Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 22.М04-мм

Транспиляция Python в EO

Андреев Илья Алексеевич

Отчёт по учебной практике

Научный руководитель:
проф. каф. СП, д.ф.-м.н., проф. А. Н. Терехов

Санкт-Петербург
2024

Оглавление

Введение

В настоящее время продолжаются попытки формализовать объектно-ориентированное програмирование. Термин ООП был предложен еще в 1966 году [3], но с тех пор так и не был формально определен. Не существует единообразия или согласия в отношении набора характеристик и механизмов, присущих объектно-ориентированному языку, поскольку «сама парадигма является слишком общей», как пришел к выводу О. Нирстраш в своем обзоре концептов парадигмы объектно-ориентированного программирования [4].

Одной из таких попыток формализации является φ-исчисление и реализующий его язык программирования EO [1] (также иногда называемый EOlang). Предлагаемое исчисление представляет объектную модель через данные и объекты, при этом операции с ними возможны через абстракцию, приложение и декорирование. Исчисление вводит формальный аппарат для манипуляций с объектами. Предлагаемый язык программирования EO полностью реализует все элементы исчисления и позволяет реализовать объектную модель на любой вычислительной платформе. Будучи языком объектно-ориентированного программирования, EO реализует четыре ключевых принципа ООП: абстракцию, наследование, полиморфизм и инкапсуляцию. По мнению авторов, такая модель позволит снизить сложность создаваемых программ, а также позволит эффективно проводить их статический анализ.

Статический анализ для языка программирования EO реализуется в проекте Polystat [5], а также в работе [2]. Утверждается, что предложенный инструментарий сможет эффективнее обнаруживать ошибки специфичные для программ, написанных в объектно-оринтированной парадигме. Существование трансляторов из широкоиспользуемых языков программирования в EO позволило бы использовать такой анализ на большем количестве программных продуктов.

Темой данной работы была выбрана трансляция одного из самых популярных [7] языков программирования Python [6]. В работе про-

водится анализ вариантов трансформации программных конструкций языка программирования Python в объекты языка программирования EO. Для анализа полученных результатов реализован транспилятор, обеспечивающий необходимые преобразования.

В работе будет часто использоваться термины *транспиляция* и *транспилятор*. Транспиляция — это такой вид трансляции, при котором исходный и целевой языки работают примерно на одинаковых уровнях абстракции. Целью транспиляции могут быть, например, перевод программ на новые языки программирования, или возможность использовать инструментарий, недоступный для исходного языка программирования.

1. Постановка задачи

Целью работы является реализация транспилятора языка программирования Python в язык программирования EO. Для её выполнения были поставлены следующие задачи:

1. ограничить транспилируемое подмножество языка Python;
2. описать способы проекции конструкций языка Python на конструкции языка EO;
3. реализовать соответствующие трансформации конструкций языка Python;
4. протестировать реализованный транспилятор на примере существующих проектов.

2. Обзор

2.1. Обзор языка программирования EO

Язык программирования EO был создан с целью устранения проблемы сложности кода объектно-ориентированной парадигмы, предлагая формальное исчисление объектов. EO — это объектно-ориентированный язык програмирования с ленивыми вычислениями. Будучи объектно-ориентированным, язык EO предоставляет 4 главные черты своей парадигмы: абстракцию, наследование, полиморфизм и инкапсуляцию.

Основной сущностью этого языка является объект — набор атрибутов, которые связаны с другими объектами. При этом в языке не существует понятия типа. В случае попытке обращения к атрибуту, которым объект не обладает, произойдет прерывание программы во время исполнения.

Результат вычисления объекта в языке EO происходит в формате операции “датаизации” — процесса, при котором объект превращается в данные, которые он представляет. При этом каждый объект либо знает, какие данные он представляет, либо знает, где он может получить такую информацию. Исполнение всей программы — это датаизация объекта верхнего уровня композиции. Ориентированность языка на ленивые вычисления обусловлена тем, что датаизация происходит не при инициализации объектов, а при обращении к ним.

Следующие четыре принципа лежат в устройстве языка EO:

- Объект — это коллекция атрибутов, связанных с объектом по их именам. Объект называется атомарным, если его реализация представлена средой исполения. Примерами таких объектов могут быть целочисленный объект `int` или объект стандартного вывода `stdout`.

- Объект называется абстрактным, если хотя бы один его атрибут является свободным — то есть не связанным ни с одним объектом. Применением объекта называется копирование абстрактного

объекта со связыванием одного или нескольких его свободных атрибутов с объектами. Операция применения может приводить к появлению как абстрактного, так и закрытого (то есть не имеющего свободных атрибутов) объекта.

