Intermediate Functional Programming in Haskell

Universität Bielefeld, Sommersemester 2015

Jonas Betzendahl & Stefan Dresselhaus

Überblick für Heute:

Organisatorisches

Tools & Ressourcen

Funktionale Programmierung 101

Haskell Crash-Kurs

Orga-Krams

Orga-Krams (1): Veranstaltungen

Es gibt Vorlesungen ($wochentag1, AA - BB Uhr in $Raum1)
und Übungen ($wochentag2, CC - DD Uhr in $Raum2).

Die Vorlesungen starten mit einem eher praktisch orientierten Abschnitt ab Heute gefolgt in der zweiten Semesterhälfte von einem eher theoretischen Teil.

Teilnahme an den Übungen ist nicht verpflichtend, aber wahrscheinlich für euch von großem Vorteil.

Orga-Krams (2): Input / Output

Für das Modul gibt es 5 (echte) Leistungspunkte
Als Leistung müsst ihr ein kleines Programmierprojekt abschließen. Details dazu gibt es in der Übung

Wegen bürokratischer Hürden können die Leistungspunkte nur in der *individuellen* Ergänzung eingebracht werden. Nicht in Wahlpflichtbereichen und auch nicht in der strukturierten Ergänzung.

Orga-Krams (3): Personenkult

Wir, das sind Jonas Betzendahl und Stefan Dresselhaus.
Mailadressen: `{jbetzend,sdressel}@techfak...`

Wir sind zwei Masterstudenten an der TechFak, die ihren Wunsch nach mehr und besserer Lehre zu funktionaler Programmierung und Haskell hier in Bielefeld jetzt selbst in die Hand nehmen.

Formal verantwortlich ist Dr. Alexander Sczyrba (`asczyrba@techfak...`).
Er ist eure Anlaufstelle für Fragen im Kontext der Fakultät und Beschwerden zu uns.

Orga-Krams (4): Material von uns für euch

Alle Übungsblätter, Foliensätze, Beispiele, Vorlagen und sonstige Unterlagen findet ihr entweder im ekVV oder auf der Website dieser Veranstaltung. Die URL ist:

`www.dieseurlmussnochersetztwerden.de`

Audio/Video-Mitschnitte findet ihr ebenfalls im Web und zwar auf folgender Seite:

`www.irgendwiesowaswieunirekorder.de`

Login: Login, Passwort: 123456

Tools und Ressourcen

T & R (1): Haskell / GHC

Um auf euren eigenen Rechnern Haskell sinnvoll benutzen zu können benötigt ihr den `Glasgow Haskell Compiler` (GHC).

Der GHC und viele nützliche Dinge mehr sind alle im Rundum-Glücklich-Paket genannt *Haskell Platform* enthalten. Die gibt es für Linux, Mac, BSD und Windows unter folgender URL:

`https://www.haskell.org/platform/`

Wichtig:
Der Interpreter `Hugs` wird von uns explizit nicht unterstützt.

T & R (2): GHCi

Die mitgelieferte interaktive Umgebung des GHC heißt GHCi.

Der GHCi stellt ein REPL (Read - Evaluate - Print - Loop) bereit, die beim Entwickeln *sehr* nützlich sein kann (ähnlich zu `Hugs`). Mehr dazu in den Übungen.

T & R (3): Hackage

Dieser Tage gibt es eine Vielzahl an nützlichen und mächtigen Bibliotheken für Haskell. Zu Hause sind die meisten davon auf *Hackage*:

`https://hackage.haskell.org/`

Auf Hackage findet ihr übersichtliche Zusammenfassungen der Bibliotheken, detaillierte Auflistungen der exportierten Funktionen und Datentypen und direkte Links zu den jeweiligen Implementationen (!).

T & R (4): cabal

Ebenfalls in der *Haskell Platform* enthalten ist `cabal`, Haskells eigenes Paket-Management-System. `cabal` erlaubt es euch, einfach via Terminal Bibliotheken von Hackage lokal (!) und im Zweifelsfall auch problemlos in einer Sandbox zu installieren. `cabal` ist außerdem ein Build-System, kann Test-Suites ausführen und und und. Aber dazu im Zweifelsfall später mehr.

