Фреймворк для разработки межпроцедурных статических анализаторов на основе KC-достижимости

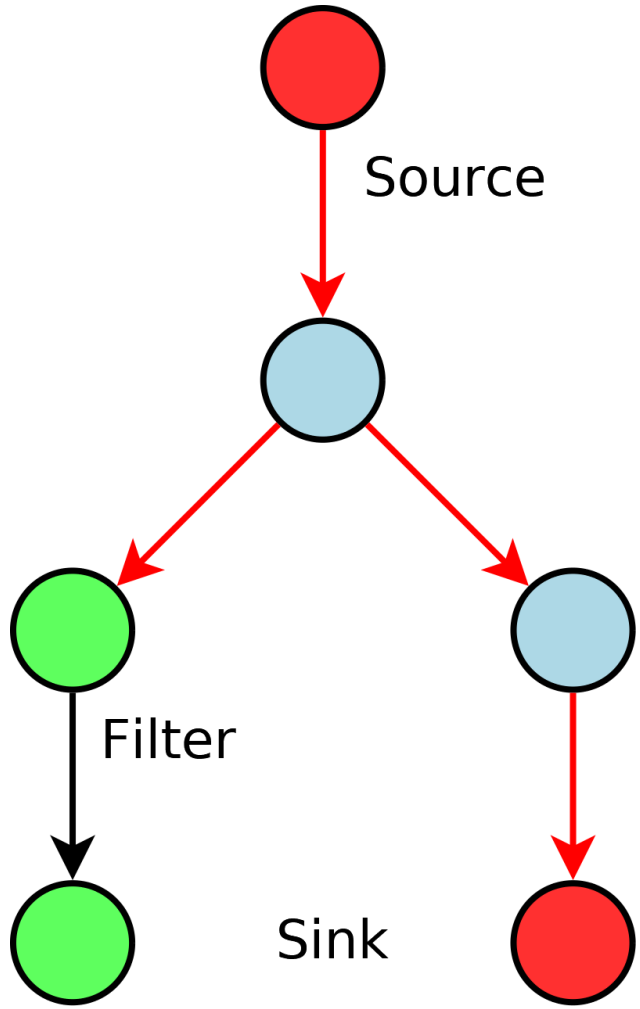
Илья Ножкин, Семён Григорьев

JetBrains Research, Programming Languages and Tools Lab
Saint Petersburg University

14 ноября 2019

Пример

- Данные поступают из недоверенного источника
- Данные обрабатываются и проверяются на корректность
- Данные используются для дальнейших вычислений



Внутрипроцедурный и межпроцедурный анализ

File 1:

```
1  class A {
2    [Tainted]
3    int Source;
4  }
5
6  class B {
7    [Filter]
8    static int Filter(int d);
9  }
10
11 class C {
12   [Sink]
13   void Sink(int d);
14 }
```

File 2:

```
1  class D {
2    void Process(A a) {
3      int d = Read(a);
4      int f = B.Filter(d);
5      Consume(d);
6      AnotherConsume(f);
7    }
8
9    int Read(A a) {
10     return a.Source;
11   }
12
13   void Consume(int d) {
14     C c = new C();
15     c.Sink(d);
16   }
17
18   void AnotherConsume(int d) { ... }
19 }
```

Внутрипроцедурный и межпроцедурный анализ

File 1:

```
1  class A {
2    [Tainted]
3    int Source;
4  }
5
6  class B {
7    [Filter]
8    static int Filter(int d);
9  }
10
11 class C {
12   [Sink]
13   void Sink(int d);
14 }
```

File 2:

```
1  class D {
2    void Process(A a) {
3      int d = Read(a);
4      int f = B.Filter(d);
5      Consume(d);
6      AnotherConsume(f);
7    }
8
9    int Read(A a) {
10     return a.Source;
11   }
12
13   void Consume(int d) {
14     C c = new C();
15     c.Sink(d);
16   }
17
18   void AnotherConsume(int d) { ... }
19 }
```

Внутрипроцедурный и межпроцедурный анализ

File 1:

```
1  class A {
2    [Tainted]
3    int Source;
4  }
5
6  class B {
7    [Filter]
8    static int Filter(int d);
9  }
10
11 class C {
12   [Sink]
13   void Sink(int d);
14 }
```

