Funktionale Programmierung

Der ghci Interpreter

Hans-Wolfgang Loidl, Andreas Abel

LFE Theoretische Informatik, Institut für Informatik,
Ludwig-Maximilians Universität, München

April 27, 2009

Der GHCi Interpreter

GHCi ist ein Interpreter für Haskell und kommt zusammen mit dem Glasgow Haskell Compiler (GHC).

Aufruf von der Kommandozeile: `ghci`

Dokumentation:
http://haskell.org/ghc/docs/latest/html/users_guide/ghci.html

Download: http://haskell.org/ghc/download_ghc_6_10_2.html#binaries

Installation

Installation in Kurzform:

Kommandozeile

```
# download
wget http://haskell.org/ghc/dist/6.10.2/ghc-6.10.2-i386-unknown-linux.t
# unpack
tar xvfj ghc-6.10.2-i386-unknown-linux.tar.bz2
# install
cd ghc-6.10.2
./configure --prefix=${HOME}/DISTS/ghc-6.10.2
make install
export PATH=${HOME}/DISTS/ghc-6.10.2/bin:${PATH}
```

Erstes Beispiel

GHCi Interaktion

```
ghci
Prelude> 2*2
4
Prelude> let square x = x*x
Prelude> square (2+2)
16
Prelude> [x*x | x <- [1..9], even x]
[4,16,36,64]
Prelude> sum it
120
:quit
```

Beispiele der 1. Vorlesung

GHCi Interaktion

```
ghci Types.hs
*Types> sqs1 [1..10]
[1,4,9,16,25,36,49,64,81,100]
*Types> sqs_even [1..10]
[4,16,36,64,100]
*Types> :type sqs_even
sqs_even :: [Int] -> [Int]
*Types> sqs_even [1..10] == sqs_even' [1..10]
True
*Types> :info sqs_even
sqs_even :: [Int] -> [Int]      -- Defined at Types.hs:116:0
*Types> :help
...
*Types> :quit
```

Weitere Beispiele

GHCi Interaktion

```
ghci
*Types> :load "Types.hs"
*Types> let add' x y = x+y
*Types> let inc' = add 1
*Types> map' inc' [1..5]
[2,3,4,5,6]
*Types> :{
*Types| let { sqs' [] = []
*Types|     ; sqs' (x:xs) = x*x : (sqs' xs)
*Types|     }
*Types| :}
*Types> sqs' [1..10]
[1,4,9,16,25,36,49,64,81,100]
*Types> :quit
```

Debugging

GHCi Interaktion

```
ghci Types.hs
*Types> :break 28
Breakpoint 0 activated at Types.hs:28:12-15
*Types> foo (BI { b=True, i=2})
Stopped at Types.hs:28:12-15
_result :: Int = _
bi :: BI = BI True 2
[Types.hs:28:12-15] *Types> :list
27  foo :: BI -> Int
28  foo bi = if b bi
29              then i bi
[Types.hs:28:12-15] *Types> bi
BI {b = True, i = 2}
[Types.hs:28:12-15] *Types> :step
Stopped at Types.hs:29:16-19
_result :: Int = _
bi :: BI = BI True 2
[Types.hs:29:16-19] *Types> :step
2
*Types> :quit
```