Prof. Dr. R. Loogen, M. Dieterle
Fachbereich Mathematik und Informatik
Hans-Meerwein-Straße
D-35032 Marburg

7. Dezember 2010

Philipps Universität Marburg

Übungen zu „Parallelität in funktionalen Programmiersprachen“, WS 2010

Nr. 8, Abgabe der Aufgaben: 21. Dezember 2010 vor der Vorlesung

---

Hinweis: Dies ist ein Zweiwochenblatt.

Aufgaben

8.1 Abstrakte Semantik — 8 Punkte

Gegeben seien die folgenden Funktionen in Mini-Haskell :

(a) **foldr1**

```
foldr1Int          :: (Int → Int → Int) → [Int] → Int
foldr1Int f [x]    =  x
foldr1Int f (x:xs) =  f x (foldr1Int f xs)
foldr1Int f []     =  errorEmptyList "foldr1"
```

und

(b) **scanr1**

```
scanr1Int          :: (Int → Int → Int) → [Int] → [Int]
scanr1Int f []     =  []
scanr1Int f [x]    =  [x]
scanr1Int f (x:xs) =  (:) (f x (foldr1Int f xs)) (scanr1Int f xs)
```

Bestimmen Sie formal die abstrakte Semantik von `foldr1Int` und `scanr1Int`, sowie die sicheren Auswertungsgrade der Argumente zu den möglichen Auswertertungsgraden der Ergebnisse (*Evaluation Transformer* - ET(**foldr1**) und ET(**scanr1**)).

8.2 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 `parListSegments` (vom *alten* Typ `a → ()`), um Listen *blockweise* parallel auszuwerten, wobei die Anzahl der Blöcke einstellbar sein soll. Ein weiterer Parameter ist die Auswertungsstrategie für die Listenelemente. — / 3

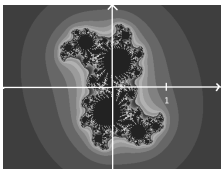
```
parListSegments :: Int → Strategy a → Strategy [a]
```

(b) Eine zweite Auswertungsstrategie `parListRR` soll die Listenelemente reihum (*Round Robin*) aufteilen und die Teile parallel auswerten. Auch hier sei die Anzahl der Teile parametrisierbar. — / 3

```
parListRR :: Int → Strategy a → Strategy [a]
```

Bitte wenden!

(c) Parallelisieren Sie mit den Listenstrategien ein Programm zur Visualisierung von *Julia-Mengen*. Erstellen Sie das Programm aus der sequentiellen Vorlage auf der Vorlesungsseite, parametrisierbar mit den 2 verschiedenen Auswertungsstrategien (`parListSegments`, `parListRR`, jeweils mit Strategie `rnf`). Messen Sie, welche parallele Strategie die beste Laufzeit ergibt. Wählen Sie dazu die Problemgröße 2500 (4. Parameter) und Ausschnitt b (5. Parameter).



JuliaSets.hs, Ausschnitt b

i. Messen Sie mit `+RTS -H1000M -N`i, wobei $i \in \{1, 2, 4, 7\}$ und die Anzahl der Sparks $s = i$ (2. Parameter). Führen Sie auch eine Vergleichsmessung mit der sequentiellen Programmversion (1. Parameter = 0) durch.

ii. Wählen Sie in einer zweiten Messreihe die Laufzeitparameter fix mit `+RTS -H1000M -N7` und die Anzahl der Sparks s als ein Vielefaches der Threadanzahl: $s = k * 7$ mit $k \in \{1, 2, 4\}$ sowie $s = 2500$.

Bei den Messungen sollte die Ausgabe ausgeschaltet sein (3. Parameter = 0). Präsentieren Sie Ihre Messergebnisse in Tabellen. Interpretieren Sie die Ergebnisse.

8.3 Monadische Auswertungsstrategien

6 Punkte

Definieren Sie die Auswertungsstrategien für Listen aus Aufgabe 8.2 mit den monadischen Auswertungsstrategien aus Control.Parallel.Strategies > 3.0. Führen Sie ausführliche Vergleichsmessungen der alten und der monadische Strategien mit dem *Julia-Mengen* Programm durch.