

tech.skbbkontur.ru

bit.ly/2wQZF1u

IN PRODUCTION

Как писать код, который пишет код на MSIL

Евгений Юрьев, СКБ Контур

План

- Задача
- Способы решения
- Сферы применения кодогенерации

Контекст

- Контур.Экстерн
- Более 1 000 разных отчетов (Xml)
- Проверка, редактирование,
сбор свойств

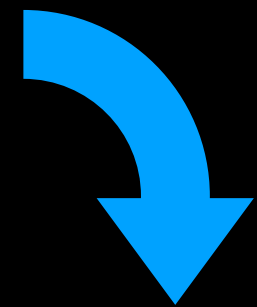
Корректное имя файла отчета

Корректное имя файла отчета



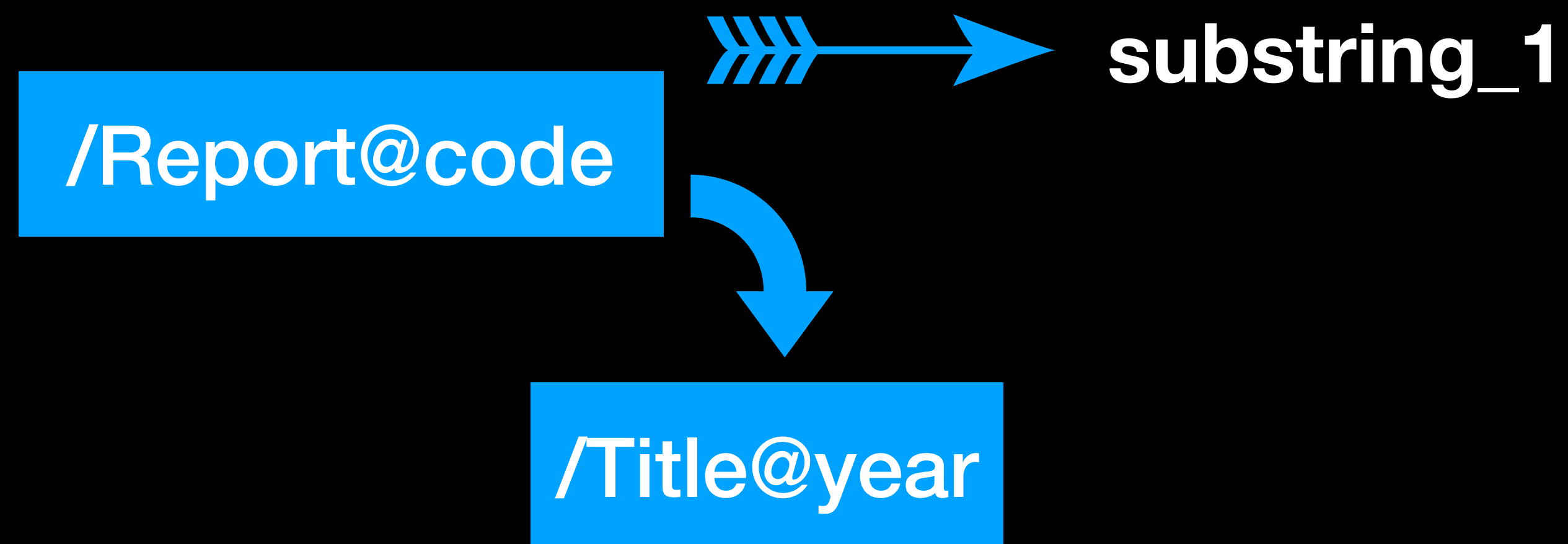
Корректное имя файла отчета

/Report@code

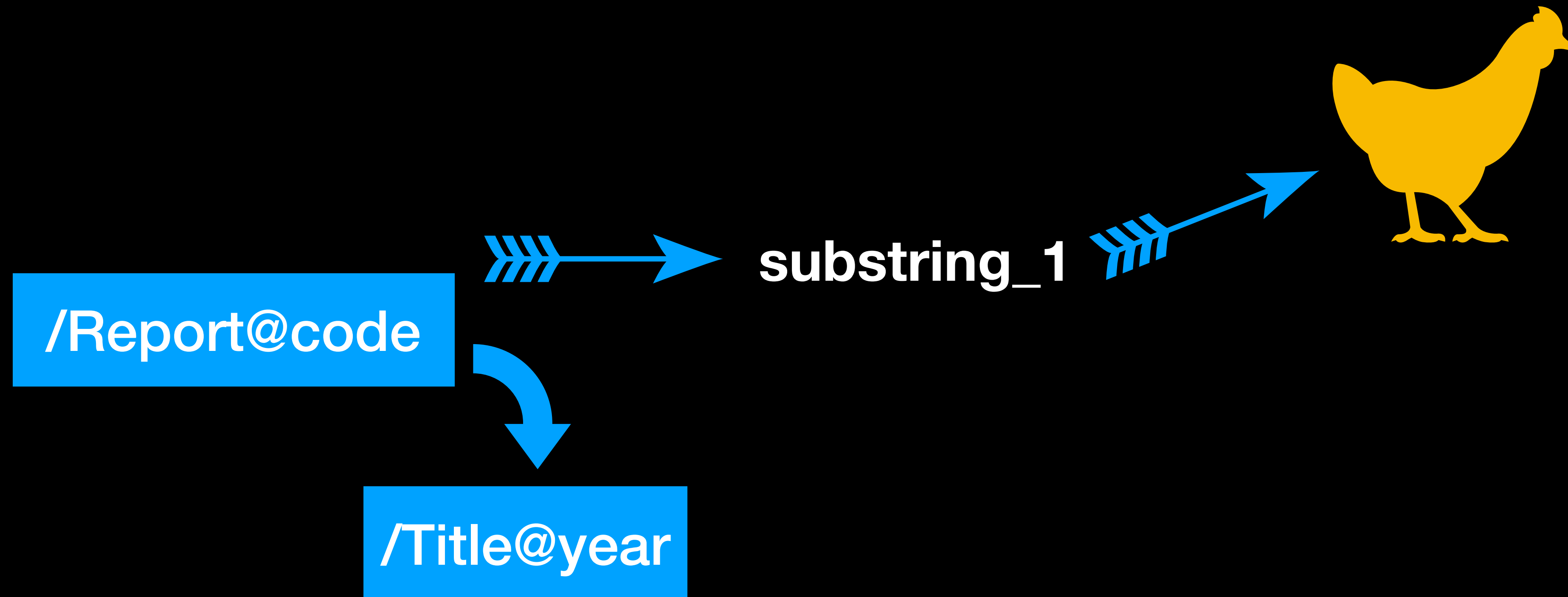


