Аннотация к магистерской диссертации “Структурирование вычислений с эффектами”

Г. А. Лукьянов

12 Июня 2017

1 Введение

Необходимо, чтобы программное и аппаратное обеспечение обеспечение были надёжными. Например, медицинские импланты должны автономно работать в организме пациентов, самостоятельно адаптироваться к изменениям условий работы и иметь время службы порядка десятилетий, а минутное неисправность системы обработки банковских транзакций может привести к значительным издержкам. Безопасность и надёжность программного и аппаратного обеспечения может быть достигнута тестированием и контролем качества на поздних этапах разработки, однако всегда существует возможность оставить вне рассмотрения сценарий работы, потенциально ведущий к катастрофе.

В то время как позднее тестирование не предоставляет полных гарантий надёжности, формальные методы являются систематическим подходом к разработке сложных систем с гарантией корректности “по построению”. Например, методы, известные как *проверка моделей* (англ. *model checking*) производят исчерпывающий анализ множества состояний системы и оценивают её статус в каждом из них, полностью исключая возможность непредвиденного поведения (при условии адекватности спецификации).

Одним из классов методов формальной верификации является использование языков программирования с богатыми системами типов для спецификации свойств разрабатываемой системы и последующего доказательства корректности путём проверки типов. Это возможно благодаря соответствию Карри-Говарда — прямому соотношению между типами и высказываниями, а также программами и доказательствами. Более подробную информацию можно получить из статьи Филлипа Вадлера “Высказывания как типы” [1] — замечательная компиляция исторических фактов и точек зрения на соответствию Карри-Говарда.

Развитие теории типов и теории языков программирования привело к появлению и распространению систем контроля вычислительных эффектов — формализмов, позволяющих классифицировать программы по наличию в них побочных эффектов: отделить *чистые* вычисления от обременённых поведением, требующем особого контроля. Таким поведением может быть файловый ввод-вывод, сетевое взаимодействие, работа изменяемыми переменным или нечто иное. Впервые формализация вычислений побочных эффектов и монадическая детонационная семантика для $lambda$-исчисления с эффектами была предложена Могги [2]. Впоследствии, Вадлер [3] реализовал монады в языке программирования Haskell. Было обнаружено, что многие привычные концепции функционального программирования, такие как, например, списки, опциональные типы и комбинаторы парсеров, могут быть единообразно структурированы с помощью монад. Со временем, возникли новые требования к структурированию вычислений с эффектами, например, модульная комбинация нескольких эффектов в одном вычислении. Это привело к развитию как монадического подхода — изобретению преобразователей монад [4], так и альтернативных подходов, в частности, алгебраических эффектов и обработчиков эффектов [5].

2 Постановка задачи

Данная работы нацелена на анализ и развитие методов структурирования вычислений с эффектами. Производится анализ и выяснение границ выразительности существующих методов путём решения модельной задачи: реализации библиотек комбинаторов парсеров. Также целью является апробировать рассматриваемые методы на более масштабном программном обеспечении.

Для достижения поставленных целей предлагается решить следующий задачи:

- Разработать библиотеку монадических комбинаторов парсеров для иллюстрации использования преобразователей монад в Haskell.
- Разработать библиотеку комбинаторов парсеров на основе расширяемых эффектов — реализации концепции расширяемых эффектов в Haskell.
- Выделить отличия преобразователей монад и расширяемых эффектов.
- Проанализировать возможности экспериментального языка программирования Frank, поддерживающего концепцию алгебраических эффектов на уровне ядра, для реализации библиотек комбинаторов парсеров.
- Произвести анализ производительности разработанных библиотек, сравнить с популярными аналогами и представить отчёт в виде приложения к диссертации.
- Продемонстрировать преимущества типизированного функционального языка программирования с системой эффектов при реализации крупных систем: разработать полномасштабное веб-приложение на Haskell.

3 Результаты

В данной диссертации разработаны библиотеки комбинаторов парсеров на основе различных формализмов для контроля вычислительных эффектов: Haskell-библиотека на основе преобразователей монад [6] — наиболее широко распространённого в Haskell-сообществе подхода к структурированию вычислений с эффектами; Haskell-библиотека на основе расширяемых эффектов [7] — альтернативному преобразователям монад подходу к контролю побочных эффектов, являющимся Haskell-реализацией алгебраических эффектов и обработчиков; прототипные реализации библиотек для экспериментального языка программирования Frank [8], предоставляющего средства для естественного программирования с использованием алгебраических эффектов. Для Frank реализована библиотека, трактующая парсер как комбинацию нескольких эффектов, а также проанализирована возможность реализации парсера как обособленного эффекта.

