FUZZING ДЛЯ ТЕСТИРОВАНИЯ JVM: ЗАЧЕМ И КАК

Макс Казанцев
max.kazantsev@azul.com

Что вернёт эта функция?

```
char foo1(char c) {
    for (int i = 0; i < 193; i++) {
        c &= (char) 1;
        c += (char) 2;
    }
    return c;
}
```

clang v3.8-5.0, -O2

A) 0 или 1	B) 1 или 2
C) 2 или 3	D) 3 или 4

А эта?

```
char foo2(char c) {
    for (int i = 0; i < 193; i++) {
        c &= (char) 1;
        c += (char) 3;
    }
    return c;
}
```

clang v3.8-5.0, -O2

A) 0 или 1	B) 1 или 2
C) 2 или 3	D) 3 или 4

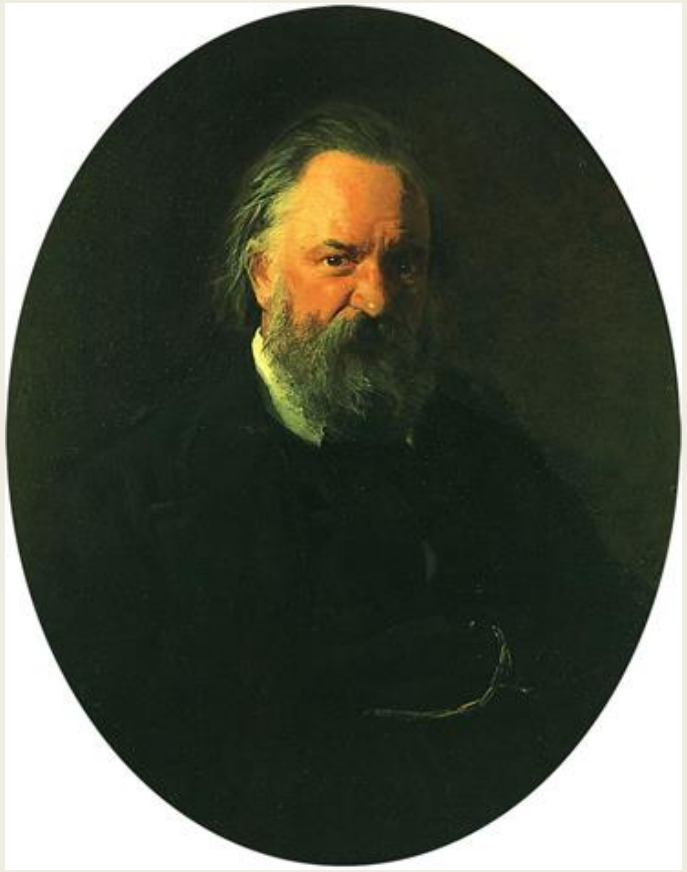
https://godbolt.org/g/kG6qWA

А здесь бага нет

```
char foo3(char c) {
    for (int i = 0; i < 193; i++) {
        c &= (char) 2;
        c += (char) 3;
    }
    return c;
}

char foo4(char c) {
    for (int i = 0; i < 193; i++) {
        c &= (char) 1;
        c += (char) 2;
    }
    return c;
}
```

Два главных вопроса



Кто виноват?

Что делать?

О чём поговорим

- Почему в компиляторах и VM бывают баги
- Что такое Fuzzing-тестирование
- Как фаззер помогает искать баги
- Как использовать один фаззер для разных языков

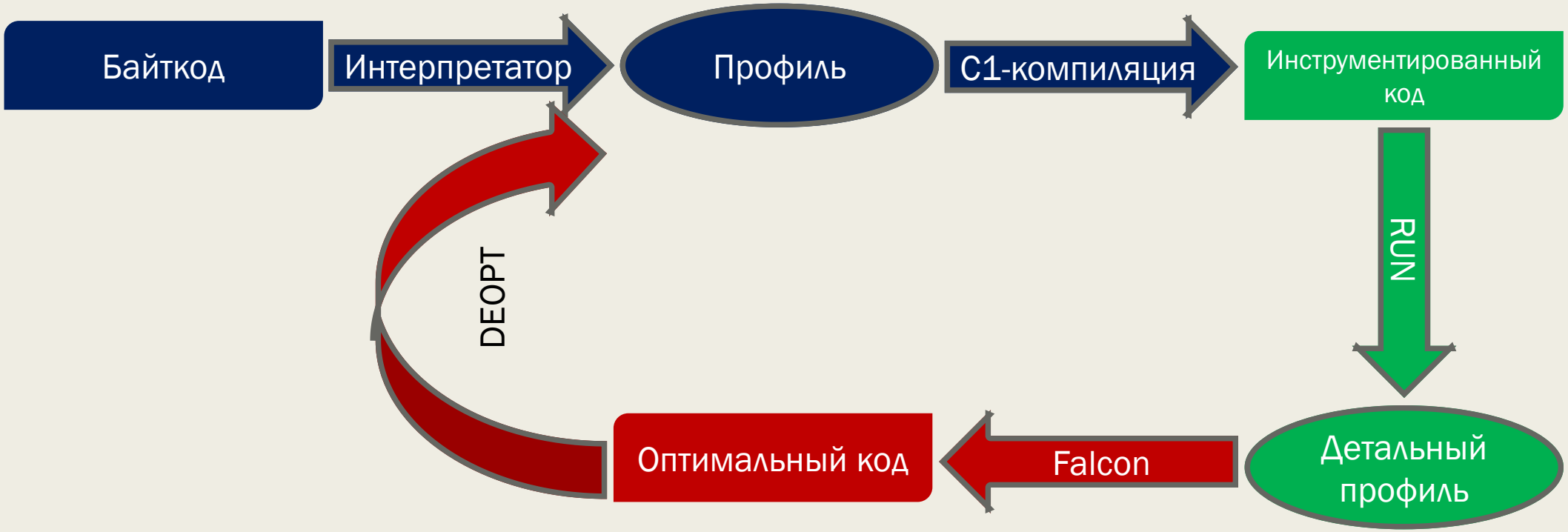
Чем мы занимаемся

- Azul Systems
  - *Zing VM*
  - *Zulu – OpenJDK fork*
  - *Zulu Embedded*
- Falcon – Tier-2 JIT-компилятор на основе LLVM в Zing VM
  - *Пишем новые оптимизации*
  - *Расширяем область применимости старых оптимизаций*
  - *Исправляем ошибки*

Немного об LLVM

- Фреймворк для построения компиляторов
  - *Внутреннее представление*
  - *Высокоуровневые оптимизации*
  - *Кодогенератор и платформенно-зависимые оптимизации*
  - *Инструменты*
- Широкая область применения
  - *clang*
  - *Xcode*
  - *Swift, Rust, Kotlin Native и др.*
- ~50k строк изменений в неделю
- Больше деталей: https://llvm.org/

Компиляция в Zing VM



Виды багов, которые мы ищем

- Падения VM во время исполнения
- Бесконечное/неприемлемо долгое время компиляции
- Некорректные результаты компиляции

Почему в компиляторах есть баги?

- Объём кода
- Скорость изменения кода
- Некоторые баги проявляются только при соблюдении особых условий
- Даже если компилятор сделал ошибку, она не обязательно наблюдаема

Особенности JIT-компиляции

- Может использовать данные времени исполнения
  - *Горячие участки кода*
  - *Информация о реальных классах объектов в виртуальных вызовах*
  - *Информация о загруженных классах и т.п.*
- Порядок компиляции/инлайнинга может отличаться от запуска к запуску
- Возможны спекулятивные оптимизации
- Методы перекомпилируются несколько раз

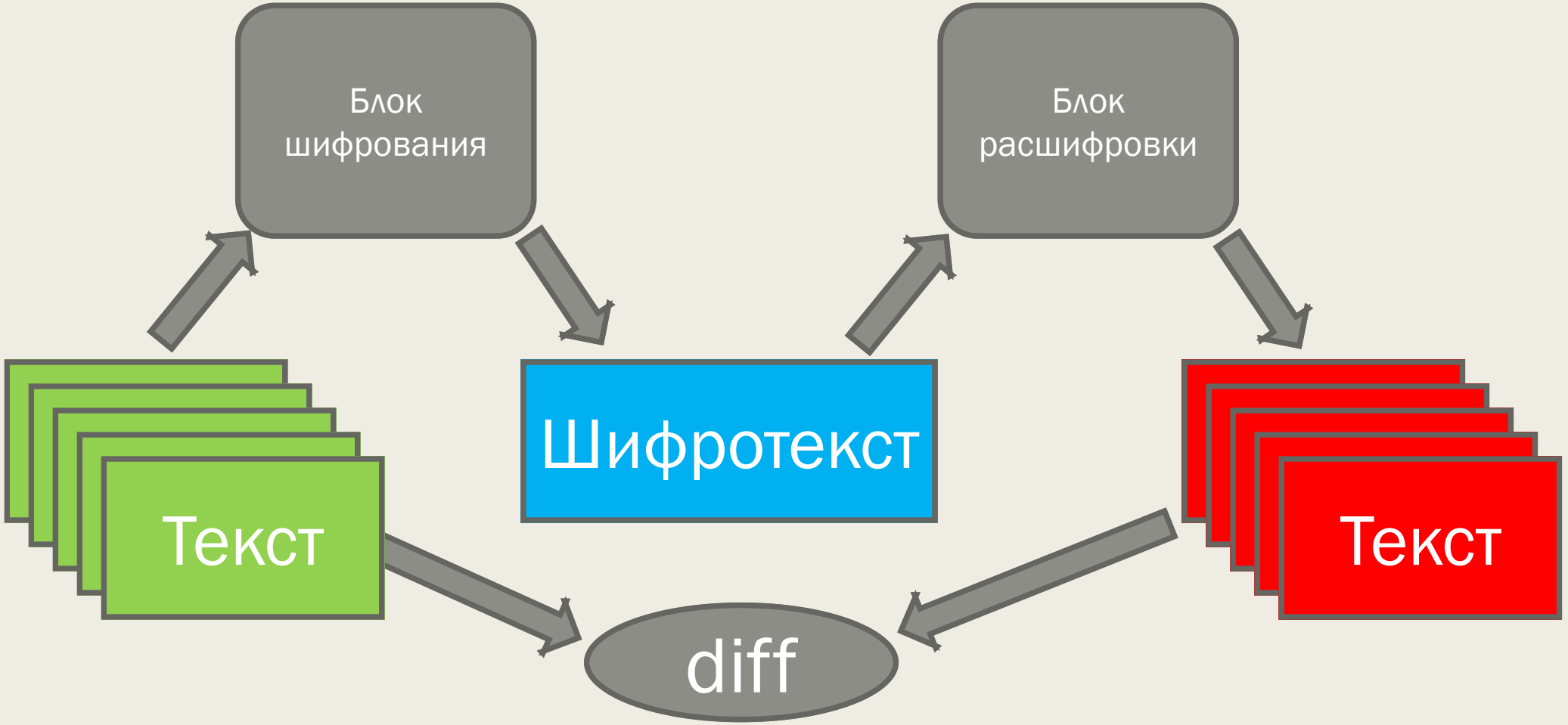
Как тестируют компиляторы в VM

- Наборы небольших статических тестов (regression, unit и т.п.)
- Реальные приложения
- Coverage-тестирование
- Fuzzing-тестирование

Что такое Fuzzing

- От англ. Fuzzy – нечёткий, неопределённый, пушистый
- Состоит в генерации и подаче на вход случайных данных
- Широко применяется там, где требуется высокая степень надёжности

Как можно тестировать шифрование?



