Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Факультет информационных технологий и программирования
Кафедра компьютерных технологий

Рыбак Андрей Викторович

Представление структур данных индуктивными семействами и доказательства их свойств

Научный руководитель: Я. М. Малаховски

Санкт-Петербург
2014

Содержание

Введение

Структуры данных используются в программировании повсеместно для упрощения хранения и обработки данных. Свойства структур данных происходят из инвариантов, которые эта структура данных соблюдает.

Практика показывает, что тривиальные структуры и их инварианты данных хорошо выражаются в форме индуктивных семейств. Мы хотим узнать насколько хорошо эта практика работает и для более сложных структур.

В данной работе рассматривается представление в форме индуктивных семейств структуры данных приоритетная очередь типа «двоичная куча».

Глава 1. Обзор

В данной главе производится обзор предметной области и даются определения используемых терминов.

1.1. Лямбда-исчисление

Лямбда-исчисление (λ-calculus) — вычислительный формализм с тремя синтаксическим конструкциями, называемыми *пре-лямбда-термами*:

- *вхождение переменной*: v. При этом $v \in V$, где V — некоторое множество имён переменных;
- *лямбда-абстракция*: $\lambda x.A$, где x — имя переменной, а A — пре-лямбда-терм. При этом терм A называют *телом абстракции*, а x перед точкой — *связыванием*.
- *лямбда-аппликация*: BC;

и одной операцией *бета-редукции*. При этом говорят, что вхождение переменной является *свободным*, если оно не связано какой-либо абстракцией. *Лямбда-термы* — это пре-лямбда-термы, факторизованные по отношению *альфа-эквивалентности*.

Альфа-эквивалентность (α-equality) отождествляет два пре-лямбда-терма, если один из них может быть получен из другого путём некоторого *корректного* переименовывания переменных — переименования не нарушающего отношение связанности.

Бета-редукция (β-reduction) для лямбда-терма A выбирает в нём некоторую лямбда-аппликацию BC, содержащую лямбда-абстракцию в левой части A, и заменяет свободные вхождения переменной, связанной A, в теле самой A на терм C.[1]

Два лямбда-терма A и B называются *конвертабельными*, когда существует две последовательности бета-редукций, приводящих их к обще-

[1] В терминах пре-лямбда-термов это означает замену свободных вхождений в теле A на пре-терм C так, чтобы ни для каких переменных не нарушилось отношение связанности. То есть, в пре-терме A следует корректно переименовать все связанные переменные, имена которых совпадают с именами свободных переменных в C.

му терму C. Или, эквивалентно, когда термы A и B состоят с друг с другом в рефлексивно-симметрично-транзитивном замыкании отношения бета-редукции, также называемом отношением *бета-эквивалентности*.

За более подробной информацией об этом формализме следует обращаться к [1] и [2].

1.2. Функциональное программирование

Функциональное программирование — парадигма программирования, являющаяся разновидностью декларативного программирования, в которой программу представляют в виде функций (математическом смысле этого слова, а не в смысле, используемом в процедурном программировании), а выполнением программы считают вычисление значений применения этих функций к заданным значениям. Большинство функциональных языков программирования используют в своём основании лямбда-исчисление (например, Haskell [3], Curry [4], Agda [5], диалекты LISP [6–8], SML [9], OCaml[10]), но существуют и функциональные языки явно не основанные на этом формализме (например, препроцессор языка C и шаблоны в C++).

1.3. Алгебраические типы данных и сопоставление с образцом

Алгебраический тип данных — вид составного типа, то есть типа, сформированного комбинированием других типов. Комбинирование осуществляется с помощью алгебраических операций — сложения и умножения.

Сумма типов A и B — дизъюнктное объединение исходных типов. Значения типа-суммы обычно создаются с помощью *конструкторов*.

Произведение типов A и B — прямое произведение исходных типов, кортеж типов.

1.3.1. Сопоставление с образцом

Сопоставление с образцом — способ обработки объектов алгебраических типов данных, который идентифицирует значения по конструктору и извлекает данные в соответствии с представленным образцом.

1.4. Теория типов

Теория типов — раздел математики изучающий отношения типизации вида $M : \tau$ и их свойства. M называется *термом* или *выражением*, а τ — типом терма M.

Теория типов также изучает правила для *переписывания* термов — замены подтермов в выражениях другими термами. Такие правила также называют правилами *редукции* или *конверсии* термов. Редукцию терма x в терм y записывают: $x \to y$. Также рассматривают транзитивное замыкание отношения редукции: $\xrightarrow{*}$. Например, термы $2 + 1$ и 3 — разные термы, но первый редуцируется во второй: $2 + 1 \xrightarrow{*} 3$. Если для терма x не существует терма y, для которого $x \to y$, то говорят, что терм x — в *нормальной форме*.

1.4.1. Отношение конвертабельности

Два терма x и y называются *конвертабельными*, если существует терм z такой, что $x \xrightarrow{*} z$ и $y \xrightarrow{*} z$. Обозначают $x \xleftrightarrow{*} y$. Например, $1 + 2$ и $2 + 1$ — конвертабельны, как и термы $x + (1 + 1)$ и $x + 2$. Однако, $x + 1$ и $1 + x$ (где x — свободная переменная) — не конвертабельны, так как оба представлены в нормальной форме. Конвертабельность — рефлексивно-транзитивно-симметричное замыкание отношения редукции.

1.4.2. Интуиционистская теория типов

Интуиционистская теория типов основана на математическом конструктивизме [11].

Операторы для типов в ИТТ: