План

Проект “Кеплер”

α-Кеплер

Евгений Бурмако

EPFL, LAMP

14 января 2012

Вступительные слова

Меня зовут Евгений Бурмако, с прошлой осени я работаю в Scala Team и параллельно учусь в аспирантуре EPFL на кафедре Мартина Одерского.

Сегодня мы обсудим прогресс проекта “Кеплер”, в рамках которого реализуются **макросы** и **квазицитаты** - средства метапрограммирования времени компиляции для Скалы. В процессе общения мы реализуем одну штуку, о которой я мечтал со времен знакомства с Немерле. Также мы поговорим о том, какие незапланированные применения макросов нашлись в процессе работы над проектом.

В предыдущем выступлении о макросах я рассматривал макросы с теоретической точки зрения, а сегодня будет, в основном, практика. Поэтому перед тем, как продолжить, может быть полезно зачесть слайды прошлого рассказа.

Важное замечание

Уже очень долгое время макросы в популярных языках программирования ассоциируются с макросами C/C++. Неудивительно, что многие автоматически воспринимают слово “макрос” в штыки.

В отличие от макросов препроцессора, наши макросы:

- Представляют собой код на полноценной Скале
- Работают с высокоуровневыми и типизированными деревьями выражений
- Выполняются в контексте компилятора, поэтому имеют доступ ко всей семантической информации, доступной компилятору
- Не изменяют синтаксис Скалы

Наши макросы напоминают макросы Лиспа, доработанные для поддержки богатого синтаксиса и статической типизации.

Начнем издалека

Одной из моих недавних задач в разработке scalac было допиливание `Code.lift`.

Лифт - это такая магическая функция, которая принимает на вход один-единственный аргумент и возвращает AST, которое соответствует этому аргументу. Например:

```
scala> lift{a + b}

val fv_a = freeVar("a", staticClass("scala.Int"), a)
val fv_b = freeVar("b", staticClass("scala.Int"), b)
Apply(Select(Ident(fv_a), newTermName("$plus")),
    List(Ident(fv_b))))
```

Если что, лифт практически незаменим для создания разного рода доменно-специфичных языков - от больших (запросы в OR/M) до маленьких (урлы в MVC фреймворках).

Баг в лифте

Текущая реализация лифта имеет один недостаток - если попробовать реифицировать выражение, содержащее операции над константами, то константы схлопнутся из-за того, что `doTypedApply` (функция, типизирующая применение методов) вызывает `constfold`.

```
scala> lift{2 * 2}

Literal(Constant(4))
```

Эта оптимизация полезна при генерации исполняемого кода, но для нас она не подходит, ибо код обычно лифтят, когда хотят узнать и проанализировать точную структуру кода.

Фиксим баг

Чтобы починить этот досадный недостаток, я залез в `Typers` и немножко подпилил `doTypedApply`. Теперь перед сверткой констант тайпер убеждается в том, что он не находится в `Code.lift`, и вот - проблема решена.