Haskell Live

[01] Eine Einführung in Hugs

Bong Min Kim
e0327177@student.tuwien.ac.at

Christoph Spörk
christoph.spoerk@inode.at

Florian Hassanen
florian.hassanen@googlemail.com

Bernhard Urban
lewurm@gmail.com

8. Oktober 2010

Hinweise

Diese Datei kann als sogenanntes "Literate Haskell Skript" von `hugs` geladen werden, als auch per `lhs2TeX`[1] und LaTeX in ein Dokument umgewandelt werden.

Kurzeinfühing in `hugs`

`hugs`[2] ist ein Interpreter für die funktionale Programmiersprache Haskell. Abhängig vom Betriebssystem wird der Interpreter entsprechend gestartet, unter GNU/Linux beispielsweise mit dem Befehl `hugs`. Tabelle 1 zeigt eine Übersicht der wichtigsten Befehle in `hugs`.

Präsentiertes Skript

$$\begin{aligned} &eins :: Integer \\ &eins = 1 \\ &addiere :: Integer \to Integer \to Integer \\ &addiere\ x\ y = x + y \end{aligned}$$

[1] `http://people.cs.uu.nl/andres/lhs2tex`
[2] Haskell User's Gofer System

Befehl	Kurzbefehl	Beschreibung
`:edit name.hs`	`:e name.hs`	öffnet den Editor der in `$EDITOR` (Unix) bzw. in WinHugs in den Optionen definiert ist, mit der Datei `name.hs`
`:load name.hs`	`:l name.hs`	ladet das Skript `name.hs`
`:edit`	`:e`	öffnet den Editor mit der zuletzt geöffneten Datei
`:reload`	`:r`	erneuertes Laden des zuletzt geladenen Skripts
`:type` ***Expr***	`:t` ***Expr***	Typ von ***Expr*** anzeigen
`:info` ***Name***		Informationen zu ***Name*** anzeigen. ***Name*** kann z.B. ein Datentyp, Klasse oder Typ sein
`:cd dir`		Verzeichnis wechseln
`:quit`	`:q`	`hugs` beenden

Table 1: Einige Befehle in `hugs`

$addiere\_fuenf :: Integer \to Integer$
$addiere\_fuenf\ x = addiere\ 5\ x$

$ist1 :: Integer \to Bool$
$ist1\ 1 = True$ -- Reihenfolge beachten! (Pattern Matching)
$ist1\ x = False$

Listen können einfach erzeugt werden, zum Beispiel erzeugt der Ausdruck `[1,2,3,4]` eine Liste von gleicher Darstellung. In Tabelle 2 sind einfache Beispiele angeführt.

Ausdruck	Ergebnis	Beschreibung
`[1..5]`	`[1,2,3,4,5]`	Erzeugt eine Liste mit den Elemente 1 bis 5
`[1,4..13]`	`[1,4,7,10,13]`	Siehe nächstes Beispiel
`[`a`,`b`..`x`]`	`[`a`,`b`,`$b+(a-b)$`,`$b+2*(a-b)$`..`x`]`	Es wird ein Offset (Differenz von a und b) ermittelt. Sei b die Basis, so wird bis zum Wert x jeder Wert der die Summe der Basis plus einem Vielfachen des Offsets entspricht, der Liste hinzugefügt
`[]`	`[]`	Leere Liste aka. "nil"
`1:(2:(3:(4:[])))`	`[1,2,3,4]`	`(:)` aka. "cons"
`1:2:3:4:[]`	`[1,2,3,4]`	"cons" ist rechts-assoziativ
`"asdf"`	`"asdf"`	Liste von `Char`. Beachte, dass der Typ `String` dem Typen `[Char]` entspricht.

Table 2: Einfache Beispiele für Listen

$my\_head :: [Integer] \to Integer$
$my\_head\ [\,] = 0$

$my\_head\ (x : xs) = x$ -- Reihenfolge beachten! (Pattern Matching)
$my\_head\ (x : [\,]) = x + 1$

$laf1 :: [Integer] \rightarrow [Integer]$ -- **l**ist_**a**ddiere_**f**uenf
$laf1\ [\,] = [\,]$
$laf1\ (x : xs) = (addiere\_fuenf\ x) : (laf1\ xs)$

$laf2 :: [Integer] \rightarrow [Integer]$
$laf2\ l = [addiere\_fuenf\ x \mid x \leftarrow l, x > 10]$ -- list comprehension

$laf3 :: [Integer] \rightarrow [Integer]$
$laf3\ l = map\ (addiere\_fuenf)\ l$ -- map power

Dokumentation

- Prelude: `http://www.google.at/search?q=haskell+prelude+documentation`
- Interaktive Einführung in Haskell: `http://tryhaskell.org`