Существующие Fuzzer’ы

- libFuzzer https://llvm.org/docs/LibFuzzer.html
- American Fuzzy Lop http://lcamtuf.coredump.cx/afl/
- Mull Mutation https://github.com/mull-project/mull
- CSmith https://embed.cs.utah.edu/csmith/
- jittester (OpenJDK project)
- Java Fuzzer for Android https://github.com/android-art-intel/Fuzzer
- Java Fuzzer https://github.com/AzulSystems/JavaFuzzer
- И другие

Пример сгенерированной программы

```
public static void vMeth1() {

    int i7=54703, i8=-14, i18=-32644, i19=4, i20=8, i21=-1, iArr1[]=new int[N];
    short sArr[]=new short[N];
    long lArr1[]=new long[N];

    FuzzerUtils.init(sArr, (short)28174);
    FuzzerUtils.init(iArr1, 5);
    FuzzerUtils.init(lArr1, -50290L);

    for (i7 = 11667; i7 > 225; i7--) {
        switch ((i7 % 3) + 74) {
        case 74:
            for (i18 = 4; i18 < 175; i18++) {
                for (i20 = i7; i20 < 2; ++i20) {
                    double d1=-20.76874;
                    sArr[i18] = (short)(((long)(d1 / (i7 | 1)) >> iArr1[i7]) * i20);
                    i19 = (int)8563533763630855434L;
                    Cls1.fFld += ((lArr1[i7 + 1] - (++i8)) + i18);
                    i19 += (int)(-Cls.instanceCount);
                    if (i7 != 0) {
                        vMeth1_check_sum += i7 + i8 + i18 + i19 + i20 + i21 + FuzzerUtils.checkSum(sArr) +
                            FuzzerUtils.checkSum(iArr1) + FuzzerUtils.checkSum(lArr1);
                        return;
                    }
                    try {
                        iArr1[i20] = (240 / Test.iFld1);
                        iArr1[i20 - 1] = (-44739 % i21);
                        i19 = (Test.iFld1 / -34797);
                    } catch (ArithmeticException a_e) {}
                    i8 += (i20 | Test.instanceCount);
                    Test.iFld1 = (i21--);
                }
            }
            break;
        case 75:
            try {
                Test.iFld1 = (5441 / i18);
                iArr1[i7] = (i20 % -49571);
                i19 = (i8 % 130);
            } catch (ArithmeticException a_e) {}
        case 76:
            i8 |= (int)((-Cls.instanceCount) - (iArr1[i7]--));
            break;
        default:
            i8 -= i18;
        }
    }
    vMeth1_check_sum += i7 + i8 + i18 + i19 + i20 + i21 + FuzzerUtils.checkSum(sArr) + FuzzerUtils.checkSum(iArr1)
        + FuzzerUtils.checkSum(lArr1);
}
```

К счастью, их можно сокращать

```
1   public static void vMeth1() {
2       int i7=54703, iArr1[] = new int[N];
3       for (i7 = 228; i7 > 225; i7--) {
4           switch ((i7 % 3)) {
5           case 0:
6               for (int i18 = 4; i18 < 175; i18++) {
7                   for (int i20 = i7; i20 < 2; ++i20) {
8                       if (i7 != 0) {
9                           return;
10                      }
11                      iArr1[i20 - 1] = 1;
12                  }
13              }
14          default:
15          }
16      }
17      vMeth1_check_sum += i7;
18  }
```

Прошлые доклады про фаззеры

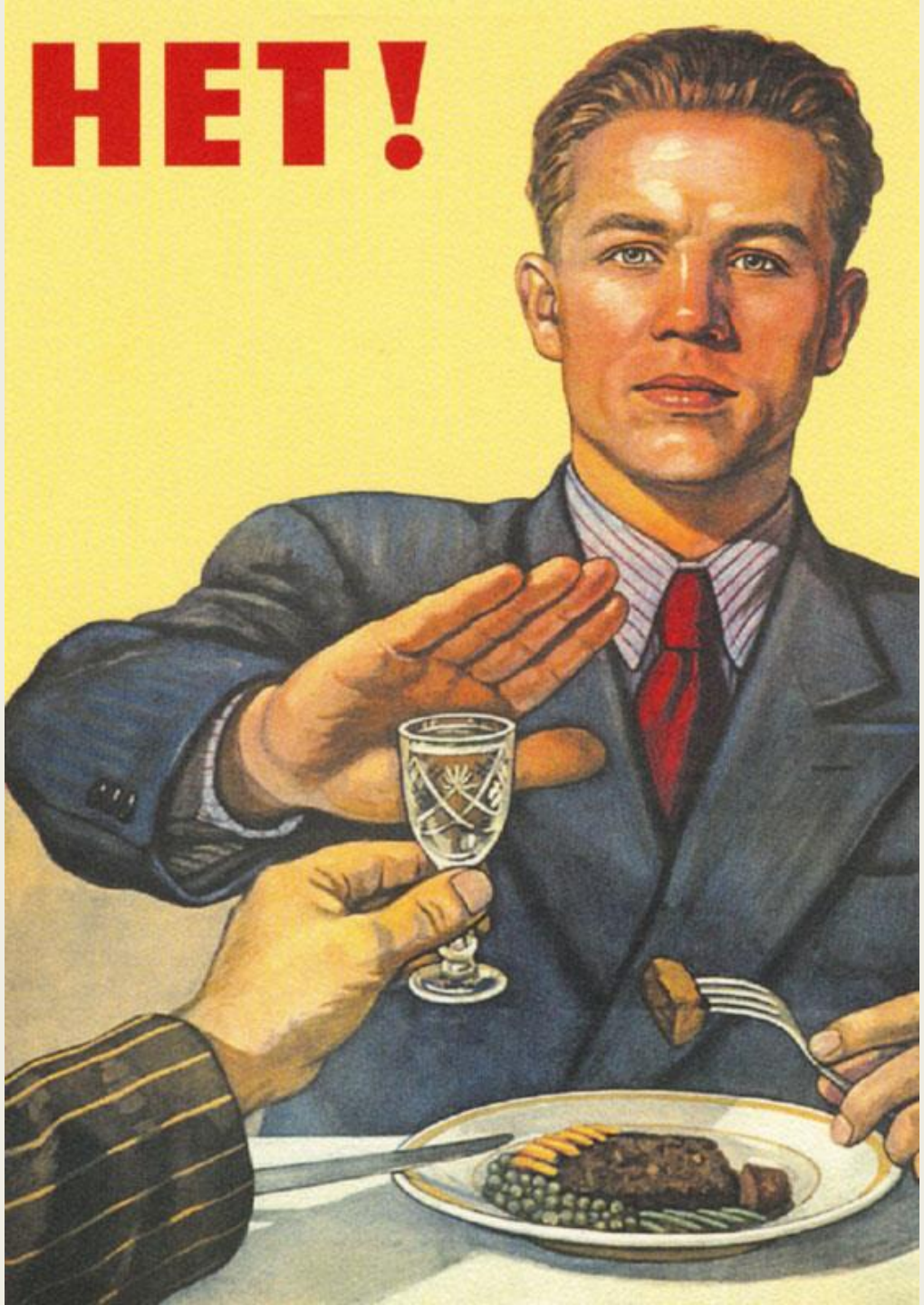
- JBreak 2018 (Новосибирск)
- Heisenbug 2018 (Санкт-Петербург)

Опыт использования Java-fuzzer-ов

- Регулярная генерация новых тестов – 150+ тыс. в неделю
- Среднее количество находимых багов – 1 в неделю (раньше – 2-3 в неделю)
- Regression suite
- Pre-commit suite
- Измерение производительности (?)

Какие возникают проблемы

- Введение новых конструкций языка (лямбды, анонимные классы и т.д.)
- Каждый Fuzzer ориентирован на конкретный язык
- Есть другие языки, ориентированные на JVM
  - *Scala*
  - *Kotlin*
  - *ByteCode*
- Пишем Fuzzer заново под каждый язык?


НЕТ!

