Dr. Rüdiger Ebendt
ebendt@tzi.de
MZH 3530

6. Übungsblatt zur Vorlesung

Optimierung von graphenbasierten Funktionsdarstellungen

Aufgabe 1

Gegeben seien Funktionen $f\colon \mathbb{N} \to \mathbb{R}^+$, $g\colon \mathbb{N} \to \mathbb{R}^+$, $h\colon \mathbb{N} \to \mathbb{R}^+$ und $k\colon \mathbb{N} \to \mathbb{R}^+$. Sei weiter $f = o(g)$ und $h = O(k)$.

Zeige oder widerlege die folgenden Aussagen:

- $\lambda n.h(n) \cdot f(n) = o(\lambda n.k(n) \cdot g(n))$
- $O(f) \subseteq o(g)$
- $g - f = \Theta(g)$

Aufgabe 2

In der Vorlesung wurde der **restrict**-Operator zur Konstruktion der BDD-Darstellung eines Kofaktors vorgestellt. Die Komplexitätsanalyse beruhte auf der Annahme, dass die Computed Table mit einem Array für Daten-Retrieval realisiert wird. Ermittele nun die Zeit- und Platzkomplexität für den worst-case erneut, und zwar unter der Voraussetzung, dass

a) überhaupt keine Computed Table verwendet wird, die entsprechende Anweisung im Pseudo-Code von **restrict** also gestrichen wird.
b) die Computed Table mit einem hash-basierten Cache realisiert wird.

Wie realistisch ist das unter b) anzuwendende worst-case Szenario? Welche der insgesamt drei möglichen Realisierungen der Computed Table, also

- gar keine Computed Table
- als Array für Daten-Retrieval
- als hash-basierter Cache

hältst Du für die in der Praxis beste Realisierung? Begründe Deine Antwort.

Aufgabe 3

In der Vorlesung wurde der **compose**-Operator zur Konstruktion der BDD-Darstellung einer Komposition von Funktionen vorgestellt. Im Pseudo-Code zu **compose** wird im Fall $\mathrm{var}(F) < v$ nach zwei rekursiven **compose**-Aufrufen auf den Kindern von BDD F $\mathbf{ITE}(\mathrm{var}(F), T, E)$ zurückgegeben. Warum ist das an dieser Stelle nötig? Wieso wäre insbesondere Eintrag des Tupels $(\mathrm{var}(F), T, E)$ in der Unique Table und anschließende Rückgabe keine korrekte Lösung?

Aufgabe 4

Die Zeitkomplexität eines Zugriffs auf eine Hash-Tabelle mit (ungeordneten) Kollisionslisten ist im worst-case von der Größenordnung der Anzahl der Einträge in der Tabelle (weil im worst-case alle Schlüssel auf denselben Hash-Wert abgebildet werden können, die Kollisionsliste also alle Einträge enthalten würde).

Vorsichtige Gemüter könnte diese hohe Komplexität dazu motivieren, beim Design eines BDD-Paketes auf eine Unique Table ganz zu verzichten. Stattdessen kann man auch unreduzierte BDDs aufbauen und anschliessend reduzieren. Sei G ein BDD. Die Zeitkomplexität eines Reduktionsalgorithmus wurde von Bryant 1986 noch mit $O(|G| \cdot \log |G|)$ angegeben und 1993 durch eine Verbesserung von Sieling und Wegener auf $O(|G|)$ verringert.

Ändern sich die Komplexitäten der in der Vorlesung besprochenen Algorithmen zur BDD-Manipulation, wenn man wie beschrieben vorgeht statt eine Unique Table zu verwenden?

Ist es Deiner Meinung nach besser, vorsichtig zu sein? Begründe Deine Antwort.