File 2:

```
1  class D {
2    void Process(A a) {
3      int d = Read(a);
4      int f = B.Filter(d);
5      Consume(d);
6      AnotherConsume(f);
7    }
8
9    int Read(A a) {
10     return a.Source;
11   }
12
13   void Consume(int d) {
14     C c = new C();
15     c.Sink(d);
16   }
17
18   void AnotherConsume(int d) { ... }
19 }
```

Сложность межпроцедурного анализа

- Реализация анализатора нетривиальна, поэтому написание нового решения для каждой задачи нерационально
- Необходимо перед началом анализа собрать данные обо всей программе
- Обработка такого количества данных может быть долгой

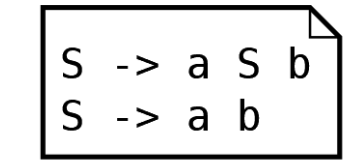
Задача

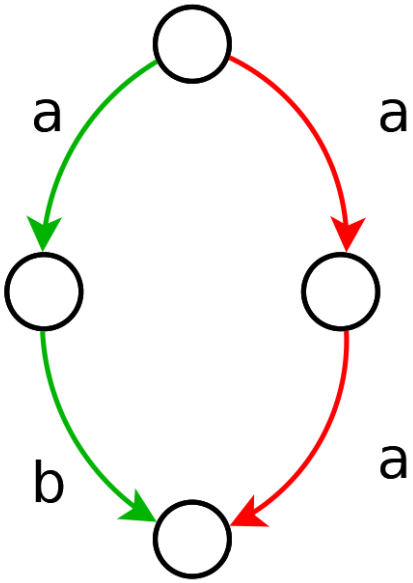
Разработать инструмент, который:

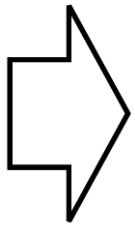
- Решает проблемы, связанные со сложностью межпроцедурного анализа
- Предоставляет информацию о найденных ошибках, позволяющую как можно точнее установить причины
- Может быть использован совместно с любой средой разработки или инструментом статического анализа

КС-достижимость (CFL-reachability)

