Контекстно чувствительный анализ алиасов

Авторы: С. А. Варивода, А. Д. Милакин, А. А. Солдатенков
Научный руководитель: С. В. Григорьев
(к. ф.-м. н., доцент кафедры информатики СПбГУ)

СПбГЭТУ «ЛЭТИ» им. В.И. Ульянова (Ленина)
Кафедра вычислительной техники

25 апреля 2018 г.

Статический анализ кода

- Развитие языков программирования усложняет анализ создаваемого кода

Обзор

- В 1995 году Томас Репс заметил, что многие задачи межпроцедурного статического анализа выразимы в терминах контекстно-свободной достижимости

- Анализ алиасов выразим в терминах КС-достижимости

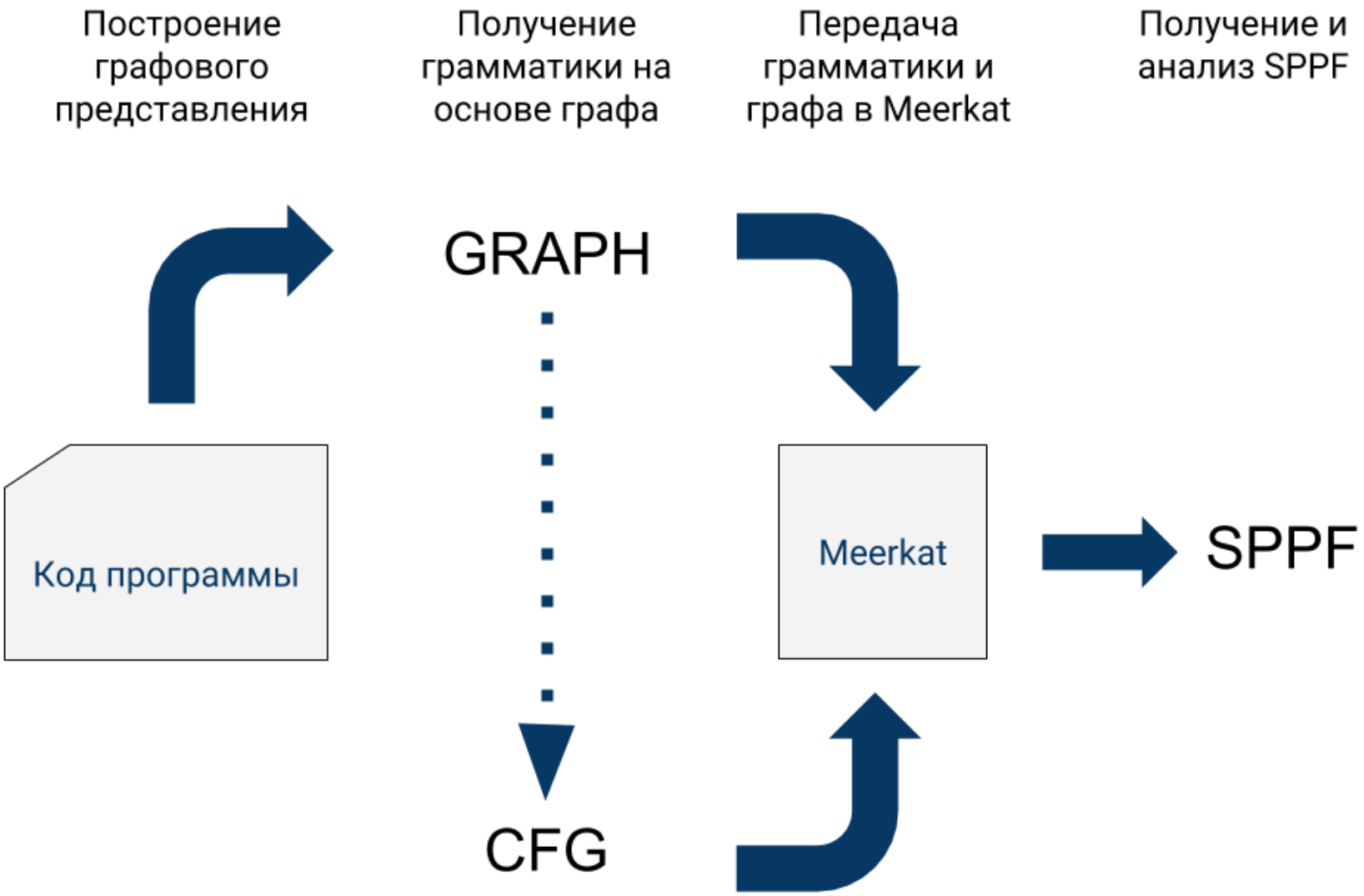
Цель

- Цель: разработка решения для контекстно чувствительного анализа алиасов для языка программирования Java на основе КС-достижимости

