Генерация описаний алгебраических типов данных на основе JSON

О. И. Маросеев, ФИ и ИТ
Научный руководитель: асс. каф. ИВЭ А. М. Пеленицын

Институт математики, механики и компьютерных наук ЮФУ

20 июня 2016 г.

JSON

JSON (JavaScript Object Notation) — простой формат обмена данными, основанный на подмножестве языка программирования JavaScript.

Пример JSON-файла

```
{
    "firstName": "John",
    "lastName" : "Smith",
    "age" : 25
}
```

Алгебраические типы данных в Haskell

Алгебраические типы данных — вид составных типов, представленных типом-произведением, типом-суммой, либо комбинацией: суммой произведений.

АТД: двоичное дерево

```
data Tree a = Leaf a
            | Node (Tree a) (Tree a)
```

Постановка задачи

Цель

Создание механизма для анализа входных данных JSON и генерации определений типов функционального языка программирования Haskell

1. Получение абстрактного синтаксического дерева (далее — AST) по исходному JSON.
2. Преобразование AST в структуру, пригодную для генерации алгебраического типа данных Haskell.
3. Добавление возможности использования полученного типа в другой программе

Альтернативы подходы к реализации

Разбор JSON

- Создание специализированного парсера «с нуля»,
- генерация максимально специализированного парсера,
- библиотека работы с JSON (Aeson).

Генерация АТД

- GHC.API (+ потенциально плагин к комиплятору).
- Template Haskell

Альтернативы подходы к реализации

Разбор JSON

- Создание специализированного парсера «с нуля»,
- генерация максимально специализированного парсера,
- библиотека работы с JSON (Aeson).

Генерация АТД

- GHC.API (+ потенциально плагин к комиплятору).
- Template Haskell

Получение AST

Data.Aeson — библиотека для работы с файлами в формате JSON, написанная на языке Haskell. Aeson позволяет получить абстрактное синтаксическое дерево по JSON.

Тип для представления AST

```
data Value
  = Object Object
  | Array Array
  | String Text
  | Number Scientific
  | Bool Bool
  | Null
```

Преобразование синтаксического дерева к [Dec]

Представление АТД в Template Haskell

```
DataD Cxt Name [TyVarBndr] [Con] Cxt :: Dec
```

Накопление результатов (монада State)

```
State [Dec] ()
```

Монадическая свертка по ключу и значению

```
foldlWithKeyM :: (Monad m) =>
(b -> k -> a -> m b) -> b -> HashMap k a -> m b
```

Обход AST

Замечание

При анализе AST можно встретить вложенные объекты

Функция, запускающая обработку AST

```
convertObject :: String -> Value -> State [Dec] ()
```

Использование сгенерированного типа

Вклейка кода — преобразование шаблона (структура Template Haskell) с данным параметром в обычный Haskell-код во время компиляции и вклеивает его на то же место. Вклейка производится оператором $(..) .

Использование вклейки

```
{-# LANGUAGE TemplateHaskell #-}
import Language.Haskell.TH
import DataCreation

$(getDataFromJSON)

main = ...
```

Результаты

1. Рассмотрены принципы метапрограмирования на языке Haskell.
2. Реализована генерация типов данных по JSON.
3. Реализация оформлена в виде cabal-пакета.
4. Исходный код доступен в Git-репозитории:
   `https://github.com/olegmaroseev/JSONtoDATA`