The TruffleRuby Compilation Pipeline

Just-in-time compiling Ruby with self-specializing ASTs and partial evaluation

Chris Seaton
Research Manager
Oracle Labs
March 2019

ORACLE

Safe Harbor Statement

The following is intended to provide some insight into a line of research in Oracle Labs. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. Oracle reserves the right to alter its development plans and practices at any time, and the development, release, and timing of any features or functionality described in connection with any Oracle product or service remains at the sole discretion of Oracle. Any views expressed in this presentation are my own and do not necessarily reflect the views of Oracle.

TruffleRuby basics

We wanted to build a Ruby that

- Runs idiomatic Ruby code faster
- Runs Ruby code in parallel
- Executes C extensions in a managed environment
- Adds fast and low-overhead interoperability with other languages
- Provides new tooling such as debuggers and monitoring
- Has very high compatibility with the standard implementation of Ruby

ORACLE

Documents — -bash — 84×15

```
$ rbenv install truffleruby-1.0.0-rc14
$ rbenv shell truffleruby-1.0.0-rc14
$ ruby -v
truffleruby 1.0.0-rc14, like ruby 2.6.2, GraalVM CE Native [x86_64-darwin]
$
```

ORACLE®

demo.rb

```ruby
require 'erb'

template = ERB.new(%{
  <h1>Hello world!</h1>
  <p>The time is <%= now %></p>
})

loop do
  start = Time.now

  100_000.times do
    now = Time.now
    puts template.result(binding)
  end

  $stderr.puts Time.now - start
end
```

Line 1, Column 1 Spaces: 2 Ruby

Documents — ruby perf.rb — 84×15

```
$ rbenv shell 2.6.2
$ ruby perf.rb > /dev/null
2.165717
2.134813
2.150839
2.139991
2.145957
2.17635
2.209064
2.196461
2.183587
```

Documents — ruby --jit perf.rb — 84×15

```
$ rbenv shell 2.6.2
$ ruby --jit perf.rb > /dev/null
2.404086
2.436718
2.618684
2.391098
2.47526
2.488115
2.427324
2.407296
2.503696
```

ORACLE

Documents — java -Djdk.home= -

```
$ rbenv shell jruby-9.2.6.0
$ ruby perf.rb > /dev/null
4.098946
3.117474
3.164635
3.18634
3.37483
3.305855
3.40638
3.088724
3.0775989999999998
```

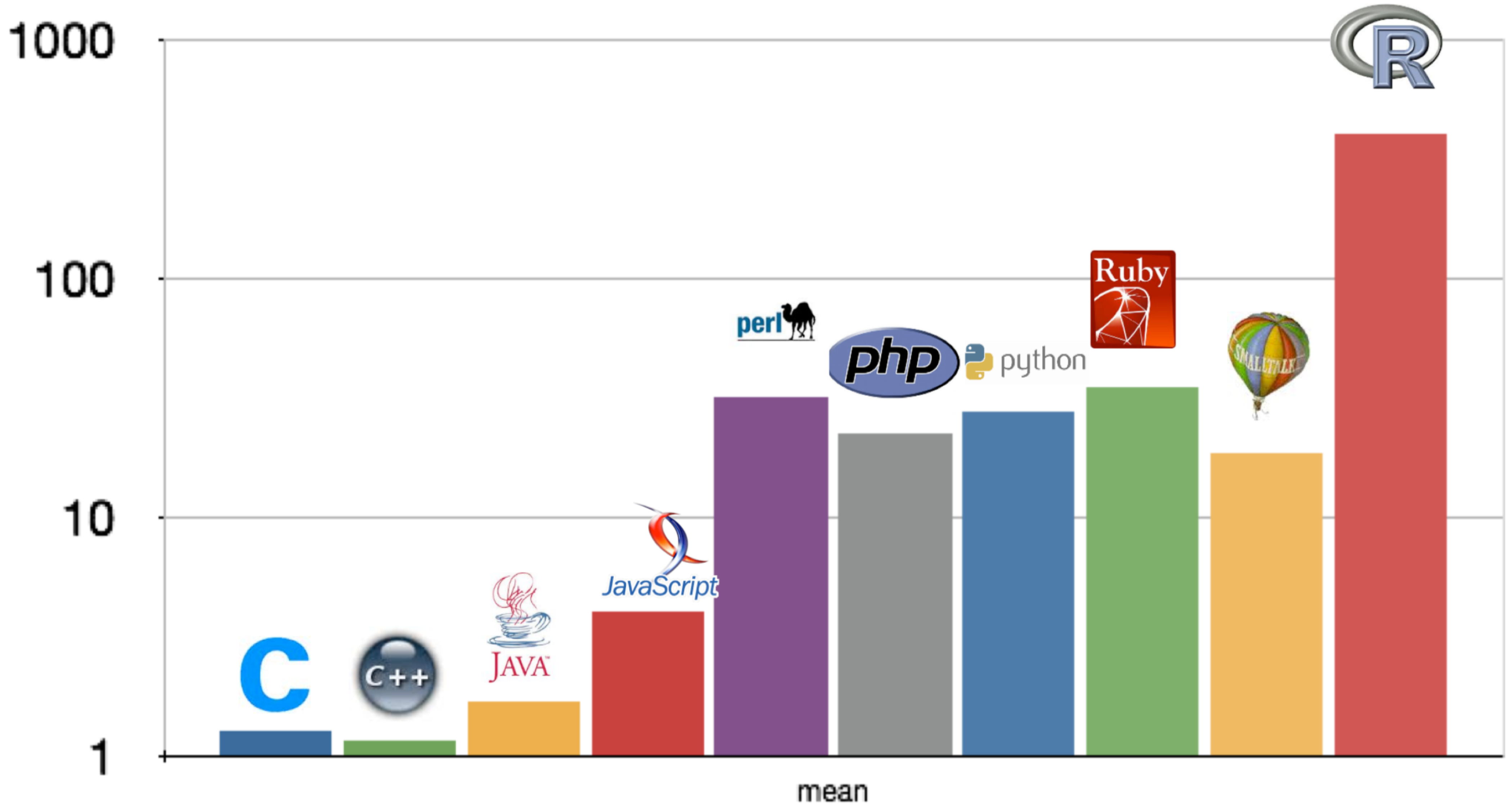
Documents — java -Djdk.home= -Djruby.home=/Users/chrisseaton/.rbenv/versi

```
$ rbenv shell jruby-9.2.6.0
$ ruby -Xcompile.invokedynamic=true perf.rb > /dev/null
3.8451880000000003
2.939869
3.010014
3.177851
3.1914369999999996
3.366082
2.927472
2.95315
2.990667
```