/Title@year

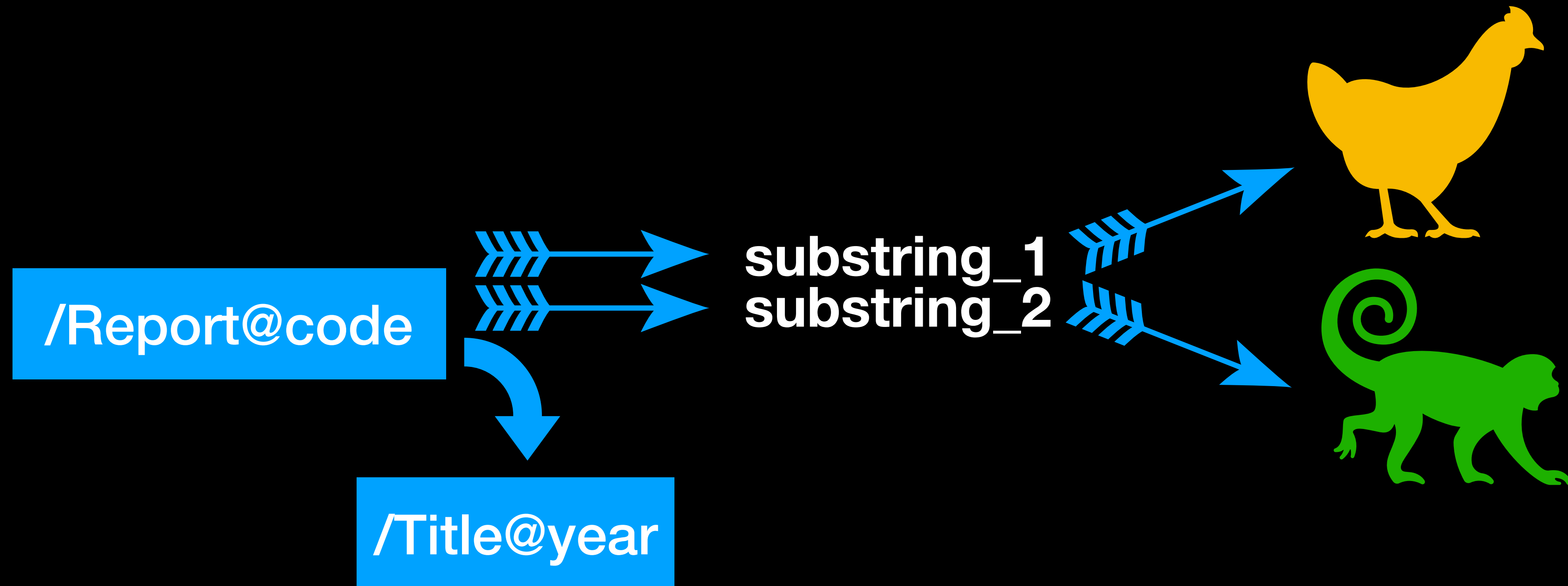
Корректное имя файла отчета



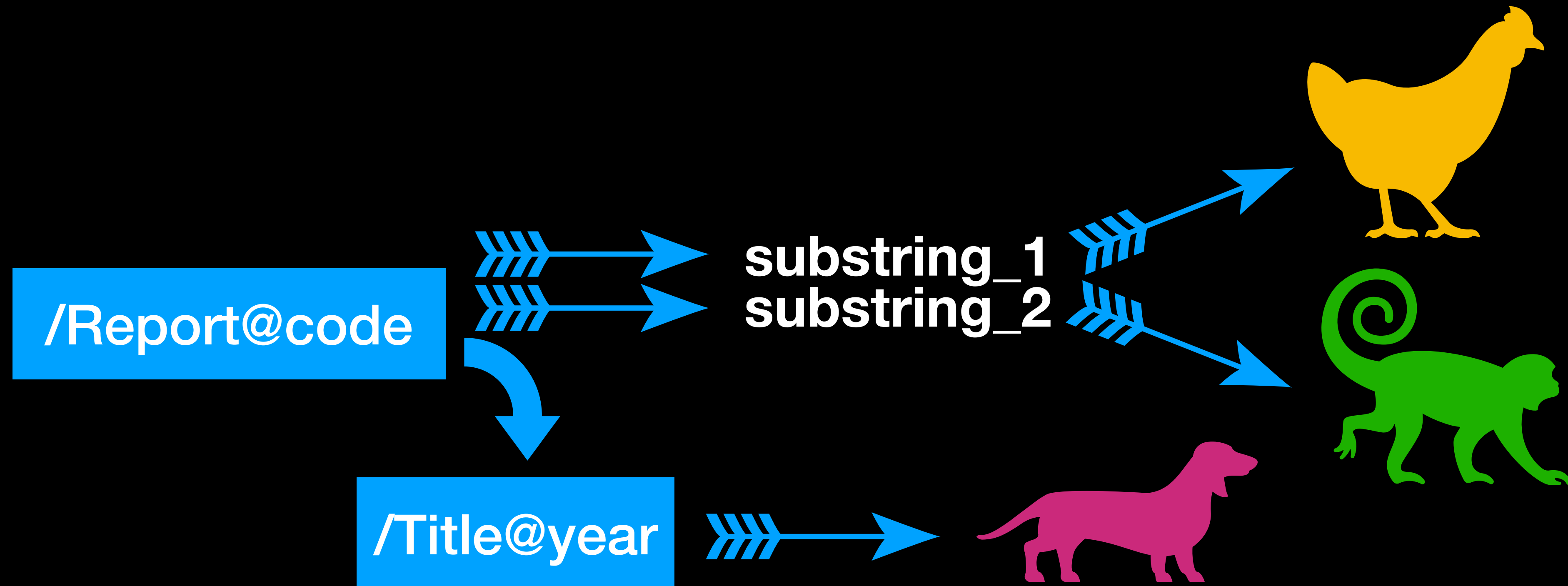
Корректное имя файла отчета



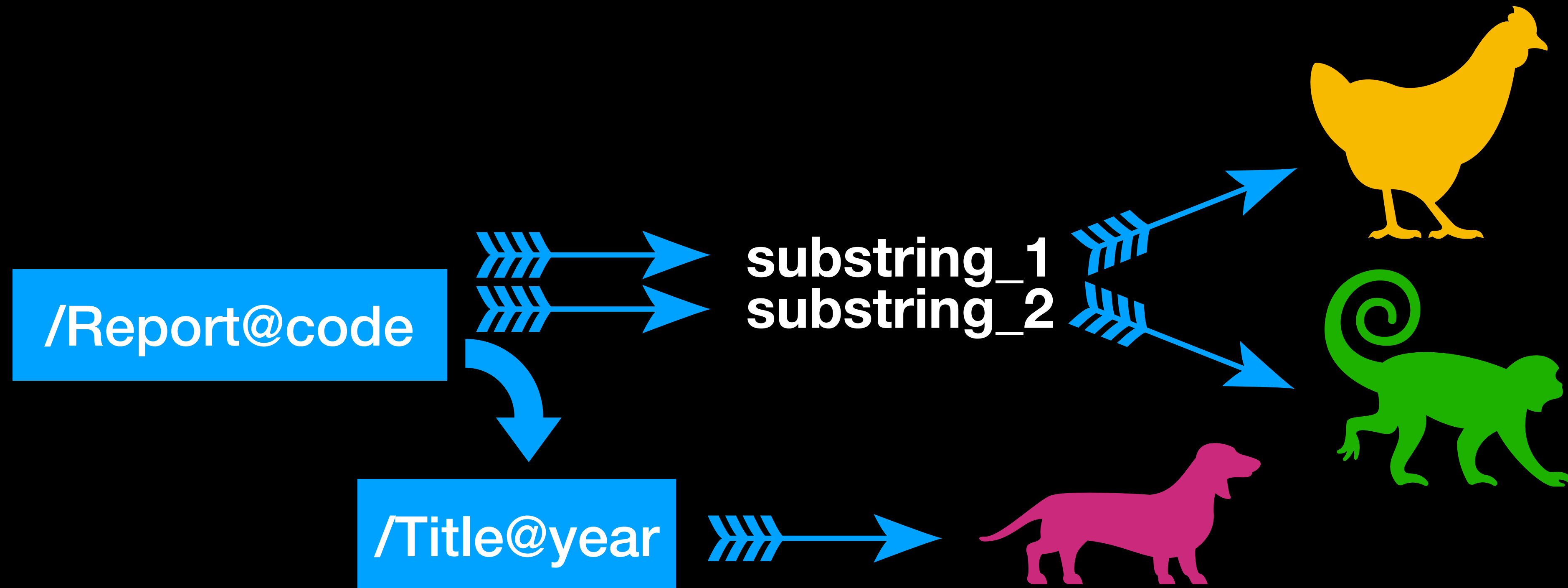
Корректное имя файла отчета



Корректное имя файла отчета



Корректное имя файла отчета



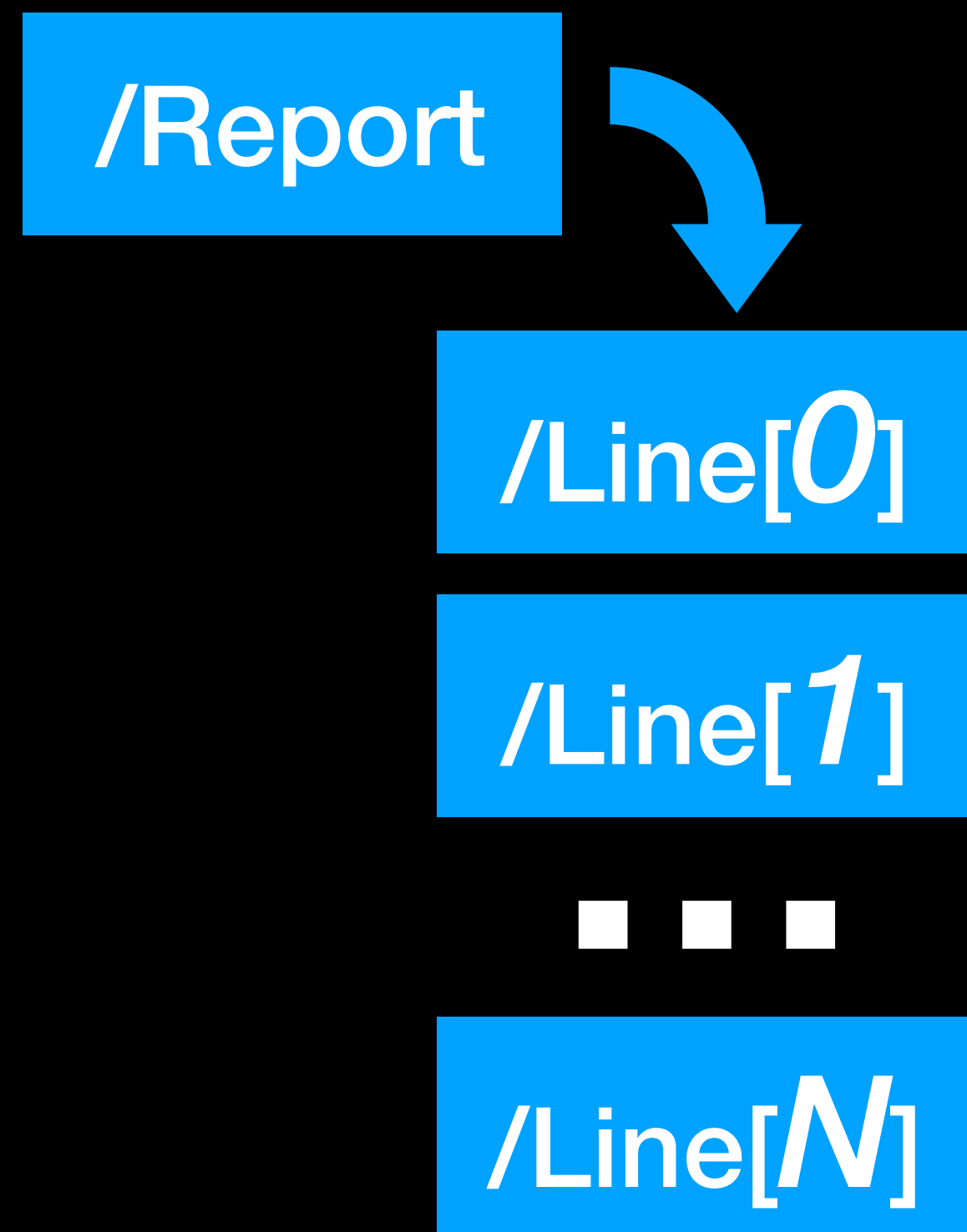
`$"{{chicken}}_{{dog}}_{{monkey}}.xml"`

Корректное имя файла отчета

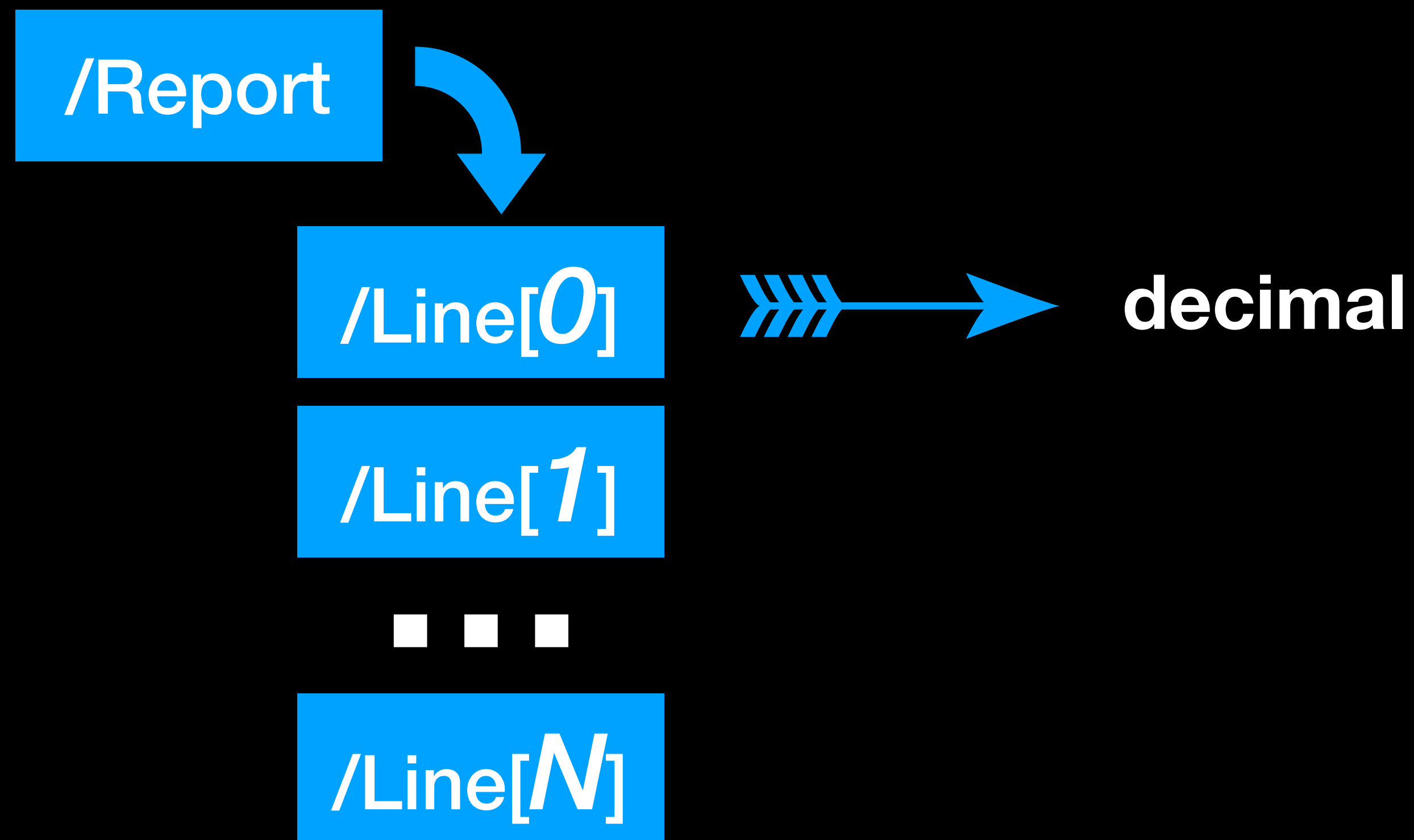
"0612008_002_001__2015_17_
201709041433
b9f38c93-afd6-413f-9b62-
f42930fa3fee.xml"

