind, ist die Klausel $\overline{u_1} \vee \overline{u_2}$ in C nicht erfüllt. Wenn beide Knoten blau sind, ist die Klausel $u_1 \vee u_2$

[1] Vgl. Vorlesung 9, 28.11.2017, Folie 14

in C nicht erfüllt. Wenn genau ein Knoten, sagen wir u_1, rot ist und der andere, also u_2, blau ist, so sind beide Klauseln $\overline{u_1} \vee \overline{u_2}$ und $u_1 \vee u_2$ erfüllt.

Folglich entsprechen die erfüllenden Belegungen der 2Sat-Instanz (U, C) genau den zulässigen 2-Färbungen der 2Color-Instanz $G = (V, E)$. Offensichtlich ist die Größe der Instanz (U, C) von 2Sat ziemlich genau zweimal die Größe der Instanz $G = (V, E)$ von 2Color, also ist dies insbesondere eine polynomielle Reduktion.

(b) Zunächst kann für jede 2-Färbung der Knoten von $G = (V, E)$ in $O(|E|)$ Zeit überprüft werden, ob mindestens k Kanten Endpunkte in beiden Farben haben. Max2Color ist also in NP.

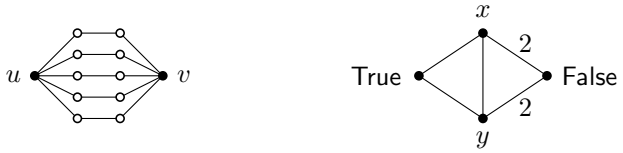
Wir zeigen nun Max2Sat $\propto$ Max2Color. Da Max2Sat NP-vollständig ist (Übung), folgt dann, dass Max2Color auch NP-vollständig ist.

Sei also (U, C, k) eine beliebige Instanz des Problems Max2Sat. Wir definieren eine Instanz $(G = (V, E), k')$ von Max2Color mit Hilfen der folgenden zwei Gadgets. Wir bezeichnen Kanten mit Endpunkten in beiden Farben als *gute Kanten*. Max2Color fragt also, ob eine 2-Färbung der Knoten mit mindestens k guten Kanten gibt.

- Seien u und v zwei Knoten. Wir fügen w unabhängige Pfade der Länge drei zwischen u und v ein und bezeichnen diese Konstruktion als "Kanten mit Gewicht w zwischen u und v". Siehe Abbildung 1 links.

  Dies hat den folgenden Effekt für eine gegebene 2-Färbung der Knoten. Wenn u und v dieselbe Farbe haben, so sind höchstens $2w$ der $3w$ Kanten in dem Gadget gut. Andererseits können die inneren Knoten so gefärbt werden, dass alle $3w$ Kanten gut sind, solange u und v mit verschiedenen Farben gefärbt sind. Wir interpretieren das so, dass wir w gute Kanten erhalten, wenn wir u und v verschieden einfärben. So gesehen sind normale Kanten gleichbedeutend mit Kanten von Gewicht 1.
- Seien x, y, True, False vier Knoten. Wir fügen ein Dreieck mit Endpunkten x, y und True ein, sowie zwei Kanten von Gewicht 2, zwischen False und x beziehungsweise y.

  Dies hat den folgenden Effekt für eine gegebene 2-Färbung der Knoten bei der True und False verschiedene Farben haben. Wenn weder x noch y die Farbe von True erhalten, so erhalten wir genau zwei gute Kanten. Wenn genau einer von x und y die Farbe von True erhält, so erhalten wir genau vier gute Kanten. Wenn sowohl x als auch y die Farbe von True erhält, so erhalten wir genau vier gute Kanten. Wir interpretieren das so, dass wir zwei gute Kanten erhalten, wenn wir mindestens einen von x und y in der Farbe von True färben.



Kante mit Gewicht 5 zwischen u und v

Gadget für Klausel $x \vee y$ mit zwei Gewicht-2-Kanten

Abbildung 1: Illustrationen der beiden Gadgets.

Nun definieren wir eine Instanz $(G = (V, E), k')$ von Max2Color.

- Die Knotenmenge V besteht aus zwei Knoten True und False, zwei Knoten x und $\overline{x}$ für jede Variable $x \in U$, sowie einigen zusätzlichen Knoten die in Kanten von Gewicht w eingeführt werden.
- Wir wählen $w = 5|C| + 1$. Wir fügen Kanten von Gewicht w zwischen True und False, sowie zwischen x und $\overline{x}$ für jedes $x \in U$ ein.