ORACLE

Documents — ruby perf.rb — 84×15

```
$ rbenv shell truffleruby-1.0.0-rc14
$ ruby perf.rb > /dev/null
1.577
1.912
0.781
0.464
0.465
0.6
0.465
0.47
0.461
```



1000
100
10
1
C
C++
JAVA
JavaScript
perl
php
python
Ruby
SMALLTALK
R
mean

ORACLE

Current situation	How it should be

Current situation

Prototype a new language

Parser and language work to build syntax tree (AST), AST Interpreter

Write a “real” VM

In C/C++, still using AST interpreter, spend a lot of time implementing runtime system, GC, …

People start using it

People complain about performance

Define a bytecode format and write bytecode interpreter

Performance is still bad

Write a JIT compiler
Improve the garbage collector

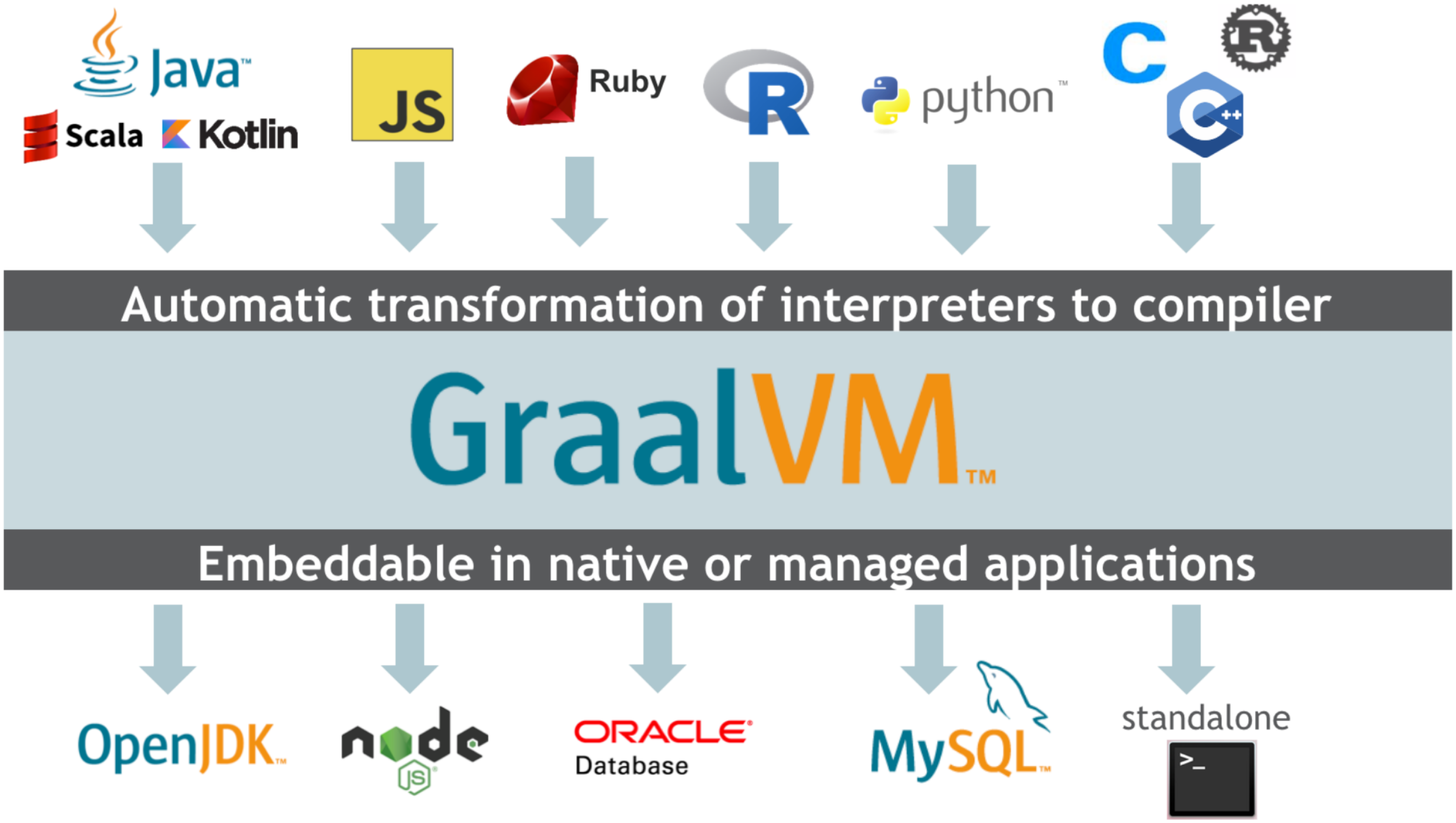
How it should be

Prototype a new language in Java

Parser and language work to build syntax tree (AST)
Execute using AST interpreter

People start using it

And it is already fast


Java
Scala
Kotlin
JS
Ruby
R
python
C
Automatic transformation of interpreters to compiler
GraalVM
Embeddable in native or managed applications
OpenJDK
node
JS
ORACLE
Database
MySQL
standalone
>_