Сумма вычета по НДС

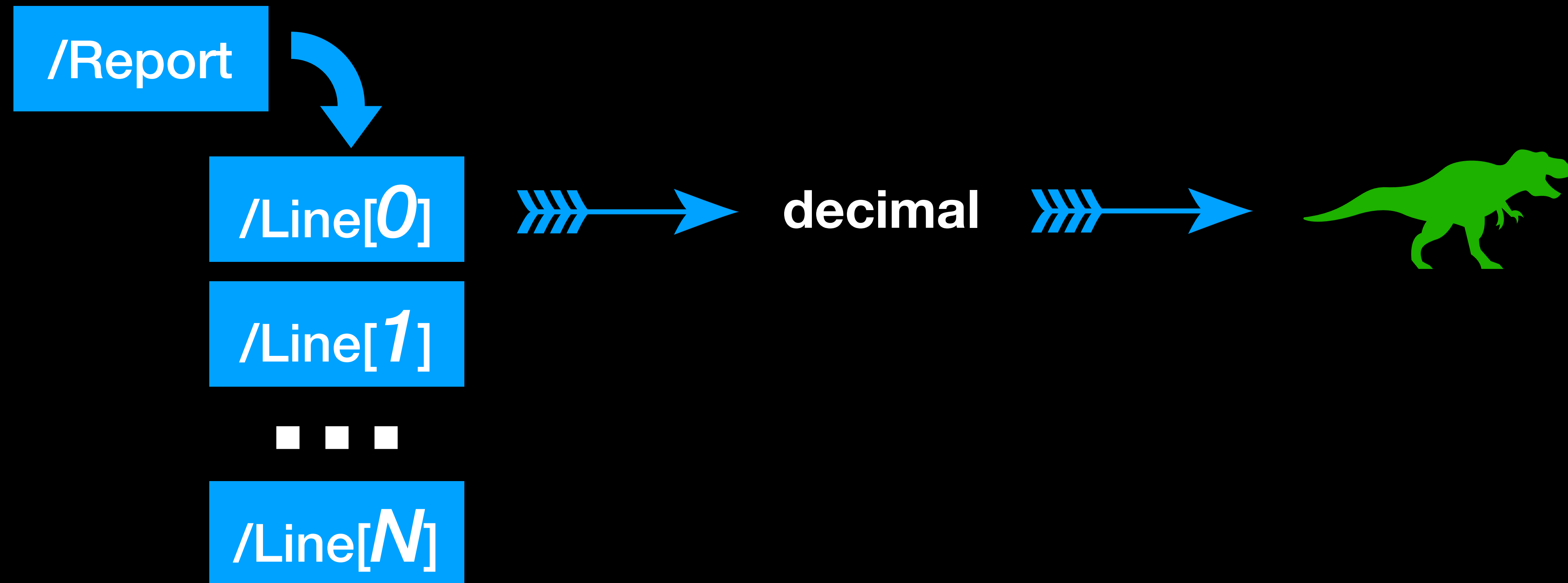
Сумма вычета по НДС



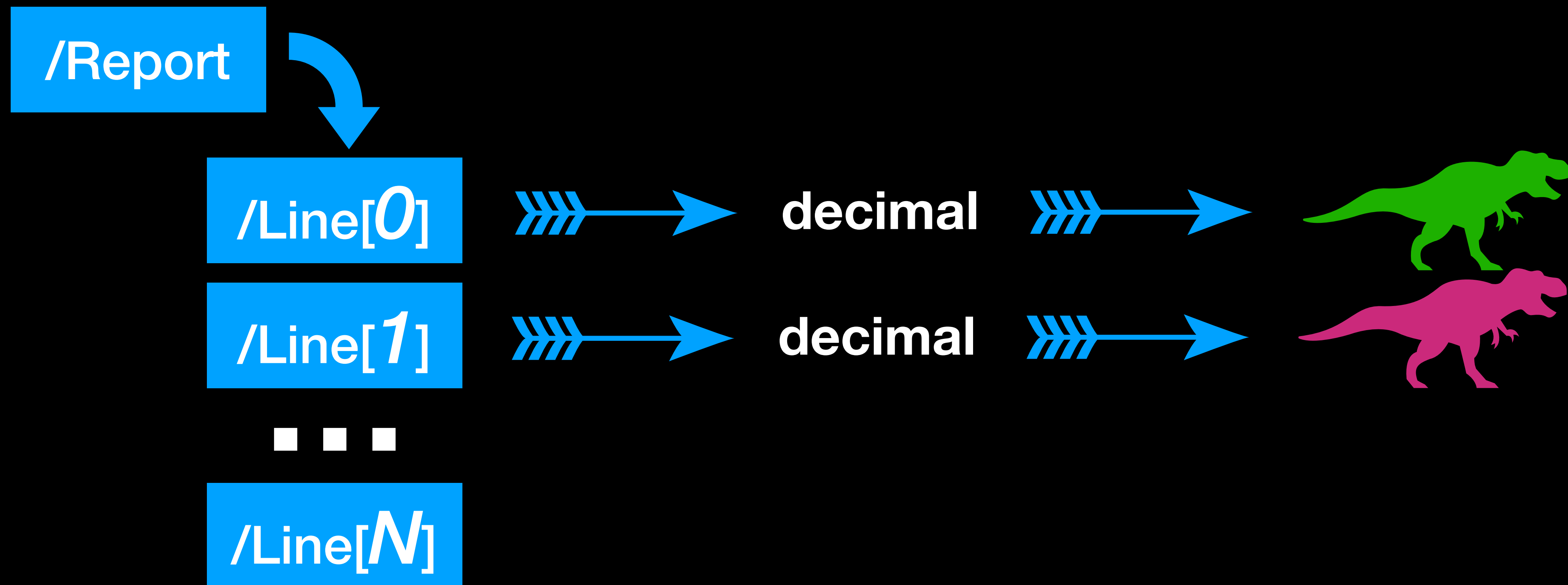
Сумма вычета по НДС



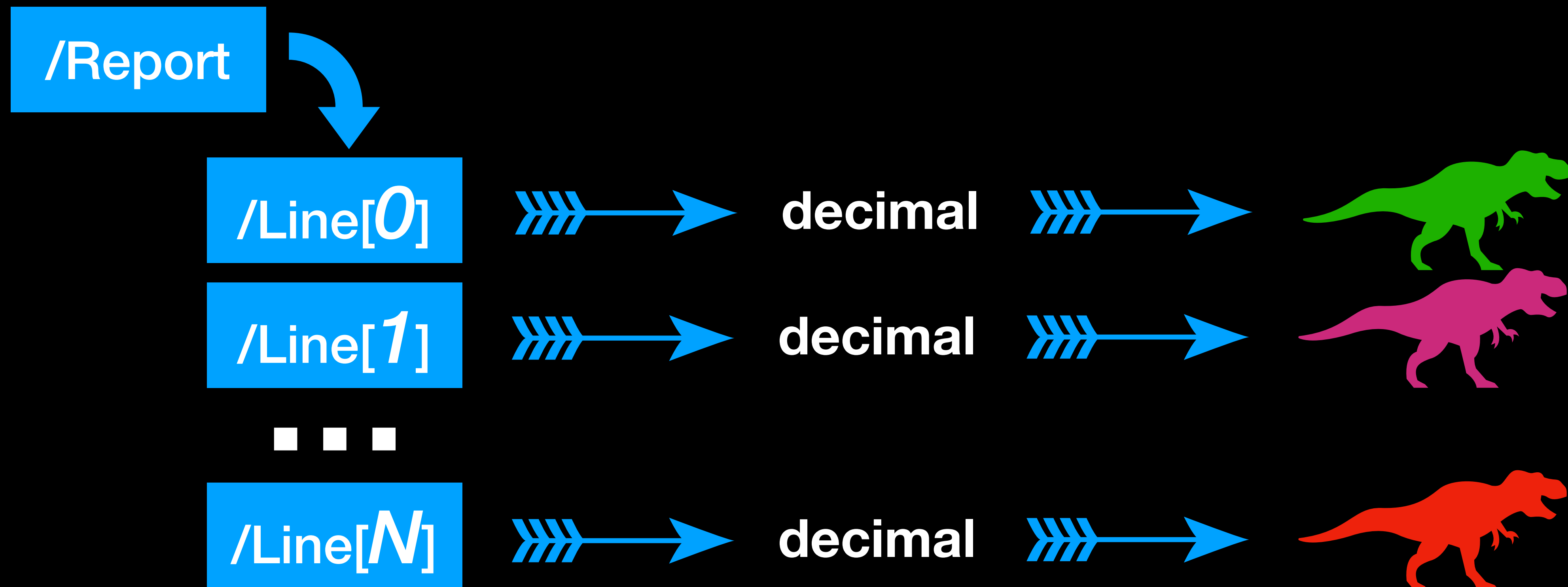
Сумма вычета по НДС



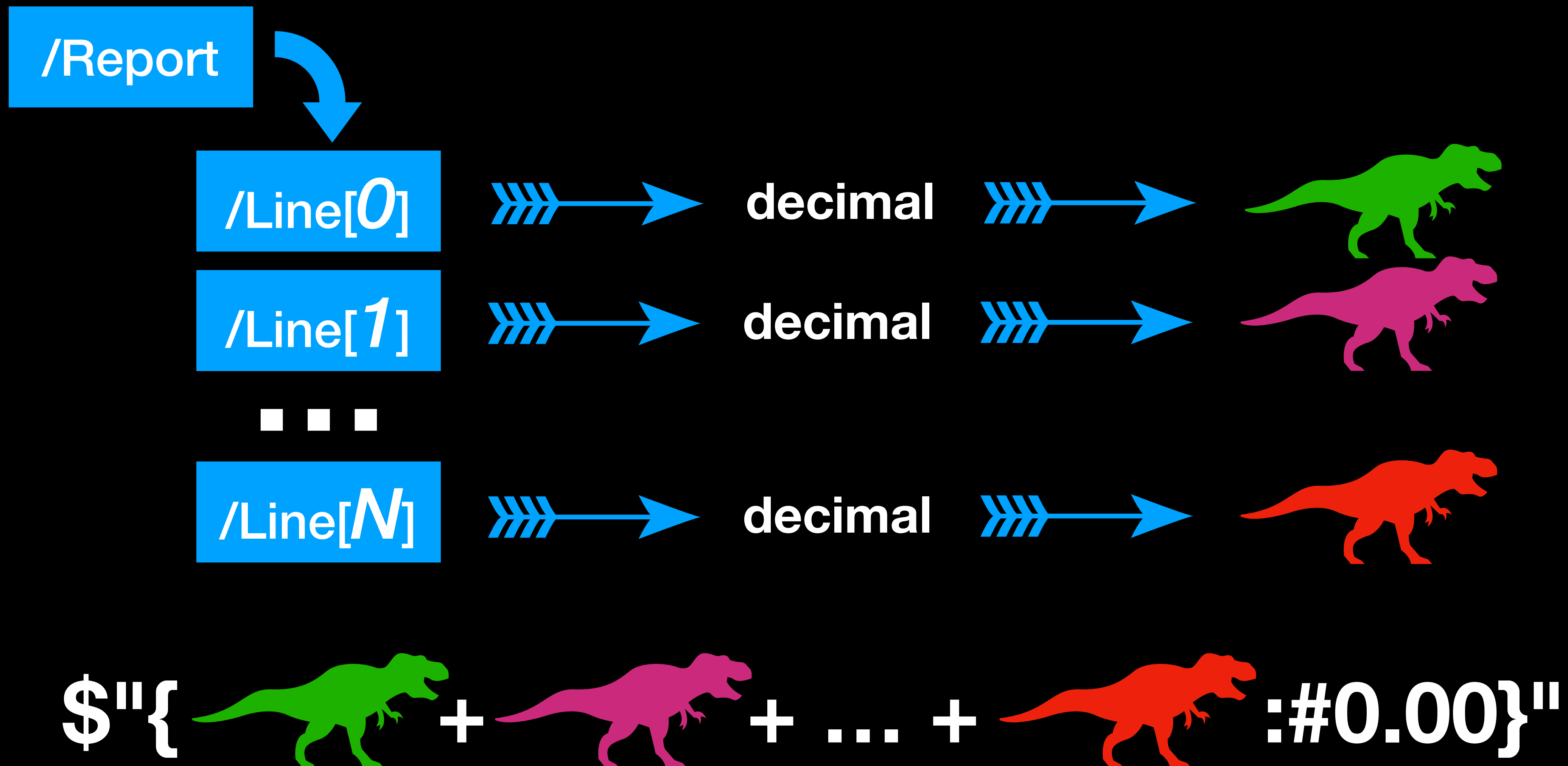
Сумма вычета по НДС



Сумма вычета по НДС



Сумма вычета по НДС



Сумма вычета по НДС

"17534871201.04"

Требования и ограничения

Требования и ограничения

- Большие отчеты (up to 10 Gb)

Требования и ограничения

- Большие отчеты (up to 10 Gb)
- Поточковая обработка

Требования и ограничения

- Большие отчеты (up to 10 Gb)
- Поточковая обработка
- Умение автоматически останавливать чтение как можно раньше

Требования и ограничения

- Большие отчеты (up to 10 Gb)
- Поточковая обработка
- Умение автоматически останавливать чтение как можно раньше
- Декларативное описание инструкций

Какие операции нужны

- Работа со строками (длина, подстрока, форматирование)
- Преобразование типов (string, date, decimal, bool)
- Арифметические операции
- Логические операции
- If - Then - Else
