Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”

Факультет Программной инженерии и компьютерной техники

Направление подготовки (специальность) Системное и прикладное программное обеспечение

ЭССЕ на тему:
Курс “Статический анализ программ”

Обучающийся Орловский М.Ю P4216
(Фамилия И.О.) **(номер группы)**

Санкт-Петербург
2024 г.

Курс представлял собой самостоятельное изучение теории статического анализа по одноименной работе Андерса Мёллера и Майкла Шварцбаха (ссылка: https://cs.au.dk/~amoeller/spa/spa.pdf). Основная цель статического анализа – выявление свойств программ без фактического их запуска, на основе исходного кода. Соответственно, статический анализатор это программа которая выявляет какие-либо факты о коде (например, ограничения на значения или типы).

Хоть книга и относительно небольшая, она представляет собой в сущности компиляцию статей, а поэтому имеет довольно сухой теоретический язык. В связи с этим она покрывает большое количество различной теории, в том числе математической. Поэтому, логичным представляется написать о том, что я узнал из книги что в итоге было использовано в моем проекте по выработке метода статического анализа.

Первое это общая теория анализа, связанная прежде всего с формальными определениями – в книге приводятся доходчивые определения корректности и полноты анализа, оценки возможности завершения алгоритма (т. к. теорема Райса определяет что никакой алгоритм не является тотальным по факту построения). Выводятся методы оценки полноты алгоритма, например консервативность и точность.

В контексте описания книги вводится язык, называемый TIP (tiny imperative programming language), на примере которого показываются различные методы анализа. Большая часть методов используется для анализа графа переходов (control flow), поэтому говорить об этих алгоритмах я не буду – они слабо применимы в рамках моего исследования.

Одним из очень полезных алгоритмов статического анализа является представленный в 3 главе алгоритм вывода типов для системы типов с указателями. Алгоритм представляет собой классический алгоритм унификации Хиндли-Милнера и за счет прозрачного описания он довольно просто реализуем. Именно это я и сделал в рамках работы по небольшому языку программирования – мне удалось реализовать алгоритм вывода типов для монотипов и функций. Пример работы алгоритма (исходный код программы и типизированное AST) представлен в приложении А. Ссылка на проект: https://github.com/UberDever/ctraspiler

Также, следует сказать, что при работе с различными системами типов помогла описанная в 4 главе теория решеток (lattice theory) так как она напрямую связана с типами-уточнениями и отношениями подтипизации.

За счет изучения книги получилось интегрировать честную теоретическую базу в мой метод анализа, что позволило повысить его полноту и обеспечить корректность. Также, благодаря практической реализации которую я совершил по книге, имплементация анализатора по моему методу будет проще.

ПРИЛОЖЕНИЕ А

```
fn main() {
      var a = true
      var b = a
      const c = a && b
      var d = 5.2
      return 0
}
```

```
(Source:0
    (FunctionDecl:35
        (main:1 `(FN int )`)
        (Signature:3
            (ID[]:2))
        (Block:34
            (VarDecl:9
                (ID[]:5
                    (a:4 `bool`))
                (Expr[]:8
                    (Expr:7 `bool`
                        (true:6 `bool`))))
            (VarDecl:15
                (ID[]:11
                    (b:10 `bool`))
                (Expr[]:14
                    (Expr:13 `bool`
                        (a:12))))
            (ConstDecl:23
                (ID[]:17
                    (c:16 `bool`))
```

```
    (Expr[]:22
        (Expr:21 `bool`
            (&&:20 `bool`
                (a:18)
                (b:19)))))
(VarDecl:29
    (ID[]:25
        (d:24 `float`))
    (Expr[]:28
        (Expr:27 `float`
            (5.2:26 `float`))))
(Return:33 `int`
    (Expr[]:32
        (Expr:31 `int`
            (0:30 `int`)))))))
```