Tutorium: Programmieren mit Typen in GHC

Andres Löh

HaL8 – Universität Leipzig, 21. Juni 2013

Vorkenntnisse Dieses Tutorium richtet sich an Programmierer mit Vorkenntnissen in Haskell (oder sehr interessierte Neulinge, denen das Typsystem und seine Erweiterungen eher als Herausforderung denn als Schwierigkeit erscheint).

Einführung Nimmt man die Vielzahl an Erweiterungen, die GHC mittlerweile bietet, als gegeben hin, dann hat man eine unglaublich mächtige Typ-Sprache, die auf Typniveau weitaus mehr ausdrücken kann als eine simple Klassifikation der Werte.

Wir können anfangen, komplexere Eigenschaften von Daten auszudrücken: etwa, dass eine Liste eine bestimmte Länge haben muss, dass eine natürliche Zahl einen Wert zwischen 3 und 7 haben muss, dass ein Baum balanciert ist oder dass ein datentyp für abstrakte Syntaxbäume in einer Programmiersprache typkorrekt sein muss.

Sehr schnell kommt man dabei an einen Punkt, bei dem wir Berechnungen auf Typniveau ausführen müssen, also Funktionen definieren müssen, die erklären, wie man Typen auf vielfältige Weise miteinander kombiniert und ineinander überführt.

Haskell geht dabei einen anderen Weg als sogenannte “abhngig getypte Sprachen” (dependently typed programming languages), indem es eine strikte Trennung zwischen normalen Werten und Typen beibehält. Funktionen auf Typebene sehen daher syntaktisch anders aus als normale Funktionen.

Programmieren auf Typebene hat den Vorteil, dass man sehr viel mehr Garantien über die Korrektheit der Programme bekommt, indem man so viele problematische oder illegale Werte wie möglich unrepräsentierbar macht. Der Nachteil ist allerdings, dass die Programmierung auf Typniveau mit sehr viel zusätzlicher Arbeit verbunden sein kann. Dieser Ansatz ist also kein Allheilmittel, sondern sollte mit Bedacht eingesetzt werden, wenn es wirklich zählt – oder einfach, weil es Spass macht.

Inhalt In diesem Tutorium geht es darum, eine Einführung in die Sprachmittel zu geben, die für die Programmierung auf Typebene nötig und hilfreich sind. Wir werden unter anderem die folgenden Themen behandeln:

- Promotion von Datentypen (`DataKinds`), Kind-Polymorphismus (`PolyKinds`)
- Natürliche Zahlen und beschränkte natürliche Zahlen (Nat und Fin)
- Vektoren und heterogene Listen / Umgebungen (Vec und Env)
- GADTs und Typfamilien (`GADTs`, `TypeFamilies`)
- Gleichheit auf Typniveau, (semi-)entscheidbare Gleichheit von Werten
- Nichtinjektivität von Typfamilien, Proxies, Singleton Types

Das Tutorium wird beispielorientiert sein. Das heißt unter anderem, dass aus Zeitgründen keine vollständige Einführung in die oben genannten Themen gegegeben werden kann. Wir werden statt dessen versuchen, konkrete (aber relativ einfache) Probleme mit Hilfe von Techniken der Programmierung auf Typebene zu lösen. Mittels solcher kleinerer Programmier-Übungen werden wir erlernen, wie man Programme auf Typebene schreibt, welche Problem dabei auftreten, und wie man einige dieser Probleme lösen kann.

Systemvoraussetzungen GHC 7.6.x wird vorausgesetzt. Am besten ist es, die hoffentlich bis zum HaL erschienene Haskell-Platform Version 2013.2.0.0, die auf GHC 7.6.3 basiert, zu installieren. Bibliotheken werden aller Voraussicht nach keine verwendet. Falls doch, sollten diese ohne grosse Abhängigkeiten während des Tutoriums mittels `cabal-install` (das ein Teil der Platform ist) installiert werden können.

Wenn jemand (zusätzlich!) einen development snapshot von GHC bauen möchte, so kann das für einige kleine Verbesserungen, die in der Entwicklungsversion von GHC implementiert sind, spannend sein. Aber es ist keineswegs nötig, und es kann sein, dass wir aus Zeitgründen ohnehin keine Zeit haben, auf diese Neuentwicklungen einzugehen.