- Wir fügen für jede Klausel $\ell_1 \vee \ell_2$ in C ein Gadget mit den Knoten True, False, ℓ_1 und ℓ_2 ein.
- Wir setzen $k' = 10|C| + 2k + 3w + |U|3w$.

Wir beobachten, dass $|V| = 2+2|U|+8|C|+(|U|+1)2w = O(|C||U|)$ und $k' = O(k+|C||U|)$. Die Instanz (G, k') hat also polynomielle Größe in der Instanz (U, C, k) von Max2Sat.

Es bleibt zu zeigen, dass (G, k') genau dann lösbar ist, wenn (U, C, k) lösbar ist. Sei also zunächst (U, C, k) lösbar und ϕ eine Wahrheitsbelegung von U die mindestens k Klauseln von C erfüllt. Wir färben Knoten True in rot, Knoten False in blau, Knoten x in rot genau dann wenn $\phi(x) =$ True, Knoten $\overline{x}$ in rot genau dann wenn $\phi(x) =$ False, sowie alle Knoten in gewichteten Kanten so dass wir möglichst viele gute Kanten erhalten.

Die Anzahl der guten Kanten ist dann gegeben durch:

- Jeweils $3w$ gute Kanten zwischen True und False, sowie zwischen x und $\overline{x}$. Insgesamt also $3w(|U| + 1)$ gute Kanten.
- Jeweils mindestens zehn gute Kanten für jede Klausel; zusätzlich zwei gute Kanten wenn die Klausel erfüllt ist. Insgesamt also mindestens $10|C| + 2k$ gute Kanten.

Zusammen erhalten wir mindestens $3w(|U| + 1) + 10|C| + 2k = k'$ gute Kanten. Damit ist (G, k') lösbar.

Sei also nun angenommen, dass (G, k') lösbar ist. Betrachte eine 2-Färbung der Knoten von G in rot und blau mit möglichst vielen guten Kanten. Es gibt also mindestens k' gute Kanten. In jedem Klausel-Gadget sind 15 Kanten, also höchstens 15 gute Kanten. Wenn mindestens eines der $|U| + 1$ Knotepaare $(\text{True}, \text{False}), (x, \overline{x})$ nicht unterschiedlich gefärbt ist, so gibt es höchstens $|U|3w + 2w + 15|C| < |U|3w + 2w + 10|C| + w \leq k'$ gute Kanten. (Hier haben wir $w > 5|C|$ benutzt.) Da aber mindestens k' Kanten gut sind, sind alle diese Knotenpaare unterschiedlich gefärbt. O.B.d.A. sei True rot und Knoten False blau. Wir definieren eine Wahrheitsbelegung ϕ durch $\phi(x) =$ True genau dann wenn Knoten x rot ist.

Innerhalb eines Klausel-Gadgets gibt es jetzt genau sechs gute Kanten wenn die Klausel erfüllt wird und vier wenn sie nicht erfüllt wird. Da es mindentens $k' = 3w(|U|+1)+10|C|+2k$ gute Kanten gibt, folgt dass ϕ mindentens k Klauseln erfüllt. Damit ist (U, C, k) erfüllbar und es folgt Max2Sat $\propto$ Max2Color.

Aufgabe 2 (5 Punkte)

In der Vorlesung wurde gezeigt, dass 3Sat NP-vollständig ist. Sie dürfen davon ausgehen, dass dabei niemals eine Variable mehrfach in einer Klausel vorkommt. Sei 4Sat folgendes Problem:

> Sei V Menge an Variablen und $\mathcal{C}$ eine Menge von Klauseln über V, wobei jede Klausel $C \in \mathcal{C}$ aus genau vier unterschiedlichen veroderten Literalen besteht, also z.B. $C = (x_3 \vee \neg x_5 \vee x_6 \vee x_9)$.
> Gibt es eine Wahrheitsbelegung von V, so dass jede Klausel in $\mathcal{C}$ erfüllt ist?

Zeigen Sie, dass 4Sat NP-vollständig ist.