Compilation basics

```ruby
def fib(n)
  if n <= 2
    1
  else
    fib(n - 1) + fib(n - 2)
  end
end
```

```
0x000000011a63b6e4: cmp    r10d,0x3
0x000000011a63b6e8: jl     0x000000011a63ba3f
0x000000011a63b703: sub    r13d,0x1
0x000000011a63b707: jo     0x000000011a63be54
0x000000011a63b897: call   0x000000011950bde0
0x000000011a63b8a8: sub    ebx,0x2
0x000000011a63b8ab: jo     0x000000011a63bda2
0x000000011a63b9af: call   0x000000011950bde0
0x000000011a63b9c5: add    r13d,eax
0x000000011a63b9c8: jo     0x000000011a63bd7b
0x000000011a63ba35: test   DWORD PTR [rip+0xfffffffff0e5e5cb],eax
0x000000011a63ba3e: ret
0x000000011a63ba3f: movabs rax,0x6c012dd20
0x000000011a63ba58: test   DWORD PTR [rip+0xfffffffff0e5e5a8],eax
0x000000011a63ba61: ret
```

```
def compile(ruby_source : String) : Array(UInt8)
  ...
end
```

ORACLE

Compiling Ruby

blog.headius.com

Headius

Helping the JVM Into the 21st Century

Monday, October 15, 2012

So You Want To Optimize Ruby

I was recently asked for a list of "hard problems" a Ruby implementation really needs to solve before reporting benchmark numbers. You know...the sort of problems that might invalidate early perf numbers because they impact how you optimize Ruby. This post is a rework of my response...I hope you find it informative!

Fixnum to Bignum promotion

In Ruby, Fixnum math can promote to Bignum when the result is out of Fixnum's range. On implementations that use tagged pointers to represent Fixnum (MRI, Rubinius, MacRuby), the Fixnum range is somewhat less than the base CPU bits (32/64). On JRuby, Fixnum is always a straight 64-bit signed value.

This promotion is a performance concern for a couple reasons:

- Every math operation that returns a new Fixnum must be range-checked. This slows all Fixnum operations.
- It is difficult (if not impossible) to predict whether a Fixnum math operation will return a Fixnum or a Bignum. Since Bignum is always represented as a full object (not a primitive or a tagged pointer) this impacts optimizing Fixnum math call sites.

Floating-point performance

A similar concern is the performance of floating point values. Most of the native implementations have tagged values for Fixnum but only one I know of (Macruby) uses tagged values for Float. This can

About Me

headius

View my complete profile

Followers

Obserwatorzy

So you want to optimize Ruby?

- Fixnum to Bignum promotion
- Floating-point performance
- Closures
- Bindings and eval
- callcc and Continuation
- Fiber implementation
- Thread/frame/etc local $globals
- C extension support
- Ruby 1.9 encoding support
- Garbage collection and object allocation
- Concurrency / Parallelism
- Tracing/debugging
- ObjectSpace
- Method invalidation
- Constant lookup and invalidation
- Rails

ORACLE

The TruffleRuby compilation pipeline

ORACLE

fib.rb

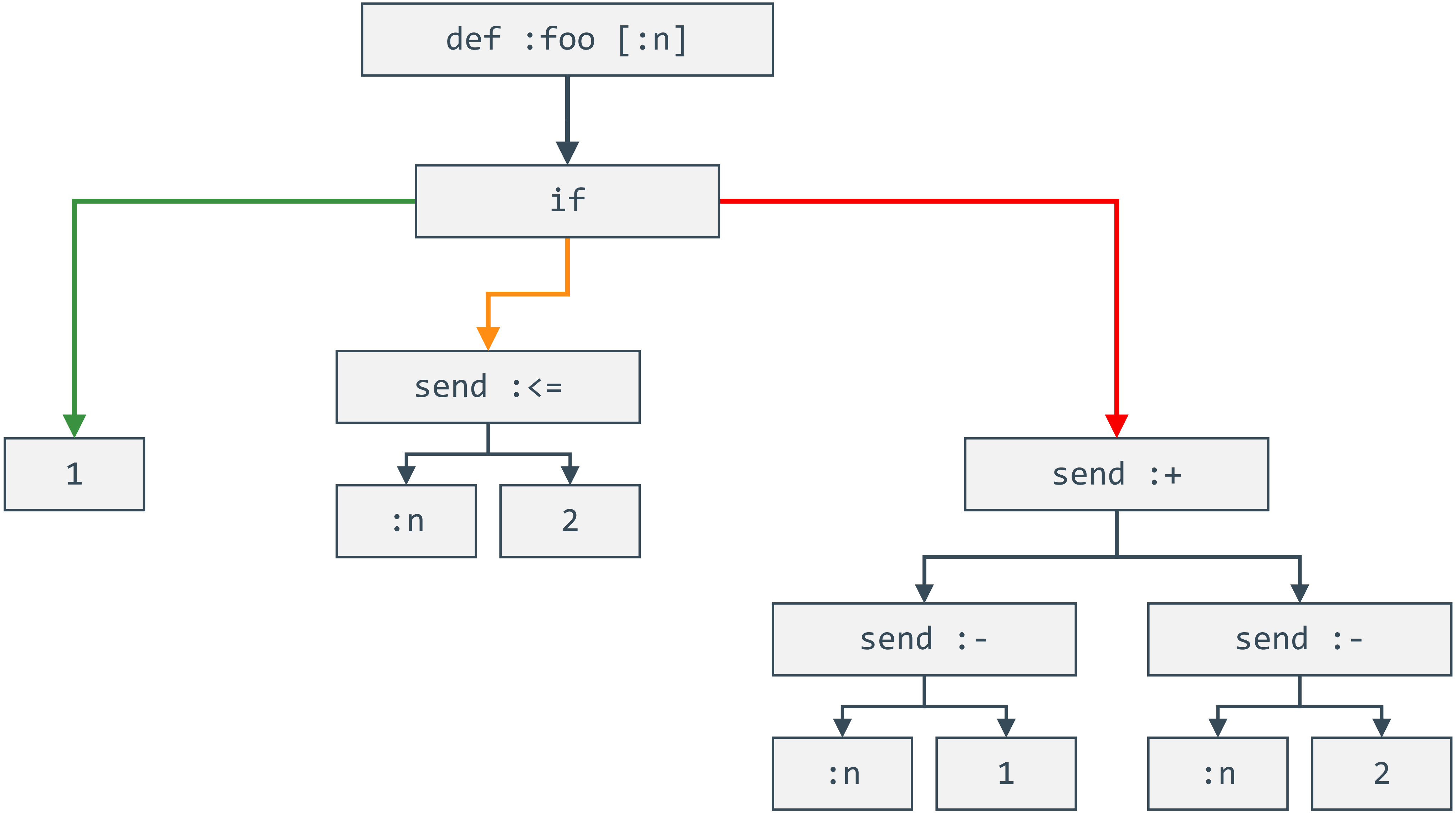
```ruby
def fib(n)
  if n <= 2
    1
  else
    fib(n - 1) + fib(n - 2)
  end
end

loop do
  puts fib(30)
end

