Query-Driven Language Server Architecture using Second-Order Abstract Syntax

Danila Danko [1] Nikita Strygin [1]

Supervisor: Nikolai Kudasov [1]

[1]Innopolis University

November 12, 2024

Problem

- Programming language researchers, enthusiasts, and students sometimes prototype new languages.
- Some of the language authors try to make their languages user-friendly by supporting the language server protocol (LSP) [1].
- Language servers for all mainstream languages support "go to definition", "lookup the type on hover", "rename all occurences".
- There exist frameworks that support building languages integrated with the LSP, e.g., Langium [2] for languages implemented in `TypeScript` and `lsp-tree-sitter` [3] for languages implemented in Python.
- However, to our best knowledge, there is no such framework for languages implemented in Haskell.

Solution

- Language servers share some features, such as "go to definition".
- Our assumption is that some of these features can be implemented at least partially in a language-agnostic framework for implementing new languages in Haskell.
- A promising approach is to work with the Second-Order Abstract Syntax (SOAS) of a language.
- SOAS allows to work with scope resolution ("go to definition"), variable bindings ("lookup the type on hover"), and substitution ("rename all occurences") in a language-agnostic way.
- We plan to use the Free Foil [4] to generate the Haskell code for SOAS of a language.

Literature review

- Read [5] and [4]

Practice with Free Foil

- Complete exercises with ASTs
- Complete them with Free Foil

Practice with a simple typed language

- Implement a type checker and interpreter for a simple typed language, e.g. Simply Typed Lambda Calculus.
- Features:
  - Single module on the input and the output
  - Use BNFC for pretty-printing and parsing
  - Use Free Foil and Template Haskell
  - Show error location in the code

[1] “Language server protocol.”
Page Version ID: 1236113985.

[2] “Langium.”

[3] “neomutt/lsp-tree-sitter.”
original-date: 2023-06-19T15:04:15Z.

[4] N. Kudasov, R. Shakirova, E. Shalagin, and K. Tyulebaeva, “Free foil: Generating efficient and scope-safe abstract syntax.”

[5] D. Maclaurin, A. Radul, and A. Paszke, “The foil: Capture-avoiding substitution with no sharp edges,” in *Proceedings of the 34th Symposium on Implementation and Application of Functional Languages*, pp. 1–10, ACM.

Thank You