Московский государственный технический университет имени Н.Э. Баумана

Разработка механизма вывода типов с использованием системы типов Хиндли-Милнера

@Подробности темы, различные детали, неуместившиеся в названии темы@

Место проведения: *факультет РК, кафедра «Системы автоматизированного проектирования (РК-6)»*
Продолжительность: *@10 минут (продолжительность доклада)@*

Никитин Владимир Леонидович, студент группы РК6-75Б

nikitinvl@student.bmstu.ru

Россия, Москва, 2023.09.15 - 17 июня 2024 г.

Содержание доклада

Работа посвящена реализации механизма вывода типов для языка программирования Kodept.

Kodept - функциональный язык программирования, разрабатываемый в рамках обучения на кафедре. Идея языка состоит в применении C-подобного синтаксиса и использовании функциональной парадигмы.

Проверка типов - процесс, когда тем или иным образом проверяется правильность типа выражения согласно системе типов языка. Вывод типов - процесс, когда компилятор самостоятельно может узнать тип выражения исходя из окружающего контекста.

Объект исследований

Система типов.

Цель исследования

Целью курсового проекта является реализация механизма вывода и проверки типов для языка программирования Kodept в качестве его дальнейшего развития.

Задачи исследования

1. спроектировать представления AST в компиляторе,
2. реализовать анализатор областей видимости,
3. написать алгоритм для вывода типов.

Определение 1

Терм x - чаще всего элемент языка программирования, будь то переменная, константа, вызов функции и др. Например, в Haskell, термами будут: лямбда-функция `x-> x + 1`, определение переменной `let x = 1 in ()` и т.д.

Определение 2

Типом A обозначается метка, приписываемая объектам. Обычно каждому терму соответствует определенный тип - $x : A$. Типы позволяют строго говорить о возможных действиях над объектом, а также формализовать взаимоотношения между ними.

Система типов определяет правила взаимодействия между типами и термами, используя суждения:

$$\frac{\Gamma \vdash t : T_1, \Delta \vdash T_1 = T_2}{\Gamma, \Delta \vdash t : T_2}$$

Системы типов в современных языках программирования можно разделить:

1) по времени проверки соответствия типам: статическая и динамическая,
2) по поддержке неявных конверсий: сильная (англ. strong) и слабая,
3) по необходимости вручную типизировать выражение: явная и неявная.

Система типов Хиндли-Милнера популярна и хорошо изучена. Согласно классификации она статическая, сильная и неявная.
К ее термам относятся:

$a, b, c ::=$

x	(переменная)
$\lambda x.a$	(лямбда-функция)
$a(b)$	(вызов функции)
$let\ a = b\ in\ c$	(объявление переменной)
$1, 2, 3, \ldots$	(целочисленный литерал)
$1.1, 1.2, 10.0, \ldots$	(вещественный литерал)
(a, b)	(объединение)

В структуре программы почти любого компилятора можно выделить следующие части:

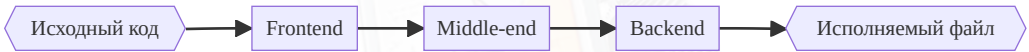

Рис. 1. Архитектура большинства современных компиляторов

Проект разрабатывается с использованием языка программирования Rust. Этот язык предлагает надежный концепт управления памятью, не имея при этом сборщика мусора. Кроме того, он соперничает по скорости с C и C++ и применяется в довольно широком спектре приложений.

Разбиение проекта на модули с указанием потока данных:

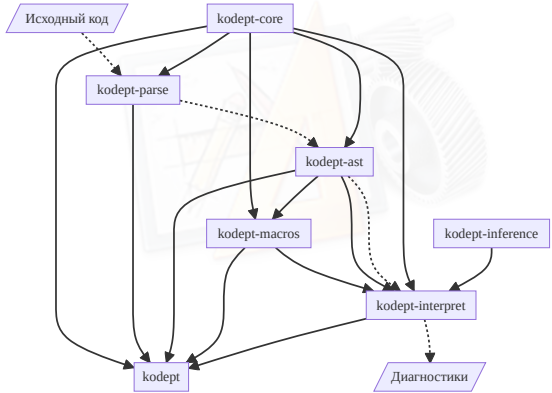

Рис. 2. Иерархия модулей в проекте

Алгоритм W является одной из реализаций системы типов Хиндли-Милнера.

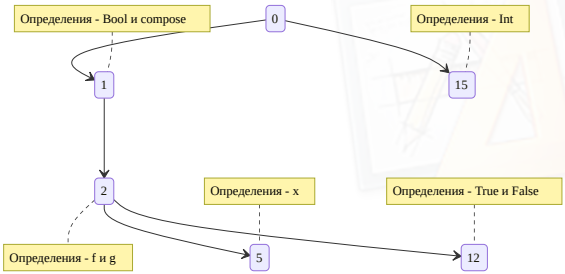

Рис. 3. Дерево областей видимости

Разбиение на области видимости позволяет убедиться, что внутри области не используются неизвестные переменные или функции.

Исходный код 1. Исходная программа на языке Kodept

```
module Testing {
    fun compose(f, g) => \x => f(
    g(x))
    enum struct Bool { True,
    False }
}
module Testing2 {
    struct Int
}
```

Компилятор Kodept является консольным приложением, поэтому для него был разработан интерфейс командной строки (CLI). С помощью него можно настроить вид выходных данных, поведение работы и др. Кроме того, можно получить внутреннюю модель в виде графа. Для этого следует использовать следующую команду (рис. 4):

```
./kodept graph examples/test.kd
```

Для демонстрации работы механизма вывода типов можно использовать следующее:

```
./kodept -d examples/test.kd
```

В таком случае тип функции `compose` будет таким:

$$[\texttt{compose: } \forall a, b, c => (' b - >' a) - > (' c - >' b) - >' c - >' a]$$

Проект можно развивать и дальше - улучшая как саму работу механизма, так и добавляя другие функции, например этап компиляции.

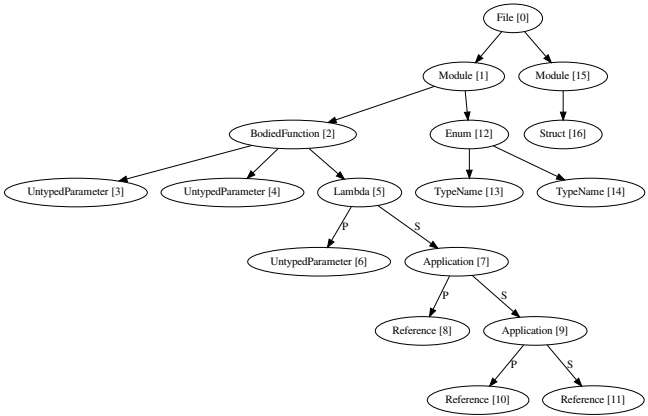

Рис. 4. Изображение структуры абстрактного синтаксического дерева

Заключение

В результате данной работы был реализован механизм вывода типов для языка программирования Kodept. Показано, что программа действительно может правильно определить тип выражения. Успешно решены все поставленные задачи, а именно:

1) сформирована модель абстрактного синтаксического дерева,
2) реализован семантический анализатор, включающий в себя анализатор областей видимости, преобразователь в термы системы типов Хиндли-Милнера и непосредственно механизм вывода типов на основе этой системы.

Спасибо за внимание!

Вопросы?

nikitinvl@student.bmstu.ru