PHILIPPS-UNIVERSITÄT MARBURG D-35032 Marburg
Fachbereich Mathematik und Informatik Hans Meerwein Straße
Prof. Dr. R. Loogen
J.Berthold

08.Juni 2006

8. Übung zu "Parallelität in funktionalen Sprachen", SS 2006

Abgabe schriftlicher Aufgaben: Do 22.Juni 2006 (vor der Vorlesung)

Hinweise:
Die Bearbeitungszeit für dieses Übungsblatt beträgt zwei Wochen. In der Woche vom 12.6.-18.6. ist am Mittwoch ein Gastvortrag, der Donnerstag ist Feiertag.

Aufgaben

8.1 **Auswertungsstrategien für Listen** 10 Punkte

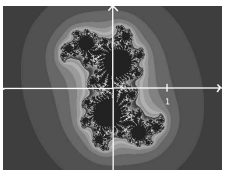
Oftmals enthalten die auszuwertenden Listen in parallelen funktionalen Programmen sehr viele Elemente. Jedes Element einzeln parallel auszuwerten ist daher ungünstig ("zu feine Granularität"), besser ist eine Aufteilung der Liste in Abschnitte bestimmter Größe, was auf verschiedene Weise erfolgen kann.

(a) Schreiben Sie eine Auswertungsstrategie `parListPart`, um Listen *abschnittweise* parallel auszuwerten, wobei die Abschnitte eine durch den Parameter vorgegebene Größe haben sollen. Weiterer Parameter ist die Auswertungsstrategie für die Listenelemente. / 2

(b) Schreiben Sie eine Hilfsfunktion `splitIntoN`, welche eine Liste *elementweise reihum* auf `n` Teillisten verteilt. Bauen Sie die Hilfsfunktion in eine Strategie `parListRR` ein, welche die erzeugten Teillisten parallel auswertet. / 3

```
parListPart, parListRR:: Int -> Strategy a -> Strategy [a]
splitIntoN :: Int -> [a] -> [[a]]
```

(c) Parallelisieren Sie mit Listenstrategien ein Programm zur Visualisierung von *Julia-Mengen*. Erstellen Sie aus der sequenziellen Vorlage auf der Vorlesungsseite parallele Programme mit verschiedenen Auswertungsstrategien und messen Sie, welche Strategie und welche Parameter die beste Laufzeit ergeben. / 4



JuliaSets.hs, Ausschnitt b

8.2 **Auswertungspläne** 8 Punkte

(a) Gegeben sei das folgende Programmfragment mit Ausdrücken `e1` und `e2`. Beschreiben und bewerten Sie die Effekte der angegebenen Auswertungspläne. / 3

```
...= let a=e1
         b=e2
         x= if (a >= 0) then b else 0
     in x `sched` ...<siehe rechts>...
```

i. `( d b | e a . e x )`

ii. `( e x | d b )`

iii. `( e a . ( e x | d b ))`

(b) Definieren Sie mit Auswertungsplänen eine zur Klasse `NFData` äquivalente Klasse `NFData2` und Instanzen dieser Klasse für Tupel und Listen. / 3

(c) Definieren Sie mit Auswertungsplänen eine zu `seqList r0` analoge Funktion. / 2