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

29.Juni 2006

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

Abgabe schriftlicher Aufgaben: Do, 6.Juli 2006 (vor der Vorlesung)

Aufgaben

10.1 Pipeline-Skelette

6 Punkte

Schreiben Sie analog zu Ringskeletten 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.

10.2 Implementierung von Eden

6 Punkte

Die Sprache **EVIL** (Eden's Vital Implementation Language) umfasse Haskell 98 und die folgenden Aktionen in der IO-Monade (implementiert in einem Modul *ParPrim*):

Name	Typ	Beschreibung
noPe	`IO Int`	liefert die Gesamtzahl der Prozessoren
selfPe	`IO Int`	liefert die Prozessornummerdes Aufrufers
expectData	`IO ( ChanName' a, a )`	erzeugt einen Kanal und einen Platzhalter vom Typ `a` im Heap
connectToPort	`ChanName' a -> IO ()`	legt den übergebenen Kanal als Empfänger zukünftiger Nachrichten fest
sendData	`Mode -> a -> IO ()`	sendet die übergebenen Daten als Graph von Closures (ggf. unausgewertet!) an den festgelegten Empfänger. *Blockiert*, falls im Subgraphen ein Platzhalter gefunden wird.

Die Nachricht wird in einem der folgenden Modi versandt:

Connect	Empfänger "registriert" Sender, keine Daten.
Data	Daten als einzelner Wert empfangen.
Stream	Daten als Listenelement empfangen, Kanal bleibt offen zum Empfang weiterer Daten.
Instantiate	Daten als Prozess empfangen, nach Empfang wird sofort ein Thread zur (whnf-)Auswertung gestartet.

Definieren Sie in einem Modul *Eden.hs* die Konstrukte der Sprache Eden mit Hilfe der angegebenen EVIL-Operationen. [1] Um Ergebnisse der IO-Aktionen außerhalb der IO-Monade verwenden zu können, benutzt man:
`System.IO.Unsafe::unsafePerformIO :: IO a -> a`

[1] Sie können mit dem Compiler `/net/maseru/berthold/ghc-6.x-eden` und dem Modul *ParPrim.hs* von der Vorlesungsseite Ihre Eden-Implementierung testen.
Das zu erstellende Eden-Modul muss *ParPrim* importieren Zunächst werden *Eden-* und *ParPrim*-Modul separat mit Optionen `-fglasgow-exts -c -parpvm --make` übersetzt, danach kann ein herkömmliches Eden-Programm dieses Eden-Modul importieren (mit Option `-parpvm --make` übersetzen).