Prof. Martin Hofmann
Steffen Jost

Ludwig-Maximilians-Universität München
Institut für Informatik

Wintersemester 2002/03
23. Januar 2003

Übung 11

Komplexitätstheorie

Aufgabe G37 (LFPL):

a) Schreibe eine LFPL Funktion f, die ausschließlich eine Liste von boolschen Werten als Eingabe bekommt und eine gleich lange Liste von Tripeln boolscher Werte zurückliefert, so dass jedes Tripel dreimal den gleichen boolschen Wert enthält wie das korrespondierende Element der Eingabeliste.

   Beispiel: $f\big([0,1,0,1,1]\big) = \big[(0,0,0),(1,1,1),(0,0,0),(1,1,1),(1,1,1)\big]$

   Beachte dabei, dass LFPL linear typsiert ist, d.h. eine Variable darf nur einmal benutzt werden!

b) Jede Akzeptor Turing Maschine mit linearem Platzverbrauch (und zusätzlichem unbegrenzten Stapelband), läßt sich durch ein LFPL-Programm implementieren.

   In LFPL läßt sich jedoch kein zusätzlicher Speicherplatz allokieren. Erkläre mithilfe der Erkenntnis aus Aufgabe a) warum es dennoch möglich ist, das Ergebnis einer Funktion in LFPL zu berechnen, die durch eine Turing Maschine mit Platzverbrauch $5n$ (bei einer Eingabe der Größe n) berechnet wird.

c) Implementiere in LFPL eine Funktion $\mathtt{g} : \mathsf{list}(\mathsf{bool}) \to \mathsf{list}(\mathsf{bool} \otimes \mathsf{bool})$, welche ihre Eingabelist nocheinmal hinten anfügt und in Paare komprimiert. Beispiel:

$$\mathtt{g}\big([0,1,1]\big) = \big[(0,1),(1,0),(1,1)\big]$$

   Vorsicht: Die Aufgabe ist etwas schwieriger, läßt sich aber doch noch einfacher lösen als von Steffen in der Übung vermutet. Belohnung!

d) Implementiere in LFPL eine Funktion $\mathtt{clone} : \mathsf{list}(\mathsf{bool} \otimes \Diamond) \to \mathsf{list}(\mathsf{bool} \otimes \mathsf{bool})$, welche ihre Eingabe verdoppelt. Beispiel: $\mathtt{clone}\big([0,1,1]\big) = \big([0,1,1],[0,1,1]\big)$

Aufgabe G38: Zeige, dass das Produkt von zwei Matrizen (32-bit Integer Einträge) mit logarithmischen Platzverbrauch berechnet werden kann!

Aufgabe G39 (Solvable Path Systems): Gegeben sei eine endliche Menge X, eine Menge $R \subseteq X^3$, sowie $x_s, x_t \in X$.

Zeige, dass die Frage, ob $x_t \in A$ ist, in polynomieller Zeit beantwortet werden kann, wobei die Menge A als die kleinste Teilmenge von X defniert sei, welche die Bedingungen $x_s \in A$ und $\forall u,v,w.\big(u \in A \wedge v \in A \wedge (u,v,w) \in R\big) \implies w \in A$ erfüllt.

Wieviel Platz braucht Dein Verfahren?

Hausaufgabe:

Aufgabe H9 (Huffman Coding Tree): Wir wollen einen gegebenen Text nach der Methode von Huffman komprimieren.

Gegeben sei eine List von Paaren, bestehen jeweils aus einem Zeichen und dessen absolute Häufigkeit im betrachteten Text. Implementiere nun eine Funktion in LFPL[3] die dazu den binären Kodierungsbaum nach Huffman berechnet.

Dabei geht man wie folgt vor: Jedes Zeichen wird als Baum betrachtet, dessen Wurzel die absolute Häufigkeit enthält und als Blatt das Zeichen selbst. Jetzt werden sukzessive die beiden Bäume mit den geringsten Häufigkeiten durch einen neuen Baum ersetzt: die Wurzel des neuen Baumes enthält die Summe der Werte der Wurzeln der beiden ursprünglichen Bäume, das linke Kind der Wurzel ist der ursprüngliche Baum mit dem kleineren Wert und das rechte Kind ist der anderen Baum. Dies wird solange wiederholt bis nur noch ein Baum übriggeblieben wird.

Der Text kann nun dadurch komprimiert werden, das man jedes Zeichen durch seinen Pfad im Kodierungsbaum ersetzt. Zur Dekodierung benötigt man natürlich wieder den Kodierungsbaum. Eine Kompression ist nun daurch möglich, das häufige Zeichen durch kürzere Pfade kodiert werden als seltenere Zeichen.[4]

Ein lückenhafter Programmtext findet sich auf der Webseite der Vorlesung. Fülle die Lücken aus und übersetze Dein Programm mit dem ebenfalls auf der Webseite erhältlichen Compiler. Es wird compilierbarer, lauffähiger Code erwartet. Die Abgabe muß bis Montag, den 3.2. erfolgen und kann auch per eMail an `jost@informatik.uni-muenchen.de` übermittelt werden.

---

[3] `z.B. mit der Implementation von Robert Atkey,`
`http://www.dcs.ed.ac.uk/home/resbnd/prototypes/by_Robert_Atkey/`

[4] Um wirklich eine Kompression zu erreichen, muss man jedes Zeichen (oder besser Gruppen von Zeichen) als Binärsequenzen interpretieren, die man damm durch die Binärsequenz des Pfades im Kodierungsbaum ersetzt, doch das soll hier nicht von Bedeutung sein.