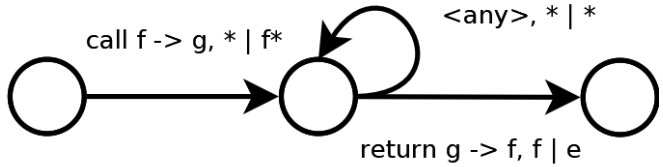
КС-достижимость в общем случае

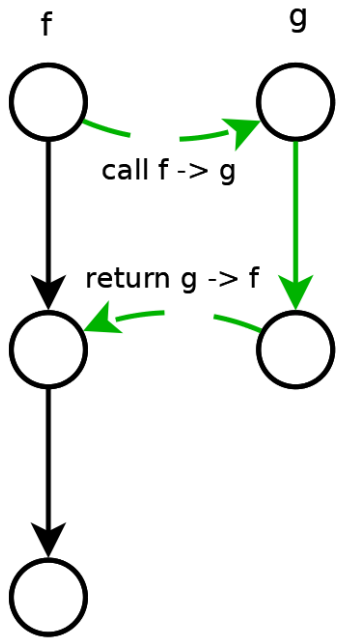






КС-достижимость в нашем случае

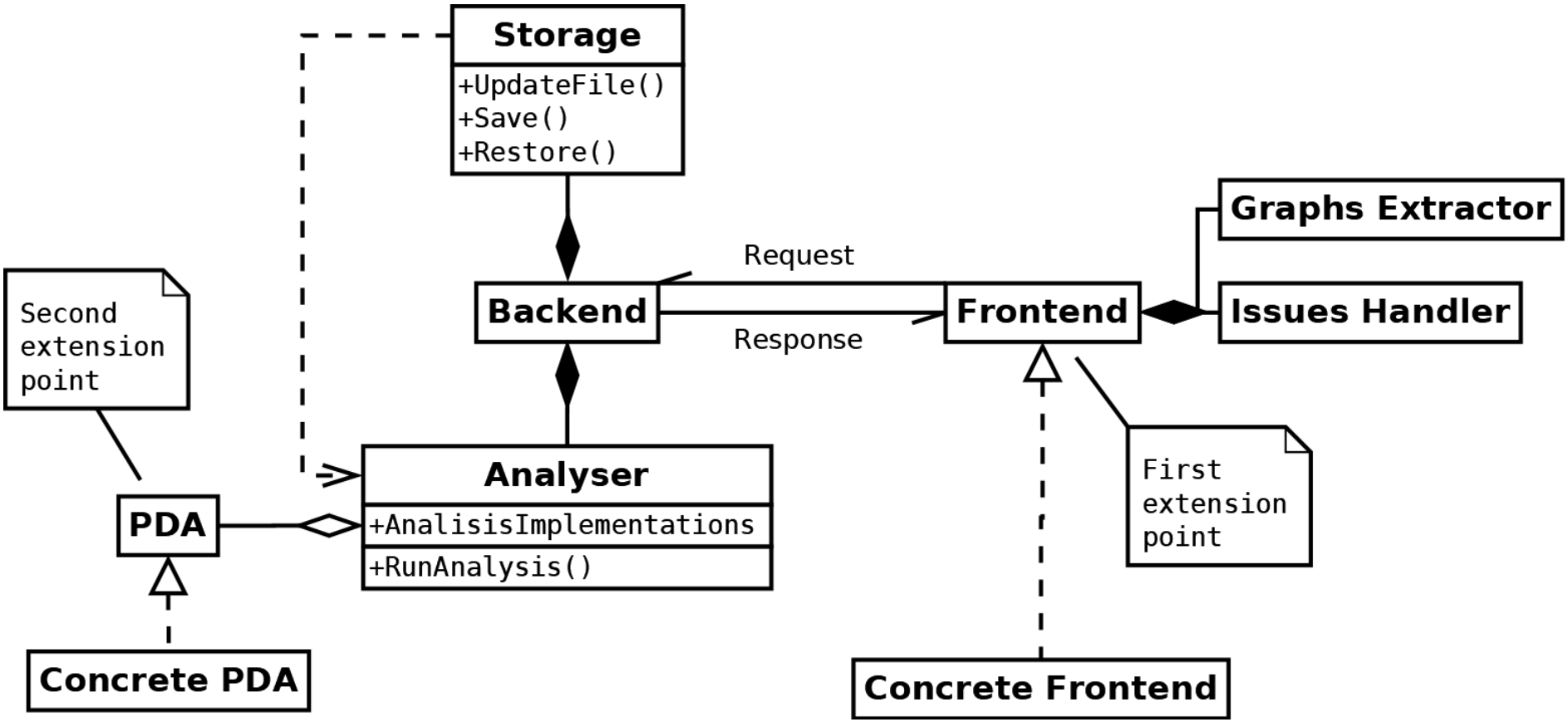




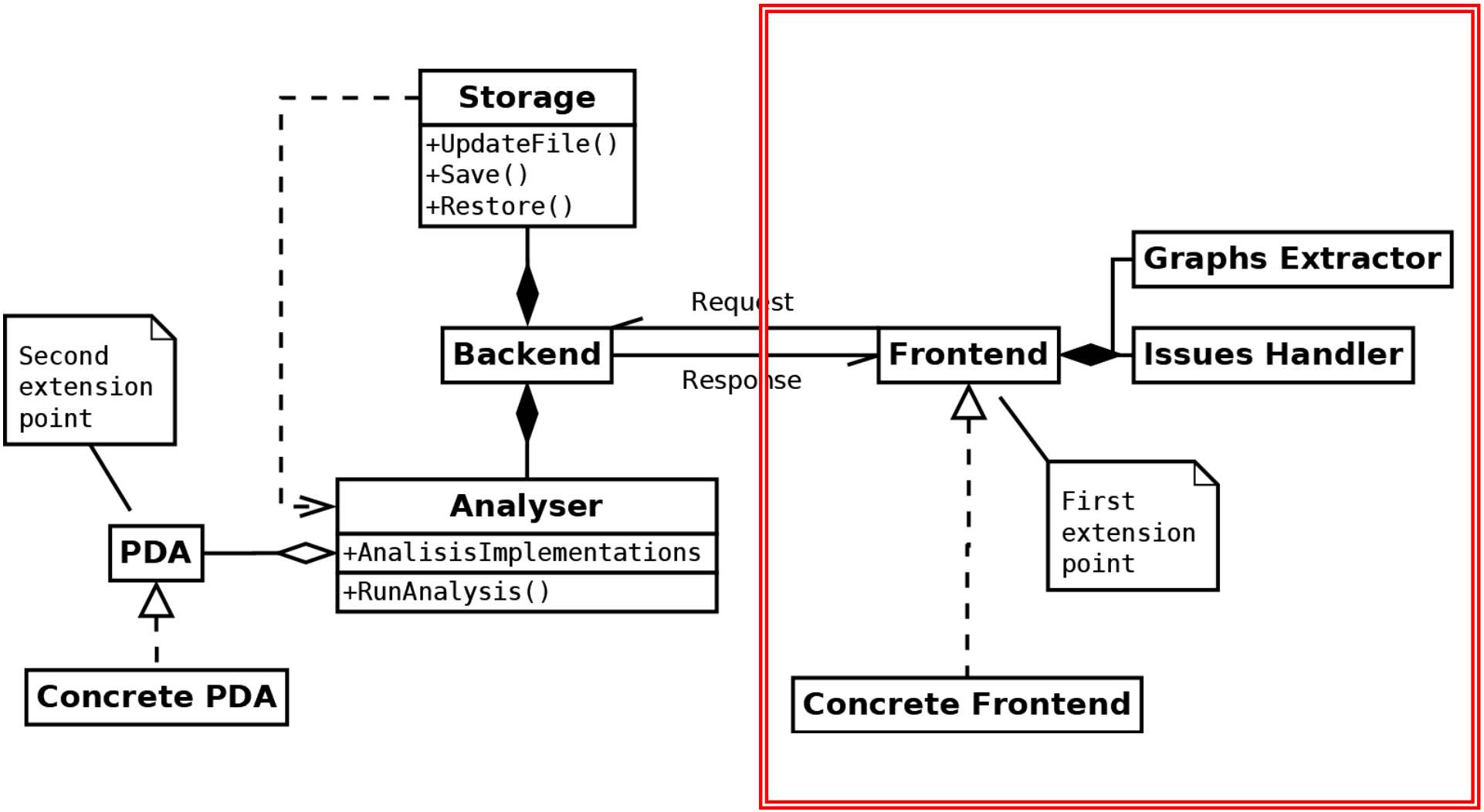
Аналоги

- Cauliflower: a Solver Generator for Context-Free Language Reachability
  Nicholas Hollingum and Bernhard Scholz. 2017
- Graspan: A Single-machine Disk-based Graph System for Interprocedural Static Analyses of Large-scale Systems Code
  Kai Wang, Aftab Hussain, Zhiqiang Zuo, Guoqing Xu, and Ardalan Amiri Sani. 2017

