Генерация расширяемого кода по типам данных на языке OCaml

Обобщенное программирования с комбинаторами и объектами

Косарев Дмитрий Сергеевич

Научный руководитель:
доктор физико-математических наук, профессор,
Терехов Андрей Николаевич

Рецензент:
доктор технических наук,
Новиков Федор Александрович

18 июня 2019

Предыстория

- Встраиваемый DSL для реляционного (логического) программирования OCanren
- Статья “Typed Embedding of a Relational Language in OCaml” DOI ссылка
- Работа раскрыла несовершенство имеющихся подходов к обобщённому программированию

Обобщённое программирование (1/2)

- Свойственно флагманским языкам функционального программирования: Haskell, OCaml
- Скорее не свойственно (но возможно) для статически типизированных OO языков: Java, C#, C++.
- Вообще не применимо к динамическим языками, там нет этой проблемы.

Обобщённое программирование (1/2)

Идея: генерировать (во время компиляции) некоторые функции по описаниям типов данных.
Например, преобразование в строковое понятное человеку представление; сериализация и десериализация; преобразования похожие свёртки и т.д

Пример

$$\textbf{type}\ (\alpha, \beta, \ldots)\ \mathrm{t} = \ldots$$

$$\textbf{val}\ \mathrm{show}: (\alpha \to \mathrm{string}) \to (\beta \to \mathrm{string}) \to \ldots \to (\alpha, \beta, \ldots)\ \mathrm{t} \to \mathrm{string}$$

Реализовывать такой однотипный код руками неудобно

Избранные работы про обобщённое программирование

Ralf Lämmel and Simon Peyton Jones (2003): *Scrap Your Boilerplate: A Practical Design Pattern for Generic Programming*, doi:10.1145/640136.604179

Dmitry Boulytchev and Sergey Mechtaev (2011): *Efficiently Scrapping Boilerplate Code in OCaml.*

François Pottier (2017): *Visitors Unchained*. doi:10.1145/3110272.

Основная задача

Разработка подхода для обобщенного программирования, который позволит порождаться *расширяемые* преобразования

На сегодняшний день такое умеет делать только Visitors

Результаты

Библиотека обобщённого программирования GT (Generic transformers) для языка OCaml

- Позволяет создавать *расширяемые* трансформации с помощью объектов, как и у Visitors
- По-другому типизируются объекты
- Комбинаторный интерфейс
- Поддерживаются большее многообразие типов
- Нет потенциальных проблем с безопасностью, которые есть у Visitors и Scrap Your Boilerplate

Расширяемость с помощью объектов

```
type ... t = C1 of ... | ... | Cn of ...
class old_transformation = object
  method C1 = ...
  ...
end
```

Конструкторы кодируются в методы один к одному

```
class new_transformation = object
  inherit old_transformation
  method Ci = (* new implementation *)
end
```

Типизация

Visitors полагаются на трюк, для сокращения количества типовых параметров до одного

```
class [σ] new_transformation = object (this : σ)
  ...
end
```

Недостатки:

- Такая типизация Visitors неприменима в файлах интерфейса
- Она не позволяет реализовать поддержку полиморфных вариантных типов языка OCaml

В Generic Transformers используется стандартный способ использования типовых параметров (они указываются явно). Полиморфные вариантные типы поддержаны

Комбинаторный интерфейс

В Visitors вызов преобразования типа $(\alpha, ...)\tau$ выглядит как

(**new** classname)#visit_τ

В то время как все остальные библиотеки используют (комбинаторный интерфейс

transformτ transformα ... (x: τ)

Виды методов и безопасность

SYB и Visitors позволяют применять преобразования *ко всем вхождениям типа* (например, `int`) в структуре данных.

Это позволяет преодолевать барьер инкапсуляции.

В Generic Transformers конструкторы алгебраических типов кодируются в методы один к одному. Преобразования типов нельзя переопределить.

Результаты

Библиотека обобщённого программирования GT (Generic transformers) для языка OCaml

- Позволяет создавать *расширяемые* трансформации с помощью объектов
- Другая типизация объекты, по сравнению с конкурентом
- Комбинаторный интерфейс
- Поддерживаются большее многообразие типов
- Нет потенциальных проблем с безопасностью

Результаты

Библиотека обобщённого программирования GT (Generic transformers) для языка OCaml

- Позволяет создавать *расширяемые* трансформации с помощью объектов
- Другая типизация объекты, по сравнению с конкурентом
- Комбинаторный интерфейс
- Поддерживаются большее многообразие типов
- Нет потенциальных проблем с безопасностью

Конец

Ссылки I

- Ralf Lämmel & Simon Peyton Jones (2003): *Scrap Your Boilerplate: A Practical Design Pattern for Generic Programming.*
  Доступно по DOI ссылке
- François Pottier (2017): *Visitors Unchained.*
  Доступно по DOI ссылке
- Dmitry Boulytchev & Sergey Mechtaev (2011): *Efficiently Scrapping Boilerplate Code in OCaml.*
  Доклад на ML Workshop при ICFP 2011
- Dmitry Kosarev & Dmitry Boulytchev (2016): *Typed Embedding of a Relational Language in OCaml.*
  Доступно по DOI ссылке