Программа на Java и на Scala

```java
public class JavaTest {
    private int fld = 12;
    void foo(int n) {
        int a = 10;
        if (a > fld) fld += n;
        float x = (a + fld != a * fld) ? a / 1.25f : fld * 1.25f;
        System.out.println(x);
    }
    public static void main(String[] args) {
        new JavaTest().foo(123);
    }
}
```

```scala
class ScalaTest {
  private var fld = 12
  def foo(n: Int) = {
    val a = 10
    if (a > fld) fld += n
    val x = if (a + fld != a * fld) a / 1.25f
    else fld * 1.25f
    System.out.println(x)
  }
}
object ScalaTest {
  def main(args: Array[String]) = {
    new JavaTest().foo(123)
  }
}
```

Общие свойства подобных языков

- Грамматика –контекстно-свободная
- Семантика – разная, но есть общие черты:
  - Небольшое число примитивных типов
  - Есть идентификаторы (классы, функции, переменные и т.п.)
  - Есть области видимости идентификаторов
- Идентификаторы объявляются перед использованием
- Тип известен в момент объявления
- Нужно «всего лишь» использовать идентификаторы в правильном контексте

Что нужно для Fuzz-генерации тестов на подобных языках

- Генерировать синтаксические конструкции
- Генерировать имена для классов, переменных, функций и т.п.
- Отслеживать области видимости
- Правильно сводить типы

Контекстно-свободные грамматики

```
@Adjective → Большой | Зелёный | Мягкий
@Animal → Крокодил | Заяц | Волк
@Adjectives → @Adjective @Adjectives | ε
@Text → @Adjectives @Animal и @Adjectives @Animal
```

Контекстно-свободные грамматики

```
@Adjective → Большой | Зелёный | Мягкий
@Animal → Крокодил | Заяц | Волк
@Adjectives → @Adjective @Adjectives | ε
@Text → @Adjectives @Animal и @Adjectives @Animal
```

Контекстно-свободные грамматики

@Adjective → Большой | Зелёный | Мягкий

@Animal → Крокодил | Заяц | Волк

@Adjectives → @Adjective @Adjectives | ε

@Text → @Adjectives @Animal и @Adjectives @Animal

Контекстно-свободные грамматики

```
@Adjective → Большой | Зелёный | Мягкий
@Animal → Крокодил | Заяц | Волк
@Adjectives → @Adjective @Adjectives | ε
@Text → @Adjectives @Animal и @Adjectives @Animal
```

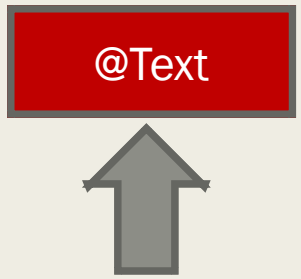
Генерация текста по грамматике

@Adjective → Большой | Зелёный | Мягкий

@Animal → Крокодил | Заяц | Волк

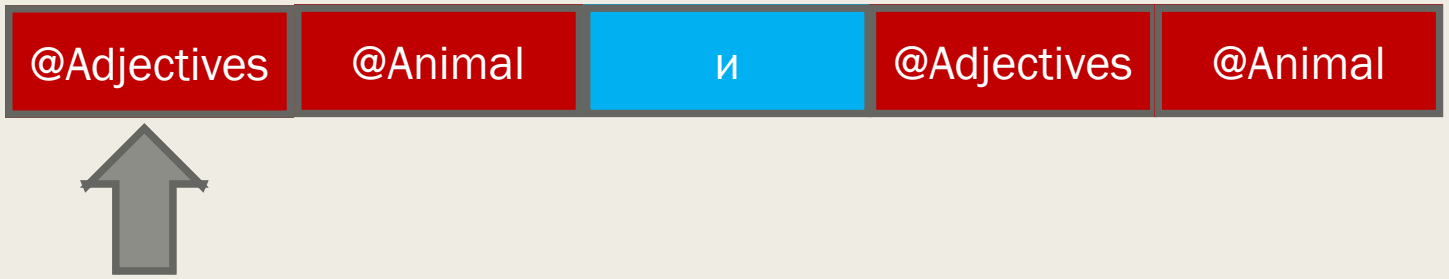
@Adjectives → @Adjective @Adjectives | ε

@Text → @Adjectives @Animal и @Adjectives @Animal



Генерация текста по грамматике

```
@Adjective → Большой | Зелёный | Мягкий
@Animal → Крокодил | Заяц | Волк
@Adjectives → @Adjective @Adjectives | ε
@Text → @Adjectives @Animal и @Adjectives @Animal
```



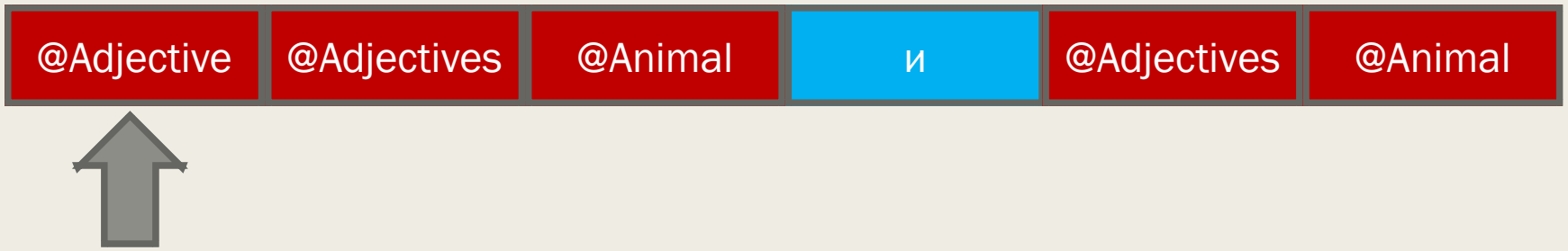
Генерация текста по грамматике

@Adjective → Большой | **Зелёный** | Мягкий

@Animal → Крокодил | Заяц | Волк

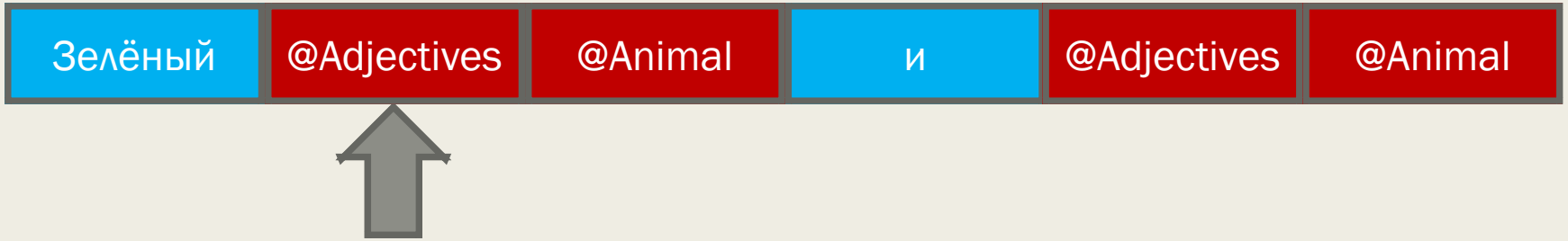
@Adjectives → @Adjective @Adjectives | ε

@Text → @Adjectives @Animal и @Adjectives @Animal



Генерация текста по грамматике

```
@Adjective → Большой | Зелёный | Мягкий
@Animal → Крокодил | Заяц | Волк
@Adjectives → @Adjective @Adjectives | ε
@Text → @Adjectives @Animal и @Adjectives @Animal
```



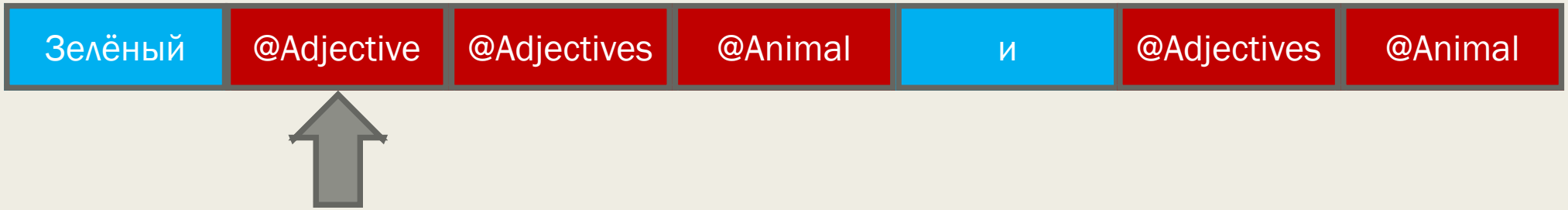
Генерация текста по грамматике

@Adjective → Большой | Зелёный | **Мягкий**

@Animal → Крокодил | Заяц | Волк

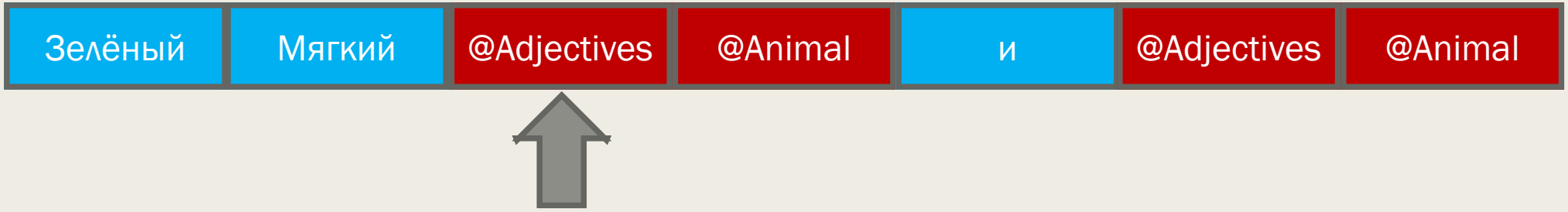
@Adjectives → @Adjective @Adjectives | ε

@Text → @Adjectives @Animal и @Adjectives @Animal



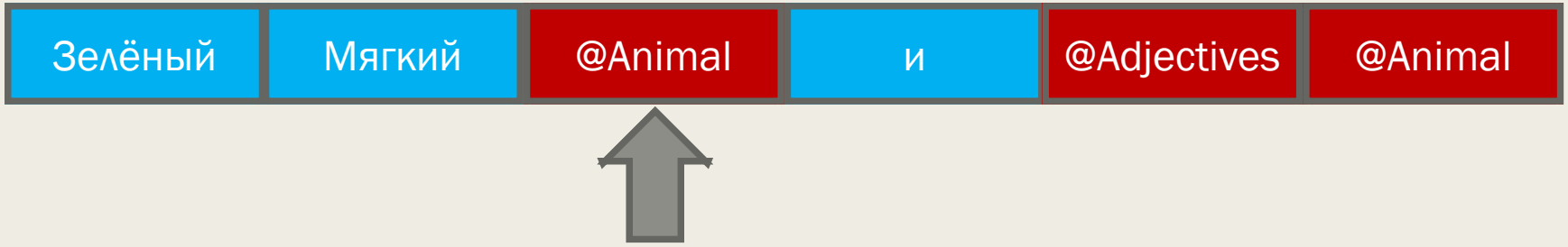
Генерация текста по грамматике

```
@Adjective → Большой | Зелёный | Мягкий
@Animal → Крокодил | Заяц | Волк
@Adjectives → @Adjective @Adjectives | ε
@Text → @Adjectives @Animal и @Adjectives @Animal
```



