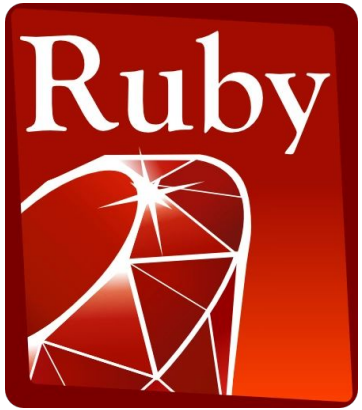
Динамическое выделение типов в Ruby

Вьюгинов Николай, 444

Научный руководитель: Я.А. Кириленко, ст. преп

Область применения



Почему анализ кода так важен?

- Обнаружение реальных и потенциальных ошибок в коде
- Ускорение разработки
- Поддержка автодополнения и автоматического рефакторинга

```
x = "123"
p x.length, x.downcase

x = {:a ⇒ '1', :b ⇒ '2', :c ⇒ '3'}
p x.length, x.downcase
```

Cannot find 'downcase' more... (⌘F1)

YARD/RDoc

```ruby
# Converts the object into textual markup given a specific format.
#
# @param format [Symbol] the format type, `:text` or `:html`
# @return [String] the object converted into the expected format.
def to_format(format = :html)
  # format the object
end
```

