Prof. Martin Hofmann
Steffen Jost

Ludwig-Maximilians-Universität München
Institut für Informatik

Wintersemester 2002/03
24. Oktober 2002

Lösungsvorschläge zur 1. Übung

Komplexitätstheorie

Lösung zu G1: Sei $L_1, L_2 \in \mathcal{P}$. Gemäß der Definition von $\mathcal{P}$ gibt es daher deterministische Turing Maschinen M_1, M_2 mit $L(M_i) = L_i$ deren Laufzeit auf eine Eingabe der Größe n jeweils durch die Polynome $p_1(n)$ und $p_2(n)$ beschränkt ist.

$L_1 \cap L_2$: Wir konstruieren eine deterministische Turing Maschine M, die zuerst ihre Eingabe auf einem zusätzlichen Band sichert und danach M_1 simuliert. Falls M_1 verwirft, beenden wir die Berechnung ebenfalls mit Ablehnung. Ansonsten wird die Berechnung von M_2 auf die Kopie der Eingabe simuliert. Endet diese mit Annahme, akzeptiert auch M und verwirft sonst. Somit akzeptiert M nur solche x, für die sowohl $x \in L_1$ als auch $x \in L_2$ gilt.

Für das Kopieren und Zurückspulen der Eingabe werden $2n$ Schritte benötigt, die Simulation von M_i benötigt jeweils höchstens $p_i(n)$ Schritte, also insgesamt höchstens $2n + p_1(n) + p_2(n)$ Schritte. Dies ist offensichtlich ein Polynom in n, somit $L(M) = L_1 \cap L_2 \in \mathcal{P}$.

$L_1 \cup L_2$: Analog zum vorhergehenden Fall. Nach der Simulation von M_1 wird allerdings M_2 nur dann simuliert, wenn M_1 nicht bereits akzeptierte. Ansonsten wird die Berchnung ebenfalls beendet, allerdings im akzeptierenden Endzustand.

Bemerkung: Die Anzahl der Bänder spielt in der Definition von $\mathcal{P}$ keine Rolle, für die Konstruktion von M dürfen wir also beliebig (konstant) viele Bänder hinzunehmen. Dies wird durch Aufgabe G4 gerechtfertigt.

Lösung zu G2: Aus $L' \in \mathcal{P}$ folgt wiederum die Existenz einer deterministischen Turing Maschine M' mit polynomialer Laufzeitbeschränkung, die L' erkennt. Damit können wir eine deterministische Turing Maschine $\hat{M}$ konstruieren, welche L in polynomialer Laufzeit ohne Verwendung eines Orakels erkennt. Daraus folgt $L \in \mathcal{P}$.

$\hat{M}$ wird folgendermaßen konstruiert: Zuerst wird M simuliert. Bei Erreichen des Orakelzustandes von M wird jedoch nun M' mit dem ehemaligen Orakelband als Eingabe simuliert. Weiterhin benutzen wir für die Simulation von M' nur zuvor ungenutzte Bänder, die nach der Benutzung wieder gelöscht werden. Entsprechend des Ausgangs der Simulation von M' setzen wir die Simulation von M anschließend fort.

Es sei die Laufzeit von M auf x durch das Polynom $p(|x|)$ beschränkt; sowie die Laufzeit von M' auf y durch $q(|y|)$. M kann bei der Berechnung von x das Orakel höchstens $p(|x|)$-mal aufrufen. Jeder dieser Aufrufe benötigt bei der Simulation in $\hat{M}$ höchstens $q\big(p(|x|)\big)$ Schritte für die Simulation von M', da die Eingabe auf dem Orakelband höchstens $p(|x|)$

Zeichen enthalten kann. Nochmals soviele Schritte sind zum Aufräumen der Bänder nach der Beendung der Simulation von M' notwendig. Insgesamt ergibt sich also eine Laufzeitbeschränkung für $\hat{M}$ von $2p\big(|x|\big)q\big(p(|x|)\big)$, was offensichtlich ein Polynom in $|x|$ ist.

Lösung zu G3 (Wiederholung): Eine Implementation einer deterministischen 2-Band Turing Maschine, welche L erkennt ist (beide Bänder sind semi-infinit wie üblich): Q bezeichene die Zustandsmenge, s_I bzw. s_A bezeichne das gelesene Zeichen auf dem Input- bzw. Arbeitsband, s'_I und s'_A bezeichnen das zu schreibende Zeichen auf dem jeweiligen Band an der aktuellen Position und m_I, m_A die anschließend durchzuführenden Kopfbewegungen.