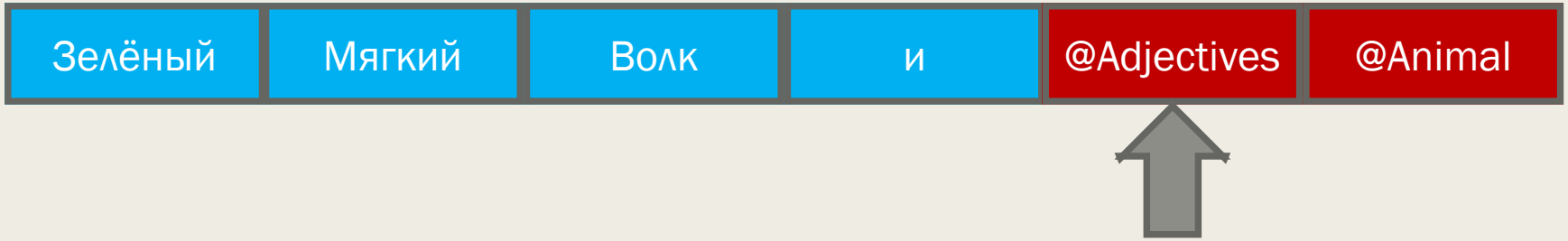
Генерация текста по грамматике

```
@Adjective → Большой | Зелёный | Мягкий
@Animal → Крокодил | Заяц | Волк
@Adjectives → @Adjective @Adjectives | ε
@Text → @Adjectives @Animal и @Adjectives @Animal
```



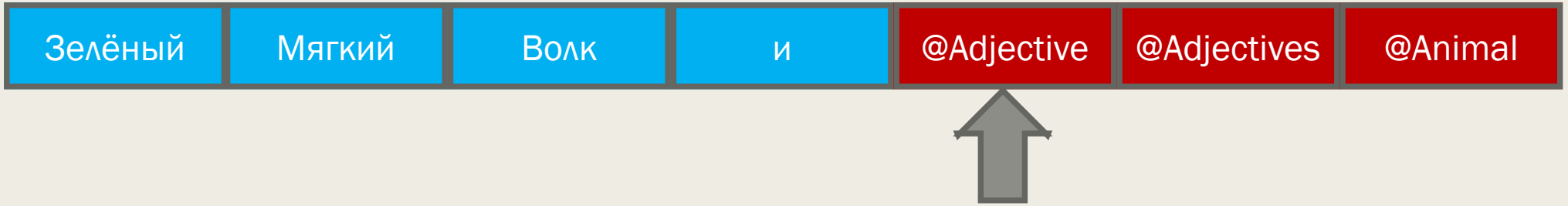
Генерация текста по грамматике

```
@Adjective → Большой | Зелёный | Мягкий
@Animal → Крокодил | Заяц | Волк
@Adjectives → @Adjective @Adjectives | ε
@Text → @Adjectives @Animal и @Adjectives @Animal
```



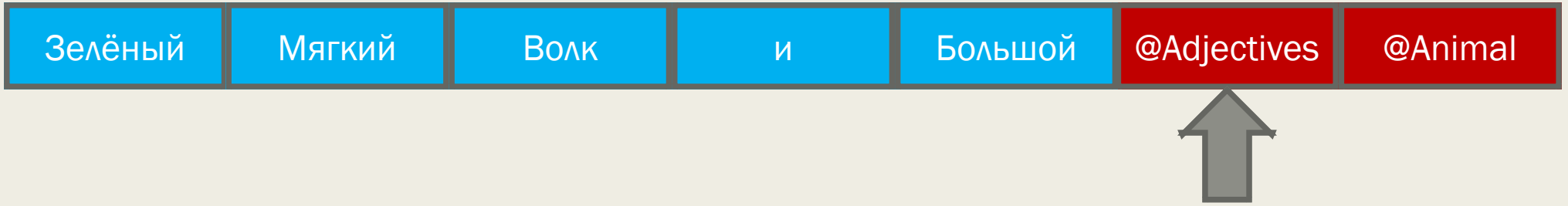
Генерация текста по грамматике

```
@Adjective → Большой | Зелёный | Мягкий
@Animal → Крокодил | Заяц | Волк
@Adjectives → @Adjective @Adjectives | ε
@Text → @Adjectives @Animal и @Adjectives @Animal
```



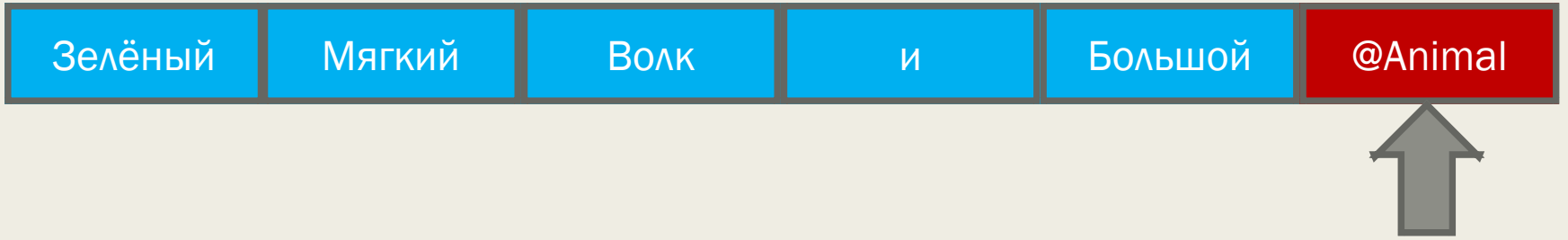
Генерация текста по грамматике

```
@Adjective → Большой | Зелёный | Мягкий
@Animal → Крокодил | Заяц | Волк
@Adjectives → @Adjective @Adjectives | ε
@Text → @Adjectives @Animal и @Adjectives @Animal
```



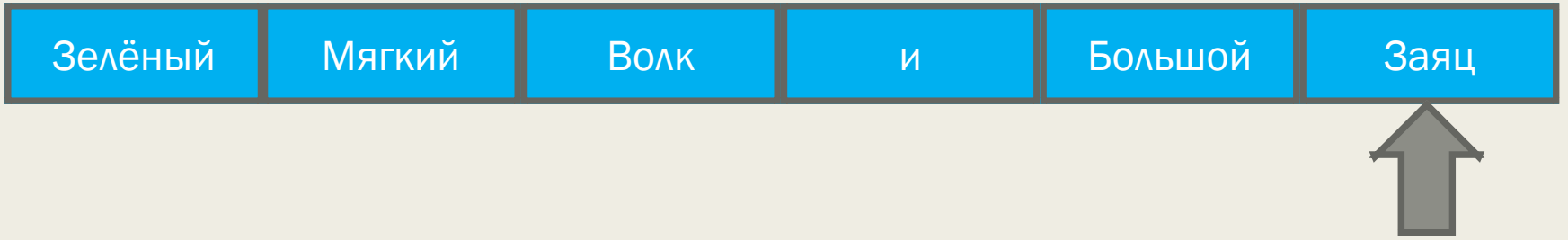
Генерация текста по грамматике

```
@Adjective → Большой | Зелёный | Мягкий
@Animal → Крокодил | Заяц | Волк
@Adjectives → @Adjective @Adjectives | ε
@Text → @Adjectives @Animal и @Adjectives @Animal
```



Генерация текста по грамматике

```
@Adjective → Большой | Зелёный | Мягкий
@Animal → Крокодил | Заяц | Волк
@Adjectives → @Adjective @Adjectives | ε
@Text → @Adjectives @Animal и @Adjectives @Animal
```



Правила грамматики

```
@TEXT
        #BEGIN_RULE @Adjectives @Animal и @Adjectives @Animal #END_RULE
@Adjective
        #BEGIN_RULE Большой #END_RULE
        #BEGIN_RULE Зелёный #END_RULE
        #BEGIN_RULE Мягкий #END_RULE
@Adjectives
        #BEGIN_RULE @Adjective @Adjectives #END_RULE
        #BEGIN_RULE #END_RULE
@Animal
        #BEGIN_RULE Крокодил #END_RULE
        #BEGIN_RULE Заяц #END_RULE
        #BEGIN_RULE Волк #END_RULE
```

Грамматикой описывается синтаксис

@IntFunction → int @ID(@Params) { @IntFunctionBody }

@Params → @Param @MoreParams | ε

@MoreParams → , @Param @MoreParams| ε

@Param → @Type @Id

@IntFunctionBody → @Statements @IntReturn

@Statements → @Statement @Statements | ε

@Statement → @DeclareVar | @Loop | @If | …

@IntReturn → return @IntExpression ;

И т.д.

Генерация синтаксиса

```
@IntFunction
```

Генерация синтаксиса

```
int @ID(@Params) {
    @IntFunctionBody
}
```

Генерация синтаксиса

```
int @ID(@Param @MoreParams) {
    @IntFunctionBody
}
```

Генерация синтаксиса

```
int @ID(int @ID @MoreParams) {
    @IntFunctionBody
}
```

Генерация синтаксиса

```
int @ID(int @ID, @Param @MoreParams) {
    @IntFunctionBody
}
```

Генерация синтаксиса

```
int @ID(int @ID, int @ID @MoreParams) {
    @IntFunctionBody
}
```

Генерация синтаксиса

```
int @ID(int @ID, int @ID, @Param @MoreParams) {
    @IntFunctionBody
}
```

Генерация синтаксиса

```
int @ID(int @ID, int @ID, int @ID @MoreParams) {
    @IntFunctionBody
}
```

Генерация синтаксиса

```
int @ID(int @ID, int @ID, int @ID) {
    @IntFunctionBody
}
```

Генерация синтаксиса

```
int @ID(int @ID, int @ID, int @ID) {
    @Statements
    @IntReturn
}
```

Генерация синтаксиса

```
int @ID(int @ID, int @ID, int @ID) {
    @Statement
    @Statements
    @IntReturn
}
```

Генерация синтаксиса

```
int @ID(int @ID, int @ID, int @ID) {
    int @ID = 123;
    @Statements
    @IntReturn
}
```

Генерация синтаксиса

```
int @ID(int @ID, int @ID, int @ID) {
    int @ID = 123;
    @Statement
    @Statements
    @IntReturn
}
```

Генерация синтаксиса

```
int @ID(int @ID, int @ID, int @ID) {
    int @ID = 123;
    @ID = @ID * @ID + @ID;
    @Statements
    @IntReturn
}
```

Генерация синтаксиса

```
int @ID(int @ID, int @ID, int @ID) {
    int @ID = 123;
    @ID = @ID * @ID + @ID;
    @Statement
    @Statements
    @IntReturn
}
```