- Обращения к внешнему коду
- Агрегации, фильтры

Legacy решение

```
var value = reader.Read("parent/title@period");
```

Legacy решение

```
class Reader
{
    → private readonly Stream xmlStream;
    → private Dictionary<string, string> pathToValue;

    1 reference
    → public Reader(Stream xmlStream)
    → {
    →     → this.xmlStream = xmlStream;
    → }
}
```

Legacy решение

```
public string Read(string path)
{
    → while (!endOfStream && !pathToValue.ContainsKey(path))
    →     → ReadNextToken(); //fill dictionary, concat strings


    → return pathToValue.ContainsKey(path)
    →     → pathToValue[path]
    →     → null;
}
```

Новое решение

- Dictionary -> fields

Новое решение


- Dictionary -> fields

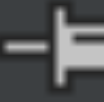


Run  Benchmarks

Method	Mean	Error	StdDev
Dict	36.6184 ns	0.7821 ns	1.0705 ns
Field	0.3908 ns	0.0725 ns	0.0678 ns

Новое решение

- Dictionary -> fields

Run  Benchmarks



Method	Mean	Error	StdDev
-----	-----:	-----:	-----:
Dict	36.6184 ns	0.7821 ns	1.0705 ns
Field	0.3908 ns	0.0725 ns	0.0678 ns