```

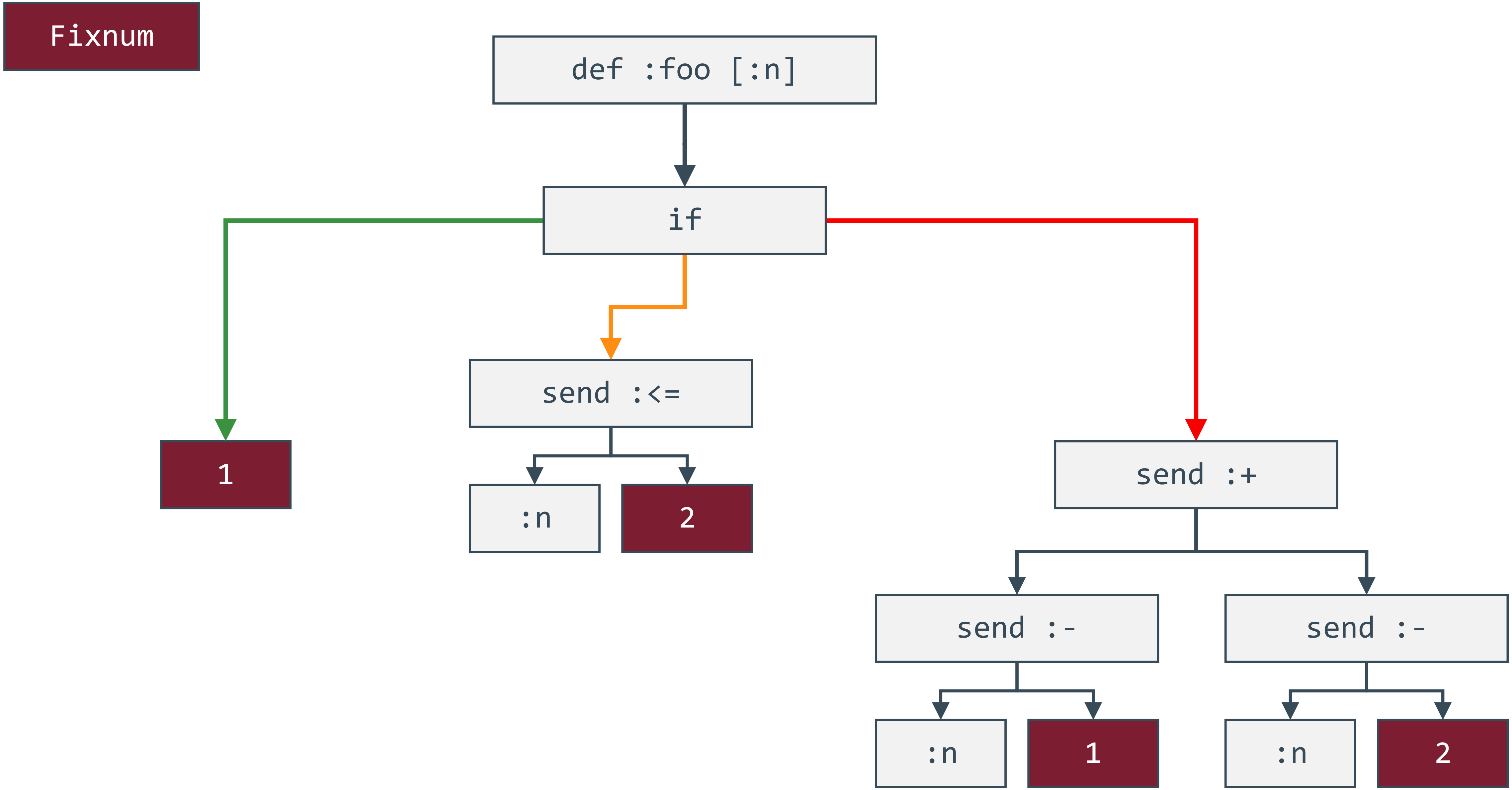
Line 1, Column 1 master Spaces: 2 Ruby

truffleruby — -bash — 110×58

```
$ rbenv shell 2.6.2
$ ruby --dump=parse fib.rb
###########################################################
## Do NOT use this node dump for any purpose other than  ##
## debug and research.  Compatibility is not guaranteed. ##
###########################################################

# @ NODE_SCOPE (line: 1, location: (1,0)-(11,3))
# +- nd_tbl: (empty)
# +- nd_args:
# |   (null node)
# +- nd_body:
#     @ NODE_BLOCK (line: 1, location: (1,0)-(11,3))
#     +- nd_head (1):
#     |   @ NODE_DEFN (line: 1, location: (1,0)-(7,3))*
```