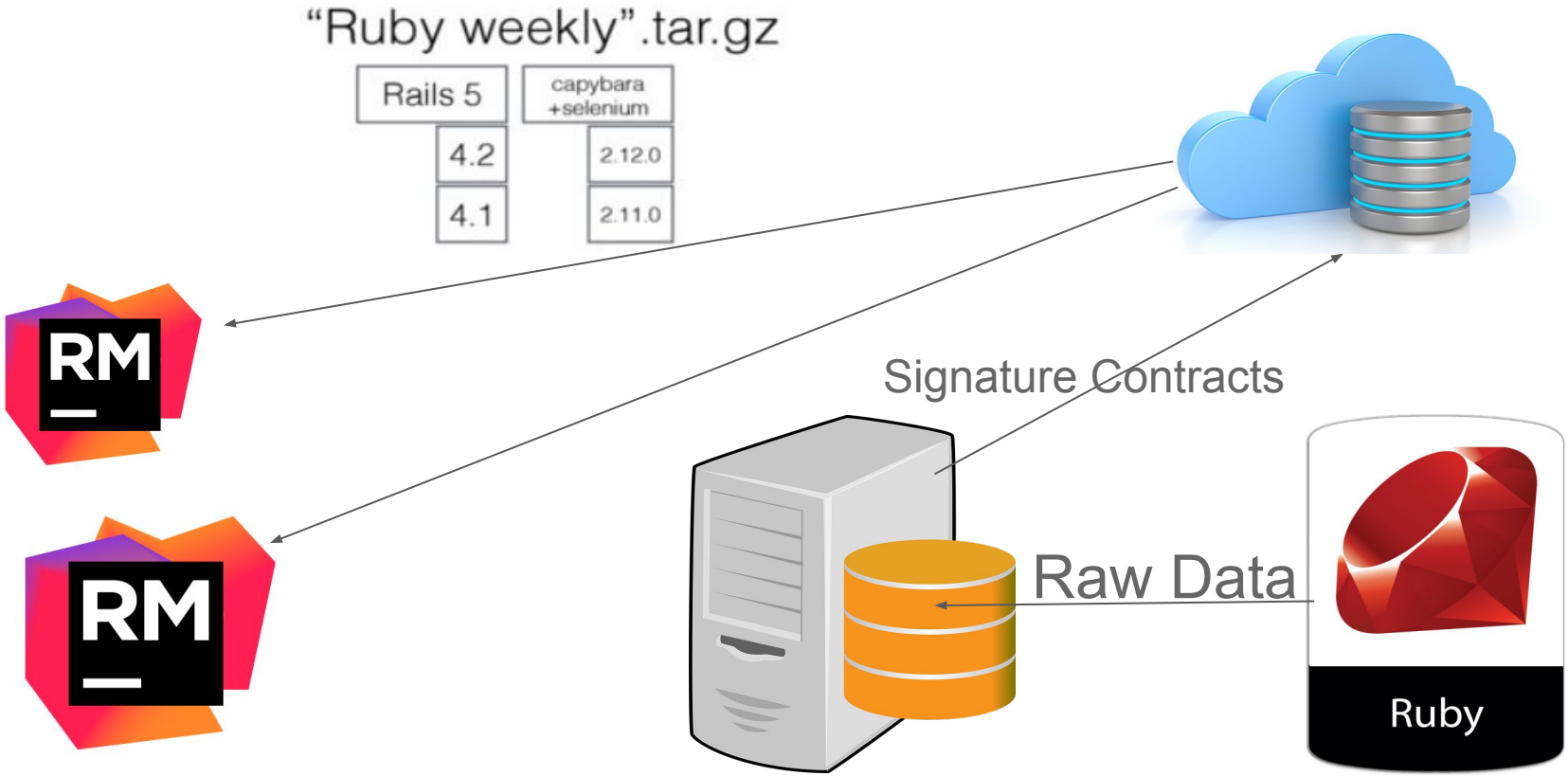
Статические анализаторы для Ruby

- Rubocop
- Ruby-lint
- Breakman
- Diamondback Ruby (DRuby)

Задачи работы

- Разработать архитектуру
- Реализовать окружение для запуска программ, собирающее данные о типах
- Разработать систему для вывода контрактов, описывающих сигнатуру метода
- Реализовать плагин для IDE, использующий контракты в системе статического анализа
- Протестировать модуль и провести нагрузочное тестирование

Схема работы



Данные о методах

- Имя метода
- Количество переданных аргументов
- Список имён именованных параметров, переданных в метод
- Тип объекта, у которого был вызван метод
- Информация о сигнатуре метода(имена, типы параметров и типы аргументов)
- Выходной тип метода
- Имя гема(библиотеки) в котором метод был декларирован
- Версия гема
- Видимость метода
- Путь к файлу, где был задекларирован метод
- Номер строки, где был задекларирован метод

TracePoint API

```ruby
def foo(a, b = 1)
  b = '1'
end

TracePoint.trace( *events :call, :return) do |tp|
  binding = tp.binding
  p tp.defined_class.method(tp.method_id).parameters
  puts tp.event, (binding.local_variables.map do |v|
    "#{v} → #{binding.local_variable_get(v).inspect}"
  end.join ', ')
end

foo(2)
```

```
[[:req, :a], [:opt, :b]]
call
a -> 2, b -> 1
[[:req, :a], [:opt, :b]]
return
a -> 2, b -> "1"

Process finished with exit code 0
```

```
code = <<~EOF
  def foo(a, b=239)
    p a, b
  end
  foo(1)
EOF
puts RubyVM::InstructionSequence.compile(code).disasm
```

```
== disasm: #<ISeq:<compiled>@<compiled>>================================
0000 trace            1                                               (   1)
0002 putspecialobject 1
0004 putobject        :foo
0006 putiseq          foo
0008 opt_send_without_block <callinfo!mid:core#define_method, argc:2, ARGS_SIMPLE>, <call
0011 pop
0012 trace            1                                               (   4)
0014 putself
0015 putobject_OP_INT2FIX_O_1_C_
0016 opt_send_without_block <callinfo!mid:foo, argc:1, FCALL|ARGS_SIMPLE>, <callcache>
0019 leave
== disasm: #<ISeq:foo@<compiled>>=======================================
local table (size: 3, argc: 1 [opts: 1, rest: -1, post: 0, block: -1, kw: -1@-1, kwrest:
[ 3] a<Arg>     [ 2] b<Opt=0>
0000 putobject        239                                             (   1)
0002 setlocal_OP__WC__0 2
0004 trace            8
0006 trace            1                                               (   2)
0008 putself
0009 getlocal_OP__WC__0 3
0011 getlocal_OP__WC__0 2
0013 opt_send_without_block <callinfo!mid:p, argc:2, FCALL|ARGS_SIMPLE>, <callcache>
```