Задачи

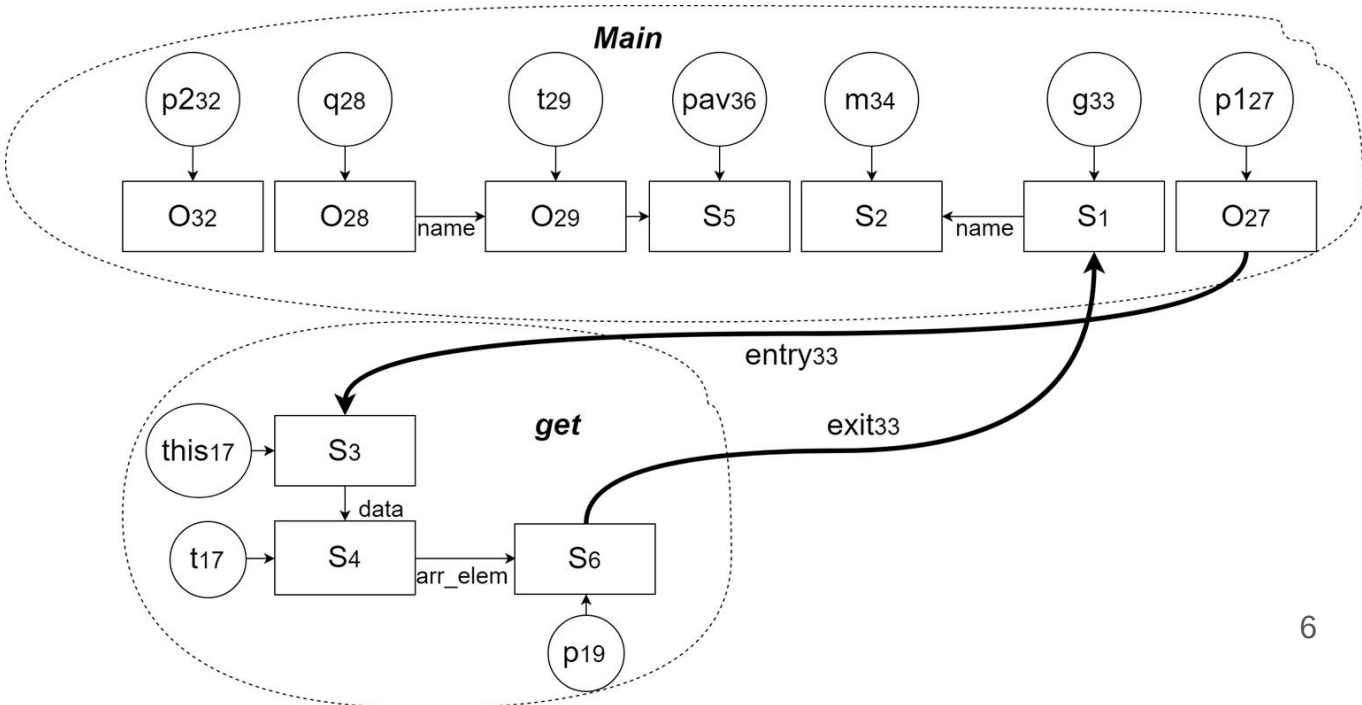
- Сформулировать алгоритм создания графа для кода Java
- Реализация алгоритма создания графа
- Построение SPPF с помощью библиотеки парсер комбинаторов Meerkat
- Анализ SPPF с выводом результатов

Решение


Построение графового представления
Получение грамматики на основе графа
Передача грамматики и графа в Meerkat
Получение и анализ SPPF
Код программы
GRAPH
CFG
Meerkat
SPPF

Графовое представление программы

```
17.  Dinosaur get(int idx) {
18.      Dinosaur[] t = this.data;
19.      Dinosaur p = t[idx];
20.      return p; }
...
25.  class Main {
26.  static void main(String[] args) {
27. JurassicPark p1 = new JurassicPark();
28. Dinosaur q = new Dinosaur();
29.      String t = "Ivan";
30.      q.name = t;
31.      p1.add(q);
32.      JurassicPark p2 = new JurassicPark();
33.      Dinosaur g = p1.get(0);
34.      String m = g.name;
35.      System.out.println(m);
36.      String pav = t;    }
```



Построение грамматики

Нетерминалы выражающие входы и выходы из функции

$<(4> \rightarrow exit33 \mid entry33^{-}$

$<)4> \rightarrow entry33 \mid exit33^{-}$

Нетерминал описывающий встречающиеся функции

$<C> \rightarrow <(i> <C> <)i> \mid <C> <C> \mid <road> \mid <road^{-}> \mid Epsilon$

Главный стартовый нетерминал описывающий местоположение алиасов

$<memAlias> \rightarrow <road^{-}> <memAlias> <road> \mid <memAlias> <memAlias> \mid <C>$

где *Epsilon* — пустой переход и $road^{-}$ — обратные пути для *road*

Технологии

В процессе разработки были использованы:

- Среда разработки IntelliJ IDEA
- Библиотека парсер комбинаторов Meerkat

Результаты

- Был разработан алгоритм построения графа программы по коду Java
- Реализован алгоритм построения графа
- Построено SPPF с помощью библиотеки парсер комбинаторов Meerkat