Генерация синтаксиса

```
int @ID(int @ID, int @ID, int @ID) {
    int @ID = 123;
    @ID = @ID * @ID + @ID;
    if (@BoolExpression) {
       @Statements
    }
    @Statements
    @IntReturn
}
```

Генерация синтаксиса

```
int @ID(int @ID, int @ID, int @ID) {
    int @ID = 123;
    @ID = @ID * @ID + @ID;
    if (@ID + @ID > @ID + @ID) {
       @Statements
    }
    @Statements
    @IntReturn
}
```

Генерация синтаксиса

Много времени спустя…

Генерация синтаксиса

```
int @ID(int @ID, int @ID, int @ID) {
    int @ID = 123;
    @ID = @ID * @ID + @ID;
    if (@ID + @ID > @ID + @ID) {
       int @ID = 456;
       @ID = @ID;
    }
    return @ID + @ID;
}
```

Шаг 1: создаём идентификаторы

```
int @ID(int @ID, int @ID, int @ID) {
    int @ID = 123;
    @ID = @ID * @ID + @ID;
    if (@ID + @ID > @ID + @ID) {
       int @ID = 456;
       @ID = @ID;
    }
    return @ID + @ID;
}
```

Шаг 1: создаём идентификаторы

```
int @IntMethodID(int @IntVarID, int @IntVarID, int @IntVarID) {
    int @IntVarID = 123;
    @ReuseIntVarID = @ReuseIntVarID * @ReuseIntVarID + @ReuseIntVarID;
    if (@ReuseIntVarID + @ReuseIntVarID > @ReuseIntVarID +
        @ReuseIntVarID) {
       int @IntVarID = 456;
       @ReuseIntVarID = @ReuseIntVarID;
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
```

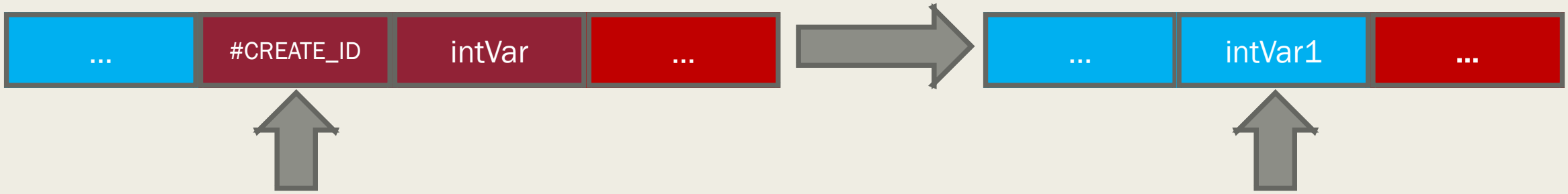
Шаг 1: создаём идентификаторы

```
@IntVarID
      #BEGIN_RULE
            #CREATE_ID intVar
      #END_RULE
```

Шаг 1: создаём идентификаторы

```
@IntMethodID → #CREATE_ID intMethod
@IntVarID → #CREATE_ID intVar
```

Шаг 1: создаём идентификаторы



Шаг 1: создаём идентификаторы

```
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int intVar4 = 123;
    @ReuseIntVarID = @ReuseIntVarID * @ReuseIntVarID + @ReuseIntVarID;
    if (@ReuseIntVarID + @ReuseIntVarID > @ReuseIntVarID +
        @ReuseIntVarID) {
       int intVar5 = 456;
       @ReuseIntVarID = @ReuseIntVarID;
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
```

Шаг 2: области видимости

```
int intMethod1(int intVar1, int intVar2, int intVar3) {
    // Доступны: intMethod1, IntVar1-3.
    int intVar4 = 123;
    // Доступны: intMethod1, IntVar1-4.
    @ReuseIntVarID = @ReuseIntVarID * @ReuseIntVarID + @ReuseIntVarID;
    if (@ReuseIntVarID + @ReuseIntVarID > @ReuseIntVarID +
        @ReuseIntVarID) {
        int intVar5 = 456;
        // Доступны: intMethod1, IntVar1-5.
        @ReuseIntVarID = @ReuseIntVarID;
    }
    // Доступны: intMethod1, IntVar1-4.
    return @ReuseIntVarID + @ReuseIntVarID;
}
```

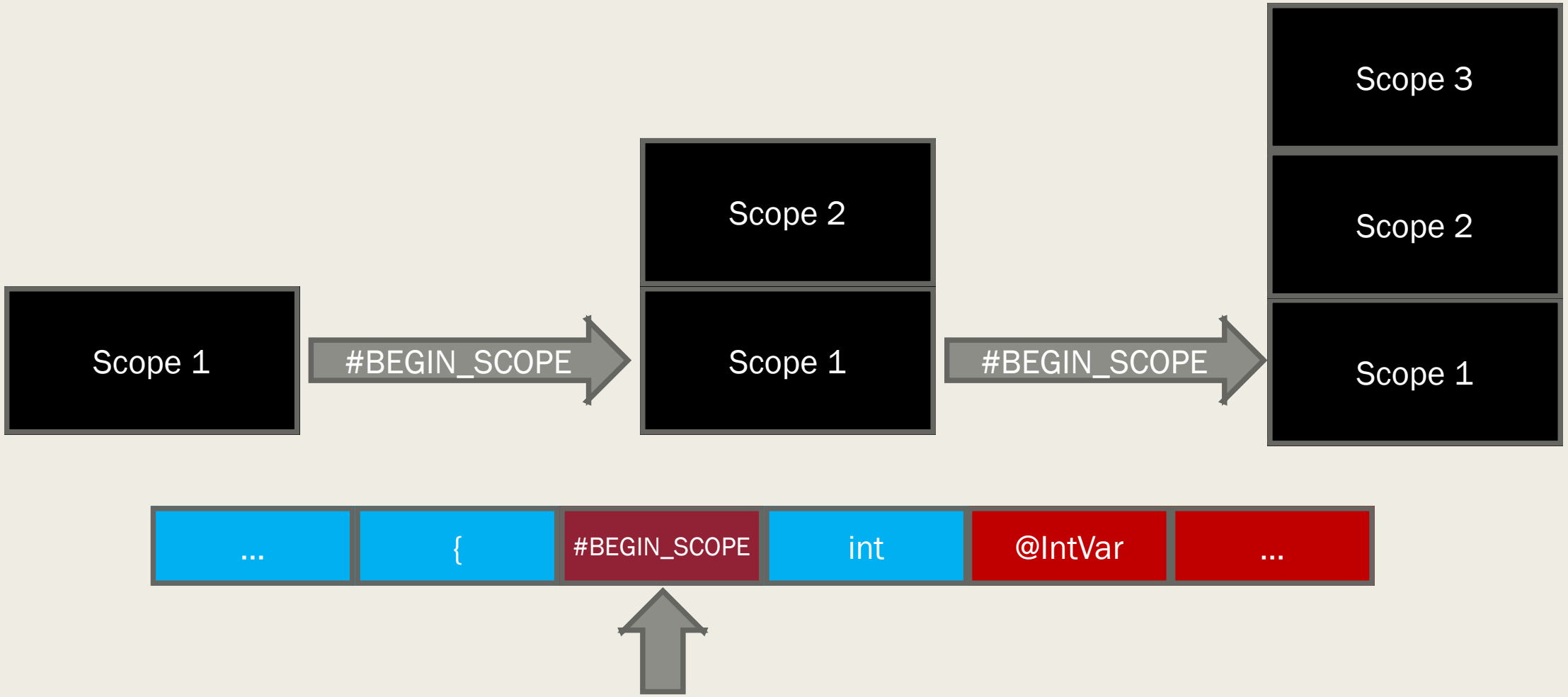
Шаг 2: области видимости

```
@INT_METHOD
      #BEGIN_RULE
            #BEGIN_SCOPE
            int @CREATE_METHOD_ID (@PARAMS) {
                  @STATEMENTS
            }
            #END_SCOPE
      #END_RULE
```

