CS314. Функциональное программирование

Институт математики, механики и компьютерных наук ЮФУ
Направление «Фундаментальная информатика и информационные технологии»
2017/2018 учебный год, осенний семестр

Программа курса

Введение в функциональное программирование на языке Haskell. Функциональные языки и важнейшие черты функционального стиля программирования. Язык Haskell и среда программирования. Рекурсивные и итеративные вычислительные процессы. Базовый синтаксис языка Haskell: операции и функции, условные выражения, охранные выражения, выражение `let` и конструкция `where`, типы и классы типов, пары и кортежи, списки, генераторы списков. Функции высшего порядка. Способы определения функций: сопоставление с образцом, каррирование и частичное применение, сечения, анонимные функции, манипуляции с аргументами, композиция, функция `($)`. Простейшие функции для обработки списков. Функции высшего порядка для работы со списками: `map`, `filter` и др. Идея свёртки, левая и правая свёртки, их варианты, сканирование списка.

Синонимы типов. Объявление типов с помощью ключевого слова `newtype`. Алгебраические типы данных: типы-перечисления, типы-контейнеры, общий случай, синтаксис записей, расширения компилятора GHC NamedFieldPuns и RecordWildCards, параметризованные и рекурсивные типы. Типы `Maybe a` и `Either a b`.

Компиляция и запуск программ. Чистые функции и действия ввода-вывода, монада `IO`. Блоки `do`. Простейший ввод-вывод, обработка текстовых файлов. Чтение аргументов командной строки. Генерация случайных чисел.

Строгие и нестрогие функции, примеры, выражение `undefined`, строгость по аргументу, нестрогая семантика языка Haskell. Частичная нестрогость. Задумки (*thunks*) и слабая головная нормальная форма, исполнение программы с учётом ленивости. Хвостовая рекурсия и её оптимизация, аккумулирующие параметры в условиях ленивости, функция `seq`, расширение BangPatterns. Исполнение свёрток, строгие свёртки, свёртки с ленивыми комбинирующими функциями. Рекурсия, ограниченная конструктором данных. Идиомы ленивых вычислений, их недостатки. Сложность операций обработки списков, влияние ленивости на сложность операций.

Важнейшие структуры данных: списки, последовательности, массивы, множества и отображения. Импорт и создание модулей. Инструменты и важнейшие библиотеки для поддержки разработки ПО на языке Haskell. Модульное тестирование, тестирование на моделях, библиотеки HUnit, QuickCheck, tasty. Профилирование кода и библиотека Criterion.

Классы типов и обобщённые вычисления. Классы типов и их экземпляры, наследование классов типов. Классы типов `Eq` и `Show`, определение собственных классов типов. Сорта типов. Класс типов Monoid и его экземпляры, законы моноидов. Класс типов `Foldable`. Класс типов Functor и его экземпляры, законы функторов. Понятие вычислительного контекста, примеры. Аппликативные функторы: определение, основные экземпляры, законы. Использование функторов, аппликативных функторов и моноидов для описания обобщённых вычислений.

Монады и комбинирование вычислений, монадическое связывание, смысл do-синтаксиса, вспомогательные функции для работы с монадами, связь монад с функторами и аппликативными функторами, монады и моноиды, классы `Alternative` и `MonadPlus`.

Экземпляры класса типов `Monad` для `IO`, `Maybe` и списков. Специальные монады: `Reader`, `Writer`, `State`, `ST`.

Функциональные парсеры: поиск типа для парсера, парсер как функтор, аппликативный функтор и монада, множество парсеров как моноид, простейшие парсеры, выбор и повторения, разбор выражений с грамматикой.

Проблема комбинирования монад. Преобразователи стандартных монад, примеры. Реализация преобразователей монад. Примеры построения и использования монадных стеков средствами преобразователей монад.

Продвинутый Haskell. Метапрограммирование и Template Haskell: манипулирование абстрактными синтаксическими деревьями, внедрение кода, реификация, квазицитирование, примеры (генерация функций-проекторов, многострочные литералы, предикаты на типах данных).

Программирование на уровне типов: виды и способы объявления семейств типов, примеры.

Компилятор GHC. Состав компилятора GHC, порядок компиляции одного модуля, внутренний язык Core, система времени исполнения RTS. Способы расширения компилятора GHC: правила переписывания термов, плагины компилятора, использование компилятора как библиотеки.

Литература

1. М. Липовача. Изучай Haskell во имя добра. ДМК Пресс, 2012.
2. А. Мена. Изучаем Haskell. Библиотека программиста. Питер, 2014.
3. B. O'Sullivan, D. Stewart, J. Goerzen. Real World Haskell. O'Reilly, 2008.
4. С. Марлоу. Параллельное и конкурентное программирование на языке Haskell. ДМК Пресс, 2014.
5. R. Bird. Thinking Functionally with Haskell. Cambridge University Press, 2014.