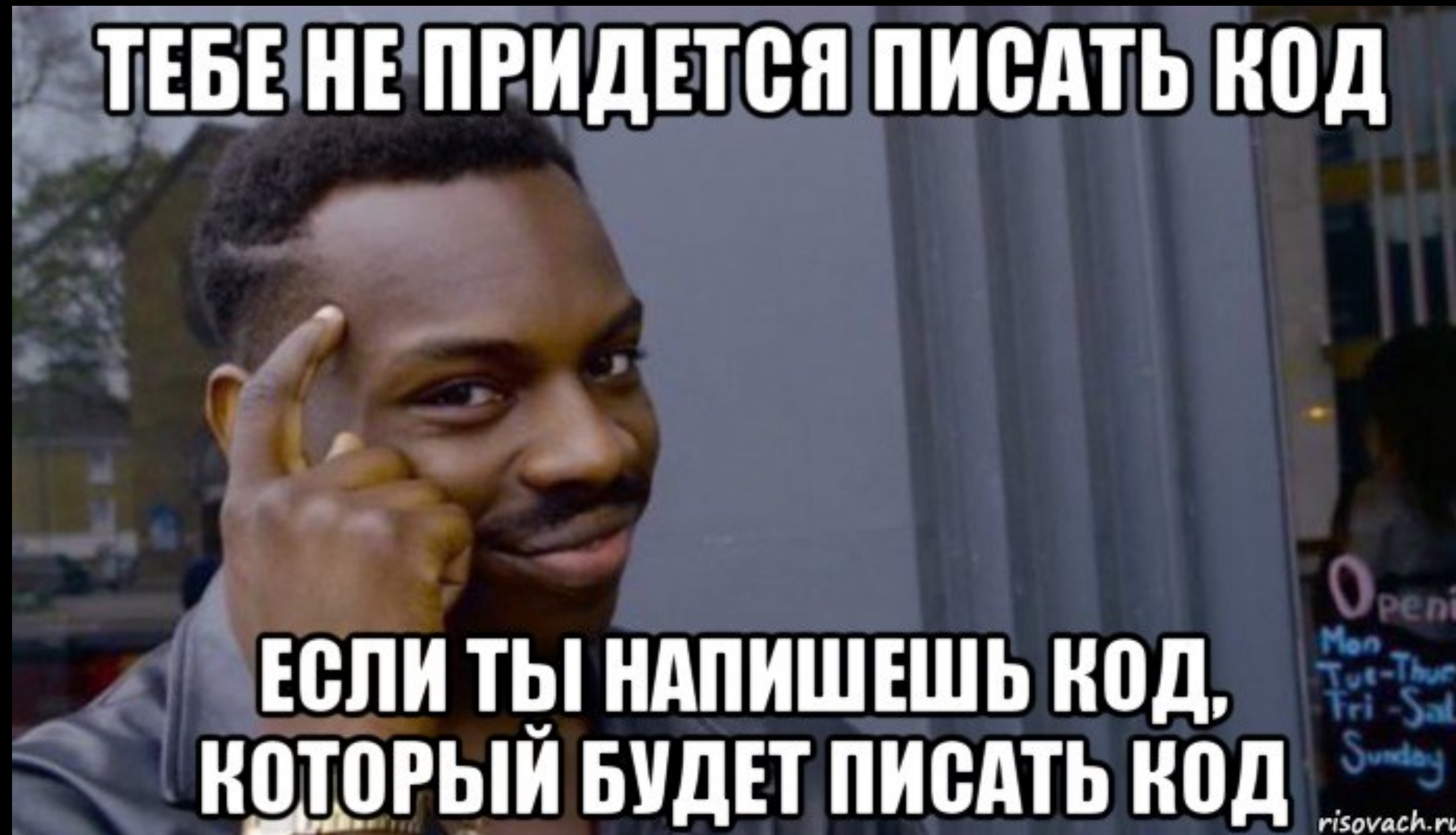
- string.Concat -> traverse tree

Новое решение

- C# -> DSL

Новое решение

- C# -> DSL



Почему DSL

Почему DSL

- Интересно!

Почему DSL

- Интересно!
- Низкий порог входа (для потребителей)

Почему DSL

- Интересно!
- Низкий порог входа (для потребителей)
- Безопасность

Почему DSL

- Интересно!
- Низкий порог входа (для потребителей)
- Безопасность
- Декларативность

Почему DSL

- Интересно!
- Низкий порог входа (для потребителей)
- Безопасность
- Декларативность
- Эффективность

Как всё устроено

Как всё устроено

Rules.Xml

Как всё устроено

Compile

Rules.Xml



IL-code

Как всё устроено

Compile

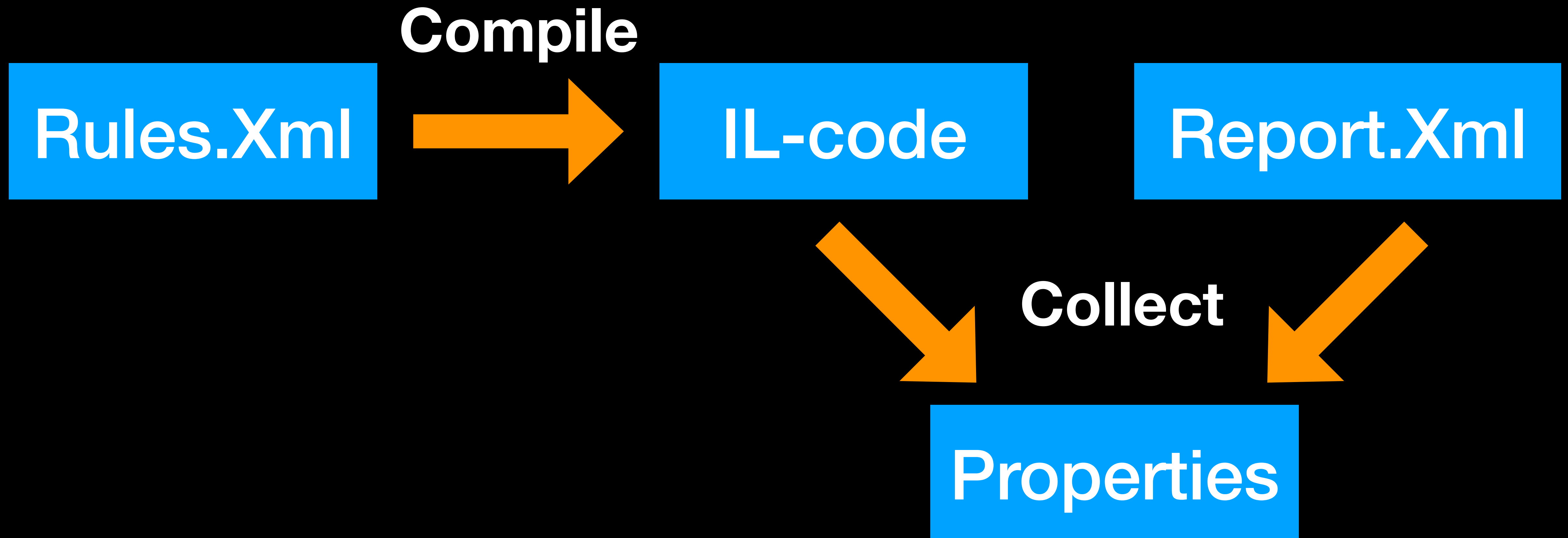
Rules.Xml



IL-code

Report.Xml

Как всё устроено



Возможности языка. *Variables*

Возможности языка. Variables

<variable

name="Foo"

path="/report@code"

/>

Возможности языка. Properties

Возможности языка. Properties

```
<property name="Bar">
```

```
  <sum>
```

```
    <var name="Foo" type="decimal">
```

```
      <const value="10" type="decimal">
```

```
    </sum>
```

```
</property>
```

Основная идея. Собираем

Основная идея. Собираем

/root/a/b@attr1

/root/a/b@attr2

/root/a/c@attr3

/root/d@attr4

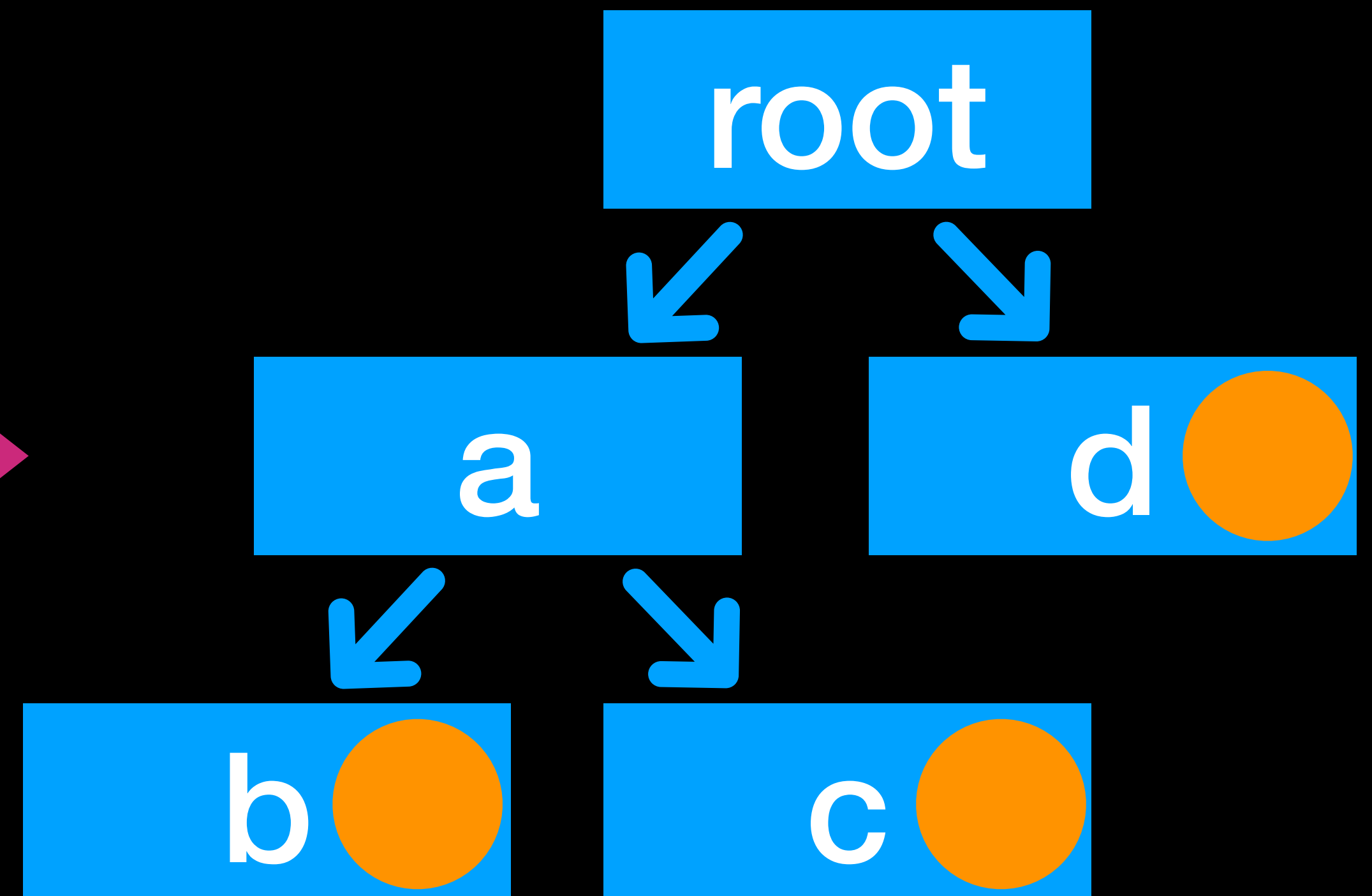
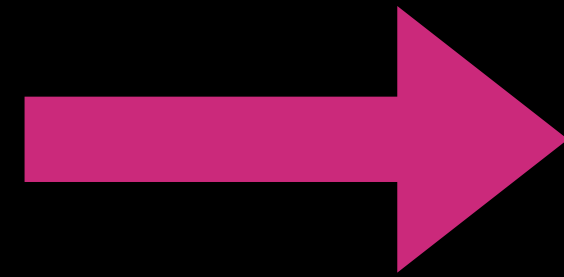
Основная идея. Собираем

/root/a/b@attr1

/root/a/b@attr2

/root/a/c@attr3

/root/d@attr4



Основная идея. Собираем



Основная идея. Вычисляем

```
<sum>
```

```
  <var name="attr1">
```

```
  <minus>
```

```
    <var name="attr2">
```

```
  </minus>
```

```
</sum>
```