Q	s_I	s_A	$\to$	Q	s'_I	m_I	s'_A	m_a
0	0	$\Box$		3	0	R	€	S
0	1	$\Box$		1	€	R	€	R
1	0	$\Box$		2	€	R	1	S
1	1	$\Box$		1	€	R	1	R
2	1	1		2	€	R	1	L
2	1	€		3	1	S	€	R
2	$\Box$	€		A	€	S	€	S
3	1	1		3	€	R	1	R
3	1	$\Box$		2	1	S	€	L
3	1	€		2	1	S	€	L
3	$\Box$	€		A	€	S	€	S

wobei 0 der Anfangszustand und A der akzeptierende Zustand sei. Alle undefinierten Tripel gehen sofort in den ablehenden Zustand über und wurden der Übersichtlichkeit geopfert.

Lösung zu G4: Wir erweitern das Arbeitsalphabet Σ von M zu $\Sigma' = \Sigma \cup \big(\Sigma \times \{0,1\} \times \Sigma \times \{0,1\}\big)$. Die Idee dabei ist, daß ein quadrupel (s_I, k_I, s_A, k_A) den Bandinhalt von je einem Feld des Input- *und* des Arbeitsbandes enthält (s_I, s_A) und durch k_I, k_A codiert ob sich der Kopf des jeweiligen Bandes gerade auf dieser Bandposition befindet oder nicht. Beachte das $\Box \neq (\Box, 0, \Box, 0)$, d.h. bei der Codierung können wir Leerzeichen "schreiben" wenn wir wollen oder müssen.

- Als erstes beginnt M' damit den Input zu codieren und den Kopf wieder an den Anfang zurück zu fahren.
- Wir scannen den beschriebenen Teil des Bandes von links nach rechts durch und merken uns die Zeichen unter dem jeweiligen Kopf. Die Position der Köpfe erkennen wir aus der Codierung.
- Damit konnen wir einen Schritt von M vorhersagen. Das Ergebnis (Die Kopfbewegungen und die Veränderungen an den beiden Bändern) schreiben wir auf das Band indem wir erneut den kompletten beschriebenen Teil des Bandes abfahren, diesmal von rechts nach links. Sollte die Operation von M ein oder zwei neue Felder (maximal 1 Feld pro Band) beschrieben haben, so tun wir dies natürlich ebenfalls.
- Endet die Simulation von M mit akzeptieren, akzeptieren wir ebenfalls und lehnen ansonsten ab.

Das Codieren des Inputs benötigt bei einer Eingabe der Länge n genau n Schritte plus nochmals soviele zum zurückspulen. Da die Laufzeit von M durch $O\big(f(n)\big)$ beschränkt ist und bei semi-infiniten Bändern nur eine Laufrichtung zur Verfügung steht, ist die Bandlänge des codierten Bandes ebenfalls durch $O\big(f(n)\big)$ beschränkt. Für jeden einzelnen Schritt von M benötigen wir daher maximal zweimal soviel Schritte wie die momentan beschriebene Bandlänge zur Simulation, da wir zweimal den gesamten beschriebenen Teil ablaufen müssen. Aus der Laufzeitbeschränkung von M folgt das maximal $O\big(f(n)\big)$ Schritte simuliert werden müssen. Somit ergibt sich für die Laufzeit von M': $O\big(2f(n)^2\big) = O\big(f(n)^2\big)$.

Lösung zu G5 (Linear Speed-up Theorem):

a) Ohne weitere Überlegungen lassen sich j viele Schritte von M nicht in einem einzigen Schritt von M' zusammengefassen. Beispielsweise könnte M gerade ständig zwischen den Zeichen j und $j+1$ hin- und herspringen und dort lesen und schreiben, so daß auch M' gleich oft zwischen Zeichen eins und zwei springen mußte. Die Anzahl solcher Sprünge zwischen zwei Blöcken läßt sich dabei nur durch $O\big(f(n)\big)$ eingrenzen.

b) Die Konstruktion M' folgt jedoch grundsätzlich dieser Idee. Wir wählen zuerst $m \in \mathbb{N}$ mit $m\epsilon \geq 7$ und erweitern das Arbeitsaplhabet Σ von M auf $\Sigma \cup \Sigma^m$ für M', so daß wir m Zeichen in der Eingabe für M durch ein einziges Zeichen codieren können. Die Anzahl der Bänder bleibt gleich ($k \geq 2$).

- Zuerst kopiert und komprimiert M' die Eingabe auf ein zweites Band. Dies benötigt n Schritte um die Eingabe zu lesen und $\lceil \frac{n}{m} \rceil$ um den Kopf des Bandes mit der komprimierten Kopie wieder zurückzufahren. Dieser Schritt ist auch der Grund für die Einschränkung auf $k \geq 2$, da das Kopieren ansonsten quadratische Zeit benötigen würde.
- Jetzt wird M simuliert, wobei konstant m Schritte von M durch 8 Schritte von M' wie folgt ausgedrückt werden:
  - Zuerst liest M' die Informationen aller Köpfe, sowie deren beider Nachbarfelder und "speichert" diese Informationen in einem entsprechendem Zustand. Dafür sind 4 Schritte notwendig: 1 mal links, 2 mal rechts und wieder zurück nach links. (In unserem Modell der Turing Maschine bewegen sich ja alle Köpfe gleichzeitig in einem Schritt.)
  - Nun können m Schritte von M in einem einzigen Schritt von M' simuliert werden: In m Schritten kann jeder Kopf von M sich höchstens um m Felder bewegen, wobei der aktuelle Zustand von M' jedoch mindestens m Felder nach links und rechts eines jeden Kopfes von M kennt. Damit ist alles Notwendige bekannt um m Schritte von M vollständig ohne Kopfbewegungen zu berechnen.