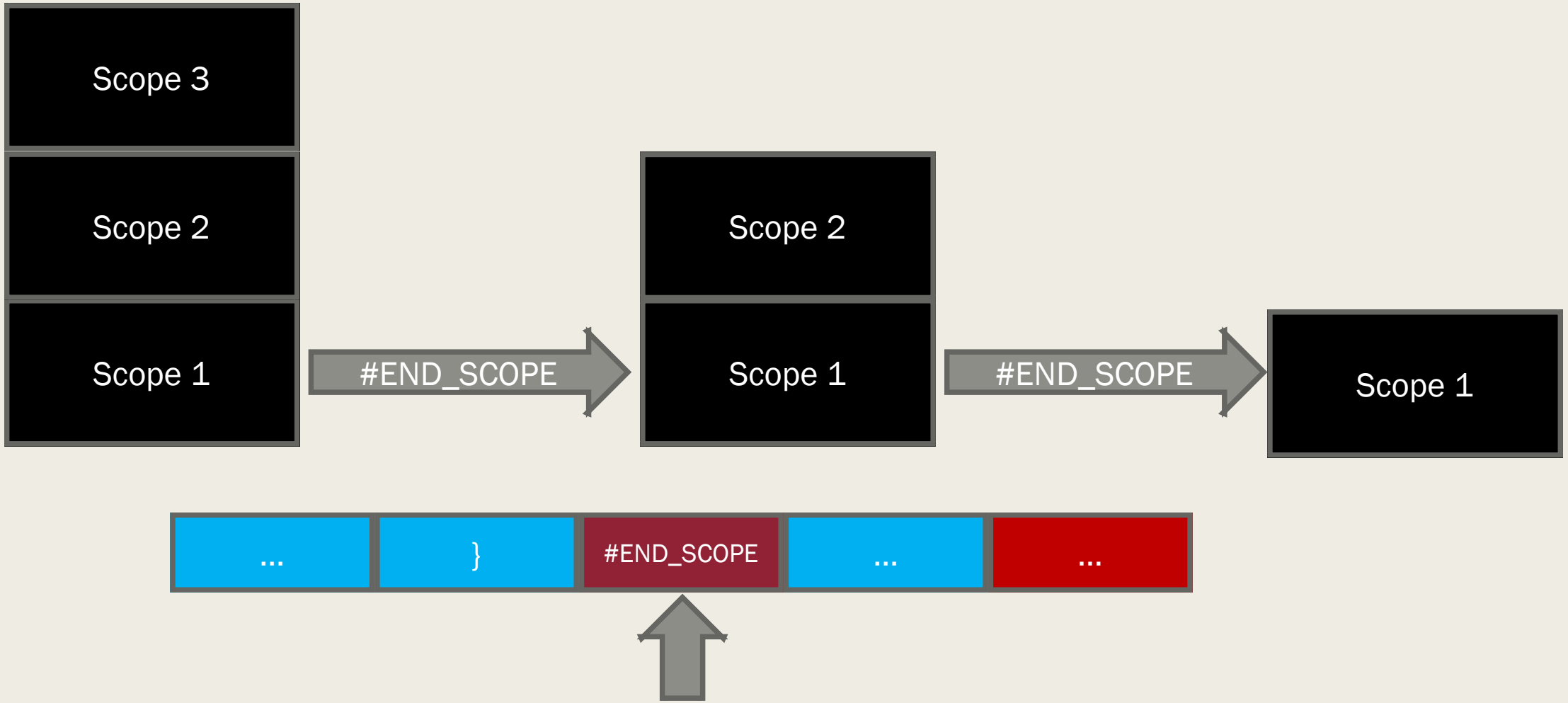
Шаг 2: области видимости

```
#BEGIN_SCOPE
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int intVar4 = 123;
    @ReuseIntVarID = @ReuseIntVarID * @ReuseIntVarID + @ReuseIntVarID;
    if (@ReuseIntVarID + @ReuseIntVarID > @ReuseIntVarID + @ReuseIntVarID) {
        #BEGIN_SCOPE
        int intVar5 = 456;
        @ReuseIntVarID = @ReuseIntVarID;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```

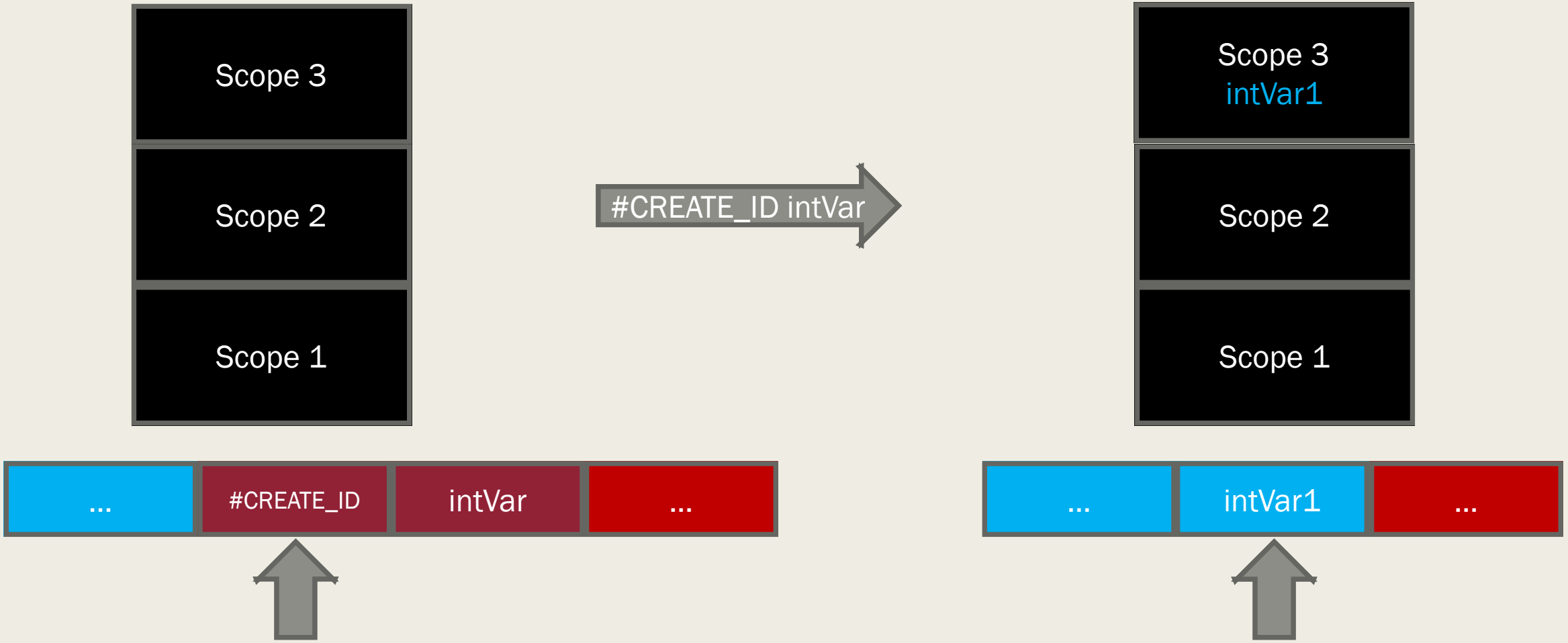
Шаг 2: области видимости



Шаг 2: области видимости



Шаг 2: области видимости



Шаг 2: области видимости

```
#BEGIN_SCOPE
int @IntMethodID(int @IntVarID, int @IntVarID, int @IntVarID) {
    int @IntVarID = 123;
    @ReuseIntVarID = @ReuseIntVarID * @ReuseIntVarID + @ReuseIntVarID;
    if (@ReuseIntVarID + @ReuseIntVarID > @ReuseIntVarID + @ReuseIntVarID) {
        #BEGIN_SCOPE
        int @IntVarID = 456;
        @ReuseIntVarID = @ReuseIntVarID;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```

Scope 1

Шаг 2: области видимости

```
#BEGIN_SCOPE
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int @IntVarID = 123;
    @ReuseIntVarID = @ReuseIntVarID * @ReuseIntVarID + @ReuseIntVarID;
    if (@ReuseIntVarID + @ReuseIntVarID > @ReuseIntVarID + @ReuseIntVarID) {
        #BEGIN_SCOPE
        int @IntVarID = 456;
        @ReuseIntVarID = @ReuseIntVarID;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```

Scope 1
intMethod1
intVar1 intVar2
intVar3

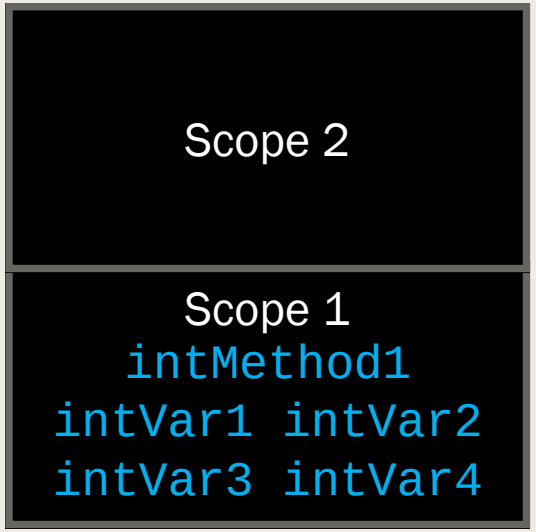
Шаг 2: области видимости

```
#BEGIN_SCOPE
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int intVar4 = 123;
    @ReuseIntVarID = @ReuseIntVarID * @ReuseIntVarID + @ReuseIntVarID;
    if (@ReuseIntVarID + @ReuseIntVarID > @ReuseIntVarID + @ReuseIntVarID) {
        #BEGIN_SCOPE
        int @IntVarID = 456;
        @ReuseIntVarID = @ReuseIntVarID;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```

Scope 1
intMethod1
intVar1 intVar2
intVar3 intVar4

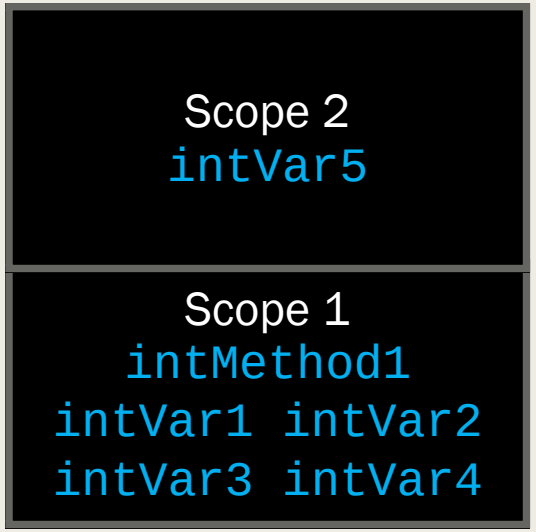
Шаг 2: области видимости

```
#BEGIN_SCOPE
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int intVar4 = 123;
    @ReuseIntVarID = @ReuseIntVarID * @ReuseIntVarID + @ReuseIntVarID;
    if (@ReuseIntVarID + @ReuseIntVarID > @ReuseIntVarID + @ReuseIntVarID) {
        #BEGIN_SCOPE
        int @IntVarID = 456;
        @ReuseIntVarID = @ReuseIntVarID;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```



Шаг 2: области видимости

```
#BEGIN_SCOPE
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int intVar4 = 123;
    @ReuseIntVarID = @ReuseIntVarID * @ReuseIntVarID + @ReuseIntVarID;
    if (@ReuseIntVarID + @ReuseIntVarID > @ReuseIntVarID + @ReuseIntVarID) {
        #BEGIN_SCOPE
        int intVar5 = 456;
        @ReuseIntVarID = @ReuseIntVarID;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```



Шаг 2: области видимости

```
#BEGIN_SCOPE
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int intVar4 = 123;
    @ReuseIntVarID = @ReuseIntVarID * @ReuseIntVarID + @ReuseIntVarID;
    if (@ReuseIntVarID + @ReuseIntVarID > @ReuseIntVarID + @ReuseIntVarID) {
        #BEGIN_SCOPE
        int intVar5 = 456;
        @ReuseIntVarID = @ReuseIntVarID;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```

Scope 1
intMethod1
intVar1 intVar2
intVar3 intVar4

Шаг 2: области видимости

```
#BEGIN_SCOPE
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int intVar4 = 123;
    @ReuseIntVarID = @ReuseIntVarID * @ReuseIntVarID + @ReuseIntVarID;
    if (@ReuseIntVarID + @ReuseIntVarID > @ReuseIntVarID + @ReuseIntVarID) {
        #BEGIN_SCOPE
        int intVar5 = 456;
        @ReuseIntVarID = @ReuseIntVarID;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```