rb_control_frame_t

```
const VALUE *pc;
const rb_iseq_t *iseq;
```

PC

Генерация контрактов

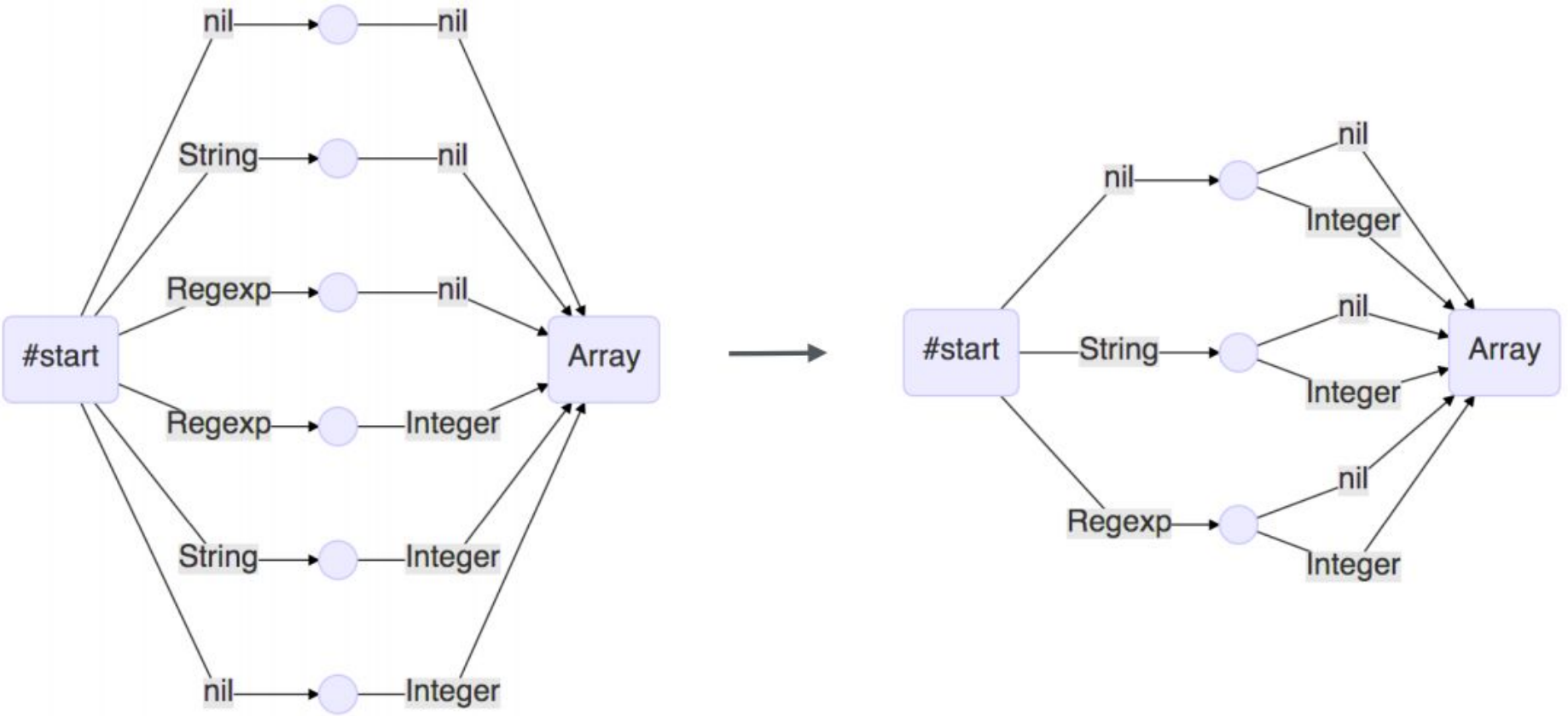
```
str.split(pattern=nil, [limit])   -> anArray
```

```
" now's  the time".split         #⇒ ["now's", "the", "time"]
" now's  the time".split(' ')    #⇒ ["now's", "the", "time"]
" now's  the time".split(/ /)    #⇒ ["", "now's", "", "the", "time"]
"1, 2.34,56, 7".split(%r{,\s*}) #⇒ ["1", "2.34", "56", "7"]
"hello".split(//)                #⇒ ["h", "e", "l", "l", "o"]
"hello".split(//, 3)             #⇒ ["h", "e", "llo"]
"hi mom".split(%r{\s*})          #⇒ ["h", "i", "m", "o", "m"]
"mellow yellow".split("ello")    #⇒ ["m", "w y", "w"]
"1,2,,3,4,,".split(',')          #⇒ ["1", "2", "", "3", "4"]
"1,2,,3,4,,".split(',', 4)       #⇒ ["1", "2", "", "3,4,,"]
"1,2,,3,4,,".split(',', -4)      #⇒ ["1", "2", "", "3", "4", "", ""]
```

