Доработка автоматического создания графа зависимостей для статического анализатора LuNA-программ

Студент ФИТ НГУ Царев
Василий Дмитриевич,
гр. 20202

Новосибирск, 2023

Задачи

- завершить построение вершин графа зависимостей по данным (DDG — data dependency graph) для LuNA-программ;
- реализовать связывание вершин графа зависимостей по данным;
- реализовать поиск неиспользуемых фрагментов данных.

Про статический анализ

- Абстрактное синтаксическое дерево — орграф, вершинами которого являются операции/данные, а рёбрами — участие операндов в операциях.
- Трёхадресный код — последовательность трёхадресных команд, строится из АСТ.
- Граф потока управления — орграф, вершинами которого являются базовые блоки (последовательности трёхадресных команд без прыжков), а рёбрами — прыжки (goto, if, …); строится из трёхадресного кода.
- Граф зависимостей по данным — орграф, вершинами которого являются операции, а рёбрами — зависимость одних операций от результатов выполнения других.

Исходный код анализируемой программы

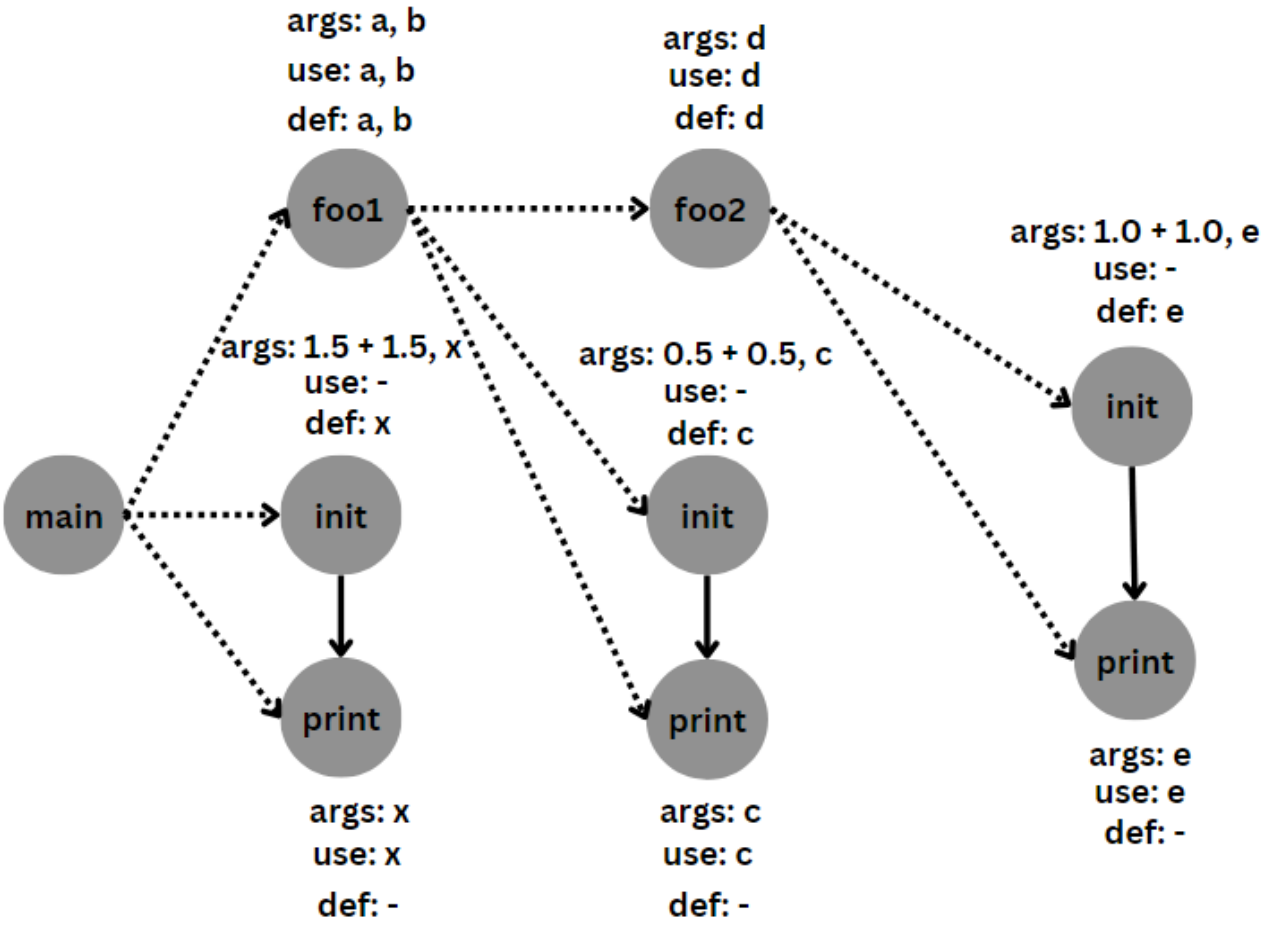
```
1   import printfl(real) as printFloat;
2   import c_init_float(real, name) as initFloat;
3
4   sub main(){
5           df a, b, x;
6
7           foo1(a, b);
8           initFloat(1.5 + 1.5, x);
9           printFloat(x);
10  }
11
12  sub foo1(name c, name d){
13          initFloat(0.5 + 0.5, c);
14          printFloat(c);
15          foo2(d);
16  }
17
18  sub foo2(name e){
19          initFloat(1.0 + 1.0, e);
20          printFloat(e);
21  }
```

Структура графа

Вершины — операторы.

Сплошные стрелки — зависимости по данным.

Пунктирные стрелки — вершины внутри блока оператора.



Алгоритм построения графа

1 шаг: анализ атомарных ФК — определение позиций аргументов, которые используются и определяются.

2 шаг: создание вершин — рекурсивный обход операторов внутри main с порождением объектов Vertice.

3 шаг: связывание вершин — рекурсивный обход созданных вершин и проведение рёбер от вершин, инициализирующих ФД до вершин, использующих ФД.

Пример вывода программы

```
Vertice number: 6
Vertice address: 0xf61c9c
Vertice type: 6 (structured CF);  name: foo1
Vertice depth: 1
Use DFs: a b
Def DFs: a b
Vertices inside: 0xf61dec 0xf61ed4 0xf6256c
Vertices before ("in"):
Vertices after ("out"):
```

```
Vertice number: 8
Vertice address: 0xf626dc
Vertice type: 5 (atomic CF); name: printFloat
Vertice depth: 1
Use DFs: x
Def DFs:
Vertices inside:
Vertices before ("in"):
Vertices after ("out"):
```

```
Vertice number: 7
Vertice address: 0xf62624
Vertice type: 5 (atomic CF); name: initFloat
Vertice depth: 1
Use DFs:
Def DFs: x
Vertices inside:
Vertices before ("in"):
Vertices after ("out"):
```

```
Vertice number: 5
Vertice address: 0xf6256c
Vertice type: 6 (structured CF);  name: foo2
Vertice depth: 2
Use DFs: d
Def DFs: d
Vertices inside: 0xf623fc 0xf624b4
Vertices before ("in"):
Vertices after ("out"):
```

Заключение

- Удалось автоматизировать построение вершин для программы, состоящей исключительно из операторов применения фрагмента кода (без циклов и ветвлений) и без индексированных ФД;
- Реализовано связывание вершин, однако лишь частично из-за ошибки в программе;
- Не удалось реализовать поиск неиспользуемых фрагментов данных.

Спасибо за внимание!