Приложение к диссертации содержит сравнение производительности предложенных библиотек комбинаторов парсеров с распространёнными аналогами.

В диссертации докладываются результаты разработки реальной программной системы с использованием чистого функционального языка программирования Haskell и подходов к контролю вычислительных эффектов, обсуждаемых ранее в диссертации. Система мониторинга активности студентов “Student’s Big Brother” [9] призвана помочь преподавателям практических занятий по программированию в реальном времени следить за производительностью студентов в течении лабораторных работ, равномерно распределять внимание между всеми студентами, не отвлекать студентов от работы для мониторинга их деятельности, и, как результат, эффективнее организовывать учебный процесс. Система организована таким образом, что мониторинг осуществляется прозрачным для студентов образом: в начале занятия им требуется запустить агент сбора данных, дальнейшая работа системы происходит в автоматическом режиме. Помимо практической ценности для учебного процесса, реализованная система имеет особенности реализации, иллюстрирующие полезность богатства системы типов для уплощения разработки и поддержки многокомпонентного программного обеспечения. Реализация системы производит разделение типов

между клиентом и сервером, таким образом исключая всякую возможность их расхождения и обеспечивая проверку соответствия на этапе компиляции. Реализация системы использует два подхода к контролю побочных эффектов: серверная часть основана на преобразователях монад, а в коде агента сбора данных применяются расширяемые эффекты. Было произведено внедрение разработанной системы в учебный процесс направления Фундаментальная информатика и информационные технологии [10] мехмата Южного Федерального Университета (Ростов-на-Дону). Преподаватель, проводивший занятия с использованием системы сообщил об улучшении показателей распределения своего внимания, а также об увеличении производительности студентов.

Выделены два направления дальнейшей работы. Проблема выразительности программирования с алгебраическими эффектами на Frank: расширение системы эффектов языка программирования Frank для поддержки GADT-синтаксиса в интерфейсах эффектов. Продолжение разработки системы “Student’s Big Brother” — развитие веб-интерфейса преподавателя: добавление возможности просмотра истории изменений, оповещений об изменениях. Для этого возможно использовать такие подходы как. например, функциональное реактивное программирование.

Библиография

1. *Wadler P.* Propositions As Types // Commun. ACM. — New York, NY, USA, 2015. — Нояб. — Т. 58, № 12. — С. 75—84. — ISSN 0001-0782. — URL: http://doi.acm.org/10.1145/2699407.
2. *Moggi E.* Notions of Computation and Monads // Inf. Comput. — Duluth, MN, USA, 1991. — Июль. — Т. 93, № 1. — С. 55—92. — ISSN 0890-5401. — URL: http://dx.doi.org/10.1016/0890-5401(91)90052-4.
3. *Wadler P.* The Essence of Functional Programming // Proceedings of the 19th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. — Albuquerque, New Mexico, USA : ACM, 1992. — С. 1—14. — (POPL ’92). — ISBN 0-89791-453-8. — URL: http://doi.acm.org/10.1145/143165.143169.
4. *Liang S.*, *Hudak P.*, *Jones M.* Monad Transformers and Modular Interpreters // Proceedings of the 22Nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. — San Francisco, California, USA : ACM, 1995. — С. 333—343. — (POPL ’95). — ISBN 0-89791-692-1. — URL: http://doi.acm.org/10.1145/199448.199528.
5. *Bauer A.*, *Pretnar M.* Programming with algebraic effects and handlers // J. Log. Algebr. Meth. Program. — 2015. — Т. 84, № 1. — С. 108—123. — DOI: 10.1016/j.jlamp.2014.02.001. — URL: http://dx.doi.org/10.1016/j.jlamp.2014.02.001.
6. Parser combinators on top of monad transformers. — URL: https://github.com/geo2a/markdown_monparsing/tree/experimental (дата обр. 11.06.2017).
7. Parser combinators on top of extensible effects. — URL: https://github.com/geo2a/ext-effects-parsers (дата обр. 11.05.2017).
8. Parser combinators in Frank. — URL: https://github.com/geo2a/frankoparsec (дата обр. 12.05.2017).
9. Student’s Big Brother source code repository. — URL: https://github.com/geo2a/students-big-brother (дата обр. 10.06.2017).
10. Сайт ФИИТ. — URL: http://it.mmcs.sfedu.ru/ (дата обр. 12.05.2017).