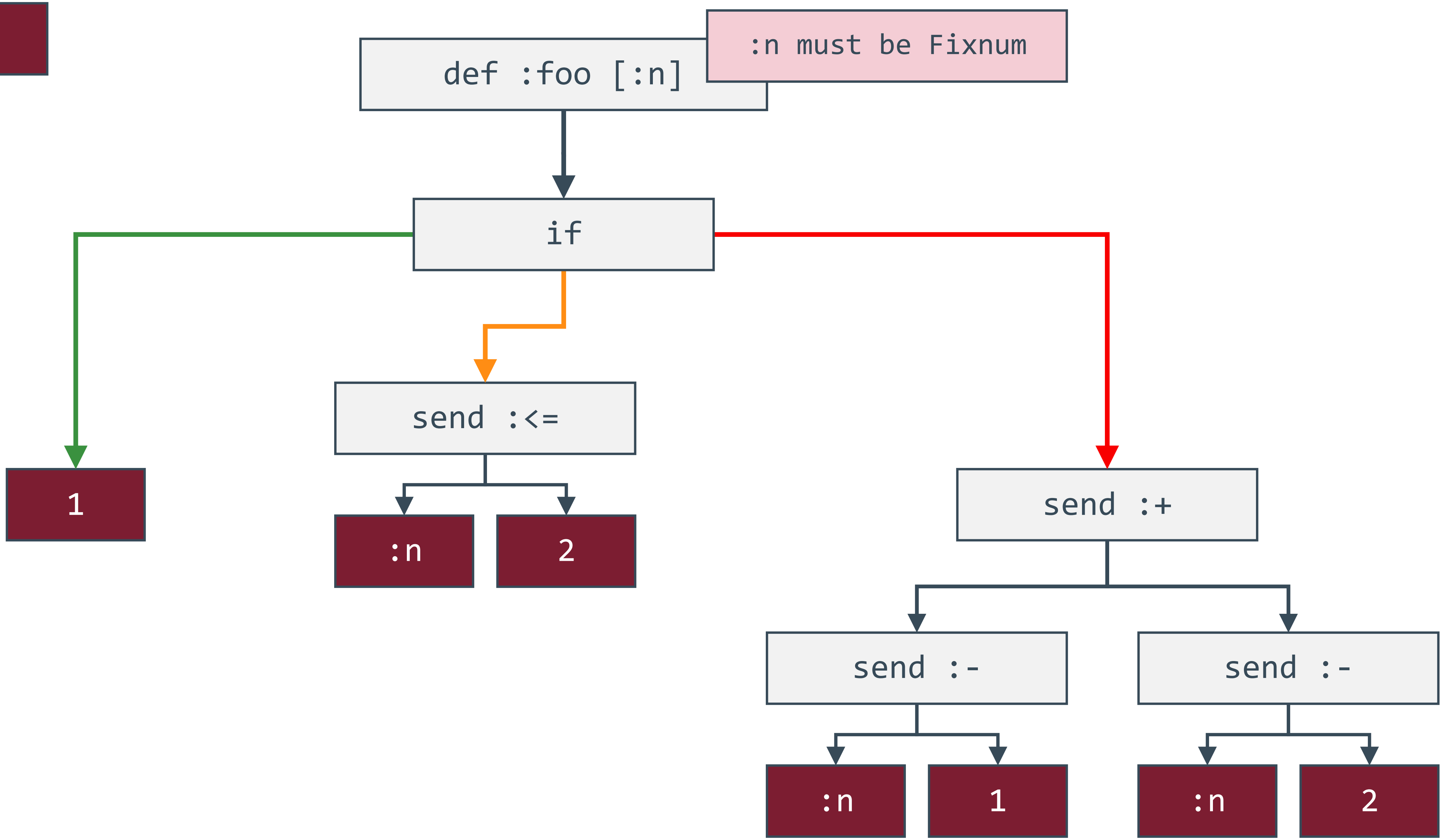
```
#                  +- nd_args:
#                      @ NODE_ARRAY (line: 10, location: (10,7)-(10,14))
#                      +- nd_alen: 1
#                      +- nd_head:
#                      |   @ NODE_FCALL (line: 10, location: (10,7)-(10,14))
#                      |   +- nd_mid: :fib
#                      |   +- nd_args:
#                      |       @ NODE_ARRAY (line: 10, location: (10,11)-(10,13))
#                      |       +- nd_alen: 1
#                      |       +- nd_head:
#                      |       |   @ NODE_LIT (line: 10, location: (10,11)-(10,13))
#                      |       |   +- nd_lit: 30
#                      |       +- nd_next:
#                      |           (null node)
#                      +- nd_next:
#                          (null node)
$ 
```