Lösung:
Um zu zeigen, dass 4Sat NP-vollständig ist, gehen wir in zwei Schritten vor:

(a) Sei $(V, \mathcal{C}) \in$ 4SAT. Dann existiert eine erfüllende Wahrheitsbelegung $\varphi : V \to \{\texttt{wahr}, \texttt{falsch}\}$. Für eine geeignete Kodierung $\langle\varphi\rangle$ ist leicht verifizierbar, dass φ tatsächlich eine erfüllende Belegung ist. Dazu wird einfach für jede Klausel $C \in \mathcal{C}$ durch Einsetzen der entsprechenden φ-Werte sicher gestellt, dass C tatsächlich erfüllt ist. Außerdem ist die Länge von $\langle\varphi\rangle$ natürlich polynomial beschränkt in der Größe der Kodierung $\langle(V, \mathcal{C})\rangle$. Damit gilt 4SAT $\in$ NP.

(b) Das Problem 4SAT ist NP-schwer. Sei $I = (V, \mathcal{C})$ eine 3SAT-Instanz. Wir konstruieren daraus in polynomialer Zeit eine 4SAT-Instanz $I' = (V', \mathcal{C}')$ so, dass $I \in$ 3SAT genau dann, wenn $I' \in$ 4SAT. Setze dazu $V' = V \dot\cup \{x_1, x_2, x_3, x_4\}$ und füge für jede Klausel $(\ell_i \vee \ell_j \vee \ell_k) \in \mathcal{C}$ die Klausel $(\ell_i \vee \ell_j \vee \ell_k \vee x_1)$ zu $\mathcal{C}'$ hinzu. Wir werden nun erzwingen, dass für jede erfüllende Wahrheitsbelegung φ' von $(V', \mathcal{C}')$ gilt $\varphi'(x_1) = \texttt{falsch}$. Dazu fügen wir folgende Klauseln zu $\mathcal{C}'$ hinzu, wobei keine Klausel eine Variable mehrfach enthält:

$$
\begin{aligned}
((x_2 \wedge x_3 \wedge x_4) &\implies \neg x_1) \equiv (\neg x_1 \vee \neg x_2 \vee \neg x_3 \vee \neg x_4)\\
((x_2 \wedge x_3 \wedge \neg x_4) &\implies \neg x_1) \equiv (\neg x_1 \vee \neg x_2 \vee \neg x_3 \vee x_4)\\
((x_2 \wedge \neg x_3 \wedge x_4) &\implies \neg x_1) \equiv (\neg x_1 \vee \neg x_2 \vee x_3 \vee \neg x_4)\\
((x_2 \wedge \neg x_3 \wedge \neg x_4) &\implies \neg x_1) \equiv (\neg x_1 \vee \neg x_2 \vee x_3 \vee x_4)\\
((\neg x_2 \wedge x_3 \wedge x_4) &\implies \neg x_1) \equiv (\neg x_1 \vee x_2 \vee \neg x_3 \vee \neg x_4)\\
((\neg x_2 \wedge x_3 \wedge \neg x_4) &\implies \neg x_1) \equiv (\neg x_1 \vee x_2 \vee \neg x_3 \vee x_4)\\
((\neg x_2 \wedge \neg x_3 \wedge x_4) &\implies \neg x_1) \equiv (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)\\
((\neg x_2 \wedge \neg x_3 \wedge \neg x_4) &\implies \neg x_1) \equiv (\neg x_1 \vee x_2 \vee x_3 \vee x_4)
\end{aligned}
$$

Sei nun φ' eine beliebige erfüllende Wahrheitsbelegung von I'. Gemäß der obigen Konstruktion gilt $\varphi'(x_1) = \texttt{falsch}$. Ist eine Klausel $(\ell_i \vee \ell_j \vee \ell_k \vee x_1)$ erfüllt, muss also auch die Klausel (ℓ_i, ℓ_j, ℓ_k) erfüllt sein, d.h. $I' \in$ 4SAT $\implies I \in$ 3SAT.

Sei umgekehrt φ eine beliebige erfüllende Wahrheitsbelegung von I. Erweitere φ zu $\varphi' : V' \to \{\texttt{wahr}, \texttt{falsch}\}$ durch $x_1, x_2, x_3, x_4 \mapsto \texttt{falsch}$. Wegen $\varphi'(x_1) = \texttt{falsch}$ sind alle Klauseln der obigen Liste erfüllt. Da φ eine erfüllende Wahrheitsbelegung von I ist sind alle anderen Klauseln erfüllt. Also gilt $I \in$ 3SAT $\implies I' \in$ 4SAT.

Diese Prozedur fügt eine konstante Anzahl an Variablen und Klauseln hinzu. Jede Klausel der Originalinstanz wird nur einmal betrachtet. Damit ist die gesamte Transformation in polynomialer Zeit durchführbar, und wir folgern, dass 4SAT NP-schwer ist.