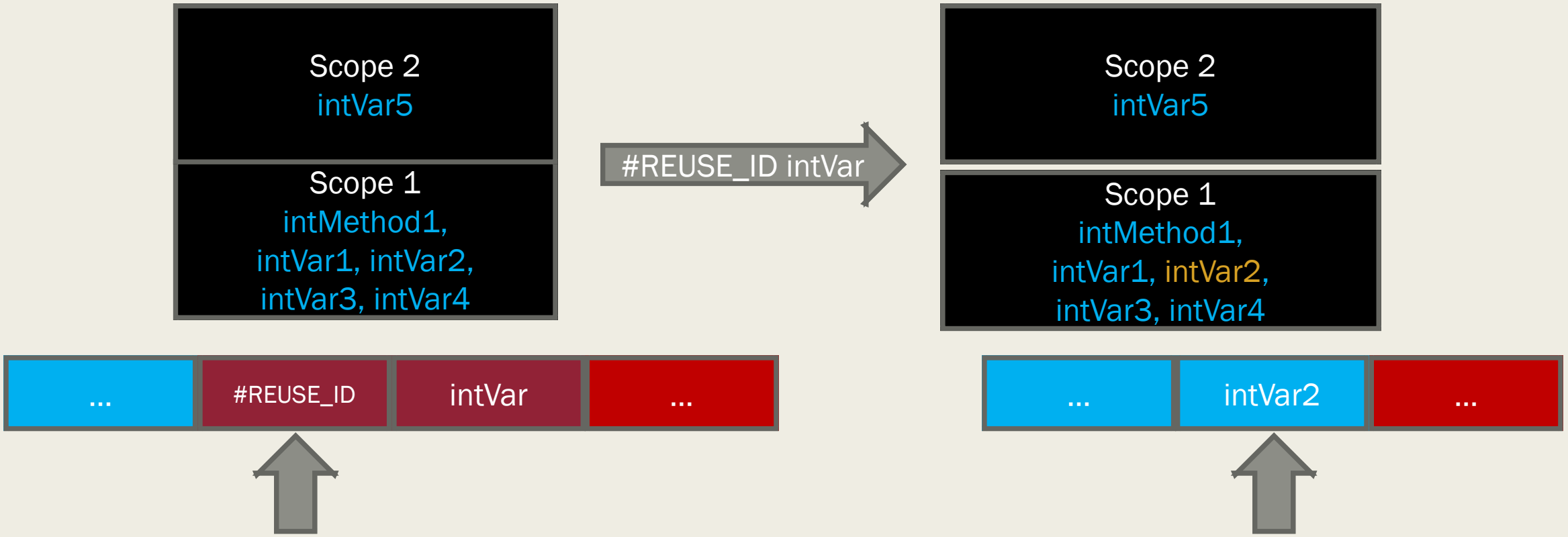
Шаг 3: используем идентификаторы

```
@ReuseIntVarID
      #BEGIN_RULE
            #REUSE_ID intVar
      #END
```

Шаг 3: используем идентификаторы

```
@ReuseMethodID → #REUSE_ID intMethod
@ReuseIntVarID → #REUSE_ID intVar
```

Шаг 3: используем идентификаторы



Шаг 3: используем идентификаторы

Шаг 3: используем идентификаторы

```
#BEGIN_SCOPE
int @IntMethodID(int @IntVarID, int @IntVarID, int @IntVarID) {
    int @IntVarID = 123;
    @ReuseIntVarID = @ReuseIntVarID * @ReuseIntVarID + @ReuseIntVarID;
    if (@ReuseIntVarID + @ReuseIntVarID > @ReuseIntVarID + @ReuseIntVarID) {
        #BEGIN_SCOPE
        int @IntVarID = 456;
        @ReuseIntVarID = @ReuseIntVarID;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```

Шаг 3: используем идентификаторы

```
#BEGIN_SCOPE
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int @IntVarID = 123;
    @ReuseIntVarID = @ReuseIntVarID * @ReuseIntVarID + @ReuseIntVarID;
    if (@ReuseIntVarID + @ReuseIntVarID > @ReuseIntVarID + @ReuseIntVarID) {
        #BEGIN_SCOPE
        int @IntVarID = 456;
        @ReuseIntVarID = @ReuseIntVarID;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```

Scope 1
intMethod1
intVar1 intVar2
intVar3

Шаг 3: используем идентификаторы

```
#BEGIN_SCOPE
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int intVar4 = 123;
    @ReuseIntVarID = @ReuseIntVarID * @ReuseIntVarID + @ReuseIntVarID;
    if (@ReuseIntVarID + @ReuseIntVarID > @ReuseIntVarID + @ReuseIntVarID) {
        #BEGIN_SCOPE
        int @IntVarID = 456;
        @ReuseIntVarID = @ReuseIntVarID;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```

Scope 1
intMethod1
intVar1 intVar2
intVar3 intVar4

Шаг 3: используем идентификаторы

```
#BEGIN_SCOPE
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int intVar4 = 123;
    @ReuseIntVarID = @ReuseIntVarID * @ReuseIntVarID + @ReuseIntVarID;
    if (@ReuseIntVarID + @ReuseIntVarID > @ReuseIntVarID + @ReuseIntVarID) {
        #BEGIN_SCOPE
        int @IntVarID = 456;
        @ReuseIntVarID = @ReuseIntVarID;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```

Scope 1
intMethod1
intVar1 intVar2
intVar3 intVar4

Шаг 3: используем идентификаторы

```
#BEGIN_SCOPE
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int intVar4 = 123;
    intVar2 = @ReuseIntVarID * @ReuseIntVarID + @ReuseIntVarID;
    if (@ReuseIntVarID + @ReuseIntVarID > @ReuseIntVarID + @ReuseIntVarID) {
        #BEGIN_SCOPE
        int @IntVarID = 456;
        @ReuseIntVarID = @ReuseIntVarID;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```

Scope 1
intMethod1
intVar1 intVar2
intVar3 intVar4

Шаг 3: используем идентификаторы

```
#BEGIN_SCOPE
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int intVar4 = 123;
    intVar2 = intVar4 * @ReuseIntVarID + @ReuseIntVarID;
    if (@ReuseIntVarID + @ReuseIntVarID > @ReuseIntVarID + @ReuseIntVarID) {
        #BEGIN_SCOPE
        int @IntVarID = 456;
        @ReuseIntVarID = @ReuseIntVarID;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```

Scope 1
intMethod1
intVar1 intVar2
intVar3 intVar4

Шаг 3: используем идентификаторы

```
#BEGIN_SCOPE
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int intVar4 = 123;
    intVar2 = intVar4 * intVar1 + @ReuseIntVarID;
    if (@ReuseIntVarID + @ReuseIntVarID > @ReuseIntVarID + @ReuseIntVarID) {
        #BEGIN_SCOPE
        int @IntVarID = 456;
        @ReuseIntVarID = @ReuseIntVarID;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```

Scope 1
intMethod1
intVar1 intVar2
intVar3 intVar4

Шаг 3: используем идентификаторы

```
#BEGIN_SCOPE
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int intVar4 = 123;
    intVar2 = intVar4 * intVar1 + intVar3;
    if (@ReuseIntVarID + @ReuseIntVarID > @ReuseIntVarID + @ReuseIntVarID) {
        #BEGIN_SCOPE
        int @IntVarID = 456;
        @ReuseIntVarID = @ReuseIntVarID;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```

Scope 1
intMethod1
intVar1 intVar2
intVar3 intVar4

Шаг 3: используем идентификаторы

И так далее...

Шаг 3: используем идентификаторы

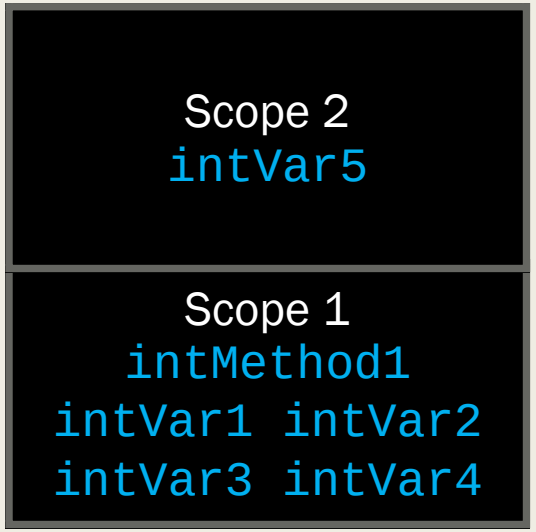
```
#BEGIN_SCOPE
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int intVar4 = 123;
    intVar2 = intVar4 * intVar1 + intVar3;
    if (intVar4 + intVar1 > intVar3 + intVar2) {
        #BEGIN_SCOPE
        int intVar5 = 456;
        @ReuseIntVarID = @ReuseIntVarID;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```

Шаг 3: используем идентификаторы

```
#BEGIN_SCOPE
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int intVar4 = 123;
    intVar2 = intVar4 * intVar1 + intVar3;
    if (intVar4 + intVar1 > intVar3 + intVar2) {
        #BEGIN_SCOPE
        int intVar5 = 456;
        intVar1 = @ReuseIntVarID;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```

Шаг 3: используем идентификаторы

```
#BEGIN_SCOPE
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int intVar4 = 123;
    intVar2 = intVar4 * intVar1 + intVar3;
    if (intVar4 + intVar1 > intVar3 + intVar2) {
        #BEGIN_SCOPE
        int intVar5 = 456;
        intVar1 = intVar5;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```



Шаг 3: используем идентификаторы

```
#BEGIN_SCOPE
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int intVar4 = 123;
    intVar2 = intVar4 * intVar1 + intVar3;
    if (intVar4 + intVar1 > intVar3 + intVar2) {
        #BEGIN_SCOPE
        int intVar5 = 456;
        intVar1 = intVar5;
        #END_SCOPE
    }
    return @ReuseIntVarID + @ReuseIntVarID;
}
#END_SCOPE
```

Scope 1
intMethod1
intVar1 intVar2
intVar3 intVar4

Шаг 3: используем идентификаторы

```
#BEGIN_SCOPE
int intMethod1(int intVar1, int intVar2, int intVar3) {
    int intVar4 = 123;
    intVar2 = intVar4 * intVar1 + intVar3;
    if (intVar4 + intVar1 > intVar3 + intVar2) {
        #BEGIN_SCOPE
        int intVar5 = 456;
        intVar1 = intVar5;
        #END_SCOPE
    }
    return intVar1 + intVar2;
}
#END_SCOPE
```

Шаг 3: используем идентификаторы

```
for (int I = 0; I < N; I++) {
  …
}
```

Шаг 3: используем идентификаторы

```
#BEGIN_SCOPE
for (int #CREATE_ID intCnt = 0; #REUSE_ID intCnt < N; #REUSE_ID intCnt ++) {
  …
}
#END_SCOPE
```

Шаг 3: используем идентификаторы