    Um jedoch alle von M in m Schritten durchgeführten Änderungen einzutragen sind zwei weitere Schritte notwendig, da jeder Kopf von M maximal zwei verschiedene Blöcke der Größe m benutzt haben kann und sich am Ende in einem beliebigem dieser beiden Blöcke aufhält. Somit sind drei weitere Schritte notwendig, insgesamt also sieben.

- Sobald M irgendwann terminiert, beendet sich M' ebenfalls mit dem gleichen Endzustand.

Wir fassen zusammen das für eine Eingabe x der Länge n gilt:

$$\mathrm{TIME}_{M'}(x) \leq n + \left\lceil \frac{n}{m} \right\rceil + 7 \cdot \left\lceil \frac{\mathrm{TIME}_M(x)}{m} \right\rceil \leq n + \frac{n}{m} + 7 \cdot \frac{\mathrm{TIME}_M(x)}{m} + 2$$

Lösung zu G6 (Gap Theorem):

a) Gemäß den Voraussetzungen gibt es nur endlich viele Inputlängen n, für die sich die Berechnungszeit von T nicht durch $f(n)$ abschätzen läßt. Die bedeutet das aber auch das es nur endlich viele Eingaben x gibt mit $\mathrm{TIME}_M(x) > f(|x|)$. Bezeichnen wir diese endliche Menge mit M.

   Die endlich vielen Information, wie T mit einem Input aus M verfährt, können wir daher in die Zustandsmenge hineinkodieren und so eine Maschine T' konstruieren mit $L(T') = L(T)$, welche jedoch immer in höchstens $f(|x|)$ Schritten terminiert, da T' sofort mit der korrekten Antwort terminiert, sobald erkannt wird das $x \in M$ gelten muß.

b) Um zu entscheiden, ob ein gegebenes Tripel (i, n, t) in H liegt oder nicht, muß man lediglich T_i auf allen möglichen Eingaben der Länge n (das sind ja nur endlich viele) simulieren, wobei man jede Simulation nach $t + 1$ Schritten abbricht, sofern sie noch nicht beendet war (man kann z.B. auf einem seperaten Band einen Zähler mitlaufen lassen). Somit kann man in endlicher Zeit entscheiden ob $(i, n, t) \in H$.

c) Wir definieren $f(n) := \min\left\{t \;\middle|\; \forall i \leq n.(i, n, t) \in H \vee (i, n, 2^t) \notin H\right\}$ Nach Abschnitt b) und der Einschränkung auf $i \leq n$ haben wir es mit einer entscheidbaren Formel zu tun. Die Funktion ist somit berechenbar, sofern das Minimum existiert.

   Dieses Minimum muß existieren, denn eine obere Schranke für t ist die längste Zeit, die alle *terminierenden* der n-vielen betrachteten Turingmaschinen jeweils maximal benötigen um eine beliebige Eingabe der Länge n zu bearbeiten.

   Für alle Zweifler an der Mathematik sei angemerkt das ein Pseudocode, einen Entscheider für H vorausgesetzt, leicht anzugeben ist.

d) Die Funktion aus der vorhergehenden Teilaufgabe ist gerade eine solche: Sei $L \in \mathrm{DTIME}\big(2^{f(n)}\big)$. Wähle ein $i \in \mathbb{N}$ mit $L = L(T_i)$. Damit folgt $\forall n.(i, n, 2^{f(n)}) \in H$. Nach der Konstruktion von f gilt dann aber auch bereits $\forall n \geq i.(i, n, f(n)) \in H$ und somit $L \in \mathrm{DTIME}\big(f(n)\big)$, denn die endlich vielen $n < i$ fallen nach dem Lemma aus Abschnitt a) nicht ins Gewicht!

   Bemerkung: Die Wahl des Sprunges von $f(n)$ auf $2^{f(n)}$ war völlig willkürlich. Das Theorem läßt sich beispielsweise auch analog für einen Sprung von $f(n)$ auf $3f(n)^4$ beweisen.

e) Das Theorem besagt, daß es beliebig große Lücken in der Hierarchie der Zeitkomplexität gibt, wenn man bei der Wahl der Komplexitätsfunktionen beliebige Funktionen zuläßt.

   Es sei noch angemerkt das es ein anloges Theorem auch für die Platzkomplexität gibt, welche wir später noch kennenlernen werden.