Universität Bremen

Praktische Informatik 3 WS 20/21

HOWTO Haskell

Kurzanleitung für den PI3-Übungsbetrieb

Christoph Lüth
Thomas Barkowsky
Tobias Brandt
Alexander Krug
Tarek Soliman
Robert Sachtleben

Diese Kurzanleitung (neudeutsch HOWTO) beschreibt erste Schritte mit Haskell, und erläutert die Benutzung des für den Übungsbetrieb in PI3 vorgegebenen Rahmenwerks.

1 Erste Schritte mit Haskell

Der Unterschied zwischen Theorie und Praxis ist in der Praxis größer als in der Theorie. Wir empfehlen, die folgenden Schritte zu befolgen, um eine lauffähige Haskell-Umgebung auf Ihrem System (bzw. einem Uni-Rechner) einzurichten und Übungsaufgaben lösen zu können.

1. Installieren Sie den Haskell Tool Stack (https://www.haskell.org/downloads/#stack)
2. Erstellen Sie ein Verzeichnis, in dem Sie mit einem Editor Ihrer Wahl eine Datei `FirstTest.hs` erstellen.
3. Fügen Sie folgenden Code (aus der Vorlesung) ein:

   ```
   fac :: Int → Int
   fac n = if n ≤ 0 then 1 else n * fac (n − 1)
   ```

   Sie können den Quellcode auch von der Veranstaltungswebseite herunterladen.
4. Starten Sie eine Kommandozeile/Terminal/Konsole (`cmd.exe` unter Windows, `bash` o.ä. unter Linux), und wechseln Sie ggf. in das erzeugte Verzeichnis.
5. Starten Sie den Interpreter über Stack mit `stack ghci`. Der Haskell Tool Stack wird beim ersten Aufruf gefühlte drei Jahre — je nach Rechnerleistung und Internetverbindung — Dateien herunterladen und übersetzen: er erzeugt dabei eine lokale Haskell-Arbeitsumgebung. Gehen Sie einen Kaffee oder Kaltgetränk Ihrer Wahl trinken. Schalten Sie den Rechner keinesfalls aus. Natürlich brauchen Sie für diesen Schritt eine aktive Internetverbindung.
6. Laden Sie Ihre Testdatei `FirstTest.hs` in den Interpreter: `:load FirstTest` bzw. kürzer `:l FirstTest`.
7. Der Interpreter erlaubt Ihnen, die in der geladenen Datei definierten Funktionen und Variablen auszuwerten. Geben Sie hierzu entsprechende Haskell-Ausdrücke auf der Kommandozeile des Interpreters ein. Beispiele:

   ```
   17 + 4          100 / 7          sqrt 100 − 1
   (−1) * (−1)     fac 10           fac (−3)
   fac "kolibri"   fac 9 − 20       foo 10
   ```

   Beachten Sie auch die hilfreichen (teilweise noch unverständlichen) Fehlermeldungen bei fehlerhaften Ausdrücken.
8. Hilfe bekommen Sie durch Eingabe von `:help`. Die meisten Optionen sind anfangs noch nicht relevant; als hilfreich werden sich aber die Kommandos `:reload` (Abk.: `:r`), `:type` (`:t`) und `:browse` erweisen.

   Ein üblicher Entwicklungszyklus sieht dann so aus: editieren und speichern im Editor, (neu) laden im Interpreter (`:l` bzw. `:r`), Fehlermeldungen analysieren bzw. Testeingaben machen, zurück zum Editor.
9. Die PI3-Webseite bietet nützliche Links zu den Themen Haskell lernen, API-Dokumentation, etc.

2 Das PI3-Rahmenwerk für Übungsaufgaben

Alle PI3-Aufgaben werden mit einem leichtgewichtigen Rahmen als Stack-Projekt ausgebeben, dessen Nutzung verbindlich ist. Es erleichtert uns die Korrektur, und Ihnen die Arbeit. Außerdem wird dieser Rahmen auch in der elektronischen Klausur genutzt.