```
// Когда это единственная переменная с префиксом intCnt,
// всё работает.
#BEGIN_SCOPE
for (int intCnt1 = 0; #REUSE_ID intCnt < N; #REUSE_ID intCnt ++) {
  …
}
#END_SCOPE
```

Шаг 3: используем идентификаторы

```
// Когда это единственная переменная с префиксом intCnt,
// всё работает.
#BEGIN_SCOPE
for (int intCnt1 = 0; intCnt1 < N; #REUSE_ID intCnt ++) {
  …
}
#END_SCOPE
```

Scope 1
intCnt1

Шаг 3: используем идентификаторы

```
// Когда это единственная переменная с префиксом intCnt,
// всё работает.
#BEGIN_SCOPE
for (int intCnt1 = 0; intCnt1 < N; intCnt1 ++) {
  …
}
#END_SCOPE
```

Scope 1
intCnt1

Шаг 3: используем идентификаторы

```
// Проблема: вложенный цикл
#BEGIN_SCOPE
for (int #CREATE_ID intCnt = 0; #REUSE_ID intCnt < N; #REUSE_ID intCnt ++) {
  #BEGIN_SCOPE
  for (int #CREATE_ID intCnt = 0; #REUSE_ID intCnt < N; #REUSE_ID intCnt ++) {
    …
  }
  #END_SCOPE
}
#END_SCOPE
```

Шаг 3: используем идентификаторы

```
// Проблема: вложенный цикл
#BEGIN_SCOPE
for (int #CREATE_ID intCnt = 0; #REUSE_ID intCnt < N; #REUSE_ID intCnt ++) {
  #BEGIN_SCOPE
  for (int #CREATE_ID intCnt = 0; #REUSE_ID intCnt < N; #REUSE_ID intCnt ++) {
    …
  }
  #END_SCOPE
}
#END_SCOPE
```

Scope 1

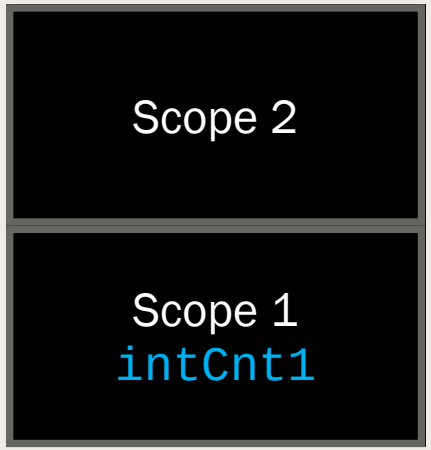
Шаг 3: используем идентификаторы

```
// Проблема: вложенный цикл
#BEGIN_SCOPE
for (int intCnt1 = 0; intCnt1 < N; intCnt1 ++) {
  #BEGIN_SCOPE
  for (int #CREATE_ID intCnt = 0; #REUSE_ID intCnt < N; #REUSE_ID intCnt ++) {
    …
  }
  #END_SCOPE
}
#END_SCOPE
```

Scope 1
intCnt1

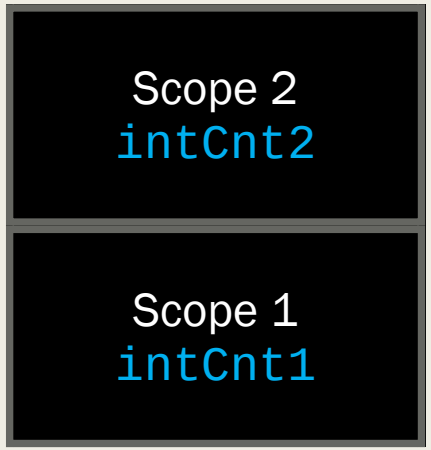
Шаг 3: используем идентификаторы

```
// Проблема: вложенный цикл
#BEGIN_SCOPE
for (int intCnt1 = 0; intCnt1 < N; intCnt1 ++) {
  #BEGIN_SCOPE
  for (int #CREATE_ID intCnt = 0; #REUSE_ID intCnt < N; #REUSE_ID intCnt ++) {
    …
  }
  #END_SCOPE
}
#END_SCOPE
```



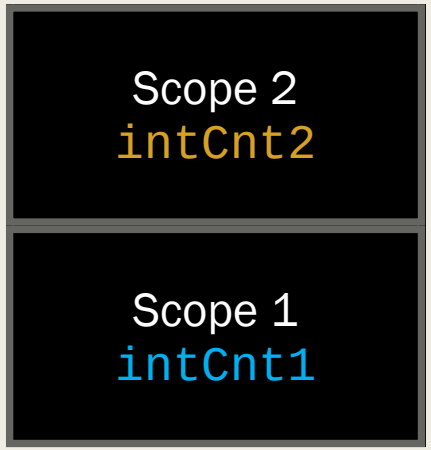
Шаг 3: используем идентификаторы

```
// Проблема: вложенный цикл
#BEGIN_SCOPE
for (int intCnt1 = 0; intCnt1 < N; intCnt1 ++) {
  #BEGIN_SCOPE
  for (int intCnt2 = 0; #REUSE_ID intCnt < N; #REUSE_ID intCnt ++) {
    …
  }
  #END_SCOPE
}
#END_SCOPE
```



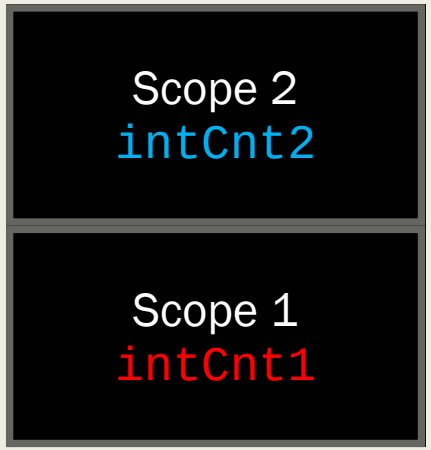
Шаг 3: используем идентификаторы

```
// Проблема: вложенный цикл
#BEGIN_SCOPE
for (int intCnt1 = 0; intCnt1 < N; intCnt1 ++) {
  #BEGIN_SCOPE
  for (int intCnt2 = 0; #REUSE_ID intCnt < N; #REUSE_ID intCnt ++) {
    …
  }
  #END_SCOPE
}
#END_SCOPE
```



Шаг 3: используем идентификаторы

```
// Проблема: вложенный цикл
#BEGIN_SCOPE
for (int intCnt1 = 0; intCnt1 < N; intCnt1 ++) {
  #BEGIN_SCOPE
  for (int intCnt2 = 0; intCnt2 < N; #REUSE_ID intCnt ++) {
    …
  }
  #END_SCOPE
}
#END_SCOPE
```



Шаг 3: используем существующие идентификаторы

Шаг 3: используем идентификаторы

```
#BEGIN_SCOPE
for (int #CREATE_ID intCnt = 0; #REUSE_LOCAL_ID intCnt < N; #REUSE_LOCAL_ID intCnt ++) {
  …
}
#END_SCOPE
```

Шаг 4: вероятности выбора правил

@IntFunction → int @ID(@Params) { @IntFunctionBody }

@ Params → @Param @MoreParams| ε

@MoreParams → , @Param @MoreParams| ε

@Param → @Type @Id

@IntFunctionBody → @Statements @IntReturn

@Statements → @Statement @Statements | ε

@Statement → @DeclareVar | @Loop | @If | …

@IntReturn → return @IntExpression ;

И т.д.

Шаг 4: вероятности выбора правил

```
@STATEMENTS
      #BEGIN_RULE:6
            @STATEMENT, @STATEMENTS
      #END_RULE
      #BEGIN_RULE:1
      #END_RULE
```

Вероятность 1 правила - $\frac{6}{7}$, второго - $\frac{1}{7}$

Шаг 5: переменные фаззинга

```
@DECLARE_VAR
      #BEGIN_RULE:1 #CREATE_ID floatVar #END_RULE
      #BEGIN_RULE:7 #CREATE_ID intVar #END_RULE
@REUSE_VAR
      #BEGIN_RULE:1 #REUSE_ID floatVar #END_RULE
      #BEGIN_RULE:7 #REUSE_ID intVar #END_RULE
```

Шаг 5: переменные фаззинга

```
@DECLARE_VAR
      #BEGIN_RULE:1 #CREATE_ID floatVar #END_RULE
      #BEGIN_RULE:9 #CREATE_ID intVar #END_RULE
@REUSE_VAR
      #BEGIN_RULE:1 #REUSE_ID floatVar #END_RULE
      #BEGIN_RULE:9 #REUSE_ID intVar #END_RULE
```

Шаг 5: переменные фаззинга

```
#SET FLOAT_PROB=1
#SET INT_PROB=7
@DECLARE_VAR
      #BEGIN_RULE:FLOAT_PROB #CREATE_ID floatVar #END_RULE
      #BEGIN_RULE:INT_PROB #CREATE_ID intVar #END_RULE
@REUSE_VAR
      #BEGIN_RULE:FLOAT_PROB #REUSE_ID floatVar #END_RULE
      #BEGIN_RULE:INT_PROB #REUSE_ID intVar #END_RULE
```

Шаг 5: переменные фаззинга

```
#SET FLOAT_PROB=1
#SET INT_PROB=9
@DECLARE_VAR
      #BEGIN_RULE:FLOAT_PROB #CREATE_ID floatVar #END_RULE
      #BEGIN_RULE:INT_PROB #CREATE_ID intVar #END_RULE
@REUSE_VAR
      #BEGIN_RULE:FLOAT_PROB #REUSE_ID floatVar #END_RULE
      #BEGIN_RULE:INT_PROB #REUSE_ID intVar #END_RULE
```

Преимущества подхода

- Можно иметь один Fuzzer для множества языков
- Можно применять не только для языков программирования
  - Возможно применять для XML, SQL, различных текстовых форматов
- Добавление языка – всего лишь добавление файла с грамматикой
- Вносить изменения в грамматики проще, чем в код

Недостатки подхода

- Грамматики получаются довольно громоздкими
  - Можно сократить, используя макросы
  - Грамматики можно генерировать
- Со сложными типами всё сложно
- Не каждый язык можно описать таким образом
- Неудачный подбор весов создаёт много «мёртвого» кода


Java Fuzzer
Универсальный
Fuzzer

Спасибо за внимание!

Макс Казанцев
max.kazantsev@azul.com