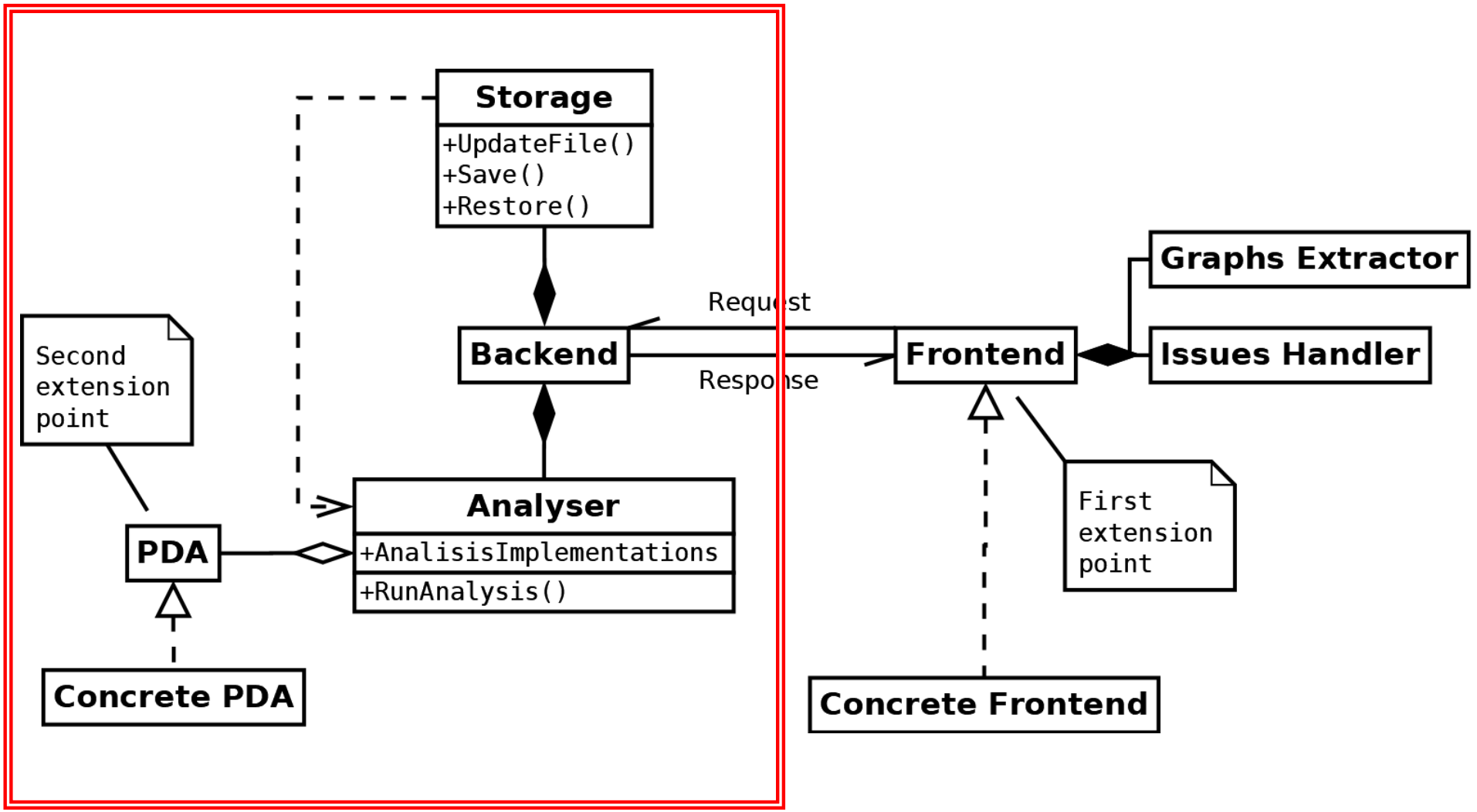
Разработанное решение



Разработанное решение



Разработанное решение



Flow sensitivity

```
17      class Program
18      {
19          [Tainted] private int A;
20          [Filter] private int Filter(int a) { return a; }
21          [Sink] private void Sink(int a) {}
22
23          private int PostSource() {
24              var b = A;
25              return b;
26          }
27
28          private int Brackets(int a) {
29              var b = a;
30              return b;
31          }
32
33          private static void Main(string[] args) {
34              var a = new Program();
35              var c = a.PostSource();
36              var d = a.Filter(c);
37              var e = a.Brackets(c);
38              var f = a.Brackets(d);
39              a.Sink(e);
40              a.Sink(f);
41          }
42      }
```

```
Tainted sink
source - Program.cs:24
assign - Program.cs:25
return <-
assign - Program.cs:35
pass -> Program.cs:37 (System.Int32)Brackets(System.Int32)
assign - Program.cs:29
assign - Program.cs:30
return <-
assign - Program.cs:37
sink -> Program.cs:39 (System.Void)Sink(System.Int32)
```

Flow sensitivity

```
17      class Program
18      {
19          [Tainted] private int A;
20          [Filter] private int Filter(int a) { return a; }
21          [Sink] private void Sink(int a) {}
22
23          private int PostSource() {
24              var b = A;
25              return b;
26          }
27
28          private int Brackets(int a) {
29              var b = a;
30              return b;
31          }
32
33          private static void Main(string[] args) {
34              var a = new Program();
35              var c = a.PostSource();
36              var d = a.Filter(c);
37              var e = a.Brackets(c);
38              var f = a.Brackets(d);
39              a.Sink(e);
40              a.Sink(f);
41          }
42      }
```

```
Tainted sink
source - Program.cs:24
assign - Program.cs:25
return <-
assign - Program.cs:35
pass -> Program.cs:37 (System.Int32)Brackets(System.Int32)
assign - Program.cs:29
assign - Program.cs:30
return <-
assign - Program.cs:37
sink -> Program.cs:39 (System.Void)Sink(System.Int32)
```

