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

30. Juni 2009

Philipps Universität Marburg

Übungen zu „Parallelität in funktionalen Programmiersprachen“, Sommer 2009

Nr. 11, Abgabe der Aufgaben: 7. Juli 2009 vor der Vorlesung

Aufgaben

11.1 Sortierender Workpool

6 Punkte

Gegeben sei der folgende Hilfs-Workpool:

```
workpoolAux :: (Trans t, Trans r) ⇒
               Int → Int → Process [t] [r] → [t] → ([Int],[[r]])
workpoolAux np prefetch worker tasks
   = (reqs,fromWorkers)
    where   fromWorkers     = parMap worker taskss
            taskss          = distribute np reqs tasks
            reqs            = map snd $ zip tasks $
                              initialReqs ++ newReqs
            initialReqs     = concat (replicate prefetch [0..np-1])
            newReqs         = merge $ mkPids fromWorkers

mkPids :: [[r]] → [[Int]]
mkPids rss = [[ i | j←rs] |(i,rs) ←zip [0..] rss]
```

(a) Erstellen Sie unter Benutzung von `workpoolAux` einen Workpool `workpoolDet`, mit Ausgaben in der Reihenfolge der korrespondierenden Eingaben. Der Typ des Workpools soll identisch mit dem der VL sein. Die Daten sollen nicht mit ID-Tags zum Sortieren versehen werden. Statt dessen soll die Verteilungsinformation `reqs` ausgenutzt werden.

(b) Was wäre die Konsequenz, wenn `reqs` ohne `map snd $ zip tasks $...` definiert wäre?

11.2 Pipeline-Skelette

6 Punkte

Schreiben Sie analog zu Ringskeletten unter Verwendung dynamischer Kanäle zwei Varianten eines *Pipeline-Skeletts* in Eden:

```
pipe, pipeR :: (NFData a, Trans a) ⇒ [ a → a ] → [a] → [a]
```

(a) Erzeugung aller Pipelineprozesse und der Verbindungen durch den Aufrufer.

(b) Rekursive Erzeugung der Pipeline über die Funktionsliste.

Skizzieren Sie ein Beispiel zur Verwendung Ihrer Pipelines und diskutieren Sie die allgemeine Anwendbarkeit dieser Skelette.