Theoretische Informatik (WS04)

Prof. Dr. Raimund Seidel
Wei Ding, Dierk Johannes
Lösungsvorschlag zu Aufgabenblatt 11

Aufgabe 1

a) Wir definieren f und g wie folgt:

$$f(n) = \begin{cases} n^3 & \text{falls } n \text{ ungerade} \\ n & \text{falls } n \text{ gerade} \end{cases}$$

$$g(n) = \begin{cases} n^3 & \text{falls } n \text{ Primzahl} \\ n^2 & \text{sonst} \end{cases}$$

1) **Behauptung:** $f \notin \mathcal{O}(g)$
Beweis: Angenommen, es gäbe ein $n_0 \in N$ und $c > 0$ mit der Eingenschaft, daß $f(n) \leq c \cdot g(n)$ für alle $n \geq n_0$. Dann setzen wir n' auf eine ungerade Zahl größer als c, die keine Primzahl ist, und erhalten $f(n') = n'^3 > c \cdot n^2 = c \cdot g(n)$, was ein Widerspruch zur Annahme ist.

2) **Behauptung:** $g \notin \mathcal{O}(f)$
Beweis: Angenommen, es gäbe ein $n_0 \in N$ und $c > 0$ mit der Eingenschaft, daß $g(n) \leq c \cdot f(n)$ für alle $n \geq n_0$. Dann setzen wir n' auf eine gerade Zahl größer als c und erhalten $g(n') = n'^2 > c \cdot n = c \cdot f(n)$, was ein Widerspruch zur Annahme ist.

Bemerkung: Eine erste Idee ist, sich die gesuchten Funktionen aus $\sin(x)$ und $\cos(x)$ zusammenzubauen. Hierbei ist zweierlei zu berücksichtigen: 1. Wir haben uns auf Funktionen beschränkt, die fast überall nichtnegativ sind. 2. Die Nullstellen müssen bei natürlichen Zahlen liegen.

b) Es sei $gerade(n) = 1$, wenn n gerade ist und $gerade(n) = 0$ sonst. Die Funktion $ungerade(n)$ definieren wir analog.

Betrachte $f(n) = n^{n+gerade(n)}$ und $g(n) = n^{n+ungerade(n)}$. Es gilt $f(n) \neq g(n)$ für $n \geq 2$ und $f(n), g(n) \in \{n^n, n^{n+1}\}$. Genauer: Für $n \geq 2$ und n gerade ist $f(n) = ng(n)$ und für n ungerade ist $g(n) = nf(n)$.

Beide Funktionen sind streng monoton steigend. Wir zeigen dies für $f(n)$. Es ist $f(n+1) = (n+1)^{n+1+gerade(n+1)} \geq (n+1)^{n+1} = (n+1)^n \cdot (n+1)^1 > n^n \cdot n^1 > n^n \cdot n^{gerade(n)} = f(n)$.

Es gilt $f \notin \mathcal{O}(g)$ und $g \notin \mathcal{O}(f)$. Wir zeigen die erste Aussage, die zweite folgt analog. Zu zeigen ist: $\forall c > 0\ \forall n_0\ \exists n \geq n_0\ f(n) > cg(n)$. Bei gegebenen c und n_0 wählen wir n als gerade Zahl größer als $\max\{2, c, n_0\}$. Dann gilt $f(n) = ng(n) > cg(n)$.

Aufgabe 2

Die richtige Anordnung für diese Funktionen ist wie folgt:

$$O(\log n) \subseteq O(\sqrt{n}) \subseteq O(n \log n) \subseteq O(n^{\frac{4}{3}}) \subseteq O(n^2 \log n) \subseteq O(n^{2+\epsilon}) \subseteq O(3^{\sqrt{n}}) \subseteq O(2^n)$$

Begründung:
Aufgrund der Transitivität von „$\subseteq$“ genügt es, die Korrektheit der Anordnung einer Funktion bzgl. ihrer direkten Nachbarn in obiger Auflistung zu zeigen.