Flow sensitivity

```
17      class Program
18      {
19          [Tainted] private int A;
20          [Filter] private int Filter(int a) { return a; }
21          [Sink] private void Sink(int a) {}
22
23          private int PostSource() {
24              var b = A;
25              return b;
26          }
27
28          private int Brackets(int a) {
29              var b = a;
30              return b;
31          }
32
33          private static void Main(string[] args) {
34              var a = new Program();
35              var c = a.PostSource();
36              var d = a.Filter(c);
37              var e = a.Brackets(c);
38              var f = a.Brackets(d);
39              a.Sink(e);
40              a.Sink(f);
41          }
42      }
```

```
Tainted sink
source - Program.cs:24
assign - Program.cs:25
return <-
assign - Program.cs:35
pass -> Program.cs:37 (System.Int32)Brackets(System.Int32)
assign - Program.cs:29
assign - Program.cs:30
return <-
assign - Program.cs:37
sink -> Program.cs:39 (System.Void)Sink(System.Int32)
```

Flow sensitivity

```
17      class Program
18      {
19          [Tainted] private int A;
20          [Filter] private int Filter(int a) { return a; }
21          [Sink] private void Sink(int a) {}
22
23          private int PostSource() {
24              var b = A;
25              return b;
26          }
27
28          private int Brackets(int a) {
29              var b = a;
30              return b;
31          }
32
33          private static void Main(string[] args) {
34              var a = new Program();
35              var c = a.PostSource();
36              var d = a.Filter(c);
37              var e = a.Brackets(c);
38              var f = a.Brackets(d);
39              a.Sink(e);
40              a.Sink(f);
41          }
42      }
```

```
Tainted sink
source - Program.cs:24
assign - Program.cs:25
return <-
assign - Program.cs:35
pass -> Program.cs:37 (System.Int32)Brackets(System.Int32)
assign - Program.cs:29
assign - Program.cs:30
return <-
assign - Program.cs:37
sink -> Program.cs:39 (System.Void)Sink(System.Int32)
```

Flow sensitivity

```
17      class Program
18      {
19          [Tainted] private int A;
20          [Filter] private int Filter(int a) { return a; }
21          [Sink] private void Sink(int a) {}
22
23          private int PostSource() {
24              var b = A;
25              return b;
26          }
27
28          private int Brackets(int a) {
29              var b = a;
30              return b;
31          }
32
33          private static void Main(string[] args)
34              var a = new Program();
35              var c = a.PostSource();
36              var d = a.Filter(c);
37              var e = a.Brackets(c);
38              var f = a.Brackets(d);
39              a.Sink(e);
40              a.Sink(f);
41          }
42      }
```

Tainted sink
source - Program.cs:24
assign - Program.cs:25
return <-
assign - Program.cs:35
pass -> Program.cs:37 (System.Int32)Brackets(System.Int32)
assign - Program.cs:29
assign - Program.cs:30
return <-
assign - Program.cs:37
sink -> Program.cs:39 (System.Void)Sink(System.Int32)

Context sensitivity

```
17      class Container {
18          private int B;
19          public void Store(int a) { B = a; }
20      }
21
22      class Program {
23          [Tainted] private int A;
24          [Filter] private int Filter(int a) { return a; }
25          [Sink] private void Sink(Container c) {}
26
27          private static void Main(string[] args) {
28              var a = new Program();
29              var b = a.A;
30              var c = a.Filter(b);
31              var d = new Container();
32              var e = new Container();
33              d.Store(b);
34              e.Store(c);
35              a.Sink(d);
36              a.Sink(e);
37          }
38      }
```

Tainted sink
source - Program.cs:29
pass -> Program.cs:33 (System.Void)Store(System.Int32)
assign - Program.cs:19
return <-
sink -> Program.cs:35 (System.Void)Sink(TaintTrackingTests.Container)

Context sensitivity

```
17      class Container {
18          private int B;
19          public void Store(int a) { B = a; }
20      }
21
22      class Program {
23          [Tainted] private int A;
24          [Filter] private int Filter(int a) { return a; }
25          [Sink] private void Sink(Container c) {}
26
27          private static void Main(string[] args) {
28              var a = new Program();
29              var b = a.A;
30              var c = a.Filter(b);
31              var d = new Container();
32              var e = new Container();
33              d.Store(b);
34              e.Store(c);
35              a.Sink(d);
36              a.Sink(e);
37          }
38      }
```