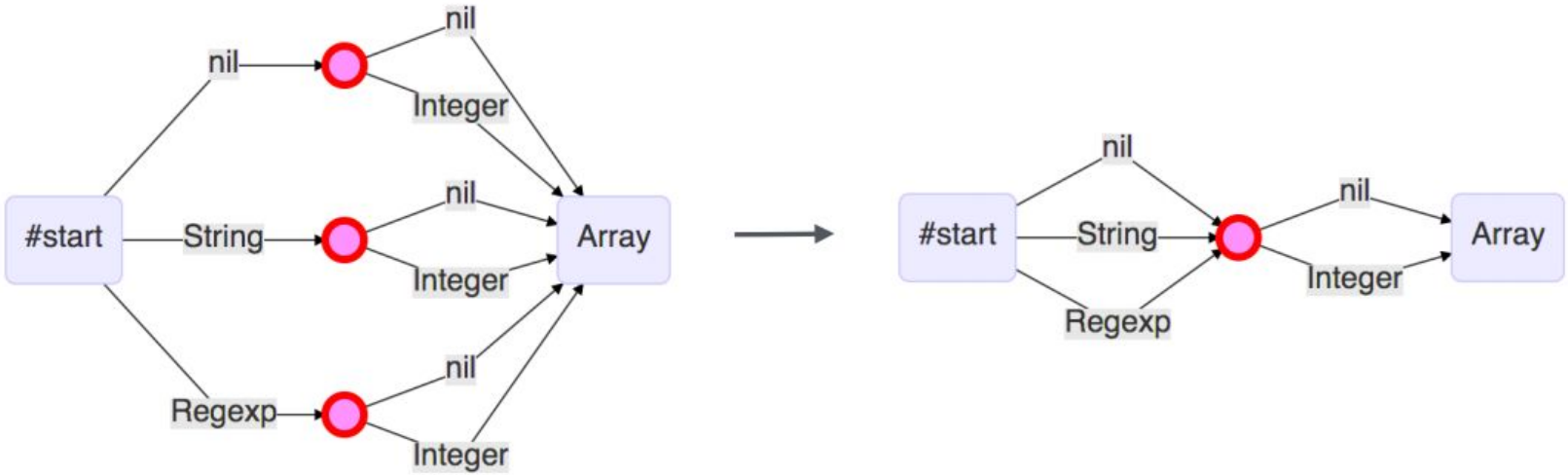
(nil, nil) → Array
(String, nil) → Array
(Regexp, nil) → Array
(Regexp, Integer) → Array
(String, Integer) → Array
(nil, Integer) → Array