Aufgabe 3

(4 + 1 = 5 Punkte)

Bei der *Quadratischen Programmierung* besteht eine Eingabe aus einer Menge von Ungleichungen mit Polynomen von höchstens Grad Zwei (mit ganzzahligen Koeffizienten) über n reellen Variablen $x_1, x_2, \ldots, x_n$. Das Problem QUADPROG ist, zu entscheiden, ob sich die Variablen so reellen Zahlen zuweisen lassen, dass alle Ungleichungen erfüllt sind.

Hier ist ein kleines Beispiel:

$$
\begin{aligned}
x_1 &\geq 0\\
x_2 &\geq 0\\
x_1^2 + x_2^2 &\leq 1\\
9x_1x_2 &\geq 1
\end{aligned}
$$

Diese Instanz ist erfüllbar, zum Beispiel durch $x_1 = x_2 = \frac{1}{2}$. Ersetzt man die letzte Ungleichung jedoch durch $x_1x_2 \geq 1$ gibt es keine erfüllende Belegung.

(a) Zeigen Sie, dass QuadProg NP-schwer ist. Benutzen Sie die NP-Schwere von 3Color, die in der Vorlesung bewiesen wurde.

(b) Um zu zeigen, dass QuadProg NP-vollständig ist, müsste man noch zeigen, dass QuadProg $\in$ NP gilt. Dies ist allerdings nicht trivial. Wieso?

Lösung:

(a) Sei $G = (V, E)$ eine Instanz von 3Color. Konstruiere eine äquivalente Instanz I' von QuadProg wie folgt. Führe für jeden Knoten $v \in V$ drei Variablen v_1, v_2, v_3 ein. Für $i = 1, 2, 3$ möchten wir die Variable v_i als

„Knoten v hat Farbe i genau dann, wenn $v_i < 0$ gilt“

interpretieren. Dazu fügen wir geeignete Ungleichungen ein. Zunächst soll jeder Knoten überhaupt eine Farbe haben:

$$\forall v \in V : v_1 + v_2 + v_3 \leq -1$$

Außerdem darf jeder Knoten $v \in V$ nur eine Farbe haben:

$$v_1 v_2 \leq 0$$
$$v_1 v_3 \leq 0$$
$$v_2 v_3 \leq 0$$

Schließlich dürfen benachbarte Knoten nicht dieselbe Farben haben, d.h. für $(u, v) \in E$:

$$u_1 v_1 \leq 0$$
$$u_2 v_2 \leq 0$$
$$u_3 v_3 \leq 0$$

Sei φ' eine Lösung von I'. Dann induziert die Interpretation

„Knoten v hat Farbe i genau dann, wenn $v_i < 0$ gilt“

gerade eine gültige Färbung von G.

Betrachte umgekehrt eine gültige Färbung $\varphi : V \to \{1, 2, 3\}$ von G. Setze

$$\varphi'(v_i) = \begin{cases} -1 & \text{falls } v \text{ Farbe } i \text{ hat und} \\ 0 & \text{sonst.} \end{cases}$$

Dann gilt für alle obigen Ungleichungen genau Gleichheit, d.h. φ' ist eine Lösung für I'.

Da für die Knoten und Kanten höchstens konstant viele Variablen bzw. Ungleichungen eingeführt werden ist die Transformation in polynomieller Zeit durchführbar. Damit haben wir gezeigt, dass QuadProg NP-schwer ist.

(b) Das Problem ist, dass ein Zeuge für die Erfüllbarkeit „klein genug“ kodiert werden muss. Reelle Zahlen können aber sehr platzaufwändig zu kodieren sein, man denke z.B. an irrationale Zahlen. Es ist deswegen nicht trivial, dass es immer einen „kleinen“ Zeugen gibt.

Aufgabe 4 (1 + 1 + 1 = 3 Punkte)

Doktor Meta forscht in seinem Labor am Geheimproblem X. Er hat bereits eine polynomielle Transformation von SAT auf X gefunden, die SAT-Instanzen der Größe n auf X-Instanzen der Größe n^3 abbildet. In einem Moment der Genialität gelingt es Doktor Meta nun, eine Laufzeitschranke für SAT von $\Omega(c^n)$ für eine Konstante $c > 1$ zu beweisen.

(a) Was folgt aus der Laufzeitschranke für SAT über die Frage $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$?