Ausgabe und Bearbeitung

Um den vorgegebene Rahmen zu nutzen, gehen Sie wie folgt vor:

1. Laden Sie die relevante Vorlage von der Webseite herunter, und entpacken Sie diese:

   ```
   unzip pi3-ws20-ueb00-VORLAGE.zip
   ```

   Dieser Befehl erzeugt die Vorlage für das 0. Übungsblatt im Verzeichnis `ueb00`.[1]

2. Die Dateien in dieser Vorlage sind immer wie folgt organisiert:
   - Das Verzeichnis `src` enthält die nötigen Quellen, die Sie mit Ihrer Implementation ergänzen müssen.
   - Das Verzeichnis `test` enthält einige Unit-Tests (die meist auch auf dem Übungsblatt stehen) als Plausibilitätskontrolle für die Lösung.
3. Mit `stack ghci` (im Verzeichnis `ueb00` aufgerufen) werden die Quellen übersetzt, und der Haskell-Interpreter aufgerufen. Sie können dann interaktiv testen.
4. Mit `stack test` werden die Dateien übersetzt, und die Unit-Tests ausgeführt. Wenn alle Tests erfolgreich sind, ist das ein gutes Zeichen, aber noch *bei weitem* keine Garantie, dass Ihre Lösung richtig ist.
5. In späteren Übungsblättern können Sie Tests zu den Dateien in `test` hinzufügen (kopieren Sie die existierenden Tests und ändern Sie diese entsprechend ab); für die ersten Übungblätter ist das noch nicht notwendig.
6. Die *Dokumentation* erfolgt, wenn nicht explizit im Aufgabenblatt anders gefordert, direkt im Code.
7. Die Dateien `uebXX.cabal` und `stack.yaml` spezifizieren den Aufbau als cabal-Projekt (die technische Grundlage des Rahmens) — bitte nicht verändern.

Dokumentation und Stil

Die Dokumentation sollte folgendes enthalten:

1. Getroffene Entwurfsentscheidungen (für die ersten Übungsblätter wahrscheinlich weniger relevant), bspw. warum werden Daten in der gewählten Weise repräsentiert, oder warum wird die Funktion wie gewählt in Hilfsfunktionen zerlegt;
2. Für jede Funktion: kurze (!) Beschreibung der Parameter, Vorbedingungen (wann ist die Funktion definiert?), Nachbedingung bzw. Ergebnis (was gibt die Funktion zurück?)

Achten Sie auf einen angemessenen *Stil* ihres Programs — verwenden Sie die richtigen Haskell-Funktionen und Datentypen, formulieren Sie kurz und knapp, vermeiden Sie Wiederholungen im Code.

Abgabe

Zur Abgabe des Übungsblattes erzeugen Sie selber ein Repository auf dem Gitlab-Server des FB3, und laden Sie Ihren Tutor für dieses Repository ein. Das Repository enhält für jedes Übungsblatt ein Verzeichnis `uebXX`, in welchem Sie die Vorlage entpackt und ihre Lösung bearbeitet haben. Laden Sie die bearbeitete Aufgabe in dieses Repository hoch (`git push`). Ihr Repository sollte `pi3-ws20-ueb-NAME` benannt sein wobei `NAME` Ihr Namenskürzel ist.

[1] Die späteren Übungsblätter heißen dann logischerweise `ueb01`, `ueb02`, etc.

Bewertung

Die Bewertung erfolgt über das Git-Repo— der Tutor korrigiert den Code und schubst die korrigierte Version wieder in das Repository, zusammen mit einer Datei `BEWERTUNG.md`, welche die Punktzahl enthält.

Die Bewertung erfolgt nach folgenden *Kriterien*:

- Korrektheit der Lösung (Gewichtung ca. 80%).
- Stil und Dokumentation (Gewichtung ca. 20%).

Viel Glück, und happy haskelling!