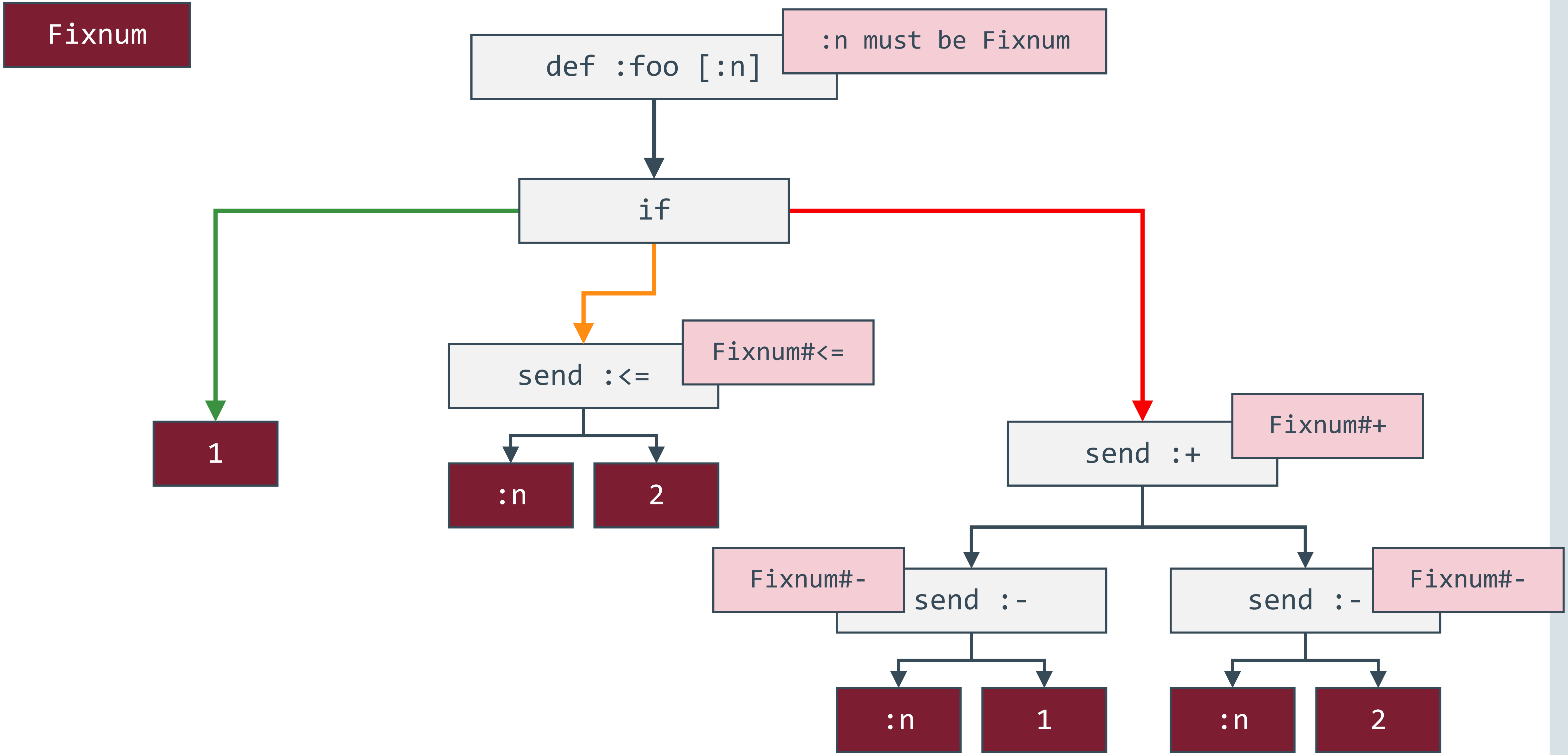

def :foo [:n]
if
1
send :<=
:n
2
send :+
send :-
send :-
:n
1
:n
2



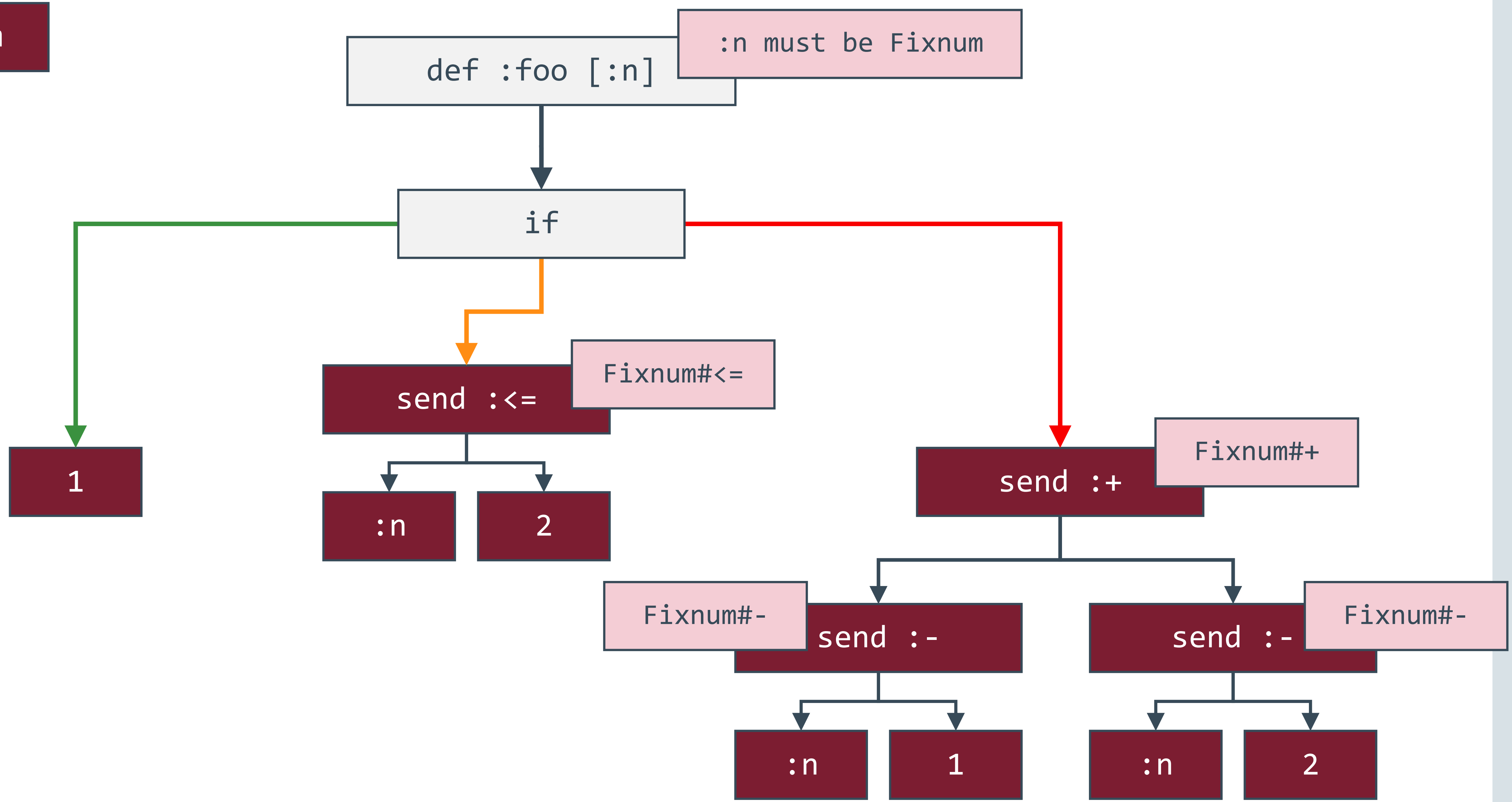
ORACLE

Fixnum

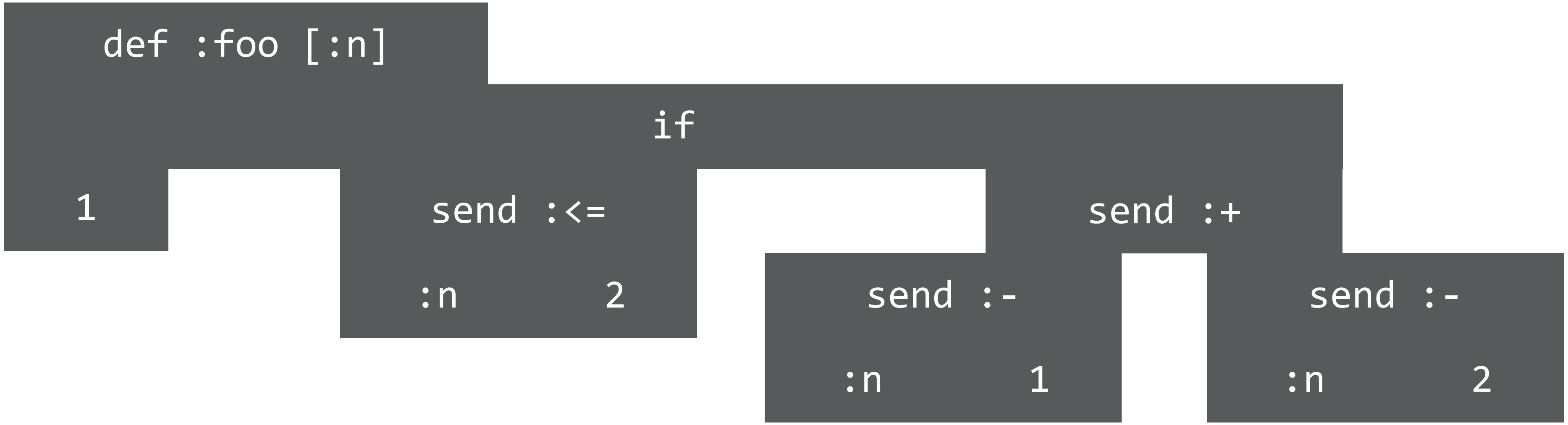



Fixnum
def :foo [:n]
:n must be Fixnum
if
1
send :<=
Fixnum#<=
:n
2
send :+
Fixnum#+
Fixnum#-
send :-
send :-
Fixnum#-
:n
1
:n
2
ORACLE