- Объект может декорировать другой объект, связывая с ним свой атрибут φ. Декорирующий объект получает все атрибуты декорируемого объекта и при этом может нести свои атрибуты.

- Атрибут Δ объекта служит для предоставления данных в процессе датаизации объекта.

EO не несет в себе ряд конструкций, которые присутствуют в языке Python. Например, нет поддержки механизма исключений, операторов переходов, изменяемых объектов и так далее. Некоторые конструкции будут транспилироваться при помощи эквивалентных преобразований дерева программы на языке Python. Для реализации транспиляции других конструкций необходимо реализовать новые объекты, которые будут инкапсулировать в себе необходимую логику.

2.2. Обзор связанных работ

Так как язык EO был недавно, еще не существует трансляторов из Python. Был разработан C2EO[8] — транспилятор из языка программирования C в язык EO, но так как концепции языка C сильно отличаются от языка Python, только некоторые идеи могут быть применены в данной работе. Например, в транспиляции Python в EO, как и в C2EO, будут использоваться объекты, оборачивающих атомарные объекты языка EO для реализации простых типов исходного языка.

3. Метод

3.1. Ограничение траспилируемого подмножества

В ходе написания статьи было принято решение не поддерживать некоторые конструкции языка Python. Это связано с тем, что эти конструкции не имеют аналогов в EO или не могут быть реализованы в статическом контексте.

Первой группой конструкций, которые не будут поддержаны, являются генераторы и корутины. Генераторы в Python позволяют создавать итераторы без явного определения класса, что упрощает написание кода. Однако в EO нет поддержки yield, что делает невозможным использование этой конструкции. Корутины также связаны с асинхронным выполнением кода и многопоточностью, что не реализовано в EO. Поэтому async, threads, futures и await также не будут поддерживаться.

Второй группой являются динамические возможности языка Python, такие как динамическое создание, изменение и удаление переменных, создание классов с метаклассами и динамические возможности импорта. Использование этих функций невозможно в контексте EO, поскольку EO является компилируемым языком, а не интерпретируемым, и состав атрибутов объекта в EO должен быть определен в момент сборки программы. Также это означает, что функции Python, которые могут интерпретировать строку как программу и исполнить ее, не будут поддержаны.

Наконец, не будет поддержана рефлексия. Как будет описано далее, некоторые конструкции языка Python не были реализованы строго по спецификации языка, а был выбран более простой подход. Поскольку главное использование данной работы — статический анализ, не зависящий от языка оригинала программы, такие инструменты языка как наследование необходимо разрабатывать в парадигмах языка EO.

Таким образом, транспилятор из Python в EO будет поддерживать большинство конструкций языка Python, но некоторые из них не смогут быть реализованы в EO. Это следует учитывать при выборе тестового

набора, на котором будет проводиться проверка реализованного транспилятора.

3.2. Метод трансляции

3.2.1. Структура программы

Поскольку в транспиляции, описанной в данной статье, не будет поддерживаться интерпретируемость программы на языке Python, как и интерактивный режим, будем рассматривать программы как набор из одного или нескольких модулей, связанных импортами. В свою очередь, модуль состоит из списка операторов языка (statement).

Программа на языке EO тоже состоит из модулей, связанных собственной реализацией импортов, но эти модули состоят из списка объектов. Это означает, что транслируемый модуль на Python необходимо обернуть в объект EO, результатом датаизации которого будет результат исполнения изначальной программы.

Импорт символов в программе Python доступен следующими способами:

- Операторы `import ...` и `import ... as ...` могут быть транслированы напрямую в мета-операторы `alias` языка EO, который поддерживает возможность задания нового имени импортируемому объекту.

- Операторы `from ... import ...` и `from ... import *` не имеют прямых аналогов в EO, и будут заменены на импорт всего модуля, а обращение к соответствующему символу должно быть уточнено с помощью dot-нотации.

3.2.2. Базовые типы и операторы

Существующим в языке Python базовым типам, например, целочисленным, строковым и так далее, в EO имеются аналоги в виде соответствующих абстрактных объектов (int, string и др.). Поскольку Python

```
package org.example
+alias org.eolang.io.stdout

[args...] > app
  stdout > @
    ''Hello, world''
```

Рис. 1: Пример простейшей программы на языке EO, демонстрирующий структуру программы и импорт объекта stdout. Датаизация данной программы приведет к выводу "Hello world!" в стандартный вывод.

```
[value] > pystring

  [x] > with-value
    pystring x > @

  [] > length
    pyint (value.length) > @

  [] > as-string
    pystring value > @
```

Рис. 2: Пример некоторых методов реализации объекта pystring. `value` является свободным атрибутом объекта, который будет связан со значением литерала во время инстанцирования.