Tainted sink
source - Program.cs:29
pass -> Program.cs:33 (System.Void)Store(System.Int32)
assign - Program.cs:19
return <-
sink -> Program.cs:35 (System.Void)Sink(TaintTrackingTests.Container)

Context sensitivity

```
17      class Container {
18          private int B;
19          public void Store(int a) { B = a; }
20      }
21
22      class Program {
23          [Tainted] private int A;
24          [Filter] private int Filter(int a) { return a; }
25          [Sink] private void Sink(Container c) {}
26
27          private static void Main(string[] args) {
28              var a = new Program();
29              var b = a.A;
30              var c = a.Filter(b);
31              var d = new Container();
32              var e = new Container();
33              d.Store(b);
34              e.Store(c);
35              a.Sink(d);
36              a.Sink(e);
37          }
38      }
```

Tainted sink
source - Program.cs:29
pass -> Program.cs:33 (System.Void)Store(System.Int32)
assign - Program.cs:19
return <-
sink -> Program.cs:35 (System.Void)Sink(TaintTrackingTests.Container)

Context sensitivity

```
17      class Container {
18          private int B;
19          public void Store(int a) { B = a; }
20      }
21
22      class Program {
23          [Tainted] private int A;
24          [Filter] private int Filter(int a) { return a; }
25          [Sink] private void Sink(Container c) {}
26
27          private static void Main(string[] args) {
28              var a = new Program();
29              var b = a.A;
30              var c = a.Filter(b);
31              var d = new Container();
32              var e = new Container();
33              d.Store(b);
34              e.Store(c);
35              a.Sink(d);
36              a.Sink(e);
37          }
38      }
```

Tainted sink
source - Program.cs:29
pass -> Program.cs:33 (System.Void)Store(System.Int32)
assign - Program.cs:19
return <-
sink -> Program.cs:35 (System.Void)Sink(TaintTrackingTests.Container)

Поддержка делегатов

```
35      private int Brackets1(int a) {
36          return a;
37      }
38
39      private int Brackets2(int a) {
40          return Filter(a);
41      }
42
43      private int Call(Brackets brackets, int v) {
44          return brackets(v);
45      }
46
47      static void Main(string[] args) {
48          Program a = new Program();
49
50          var b = a.A;
51          var c = a.Filter(b);
52
53          var d = a.Call(a.Brackets1, b);
54          var e = a.Call(a.Brackets2, b);
55
56          var f = a.Call(a.Brackets1, c);
57          var g = a.Call(a.Brackets2, c);
58
59          a.Sink(d);
60          a.Sink(e); // False positive!!!
61          a.Sink(f);
62          a.Sink(g);
63      }
```

void Program.Sink(int a)

Tainted sink
source - Program.cs:50
pass -> Program.cs:54 (System.Int32)Call(TaintTrackingTests.Brackets,System.Int32)
pass -> Program.cs:44 Invoke
assign - Program.cs:36
return <-
assign - Program.cs:44
return <-
assign - Program.cs:54
sink -> Program.cs:60 (System.Void)Sink(System.Int32)

Поддержка делегатов

```
35      private int Brackets1(int a) {
36          return a;
37      }
38
39      private int Brackets2(int a) {
40          return Filter(a);
41      }
42
43      private int Call(Brackets brackets, int v) {
44          return brackets(v);
45      }
46
47      static void Main(string[] args) {
48          Program a = new Program();
49
50          var b = a.A;
51          var c = a.Filter(b);
52
53          var d = a.Call(a.Brackets1, b);
54          var e = a.Call(a.Brackets2, b);
55
56          var f = a.Call(a.Brackets1, c);
57          var g = a.Call(a.Brackets2, c);
58
59          a.Sink(d);
60          a.Sink(e); // False positive!!!
61          a.Sink(f);
62          a.Sink(g);
63      }
```

void Program.Sink(int a)

Tainted sink
source - Program.cs:50
pass -> Program.cs:54 (System.Int32)Call(TaintTrackingTests.Brackets,System.Int32)
pass -> Program.cs:44 Invoke
assign - Program.cs:36
return <-
assign - Program.cs:44
return <-
assign - Program.cs:54
sink -> Program.cs:60 (System.Void)Sink(System.Int32)

