


Syntactic analysis and type evaluation in Haskell

Pavel Dvořák
FI MUNI

The main objective

- ▶ Design of stepwise interpreter for teaching purposes – course IB015 at FI.
 - ▶ The word *stepwise* means the ability to evaluate and show each reduction of the parse tree separately.
 - ▶ There is no known project about this subject.
 - ▶ Purpose of my thesis: creation of a basis of the interpret (i.e. syntactic and type analysis, not the evaluation itself).
- 


Features of the interpreter

- ▶ command-line interface
- ▶ loading and unloading Haskell modules
- ▶ displaying the source code of any loaded function
- ▶ inferring type of an expression
- ▶ parsing Haskell expressions

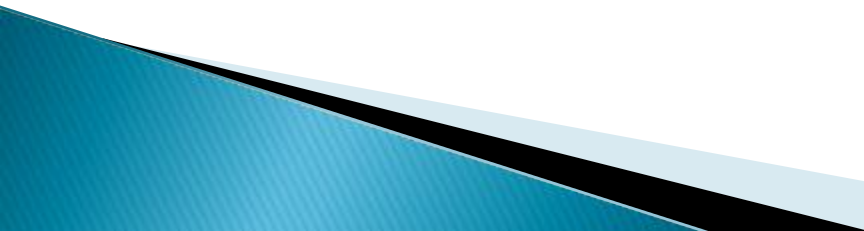


Hasky
The Haskell interpreter

Syntactic analysis

- ▶ User can load any valid Haskell 98 code.
 - ▶ The code is parsed by the `Language.Haskell` library.
 - ▶ Haskell has a little bit complicated module system, our version is slightly simplified (implemented with two associative arrays).
 - ▶ Dependencies of every module must be processed.
 - ▶ We support the syntax `Module.function`.
- 

Type evaluation

- ▶ Our type-checker is based on the one used in the paper *Typing Haskell in Haskell*.
 - ▶ The core of the analysis is the Hindley–Milner type inference algorithm. It forms a system of equations from the type constraints and tries to solve them.
 - ▶ Unfortunately, due to lack of time, our type evaluation does not infer every Haskell type yet, just a small subset.
 - ▶ Improvements are in progress.
- 



THE END >>

Any questions? Thanks for your attention.