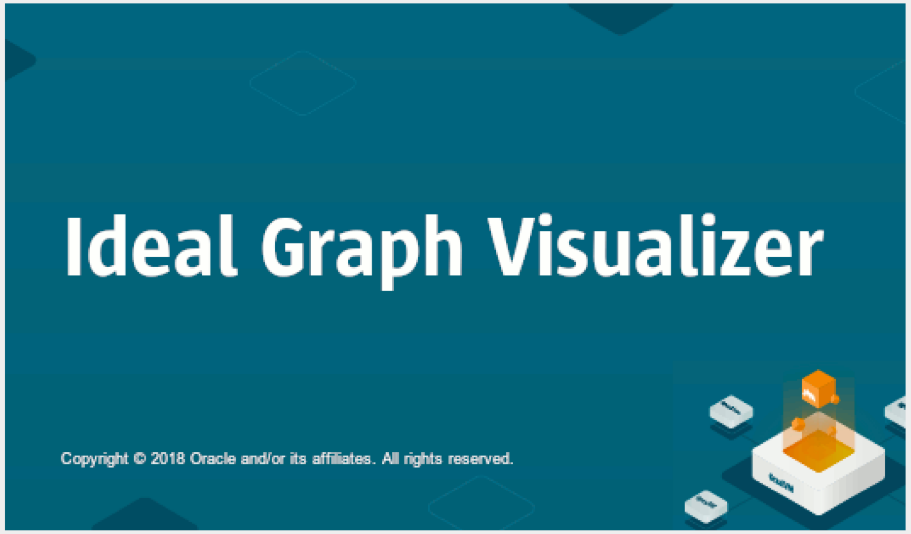
Fixnum



def :foo [:n]
:n must be Fixnum
if
send :<=
Fixnum#<=
1
:n
2
send :+
Fixnum#+
Fixnum#-
send :-
send :-
Fixnum#-
:n
1
:n
2


def :foo [:n]
if
1
send :<=
:n
2
send :+
send :-
:n
1
send :-
:n
2

About



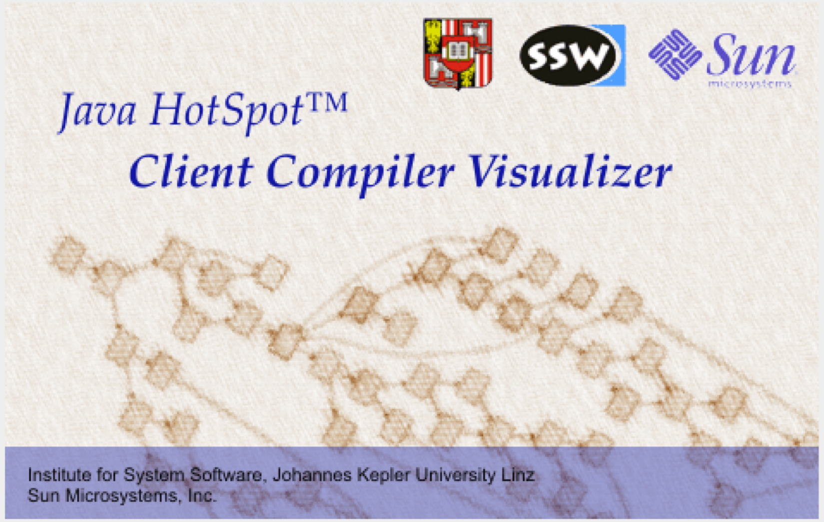
IdealGraphVisualizer, Copyright (c) 2013, 2018, Oracle and/or its affiliates.