Das folgende Beispiel installiert `lens`, eine Bibliothek die uns im Lauf der Vorlesung noch begegnen wird:

```
$ cabal update && cabal install lens
```

T & R (5): LYAHFGG

Das Buch „Learn You A Haskell“ ist die beste™ Ressource um Haskell zu lernen. Es ist verständlich und locker geschrieben und berühmt für gute Erklärungen.

Ihr findet es online frei und kostenlos verfügbar hier:
`http://learnyouahaskell.com/`

T & R (6): Hoogle

Hoogle ist eine Haskell-API-Suchmaschine, die es möglich macht, nach Funktionen zu suchen, ihren Typen angibt und zeigt, welche Imports nötig sind.
Ein besonders nützliches Feature ist, dass es auch möglich ist, nach Typsignaturen zu suchen und darauf passende Antworten zu erhalten.

Die URL lautet: `http://www.haskell.org/hoogle/`

T & R (7): Community

Die Haskell-Community ist jenseits von dedizierten Mailingslisten hauptsächlich auf Reddit (`http://www.reddit.com/r/haskell`) und im IRC (`#haskell` auf Freenode) zu finden. Sie gilt als sehr offen, hilfreich und anfängerfreundlich.

Als Illustration dazu ein inzwischen berühmt gewordener Ausschnitt aus dem IRC:

<xQ> HASKELL IS FOR ***** *****. YOU'RE ALL A BUNCH OF *****
******
<xQ> JAVASCRIPT FOR LIFE *****

<xQ> HASKELL IS FOR ***** *****. YOU'RE ALL A BUNCH OF *****
******
<xQ> JAVASCRIPT FOR LIFE *****
...
<A> xQ: Do you have any specific questions?
<A> xQ: We'd love to help you make your first steps.

<xQ> HASKELL IS FOR ***** *****. YOU'RE ALL A BUNCH OF *****
******
<xQ> JAVASCRIPT FOR LIFE *****
...
<A> xQ: Do you have any specific questions?
<A> xQ: We'd love to help you make your first steps.
<xQ> i just want to get kicked out of a bunch of channels for fun
<A> Have you seen LYAH? It's a very enjoyable book on Haskell. It also has a reputation of being very uplifting.
<xQ> why is no one cooperating with me?

<xQ> HASKELL IS FOR ***** *****. YOU'RE ALL A BUNCH OF *****
******
<xQ> JAVASCRIPT FOR LIFE *****
...
<A> xQ: Do you have any specific questions?
<A> xQ: We'd love to help you make your first steps.
<xQ> i just want to get kicked out of a bunch of channels for fun
<A> Have you seen LYAH? It's a very enjoyable book on Haskell. It
also has a reputation of being very uplifting.
<xQ> why is no one cooperating with me?
...
<xQ> what's haskell good for though?
...
<xQ> Alright thanks guys, I'll be back later to actually learn
some haskell for real

Funktionale Programmierung 101

FP-101 (1): Definition

Funktionale Programmierung ist ein *deklaratives* Programmierparadigma, in dem darauf abgezielt wird, den Programmfluss als Auswertung mehrerer (mathematischer) Funktionen auszudrücken. Dabei wird besonderer Wert darauf gelegt, so genannte *Seiteneffekte* zu vermeiden.

FP-101 (2): Deklaratives Programmieren

Im Gegensatz zur imperativen Programmierung gibt der Programmierer nur an, *welche* Werte er errechnet bekommen möchte, nicht jedoch, *wie* sie berechnet werden sollen. Das bleibt der Implementierung der Sprache (dem Compiler) überlassen.

```
triples :: (Int, Int, Int)
triples = [(a,b,c) | a <- [1..10], b <- [1..10],
                     c <- [1..10], a^2 + b^2 == c^2]
```

FP-101 (3): Seiteneffekte

```
public int fuenf()
{
    System.out.println("Seiteneffekt!");
    global_var = null;
    return 5;
}
```

Diese zwei Statements haben sehr (!) unterschiedliches Verhalten:

```
return 5 + 5;        // referential transparency
return 5 + fuenf(); // no referential transparency
```

FP-101 (4): Higher-Order Functions

Funktionen mit anderen Funktionen als Parameter.
`map` z.B. wendet eine Funktion auf alle Elemente einer Liste an.

```
ghci > map (+10) [1..10]
[11,12,13,14,15,16,17,18,19,20]

ghci > map reverse ["abcde", "123", "kayak", "haskell"]
["edcba","321","kayak","lleksah"]
```