$$\lim_{n\to\infty} \frac{\log n}{\sqrt{n}} \overset{l'Hospital^*}{=} \lim_{n\to\infty} \frac{\frac{1}{n}}{\frac{1}{2}n^{-\frac{1}{2}}} = \lim_{n\to\infty} \frac{2}{\sqrt{n}} = 0 \Rightarrow O(\log n) \subseteq O(\sqrt{n}) \quad (1)$$

$$\lim_{n\to\infty} \frac{\sqrt{n}}{n\log n} = \lim_{n\to\infty} \frac{1}{\sqrt{n}\log n} = 0 \Rightarrow O(\sqrt{n}) \subseteq O(n\log n) \quad (2)$$

$$\lim_{n\to\infty} \frac{n\log n}{n^{\frac{4}{3}}} = \lim_{n\to\infty} \frac{\log n}{n^{\frac{1}{3}}} \overset{l'Hospital^*}{=} \lim_{n\to\infty} \frac{\frac{1}{n}}{\frac{1}{3}n^{-\frac{2}{3}}} = \lim_{n\to\infty} \frac{3}{\sqrt[3]{n}} = 0 \Rightarrow O(n\log n) \subseteq O(n^{\frac{4}{3}}) \quad (3)$$

$$\lim_{n\to\infty} \frac{n^{\frac{4}{3}}}{n^2\log n} = \lim_{n\to\infty} \frac{1}{n^{\frac{2}{3}}\log n} = 0 \Rightarrow O(n^{\frac{4}{3}}) \subseteq O(n^2\log n) \quad (4)$$

$$\lim_{n\to\infty} \frac{n^2\log n}{n^{2+\epsilon}} = \lim_{n\to\infty} \frac{\log n}{n^{\epsilon}} \overset{l'Hospital^*}{=} \lim_{n\to\infty} \frac{\frac{1}{n}}{\epsilon n^{\epsilon-1}} = \lim_{n\to\infty} \frac{1}{\epsilon n^{\epsilon}} = 0 \Rightarrow O(n^2\log n) \subseteq O(n^{2+\epsilon}) \quad (5)$$

$$\lim_{n\to\infty} \frac{n^{2+\epsilon}}{3^{\sqrt{n}}} = 0 \Rightarrow O(n^{2+\epsilon}) \subseteq O(3^{\sqrt{n}}) \quad (6)$$

Bleibt noch zu zeigen, daß $O(3^{\sqrt{n}}) \subseteq O(2^n)$ gilt. Der Ansatz über Grenzwerte scheint hier nicht vielversprechend zu sein, deshalb soll nun ein anderer Ansatz demonstriert werden, und zwar gemäß folgender Aussage:
$(\exists n_0 \in \mathbb{N}, \exists c > 0 : \forall n \geq n_0 : 3^{\sqrt{n}} \leq 2^n) \Rightarrow 3^{\sqrt{n}} \in O(2^n)$
Dazu wählt man $n_0 = 4$ und $c = 1$ und zeigt den Rest durch Induktion:
Induktionsanfang $n = n_0 = 4$:
$3^{\sqrt{n}} = 3^{\sqrt{4}} = 3^2 = 9 < 16 = 2^4 = 2^n$
(Die Aussage gilt auch schon für $n_0 = 3$, jedoch hat man dabei keine so runden Zahlen.)
Induktionsschritt $n \to n+1$:
Wir zeigen diesen Schritt von rechts nach links, also beginnend bei 2^{n+1}. Die Vergleichsrelationen drehen sich entsprechend mit um!
$2^{n+1} = 2 \cdot 2^n \overset{IV}{>} 2 \cdot 3^{\sqrt{n}} = 3^{\log_3 2} \cdot 3^{\sqrt{n}} > 3^{0,63} \cdot 3^{\sqrt{n}} = 3^{\sqrt{n}+0,63} > 3^{\sqrt{n+1}}$
Letztere Ungleichung gilt schon für $n = 1$:
$\sqrt{1} + 0,63 = 1,63 > 1,414 \approx \sqrt{2}$ und somit erst recht für größere n.
Insgesamt folgt die Behauptung.

Alternative: $3^{\sqrt{n}} = 2^{\sqrt{n}\log 3}$. Hierbei ist $f(x) = 2^x$ eine streng monoton wachsende Funktion. Falls $x_1 \leq x_2$ gilt, dann auch $2^{x_1} \leq 2^{x_2}$. Zeige also, daß $\sqrt{n}\log 3 \leq n$ ab einer Stelle n_0 gilt.

$*$ Es gilt:

$$\forall \epsilon > 0 : \lim_{n\to\infty} \log n = \lim_{n\to\infty} n^{\epsilon} = \infty$$

Insbesondere gilt die Aussage auch für $\epsilon = \frac{1}{3}$ und $\epsilon = \frac{1}{2}$. Außerdem sind alle diese Funktionen auf $\mathbb{R}^+$ stetig und differenzierbar. Also ist der Satz von l'Hospital anwendbar.

Theoretische Informatik (WS04)
Prof. Dr. Raimund Seidel
Wei Ding, Dierk Johannes
Lösungsvorschlag zu Aufgabenblatt 11

Aufgabe 3

Zu zeigen ist: $g(n) \in \Theta(f(n)) \Leftrightarrow f(n) \in \Theta(g(n)) \Leftrightarrow O(g(n)) = O(f(n))$.
Wir zeigen zunächst $g(n) \in \Theta(f(n)) \Leftrightarrow O(g(n)) = O(f(n))$
und dann $f(n) \in \Theta(g(n)) \Leftrightarrow O(g(n)) = O(f(n))$.