Основная идея. Вычисляем

<sum>

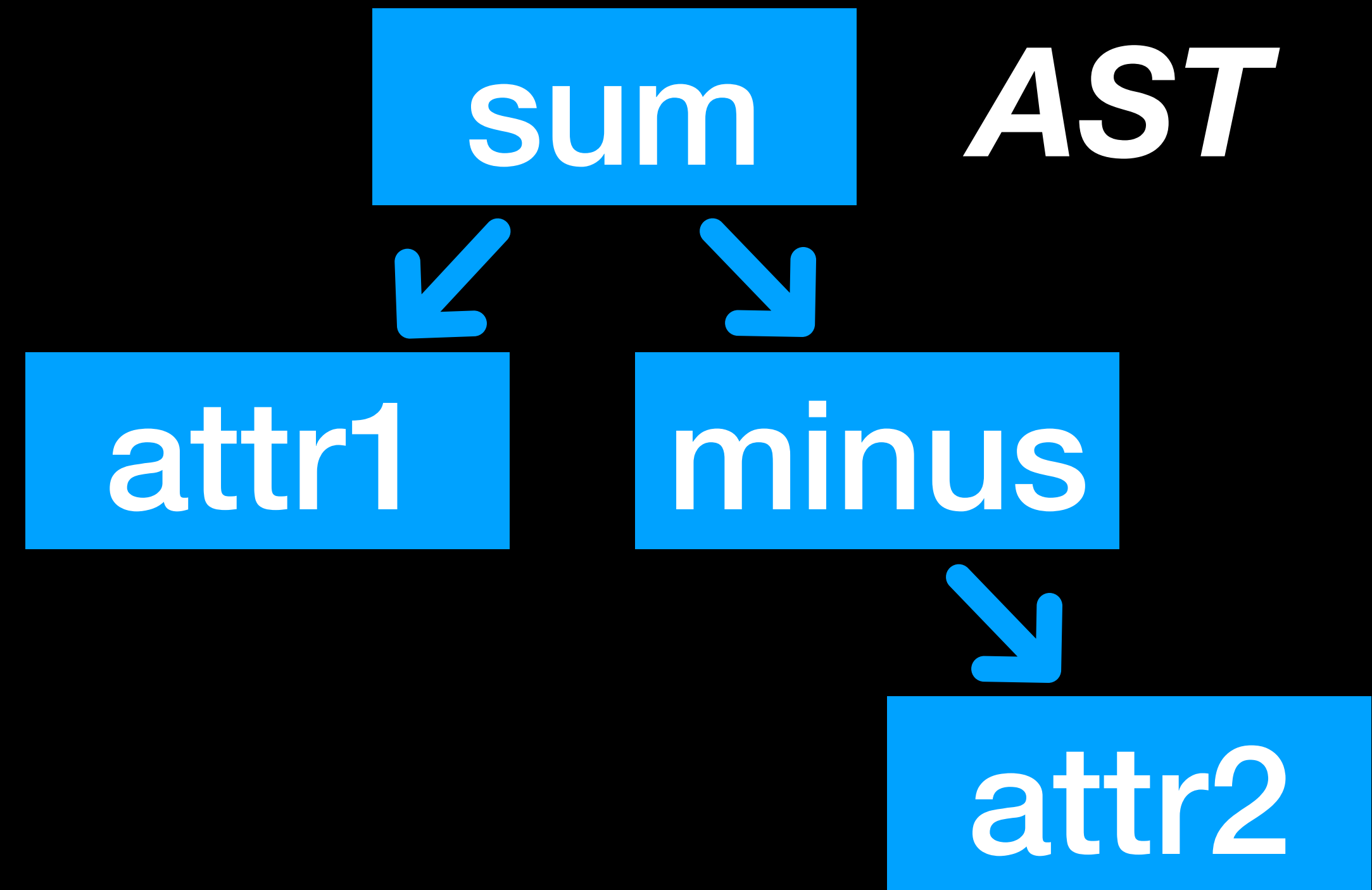
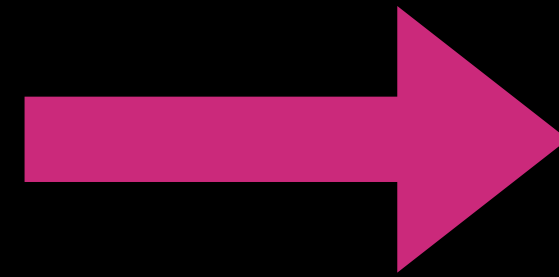
<var name="attr1">

<minus>

<var name="attr2">

</minus>

</sum>



Парсинг и компиляция

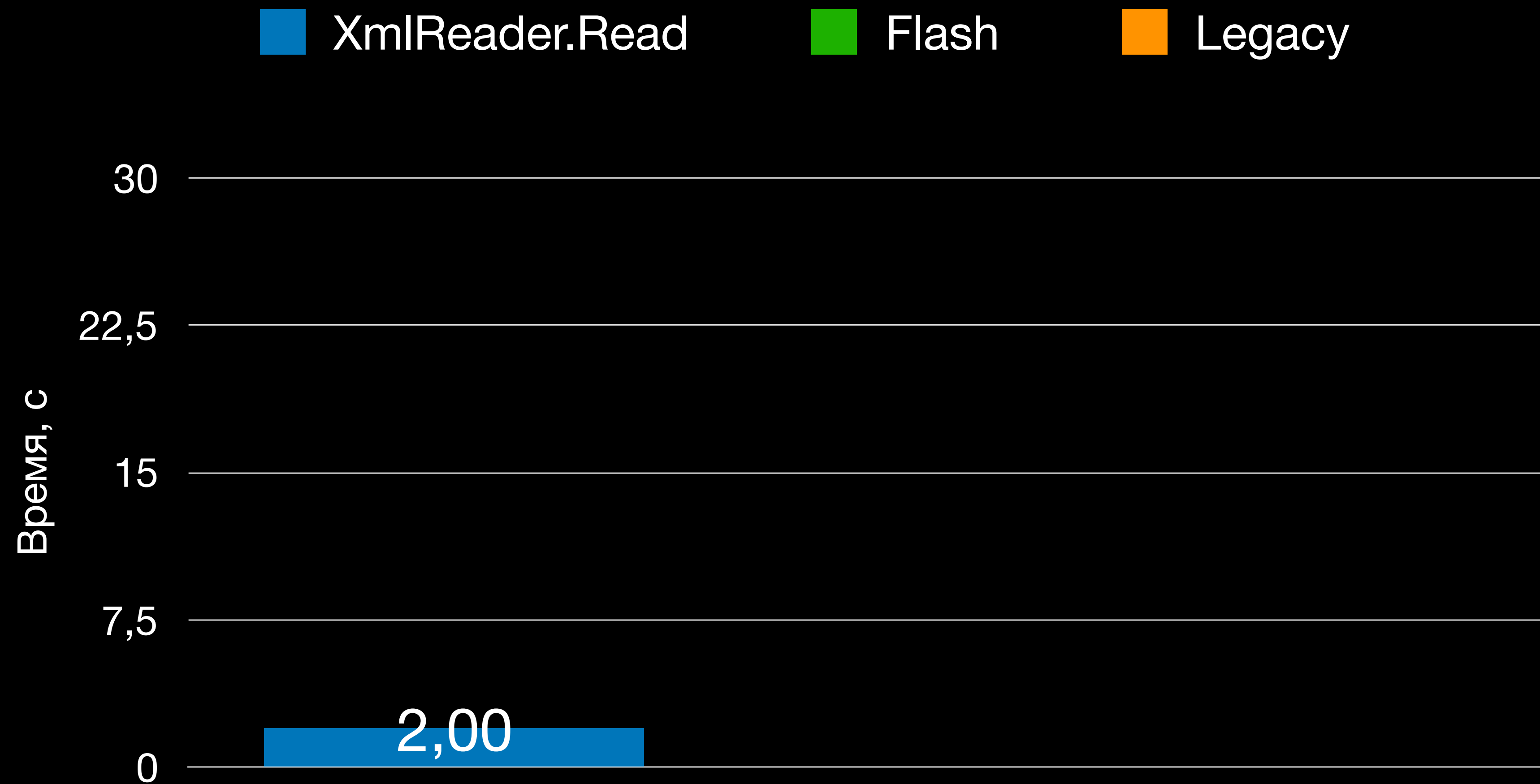
- XDocument (`rules.xml`)
- ANTLR ("`/foo[@attr1='10' and @attr2='20']`")
- Gremit

