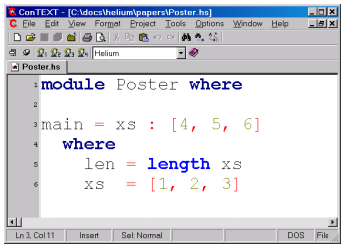
Helium
FOR LEARNING HASKELL

Bastiaan Heeren, Arjan van IJzendoorn, Daan Leijen
Rijk-Jan van Haaften, Arie Middelkoop

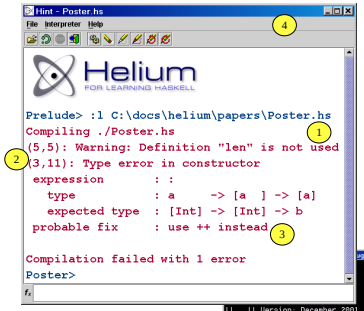
Universiteit Utrecht

1. Introduction

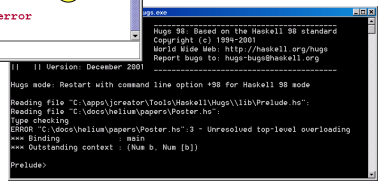
Helium is a user-friendly compiler designed especially for learning the functional programming language Haskell. Quality of the error messages has been the main concern both in the choice of the language features and in the implementation of the compiler. Helium is almost full Haskell with as the most notable difference the absence of type classes. Our goal is to let students learn functional programming more quickly and with more fun. The compiler has been successfully employed in two introductory programming courses at our institute.



A program with a type error



The message given by Helium



The message given by Hugs

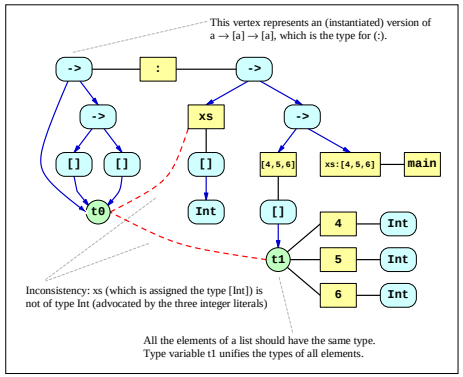
Features

1. Warnings for several common mistakes
2. Positions of error messages are exact
3. Hints and suggestions to fix programs
4. A graphical interpreter

2. The Type Inferencer

Extra effort has been spent on producing concise and understandable type error messages. As opposed to most modern compilers, the process of type inferencing is constraint-based, which clearly separates the collection of type constraints (the specification) from solving those constraints (the implementation). Therefore, the type inferencer can not only simulate well known algorithms such as W and M, but it can also solve constraints by using type graphs. A type graph is an advanced data structure to represent a substitution, which also keeps track of reasons for unifications. The extra information that is available during type inferencing paves the way for better type error messages.

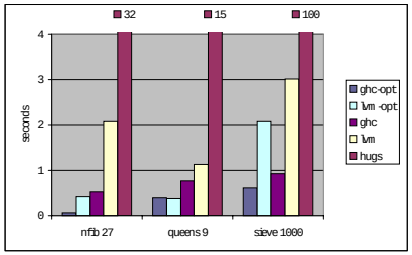
Heuristics determine the most likely site of error and create a precise error message. In many cases, a type error is accompanied by additional hints to guide the programmer in fixing ill-typed programs. The process of type inferencing can be tuned by supplying type inference directives externally.



A typegraph for the function main = xs : [4, 5, 6] (with xs :: [Int])

3. The Lazy Virtual Machine

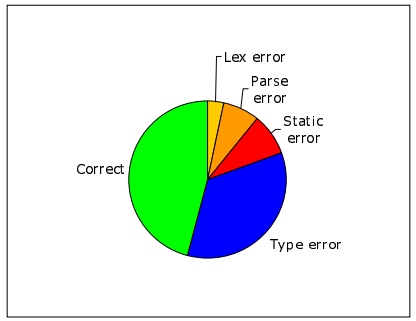
For a given source file, the Helium compiler produces an .lvm file, which can be executed by the Lazy Virtual Machine. The LVM is specifically designed to execute lazy (or non-strict) languages. It defines a portable instruction set and file format, similar to for instance the Java Virtual Machine (JVM). The instruction set is strongly based on the Spineless Tagless G- machine. At the present time, the generated code is unoptimized and rather naive. In spite of this, the runtime system performs quite well in practice. It is an order of magnitude faster than Hugs, and only about three times slower than unoptimized GHC code.



Speed comparison for three benchmarks

4. The Logger

A special logging facility has been set up to record compiled programs during lab sessions for an introductory course on functional programming. The students involved in this experiment have been informed about the logger in advance, and, if desired, it can be turned off. The collection of recorded programs not only reflects the problems that students encounter while learning Haskell, but it also gives some insights in the learning process over time. By analyzing this (anonymized) set of programs, we hope to further improve the compiler.



Results of Compilations by Students

5. Future Extensions

- Support for Haskell 98 type classes
- A basic GUI library
- Warnings and hints for incomplete pattern matches
- Recognizing standard functions, such as map and filter
- Additional heuristics to determine the most likely type error

Acknowledgements

We would like to thank Arthur Baars, Jeroen Fokker, Jurriaan Hage, Martijn Lammerts, Andres Löh, Doaitse Swierstra and all the students who used Helium for their help.

http://www.cs.uu.nl/helium