Поддержка делегатов

```
35      private int Brackets1(int a) {
36          return a;
37      }
38
39      private int Brackets2(int a) {
40          return Filter(a);
41      }
42
43      private int Call(Brackets brackets, int v) {
44          return brackets(v);
45      }
46
47      static void Main(string[] args) {
48          Program a = new Program();
49
50          var b = a.A;
51          var c = a.Filter(b);
52
53          var d = a.Call(a.Brackets1, b);
54          var e = a.Call(a.Brackets2, b);
55
56          var f = a.Call(a.Brackets1, c);
57          var g = a.Call(a.Brackets2, c);
58
59          a.Sink(d);
60          a.Sink(e); // False positive!!!
61          a.Sink(f);
62          a.Sink(g);
63      }
```

void Program.Sink(int a)

Tainted sink
source - Program.cs:50
pass -> Program.cs:54 (System.Int32)Call(TaintTrackingTests.Brackets,System.Int32)
pass -> Program.cs:44 Invoke
assign - Program.cs:36
return <-
assign - Program.cs:44
return <-
assign - Program.cs:54
sink -> Program.cs:60 (System.Void)Sink(System.Int32)

Поддержка делегатов

```
35      private int Brackets1(int a) {
36          return a;
37      }
38
39      private int Brackets2(int a) {
40          return Filter(a);
41      }
42
43      private int Call(Brackets brackets, int v) {
44          return brackets(v);
45      }
46
47      static void Main(string[] args) {
48          Program a = new Program();
49
50          var b = a.A;
51          var c = a.Filter(b);
52
53          var d = a.Call(a.Brackets1, b);
54          var e = a.Call(a.Brackets2, b);
55
56          var f = a.Call(a.Brackets1, c);
57          var g = a.Call(a.Brackets2, c);
58
59          a.Sink(d);
60          a.Sink(e); // False positive!!!
61          a.Sink(f);
62          a.Sink(g);
63      }
```

void Program.Sink(int a)

Tainted sink
source - Program.cs:50
pass -> Program.cs:54 (System.Int32)Call(TaintTrackingTests.Brackets,System.Int32)
pass -> Program.cs:44 Invoke
assign - Program.cs:36
return <-
assign - Program.cs:44
return <-
assign - Program.cs:54
sink -> Program.cs:60 (System.Void)Sink(System.Int32)

Поддержка делегатов

```
35      private int Brackets1(int a) {
36          return a;
37      }
38
39      private int Brackets2(int a) {
40          return Filter(a);
41      }
42
43      private int Call(Brackets brackets, int v) {
44          return brackets(v);
45      }
46
47      static void Main(string[] args) {
48          Program a = new Program();
49
50          var b = a.A;
51          var c = a.Filter(b);
52
53          var d = a.Call(a.Brackets1, b);
54          var e = a.Call(a.Brackets2, b);
55
56          var f = a.Call(a.Brackets1, c);
57          var g = a.Call(a.Brackets2, c);
58
59          a.Sink(d);
60          a.Sink(e); // False positive!!!
61          a.Sink(f);
62          a.Sink(g);
63      }
```

void Program.Sink(int a)

Tainted sink
source - Program.cs:50
pass -> Program.cs:54 (System.Int32)Call(TaintTrackingTests.Brackets,System.Int32)
pass -> Program.cs:44 Invoke
assign - Program.cs:36
return <-
assign - Program.cs:44
return <-
assign - Program.cs:54
sink -> Program.cs:60 (System.Void)Sink(System.Int32)

Производительность

Проект	Классов	Методов	Время выполнения (с)	Выделенная память (Гб)
Mono	21013	192745	21 ± 0.5	~ 4.2
EventStore	3828	22796	2.7 ± 0.1	~ 0.49
OpenRA	2767	15451	2.3 ± 0.1	~ 0.4

Итоги

- Реализован инструмент, позволяющий задавать правила анализа при помощи магазинных автоматов
- Его работа протестирована на двух различных видах анализа с интеграцией в ReSharper

Ссылки:

- Исходный код:
  - `https://github.com/JetBrains-Research/CoFRA`

Контакты:

- Илья Ножкин:
  - nozhkin.ii@gmail.com
- Семён Григорьев:
  - s.v.grigoriev@spbu.ru
  - semen.grigorev@jetbrains.com