Benchmark. 300M6

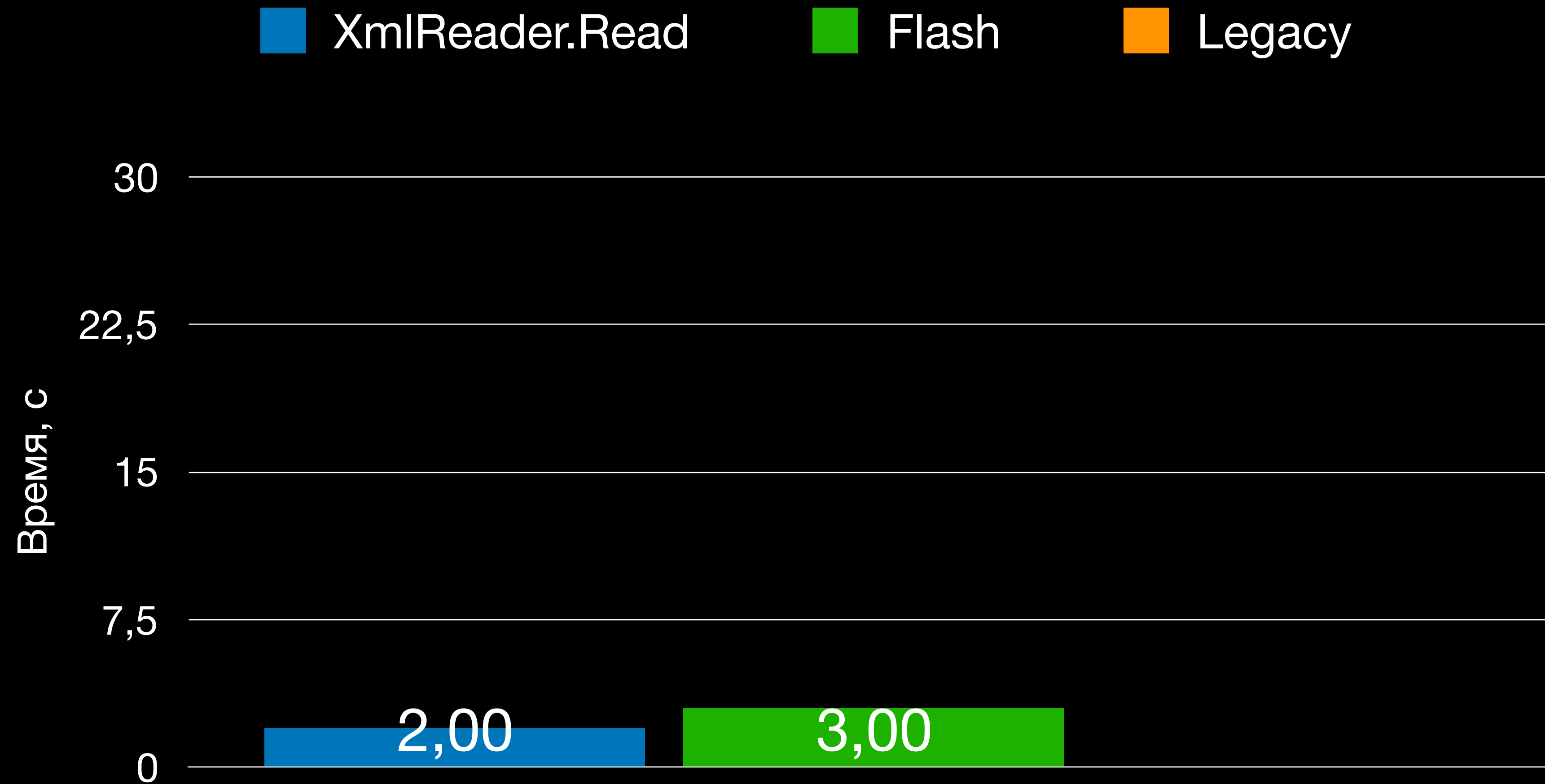
Benchmark. 300M6



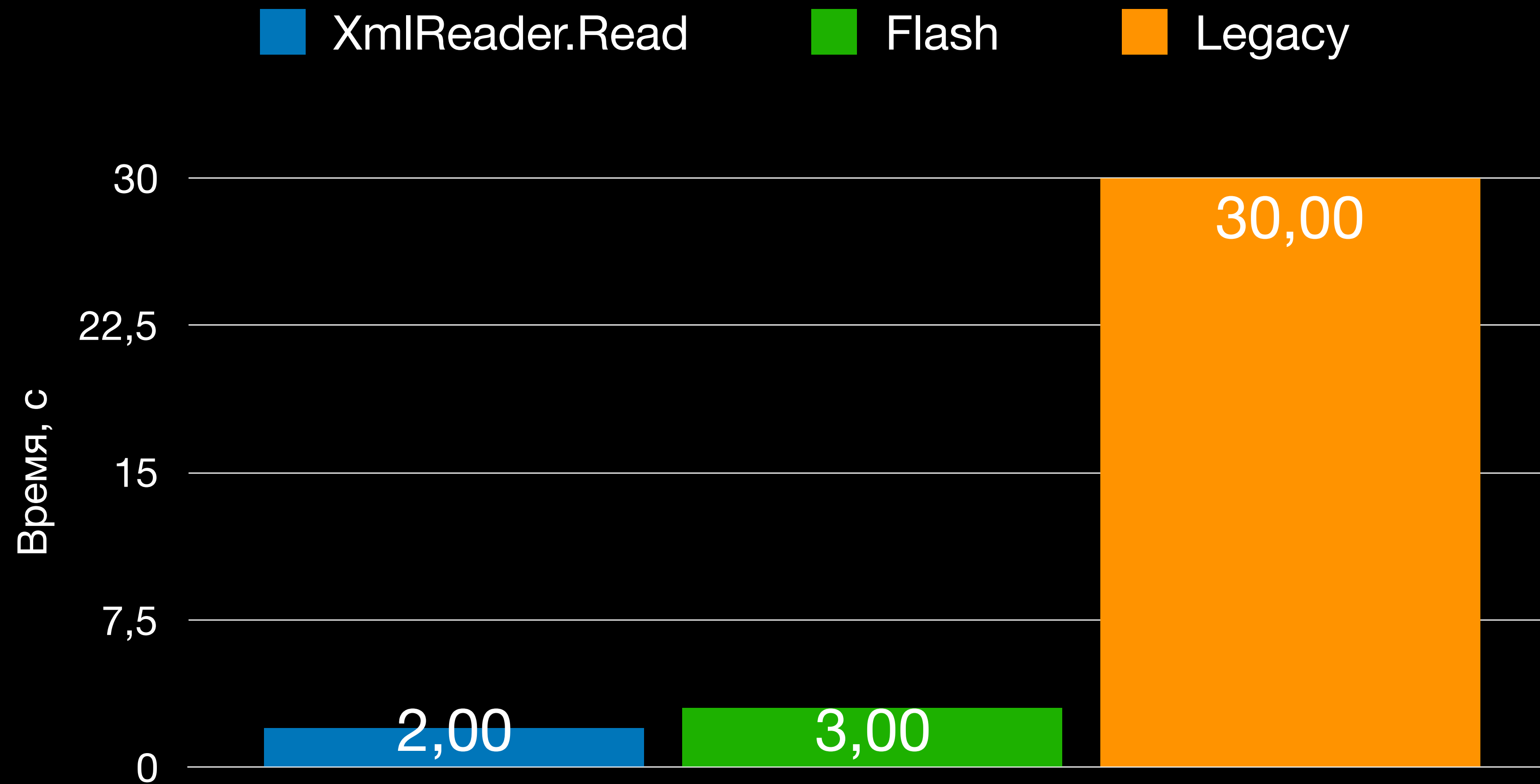
Benchmark. 300M6



Benchmark. 300M6



Benchmark. 300M6



Что такое MSIL

Что такое MSIL

- В MSIL компилируются языки высокого уровня (C#, F#)

Что такое MSIL

- В MSIL компилируются языки высокого уровня (C#, F#)
- Высокоуровневый ассемблер

Что такое MSIL

- В MSIL компилируются языки высокого уровня (C#, F#)
- Высокоуровневый ассемблер
- стек

Что такое MSIL

- В MSIL компилируются языки высокого уровня (C#, F#)
- Высокоуровневый ассемблер
- Стек
- Локальные переменные

Что такое MSIL

- В MSIL компилируются языки высокого уровня (C#, F#)
- Высокоуровневый ассемблер
- Стек
- Локальные переменные
- Метки

Пример

The image shows a screenshot of the Visual Studio IDE with two windows open: the C# source code editor and the IL Viewer.

C# Source Code (Program.cs):

```
1 class Foo
2 {
3     private static int Sum(
4         int a,
5         int b)
6     {
7         return a + b;
8     }
9 }
10
```

IL Viewer (IL_0000):

```
.method /*06000001*/ private hidebysig static
    Sum(
        /*08000001*/ int32 a,
        /*08000002*/ int32 b
    ) cil managed
{
    .maxstack 8

    // [7 3 - 7 16]
    IL_0000: ldarg.0          // a
    IL_0001: ldarg.1          // b
    IL_0002: add
    IL_0003: ret

} // end of method Foo::Sum

.method /*06000002*/ public hidebysig specialname
    .ctor() cil managed
{
    // ...
}
```

Пример

The image shows a screenshot of the Visual Studio IDE. The main editor displays the C# source code for a class `Foo` and its static method `Sum`. The code is as follows:

```
1 class Foo
2 {
3     private static int Sum(
4         int a,
5         int b)
6     {
7         return a + b;
8     }
9 }
10
```

Annotations in the code include "0 references" above the class and "0 references" above the method signature. A large white text overlay **stack: [a]** is positioned over the bottom right of the C# code.

The IL Viewer on the right shows the corresponding Intermediate Language code for the `Sum` method:

```
.method /*06000001*/ private hidebysig static
    Sum(
        /*08000001*/ int32 a,
        /*08000002*/ int32 b
    ) cil managed
{
    .maxstack 8

    // [7 3 - 7 16]
    IL_0000: ldarg.0          // a
    IL_0001: ldarg.1          // b
    IL_0002: add
    IL_0003: ret

} // end of method Foo::Sum

.method /*06000002*/ public hidebysig specialname
    .ctor() cil managed
{
```

Пример

The image shows a screenshot of the Visual Studio IDE. The main editor displays a C# class `Foo` with a static method `Sum(int a, int b)`. The code is as follows:

```
1 class Foo
2 {
3     private static int Sum(
4         int a,
5         int b)
6     {
7         return a + b;
8     }
9 }
10
```

Annotations on the code indicate the state of the stack during execution:

- Line 1: 0 references
- Line 3: 0 references
- Line 4: **stack: [a]**
- Line 5: **stack: [a, b]**

The right-hand pane shows the Intermediate Language (IL) for the `Sum` method:

```
.method /*06000001*/ private hidebysig static
    Sum(
        /*08000001*/ int32 a,
        /*08000002*/ int32 b
    ) cil managed
{
    .maxstack 8

    // [7 3 - 7 16]
    IL_0000: ldarg.0          // a
    IL_0001: ldarg.1          // b
    IL_0002: add
    IL_0003: ret

} // end of method Foo::Sum

.method /*06000002*/ public hidebysig specialname
    .ctor() cil managed
{
```


Пример

The screenshot shows the Visual Studio IDE with a C# project named 'YetAnotherSandbox'. The main editor displays the source code for a class `Foo` and its static method `Sum`. The `Sum` method takes two `int` parameters, `a` and `b`, and returns their sum. The IL Viewer on the right shows the Intermediate Language (IL) code for the `Sum` method, which includes argument loading, addition, and return instructions.