(b) Welche Laufzeitschranke ergibt sich für X?

(c) Können Sie mit der Laufzeitschranke für X die Frage $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ beantworten?

Begründen Sie Ihre Antworten!

Lösung:

(a) Die Schranke zeigt $\text{SAT} \notin \mathsf{P}$, mit $\text{SAT} \in \mathsf{NP}$ folgt $\mathsf{P} \neq \mathsf{NP}$.

(b) Es ergibt sich eine Laufzeitschranke für X von $\Omega(c^{n^{1/3}})$.

(c) Nein. Es folgt zwar, dass X ein NP-schweres Problem ist, aber man kann nicht ohne Weiteres davon ausgehen, dass X in NP liegt.

Aufgabe 5 (2 + 2 + 2 = 6 Punkte)

Wir habens uns ausführlich mit dem Konzept der NP-Vollständigkeit beschäftigt. Der Grund dafür ist, dass die NP-vollständigen Sprachen die „schwersten“ Sprachen in NP sind.

Dieses Konzept von Vollständigkeit lässt sich erweitern. In dieser Aufgabe sollen Sie einen ähnlichen Vollständigskeitsbegriff für eine andere Menge von Sprachen entwickeln. Es sei noch einmal an die Definition einer Transformation erinnert:

> Eine *Transformation* einer Sprache $L_1 \subseteq \Sigma_1^*$ in eine Sprache $L_2 \subseteq \Sigma_2^*$ ist eine Funktion $f : \Sigma_1^* \to \Sigma_2^*$ mit den Eigenschaften:
>
> - es existiert eine deterministische Turingmaschine M, die f berechnet, und
> - für alle $w \in \Sigma_1^*$ gilt $w \in L_1 \iff f(w) \in L_2$.
>
> Wir schreiben dann $L_1 \leq_M L_2$.

Diese Definition gleicht der Definition einer *polynomialen Transformation* aus der Vorlesung[2] bis darauf, dass hier keine polynomiale Laufzeit zur Berechnung von f gefordert wird[3].

Bezeichne SE die Menge der semi-entscheidbaren Sprachen. Eine Sprache L ist SE-schwer, wenn für jede Sprache L' in SE gilt, dass $L' \leq_M L$. Eine SE-schwere Sprache ist also „mindestens so schwer“ wie die schwersten Probleme in SE.

(a) Zeigen Sie, dass alle SE-schweren Sprachen nicht entscheidbar sind.

[2] Siehe z.B. Folie 2 vom 23. November.

[3] Insbesondere gilt also $L_1 \propto L_2 \implies L_1 \leq_M L_2$, die Umkehrung im Allgemeinen aber nicht.

(b) Zeigen Sie, dass wenn L SE-schwer ist und $L \leq_M L'$ gilt folgt, dass L' ebenfalls SE-schwer ist. *Hinweis:* Beweisen Sie, dass $\leq_M$ transitiv ist.

Eine Sprache L ist SE-vollständig, wenn L SE-schwer ist und $L \in$ SE gilt. Damit ist sichergestellt, dass L einerseits mindestens so schwer die schwersten Probleme in SE ist, und andererseits, dass L höchstens so schwer ist wie die schwersten Probleme in SE. Die Universelle Sprache war folgendermaßen definiert:

$$L_u = \{\langle M, w\rangle \mid M \text{ akzeptiert } w\}$$

(c) Zeigen Sie, dass L_u SE-schwer und sogar SE-vollständig ist. Damit haben Sie gezeigt, dass die Universelle Sprache zu den schwersten semi-entscheidbaren Sprachen gehört!

Lösung:

(a) Angenommen, L wäre eine entscheidbare SE-schwere Sprache. Sei M_1 eine deterministische Turingmaschine, die L entscheidet. Nach Definition gilt für jede semi-entscheidbare Sprache L' die Tranformierbarkeit $L' \leq_M L$, also für jedes $w \in \Sigma^*$ gilt $w \in L' \iff f(w) \in L$. Nun ist aber f berechenbar und L entscheidbar, wodurch sich $w \in L'$ entscheiden lässt. Damit gäbe es keine semi-entscheidbaren Sprachen, die nicht entscheidbar sind. Dies ist ein Widerspruch dazu, dass wir z.B. für die Universelle Sprache beweisen haben, dass sie semi-entscheidbar aber nicht entscheidbar ist.