- $f(n) \in \Theta(g(n)) \Leftrightarrow O(g(n)) = O(f(n))$
  - "$\Rightarrow$": $O(f(n)) = O(g(n)) \Rightarrow f(n) \in O(g(n)) \wedge g(n) \in O(f(n))$
    $\Rightarrow f(n) \in \Theta(g(n))$
  - "$\Leftarrow$": $f(n) \in \Theta(g(n))$
    - $\Rightarrow h(n) \in O(f(n))$
      $\Rightarrow \exists c \in \mathbb{N} : h(n) \leq c * f(n) \leq c' * g(n)$
      $\Rightarrow h(n) \in O(g(n))$
    - $\Rightarrow h(n) \in O(g(n))$
      $\Rightarrow \exists c \in \mathbb{N} : h(n) \leq c * g(n) \leq c' * f(n)$
      $\Rightarrow h(n) \in O(f(n))$

      $\Rightarrow O(f(n)) = O(g(n))$
- $g(n) \in \Theta(f(n)) \Leftrightarrow O(f(n)) = O(g(n))$
  - "$\Rightarrow$": $O(g(n)) = O(f(n)) \Rightarrow g(n) \in O(f(n)) \wedge f(n) \in O(g(n))$
    $\Rightarrow g(n) \in \Theta(f(n))$
  - "$\Leftarrow$": $g(n) \in \Theta(f(n))$
    - $\Rightarrow h(n) \in O(g(n))$
      $\Rightarrow \exists c \in \mathbb{N} : h(n) \leq c * g(n) \leq c' * f(n)$
      $\Rightarrow h(n) \in O(f(n))$
    - $\Rightarrow h(n) \in O(f(n))$
      $\Rightarrow \exists c \in \mathbb{N} : h(n) \leq c * f(n) \leq c' * g(n)$
      $\Rightarrow h(n) \in O(g(n))$

      $\Rightarrow O(g(n)) = O(f(n))$

Aufgabe 4 (Zusatzaufgabe)

Diese Äquivalenz ist richtig.

„$\Rightarrow$“: Es gelte $f \in \Theta(g)$ bzw. $g \in \Theta(f)$, was nach Aufgabe 3 äquivalent dazu ist. Wir zeigen $o(f) = o(g)$. Dies geschieht in zwei Schritten: $o(f) \subseteq o(g)$ und $o(g) \subseteq o(f)$. Wir zeigen nur die erste Behauptung, die zweite ist symmetrisch und wird genauso wie diese bewiesen. Es sei also $h \in o(f)$. Wir zeigen $h \in o(g)$.

Nach Voraussetzung ist $f \in \mathcal{O}(g)$, d. h. es gibt ein $c' > 0$ mit $f(n) \leq_{\text{fü}} c' \cdot g(n)$. Die Bedingung $h \in o(f)$ bedeutet, daß $h(n) \leq_{\text{fü}} c'' \cdot f(n)$ für alle $c'' > 0$ gilt. Folglich ist $h(n) \leq_{\text{fü}} c'' \cdot f(n) \leq_{\text{fü}} c'c'' \cdot g(n)$. Zu gegebenem $c > 0$ wähle $c'' = c/c'$. Damit gilt $h(n) \leq_{\text{fü}} c \cdot g(n)$, also $h \in o(g)$.

„$\Leftarrow$“: Wir zeigen die Behauptung $o(f) = o(g) \Rightarrow f \in \mathcal{O}(g)$. Dies impliziert $\mathcal{O}(f) \subseteq \mathcal{O}(g)$. Durch Vertauschen von f und g folgt dann auch $\mathcal{O}(g) \subseteq \mathcal{O}(f)$, zusammen also $\mathcal{O}(f) = \mathcal{O}(g)$. Obige Behauptung ist äquivalent zu $f \notin \mathcal{O}(g) \Rightarrow o(f) \neq o(g)$.

Die Bedingung $f \notin \mathcal{O}(g)$ bedeutet, daß

$$f(n) >_{\text{uo}} c \cdot g(n). \tag{7}$$

Hierbei bedeute das Zeichen $>_{\text{uo}}$ größer für unendlich viele Werte (unendlich oft).

