Refactoring with the Haskell-tools framework

Boldizsár Németh

Eötvös Loránd University, Budapest

nboldi@elte.hu

October 7, 2016

Before starting

What we will need:

- GHC 8.0.1: haskell.org/platform
- cabal update
- cabal install haskell-tools-refactor-0.2.0.0

Introduction

What is Haskell-tools?
: Haskell-tools is a new set of development tools for Haskell. It is built on GHC and based on a custom program representation.

What is Haskell-tools Refactor?
: A framework to write code transformations for Haskell. It is part of the Haskell-tools family.

What is ht-refact?
: A refactoring tool that supports a set of refactorings. It is based on Haskell-tools Refactor. Will provide an interface for editor integration and scripting.

Main goals of Haskell-tools Refactor

- Writing a refactoring should be easy.
- Work with the latest language features and extensions.

Requirements

- Only change what need to be changed.
- Care for indentation (and formatting).

Support writing refactorings

- Combine the information to a single representation
- A representation suited for transformation
- Semantic information available in AST nodes
- Monadic context to support refactorings
- References (a lens-like property abstraction) for accessing elements.
- Generics (currently Uniplate-based)

Representation

All information is collected into one representation:

- Lexical information from GHC
- Template Haskell and role annotations from parsed source
- Unique names from renaming
- Types and kinds from type checking
- Information about definitions in scope and imported

Monadic refactoring context

You can use the monadic context to:

- Automatically import the names that are used in the definitions
- Access information in the GHC

Keep the formatting of the code

The representation contains the original format of the source code, so layout, comments, formatting is kept during the transformation. To provide a representation that can be simply pretty-printed we go through several steps:

1. The ranges of AST elements are extracted from GHC syntax tree.
2. Cutting ranges of children from ranges of parents and replacing them with placeholders.
3. Replacing the remaining ranges with the corresponding text from source files.
4. Now the representation can be transformed and the results are simply pretty-printed.

Current status

- GHC 8.0.1 support
- 5 useful refactorings implemented
- All comments and formatting kept
- Layout sensitive
- Refactors multiple modules, can transform its own source

Implemented refactorings

- Extract binding
- Rename
- Generate type signature
- Organize imports
- Generate exports

Future plans

- 2016 - Start supporting editor integration. Refinements in project handling.
- early 2017 - Editor support. Better layout handling. Refinement of refactorings.

Demo

`haskelltools.org`

Write our own refactoring

Convert `f (g ( e ))`

to `f $ g $ e`

`https://github.com/haskell-tools`
`/haskell-tools/wiki/HaskellExchange2016`

Emberi Erőforrások Minisztériuma

Supported Through the New National Excellence Program of the Ministry of Human Capacities