Source Code (C#):

```
1 class Foo
2 {
3     private static int Sum(
4         int a,
5         int b)
6     {
7         return a + b;
8     }
9 }
10
```

IL Code (IL Viewer):

```
.method /*06000001*/ private hidebysig static
    Sum(
        /*08000001*/ int32 a,
        /*08000002*/ int32 b
    ) cil managed
{
    .maxstack 8

    // [7 3 - 7 16]
    IL_0000: ldarg.0          // a
    IL_0001: ldarg.1          // b
    IL_0002: add
    IL_0003: ret

} // end of method Foo::Sum

.method /*06000002*/ public hidebysig specialname
    .ctor() cil managed
{
    // ...
}
```

Stack State:

- stack: [a]
- stack: [a, b]
- stack: [a + b]

Пример

The screenshot shows the Visual Studio IDE with a C# project named 'YetAnotherSandbox'. The main editor displays the source code for a class 'Foo' with a static method 'Sum'. The method signature is 'private static int Sum(int a, int b)'. The method body contains a single line 'return a + b;'. The IL Viewer on the right shows the Intermediate Language (IL) for the 'Sum' method. The IL code is as follows:

```
.method /*06000001*/ private hidebysig static  
    Sum(  
        /*08000001*/ int32 a,  
        /*08000002*/ int32 b  
    ) cil managed  
{  
    .maxstack 8  
  
    // [7 3 - 7 16]  
    IL_0000: ldarg.0          // a  
    IL_0001: ldarg.1          // b  
    IL_0002: add  
    IL_0003: ret  
  
} // end of method Foo::Sum  
  
.method /*06000002*/ public hidebysig specialname  
    .ctor() cil managed  
{
```

Below the code, the stack state is shown at different points in the execution:

- stack: [a]
- stack: [a, b]
- stack: [a + b]
- stack: []

OpCodes

- ldc.i4.5
- call
- stfld / ldfld
- stloc / ldloc
- br / brtrue / brfalse
- dup
- pop

Генерируем код!

Генерируем код!

```
1 reference
private static int Emit()
{
→   var method = new DynamicMethod(
→       name: "Foo",
→       returnType: typeof(int),
→       parameterTypes: new[] {typeof(int), typeof(int)});
→   using (var il = new GroboIL(method))
→   {
→       il.Ldarg(0);
→       il.Ldarg(1);
→       il.Add();
→       il.Ret();
→   }

→   return (int)method.Invoke(null, new object[] {1, 2});
}
```

github.com/skbkontur/gremit

```
.method /*06000001*/ public static void  
    MyMethod() cil managed  
{  
    .maxstack 1  
  
    IL_0000: ldc.i4.0  
    IL_0001: ret  
  
} // end of method MyType::MyMethod
```


Исключение Reflection.Emit

```
C:\WINDOWS\system32\cmd.exe

Необработанное исключение: System.Reflection.TargetInvocationException: Адресат
вызова создал исключение. ---> System.InvalidProgramException: Компилятор JIT об
наружил внутреннее ограничение.
    в MyType.MyMethod()
    --- Конец трассировки внутреннего стека исключений ---
    в System.RuntimeMethodHandle.InvokeMethod(Object target, Object[] arguments,
Signature sig, Boolean constructor)
    в System.Reflection.RuntimeMethodInfo.UnsafeInvokeInternal(Object obj, Object
[] parameters, Object[] arguments)
    в System.Reflection.RuntimeMethodInfo.Invoke(Object obj, BindingFlags invokeA
ttr, Binder binder, Object[] parameters, CultureInfo culture)
    в System.Reflection.MethodBase.Invoke(Object obj, Object[] parameters)
    в GremitAdvantages.Program.Main(String[] args) в c:\users\yuryev\documents\vi
sual studio 2017\Projects\GremitAdvantages\GremitAdvantages\Program.cs:строка 48
```


Исключение Gremit

```
C:\WINDOWS\system32\cmd.exe

Необработанное исключение: System.InvalidOperationException: The function being
emitted is void. Thus at the end the evaluation stack must be empty
ldc.i4.0          // [Int32]
ret              //

    в GrEmit.StackMutator.ThrowError(GroboIL il, String message) в C:\BuildAgent\
work\249acc80a2fd7042\GrEmit\GrEmit\StackMutator.cs:строка 101
    в GrEmit.StackMutators.RetStackMutator.Mutate(GroboIL il, ILInstructionParame
ter parameter, EvaluationStack& stack) в C:\BuildAgent\work\249acc80a2fd7042\GrE
mit\GrEmit\StackMutators\RetStackMutator.cs:строка 10
    в GrEmit.StackMutatorCollection.Mutate(OpCode opCode, GroboIL il, ILInstructi
onParameter parameter, EvaluationStack& stack) в C:\BuildAgent\work\249acc80a2fd
7042\GrEmit\GrEmit\StackMutatorCollection.cs:строка 18
    в GrEmit.GroboIL.Emit(OpCode opCode) в C:\BuildAgent\work\249acc80a2fd7042\Gr
Emit\GrEmit\GroboIL.cs:строка 2092
    в GrEmit.GroboIL.Ret() в C:\BuildAgent\work\249acc80a2fd7042\GrEmit\GrEmit\Gr
oboIL.cs:строка 425
    в GremitAdvantages.Program.Main(String[] args) в C:\Users\yuryev\Documents\Vi
sual Studio 2017\Projects\GremitAdvantages\GremitAdvantages\Program.cs:строка 16
```

Что можно сгенерировать

- Assembly (+save)
- Class
- Method
- Field
- Property

Инструменты

- GrEmit (<https://github.com/skbkontur/gremit>)
- ildasm
- ilasm
- dotPeek (<https://www.jetbrains.com/decompiler/>)
- dnSpy (<https://github.com/0xd4d/dnSpy>)
- dnLib (<https://github.com/0xd4d/dnlib>)
- Mono.Cecil (<https://github.com/jbevain/cecil>)
- Antlr4 (<https://github.com/tunnelvisionlabs/antlr4cs>)

Области применения

Области применения

- DSL

Области применения

- DSL
- Ускорение (regex, сериализация)

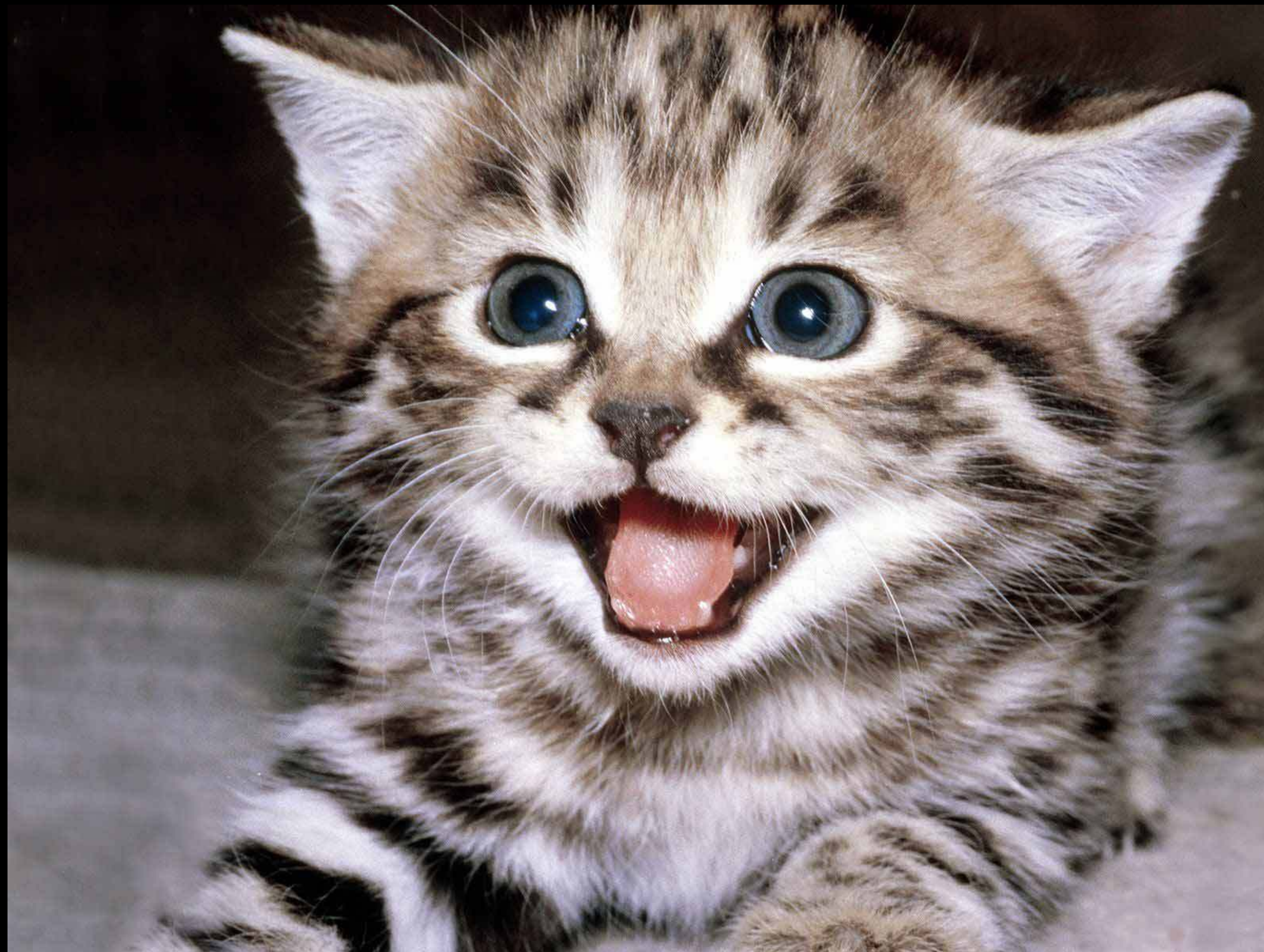
Области применения

- DSL
- Ускорение (regex, сериализация)
- AOP

Области применения

- DSL
- Ускорение (regex, сериализация)
- AOP
- Code analyzer & code fix

Вопросы?



email: odinmillion@gmail.com