Um $o(f) \neq o(g)$ zu zeigen, konstruieren wir im folgenden eine Funktion h mit $h \in o(f)$ und $h \notin o(g)$. Dazu definieren wir zunächst

$$\alpha(n) = \max\left\{ \frac{f(i)}{g(i)} \;\middle|\; i \leq n \right\}.$$

Es gelten folgende Aussagen über α:

1. Es gilt $\alpha(n) \geq f(n)/g(n)$. Dies folgt aus der Definition.
2. Die Funktion α ist monoton steigend. Dies folgt ebenfalls aus der Definition:
$$\begin{aligned} \alpha(n) &= \max\{f(i)/g(i) \mid i \leq n\} \\ &= \max\{\max\{f(i)/g(i) \mid i \leq n-1\}, f(n)/g(n)\} \\ &= \max\{\alpha(n-1), f(n)/g(n)\}. \end{aligned}$$
3. Für unendlich viele n gilt $\alpha(n) = f(n)/g(n)$.
Nehmen wir an, dies gelte nur für endlich viele Werte und n_0 sei der größte. Dann ist $\alpha(n_0 + 1) = \alpha(n_0)$ und per vollständiger Induktion folgt $\alpha(n) = \alpha(n_0)$ für alle $n > n_0$ gilt. Also gibt es eine Konstante c mit $f(n) < \alpha(n_0)g(n) \leq_{\text{fü}} cg(n)$ für alle $n > n_0$. Das steht im Widerspruch zu $f \notin \mathcal{O}(g)$.
4. Es gilt $\lim_{n\to\infty} \alpha(n) = \infty$.
Zunächst gilt wegen Gleichung (7) $\limsup_{n\to\infty} \alpha(n) > c$ für jedes $c > 0$, also $\limsup_{n\to\infty} \alpha(n) = \infty$. Da α monoton steigend ist, folgt daraus $\lim_{n\to\infty} \alpha(n) = \infty$.

Jetzt definieren wir $h(n) = f(n)/\alpha(n)$.

(a) Wir zeigen $h \in o(f)$.
Es ist $h(n) = f(n)/\alpha(n) \leq 1/\alpha(n) \cdot f(n)$. Da α monoton steigend und $\lim_{n\to\infty} \alpha(n) = \infty$ ($\lim_{n\to\infty} 1/\alpha(n) = 0$) ist, gilt $1/\alpha(n) \leq_{\text{fü}} c$ für alle $c > 0$. Deshalb ist $h(n) \leq 1/\alpha(n) \cdot f(n) \leq_{\text{fü}} c \cdot f(n)$, also $h \in o(f)$.

(b) Wir zeigen $h \notin o(g)$.
Gesucht ist eine Konstante $c > 0$ mit $h(n) >_{\text{uo}} c \cdot g(n)$. Wegen $\alpha(n) = f(n)/g(n)$ für unendlich viele n, gilt trivialerweise $f(n)/g(n) \leq_{\text{uo}} \alpha(n)$. Daraus folgt $h(n) = f(n)/\alpha(n) \leq_{\text{uo}} f(n)/(f(n)/g(n)) \leq_{\text{uo}} g(n) <_{\text{uo}} 2g(n)$.

Aufgabe 5

Wir präsentieren hier nur die Idee. Detailierter ist dieses Vorgehen beschrieben in dem Buch *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie* (Abschnitt 12.2, Lineare Beschleunigung ...) von J. Hopcroft und J. Ullmann.

Sei M eine Turing Maschine die eine Eingabe der Länge n in $\leq f(n)$ Schritten entscheidet. Es ist zu zeigen, dass es eine Turing Maschine M' gibt, die bei einer Eingabe der Länge n weniger als $n + \epsilon \cdot f(n)$ Schritte braucht, wobei wir uns auf $1 > \epsilon > 0$ beschränken können.

Beweis (konstruktiv):

Wähle ein genügend grosses k, (beispielsweise $\frac{3}{k} < \epsilon$).

Die Maschine M' hat ein Band mehr als M, dieses Band dient als „Pseudoeingabeband“. Desweiteren hat M' ein grösseres Alphabet und eine grössere Zustandsmenge. M' fasst nun k aufeinanderfolgende Symbole auf einem Band zu einem neuen Symbol zusammen ($\Gamma' = \Gamma^k$). Das ist der Grund warum man ein grösseres Alphabet braucht.

Die Maschine arbeitet nun folgendermassen:

1. Die Eingabe wird komprimiert und auf das Pseudoeingabeband kopiert.

2. M wird simuliert, wobei k Schritte von M nun in unserem Beispiel durch nur noch 3 Schritte von M' erledigt werden können.

Der Faktor k kann so gross gewählt werden, dass $\frac{c}{k} < \epsilon$ für beliebig grosses c gilt. Auf diese Weise können TM um jeden konstanten Faktor beschleunigt werden.