(b) Sei $L_1 \leq_M L_2$, wobei M die Funktion f berechnet, und $L_2 \leq'_M L_3$, wobei M' die Funktion g berechnet. Setze $h = g \circ f$. Dann gilt

$$w \in L_1 \overset{L_1 \leq_M L_2}{\iff} f(w) \in L_2 \overset{L_2 \leq_M L_3}{\iff} g(f(w)) \in L_3 \overset{h = f \circ g}{\iff} h(w) \in L_3$$

Außerdem ist h eine berechenbare Funktion. Es existiert z.B. eine Turingmaschine M'', die bei Eingabe von w zunächst M simuliert, um $f(w)$ zu berechnen. Danach simuliert M'' die Turingmaschine M' mit Eingabe $f(w)$, um $g(f(w)) = h(w)$ zu berechnen. Damit ist h eine Transformation von L_1 auf L_3, oder $L_1 \leq_{M''} L_3$.

(c) Sei $L \in$ SE und M eine Turingmaschine, die L akzeptiert. Setze $f(w) = \langle M, w\rangle$. In Vorlesung und Übung wurde gezeigt, dass die Kodierung einer Turingmaschine als Gödelnummer eine berechenbare Funktion ist. Außerdem gilt $w \in L$ genau dann, wenn $\langle M, w\rangle \in L_u$. Damit haben wir $L \leq L_u$ gezeigt, d.h. L_u ist SE-schwer.

Aus der Vorlesung ist weiter bekannt, dass L_u eine semi-entscheidbare Sprache ist. Damit ist gezeigt, dass L_u eine SE-vollständige Sprache ist.

Aufgabe 6 (4 Punkte)

In der vorherigen Aufgabe haben Sie gesehen, dass die Universelle Sprache L_u zu den schwersten semi-entscheidbaren Problemen gehört. In der Vorlesung wurde gezeigt, dass die Diagonalsprache L_d nicht semi-entscheidbar ist. Intuitiv würde man deshalb vielleicht vermuten, dass die Diagonalsprache „schwerer“ ist, als die Universelle Sprache. In dieser Aufgabe sollen Sie beweisen, dass ein solcher Schwerevergleich zumindest über Tranformationen nicht möglich ist.

Zeigen Sie dazu, dass $L_d \not\leq_M L_u$ und $L_u \not\leq_M L_d$ gilt, dass die beiden Sprachen also jeweils nicht aufeinander reduzierbar sind.

Lösung:
Angenommen, es wäre $L_d \leq_M L_u$, wobei M die Funktion f berechnet. Dann gilt $w \in L_d \iff f(w) \in L_u$. Die Universelle Sprache L_u ist semi-entscheidbar. Mit der Berechenbarkeit von f wäre dann aber auch L_d semi-entscheidbar[4], was ein Widerspruch dazu ist, dass L_d als nicht semi-entscheidbar konstruiert wurde. Also gilt $L_d \not\leq_M L_u$.

Angenommen, es wäre $L_u \leq_M L_d$. Nach Aufgabe 5 (c) ist L_u SE-vollständig. Nach Aufgabe 5 (b) ist damit L_d ebenfalls SE-vollständig. Da L_d^{c} semi-entscheidbar ist gilt somit insbesondere $L_d^{\mathsf{c}} \leq_M L_d$. Es existiert also eine durch eine TM berechenbare Funktion f, sodass $w \in L_d^{\mathsf{c}} \iff f(w) \in L_d$. Es sei $\mathcal{M}$ die TM, die L_d^{c} semi-entscheidet. Konstruiere eine TM $\mathcal{M}'$, die für Eingabe w nebenläufig $\mathcal{M}$ auf w und $\mathcal{M}$ auf $f(w)$ ausführt. Falls $w \in L_d^{\mathsf{c}}$, so akzeptiert $\mathcal{M}$ das Wort w nach endlich vielen Schritten. Falls $w \notin L_d^{\mathsf{c}}$, so gilt $f(w) \notin L_d$, also akzeptiert $\mathcal{M}$ das Wort $f(w)$ nach endlich vielen Schritten. Die TM $\mathcal{M}'$ kann also L_d^{c} entscheiden.

Aufgabe 7 (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[5] 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.

[4] Siehe auch den Beweis von Aufgabe 5 (b).
[5] am 12.12.2017

- 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 Γ^* 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 für jede Eingabe $x \in \Sigma^*$, $\mathcal{M}'$ 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 $\mathcal{M}$.