(nil | String | Regexp, nil | Integer) → Array

Генерация контрактов

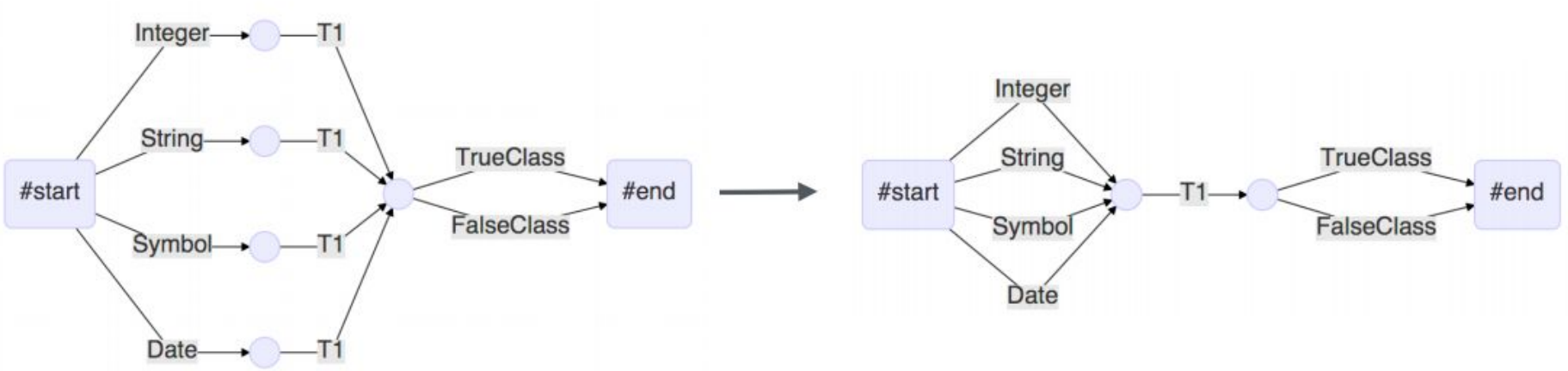


Генерация контрактов



Эвристика для автомата

```ruby
def eq(a, b)
  raise StandardError if a.class != b.class
  a == b
end
```



Тестирование подхода

Название класса (ActiveRecord::)	Название метода	n1	s1	n2	s2	n3	s3
SpawnMethods	merge	120	42	440	79	688	81
SpawnMethods	spawn	112	2	411	2	639	2
Relation	initialize_copy	112	3	411	3	639	3
Relation	reset	112	2	411	2	639	2
Relation::Merger	initialize	94	67	362	127	568	134
SpawnMethods	merge!	93	65	362	122	568	128
ModelSchema::ClassMethods	table_name=	85	3	268	3	431	3
Core::ClassMethods	relation	78	2	261	2	401	2
Scoping::Named::ClassMethods	all	78	2	269	2	414	2
Relation::Merger	merge	76	2	300	2	87	2
Scoping::Named::ClassMethods	default_scoped	76	2	259	2	399	2
Scoping::ScopeRegistry	set_value_for	62	6	218	6	332	6
Scoping::Default::ClassMethods	unscoped	60	2	247	2	378	2
Scoping::ClassMethods	current_scope=	60	3	213	3	323	3
Querying	where	60	4	291	4	459	4

Заключение

Разрабатываемый инструмент позволяет улучшить систему выделения типов, которая лежит в основе анализа кода и позволяет улучшить:

- автодополнение
- навигацию в коде
- инспекции
- построение диаграмм

Результаты

Разрабатываемый инструмент позволяет улучшить систему выделения типов, которая лежит в основе анализа кода и позволяет улучшить:

- Разработана архитектура.
- Реализовано окружение для запуска программ, собирающее данные о типах.
- Спроектирован и разработан модуль для вывода контрактов, описывающих сигнатуру метода.
- Создано облачное хранилище для сгенерированных контрактов.
- Реализован плагин для IDE, встраивающий контракты в систему статического анализа.
- Модуль протестирован и проведено нагрузочное тестирование.

github.com/jetbrains/ruby-type-inference