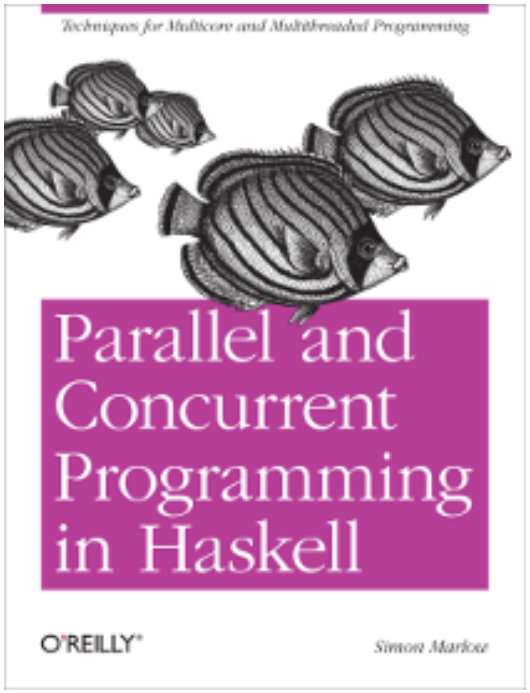
Fortgeschrittene Funktionale Programmierung in Haskell

Universität Bielefeld, Sommersemester 2015

Jonas Betzendahl & Stefan Dresselhaus

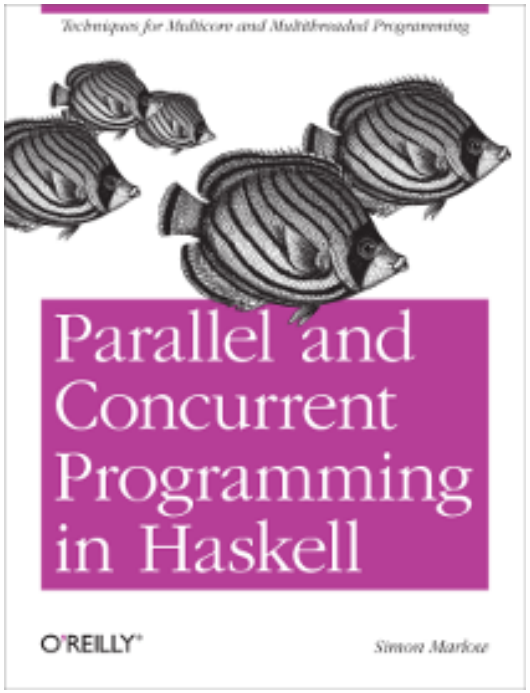
Übersicht I

Leseempfehlung:



Wunderbares Buch zum Thema von Simon Marlow.

Leseempfehlung:



Wunderbares Buch zum Thema von Simon Marlow.

Gratis im Internet verfügbar, inklusive Beispielcode.

Motivation

Free Lunch is over!

Herb Sutter (2005)

Free Lunch is over!

Herb Sutter (2005)

Die Hardware unserer Computer wird seit mehreren Jahren schon schneller breiter (*mehr* Kerne) als tiefer (*schnellere* Kerne).

Free Lunch is over!

Herb Sutter (2005)

Die Hardware unserer Computer wird seit mehreren Jahren schon schneller breiter (*mehr* Kerne) als tiefer (*schnellere* Kerne).

Um technischen Fortschritt voll auszunutzen ist es also essentiell, gute Werkzeuge für einfache und effiziente Parallelisierung bereit zu stellen.

Definitionen

Parallelism

- Die `Eval`-Monade und Strategies
- Die `Par`-Monade
- Die `RePa`-Bibliothek
- GPU-Programming mit `Accelerate`

Parallelism

- ○ Die `Eval`-Monade und Strategies
- Die `Par`-Monade
- Die `RePa`-Bibliothek
- GPU-Programming mit `Accelerate`

Das Modul `Control.Parallel.Strategies` (aus dem Paket `parallel`) stellt uns die `Eval`-Monade und einige Funktionen vom Typ *Strategy* zur Verfügung, ...

Das Modul `Control.Parallel.Strategies` (aus dem Paket `parallel`) stellt uns die `Eval`-Monade und einige Funktionen vom Typ *Strategy* zur Verfügung, ...

```
type Strategy a = a -> Eval a
```

Das Modul `Control.Parallel.Strategies` (aus dem Paket `parallel`) stellt uns die `Eval`-Monade und einige Funktionen vom Typ *Strategy* zur Verfügung, ...

```
type Strategy a = a -> Eval a
```

... insbesondere die Strategies `rpar` und `rseq`. Dazu gleich mehr.

Das Modul `Control.Parallel.Strategies` (aus dem Paket `parallel`) stellt uns die `Eval`-Monade und einige Funktionen vom Typ *Strategy* zur Verfügung, ...

```
type Strategy a = a -> Eval a
```

... insbesondere die Strategies `rpar` und `rseq`. Dazu gleich mehr.

Desweiteren stellt es die Operation `runEval`, die die monadischen Berechnungen ausführt und das Ergebnis zurück gibt, bereit.

```
runEval :: Eval a -> a
```

Das Modul `Control.Parallel.Strategies` (aus dem Paket `parallel`) stellt uns die `Eval`-Monade und einige Funktionen vom Typ *Strategy* zur Verfügung, ...

```
type Strategy a = a -> Eval a
```

... insbesondere die Strategies `rpar` und `rseq`. Dazu gleich mehr.

Desweiteren stellt es die Operation `runEval`, die die monadischen Berechnungen ausführt und das Ergebnis zurück gibt, bereit.

```
runEval :: Eval a -> a
```

Wohlgemerkt: `runEval` ist *pur!*
Wir müssen nicht gleichzeitig auch in der `IO`-Monade sein.

Parallelism

- Die `Eval`-Monade und Strategies
- ○ Die `Par`-Monade
- Die `RePa`-Bibliothek
- GPU-Programming mit `Accelerate`

Parallelism

- Die `Eval`-Monade und Strategies
- Die `Par`-Monade
- ○ Die `RePa`-Bibliothek
- GPU-Programming mit `Accelerate`

Parallelism

- Die `Eval`-Monade und Strategies
- Die `Par`-Monade
- Die `RePa`-Bibliothek
- ○ GPU-Programming mit `Accelerate`