Переоснащение неявных модулей для языка 1ML

Трилис Алексей Андреевич

научный руководитель: к.ф.-м.н. Д.А. Березун

НИУ ВШЭ — Санкт-Петербург

12 апреля 2021 г.

Введение. Ad hoc полиморфизм

- Ad hoc полиморфизм — свойство языка, позволяющее функциям иметь различную семантику в зависимости от типов аргументов
- Например, `print` и `+`
- В процедурных и объектно-ориентированных языках обычно достигается перегрузкой
- Но в языках с мощным выводом типов нужны более сложные методы

Введение. Семейство языков ML

- ML, OCaml, SML, F#, ...
- Функциональные языки с мощным выводом типов
- Активно используется в разработке и исследовании языков программирования
- А также в верификации, финансах, веб-разработке и других областях
- Отсутствует ad hoc полиформизм:
  - `+` для `int`, `+.` для `float`
  - `print_int`, `print_float`, `print_string`, ...

Введение. Модули в ML

- Продвинутая система модулей, основанная на теории зависимых типов[1]
- Язык разделён на два слоя: основной и модульный
- Модульный язык более мощный, но требует

[1] MacQueen, “Using Dependent Types to Express Modular Structure”, 1986.

Введение. 1ML

- Было показано[2], что модули могут быть выражены и без использования теории зависимых типов
- В результате этих исследований был создан экспериментальный диалект 1ML[3]

[2] Rossberg, Russo и Dreyer, “F-Ing Modules”, 2010.

[3] Rossberg, “1ML – Core and Modules United (F-ing First-Class Modules)”, 2015.

Обзор литературы. Классы типов

```haskell
class Show a where
  show :: a -> String

instance Show Int where
  show = showSignedInt

show_twice x = show x ++ show x

show_twice : Show a => a -> string
```

- Впервые в Haskell[4], затем в Agda, Rust, ...
- Требуется каноничность — не более одного экземпляра для каждого типа

[4] Wadler и Blott, “How to Make Ad-Hoc Polymorphism Less Ad Hoc”, 1989.

Обзор литературы. Неявные параметры

```
1  trait Showable [T] { def show (x: T): String }
2
3  implicit object IntShowable extends Showable [Int] {
4    def show (x: Int) = x.toString
5  }
6
7  def show [T](x : T)(implicit s: Showable [T]): String =
       {
8    s.show(x)
9  }
10
11 show(7)(IntShowable)
12 show(7)
```

Впервые в Scala[5]

[5] Oliveira, Moors и Odersky, “Type Classes as Objects and Implicits”, 2010.

Обзор литературы. Модулярные классы типов

Обзор литературы. Неявные модули

Обзор литературы. Неявные модули

```
1  module type Show = sig
2    type t
3    val show : t -> string
4  end
5
6  implicit module Show_int = struct
7    type t = int
8    let show x = string_of_int x
9  end
10
11 implicit module Show_list {S : Show} = struct
12   type t = S.t list
13   let show x = string_of_list S.show x
14 end
15
16 let show {S : Show} x = S.show x
17
18 show 5 (* show {Show_int} 5 *)
19 show [1;2;3] (* show {Show_list (Show_int)} [1;2;3] *)
```

Цель и задачи

Цель:

- Дополнить язык 1ML поддержкой неявных модулей

Задачи:

- Реализация неявных модулей, повторяющих функциональность решения для OCaml
- Расширение решения новой функциональностью
- Тестирование и сравнение с решением для OCaml

Общая схема

Поиск модулей

Генерализация типов

Проверка завершаемости

Локальные неявные модули

Порядок разрешения неявных модулей

Неявные функторы

Тестирование и сравнение

Результаты

Ссылки I

MacQueen, David B. “Using Dependent Types to Express Modular Structure”. B: Proceedings of the 13th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages. POPL ’86. St. Petersburg Beach, Florida: Association for Computing Machinery, 1986, с. 277—286. ISBN: 9781450373470.

Oliveira, Bruno C.d.S., Adriaan Moors и Martin Odersky. “Type Classes as Objects and Implicits”. B: Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications. OOPSLA ’10. Reno/Tahoe, Nevada, USA: Association for Computing Machinery, 2010, с. 341—360. ISBN: 9781450302036.

Ссылки II

Rossberg, Andreas. “1ML – Core and Modules United (F-ing First-Class Modules)”. B: Proceedings of the 20th ACM SIGPLAN International Conference on Functional Programming. ICFP 2015. Vancouver, BC, Canada: Association for Computing Machinery, 2015, с. 35—47. ISBN: 9781450336697.

Rossberg, Andreas, Claudio V. Russo и Derek Dreyer. “F-Ing Modules”. B: Proceedings of the 5th ACM SIGPLAN Workshop on Types in Language Design and Implementation. TLDI ’10. Madrid, Spain: Association for Computing Machinery, 2010, с. 89—102. ISBN: 9781605588919.

Ссылки III

Wadler, P. и S. Blott. “How to Make Ad-Hoc Polymorphism Less Ad Hoc”. B: Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. POPL ’89. Austin, Texas, USA: Association for Computing Machinery, 1989, с. 60—76. ISBN: 0897912942.