обладает значительно большим количеством методов для таких объектов, а на операторы наложено меньше ограничений (например, оператор сложения в Python может складывать целое число и число с плавающей точкой), было решено реализовать объекты для каждого типа Python в EO.

Например, строковый тип будет выражен объектом `pystring`, а каждый строковый литерал в изначальной программе заменен на копию этого объекта с нужным значением. Операторы, как встроенные, так и объявленные пользователем, будут реализованы атрибутами у объекта в EO. Их вызов будет преобразован в обращение к атрибуту через dot-нотацию.

3.2.3. Порядок вычисления выражений

Python строго специфицирует порядок вычисления, в то время как датаизация объектов в EO ленивая. Это означает, что любое выражение, состоящее более чем из одной операции (вызова или использования оператора) необходимо транслировать в последовательность более простых выражений.

Такой подход может быть реализован как преобразование кода в эквивалентный код на языке Python. При финальном преобразовании кода Python в код EO будет необходимо только обращение к специальному атрибуту `force` каждого подвыражения, который приведет к датаизации объектов в нужном порядке.

3.2.4. Функции

Функция языка Python может быть транслирована в объект, свободными атрибутами которого будут параметры изначальной функции. Вызов такой функции эквивалентен копированию объекта функции с подстановкой аргументов в свободные атрибуты, а результат вызова — это результат датаизации объекта.

Тело транслированной функции можно представить с помощью объекта `seq` языка EO. При датаизации этот объект датаизирует по порядку все атрибуты по порядку и вернет результат последнего. Любые преждевременные условные return внутри тела функции должны быть конвертированы в условное ветвление if-else.

Важно отметить, что исключения на данный момент не поддерживаются, и для их реализации потребуется более сложная обертка, чем встроенный `seq`.

3.2.5. Классы

Классы языка Python транслируются в объекты, связанные атрибуты которого являются полями класса. При этом, из-за свойств языка EO, список полей класса должен быть известен на этапе компиляции программы, и их список не может меняться во время исполнения про-

```
def func(a):
  print(a)
  return a + 5

...

[a] > func
  seq > @
    stdout (sprintf '"%i'' a)
    a.add (pyint 5)
```

Рис. 3: Пример трансляции простой функции из двух операторов в язык EO до введения поддержки исключений. Для представления тела функции используется встроенный объект `seq`.

граммы. Конструкторы оригинального класса транслируются в отдельные объекты со свободными аттрибутами-параметрами, датаизация которых возвращает "экземпляр" изначального класса. Методы классов транслируются в отдельные объекты, принимающие первым свободным аттрибутом "экземпляр" класса, чей метод вызывается в первоначальной программе.

Заключение

На данный момент изучена предметная область и стек технологий, сделан обзор, сформулирована цель и поставлены задачи.

Реализованы объекты для базовых типов языка Python (целые числа, строки), механизм сохранения порядка исчислений и трансляция функций, на данный момент без поддержки исключений.

Также реализован механизм импорта используемых в программе модулей.

Список литературы

[1] Bugayenko Yegor. EOLANG and phi-calculus // arXiv preprint arXiv:2111.13384. — 2021.

[2] Kudasov Nikolai, Olokin Mikhail, Potyomkin Oleksii et al. Detecting unanticipated mutual recursion using Elegant Objects representation of object-oriented programs. — 2022. — URL: https://arxiv.org/abs/2209.01803.

[3] FLEX—A Flexible Extendable Language.

[4] Nierstrasz Oscar. A Survey of Object-Oriented Concepts. — 1989.

[5] Polyglot Static Analyzer for Object-Oriented Programming Languages. — URL: https://github.com/polystat (дата обращения: 2023-01-08).

[6] Python Programming Languages. — URL: https://www.python.org (дата обращения: 2023-01-08).

[7] Stack Overflow Developer Survey 2022s. — URL: https://survey.stackoverflow.co/2022/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2022 (дата обращения: 2023-01-08).

[8] Трансформация модели памяти языка программирования C в объектно-ориентированное представление на языке EO / Александр Иванович Легалов, Егор Георгиевич Бугаенко, Николай Константинович Чуйкин et al. // Моделирование и анализ информационных систем. — 2022. — Vol. 29, no. 3. — P. 246–264.