IdealGraphVisualizer is based on Apache NetBeans from the Apache Software Foundation.

Product Version: IdealGraphVisualizer dev-19baa6a3fef1
Updates: Updates available
Java: 1.8.0_202; Java HotSpot(TM) GraalVM EE 1.0.0-rc14 25.202-b08-jvmci-0.56
Runtime: Java(TM) SE Runtime Environment 1.8.0_202-b08
System: Mac OS X version 10.14.3 running on x86_64; UTF-8; en_GB (idealgraphvisualizer)
User directory: /Users/chrisseaton/Library/Application Support/idealgraphvisualizer/0.27

Close

About

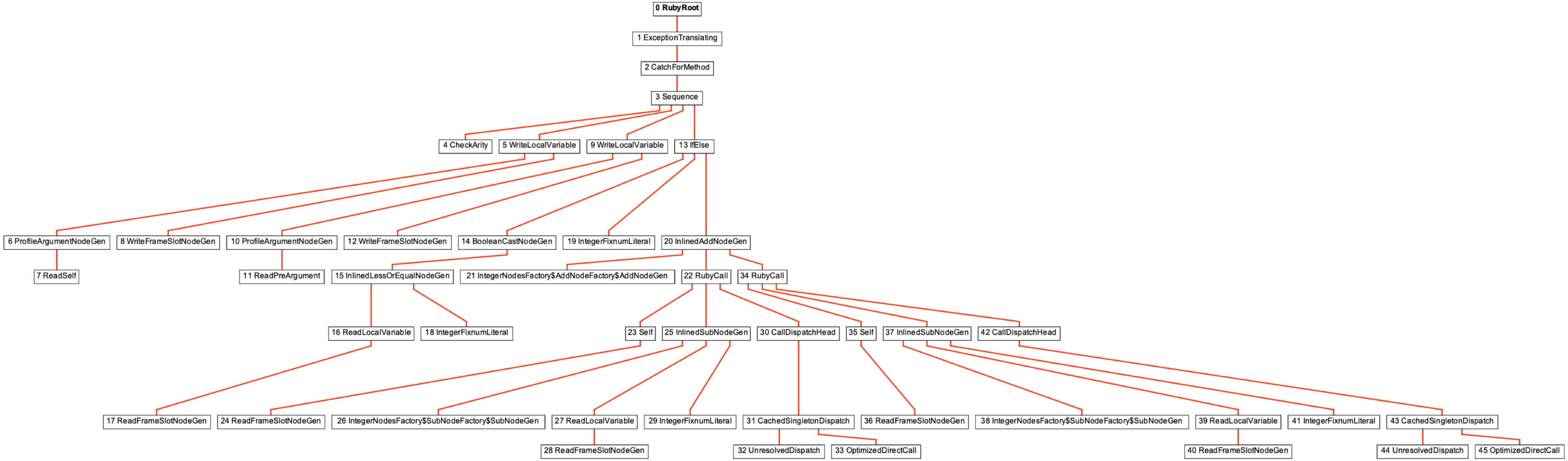


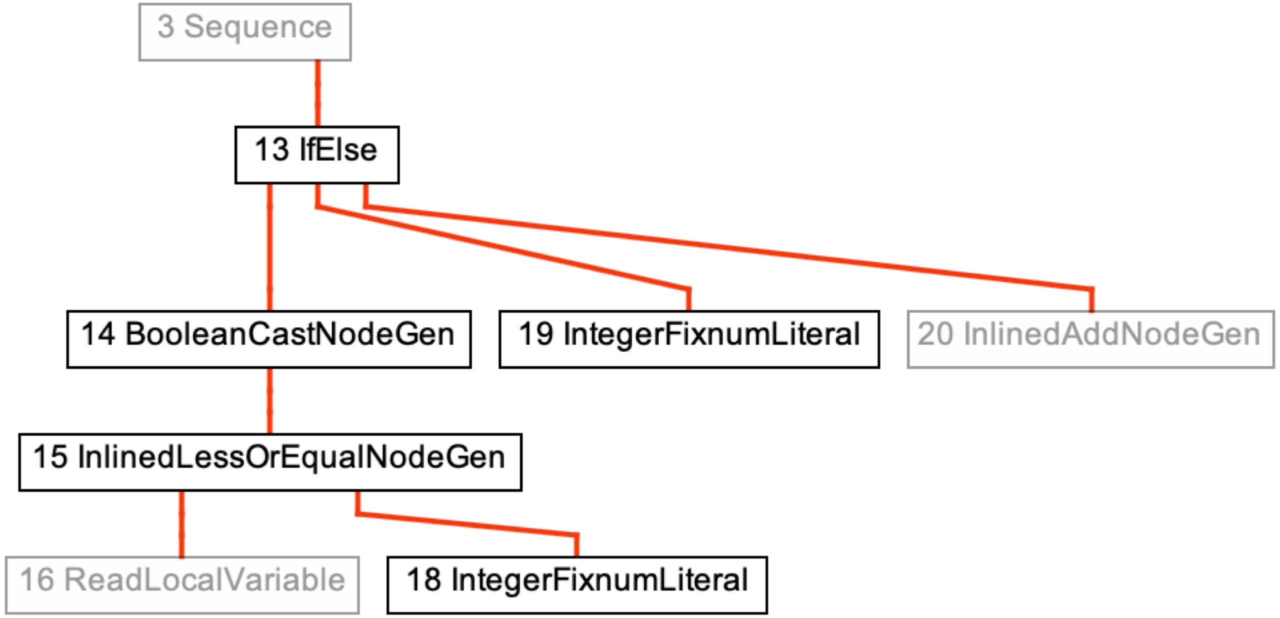
NetBeans IDE and NetBeans Platform are based on software from netbeans.org, which has been dual licensed under the Common Development and Distribution License (CDDL) and the GNU General Public License version 2 with Classpath exception. For more information, please visit www.netbeans.org.

Java: 1.8.0_201; Java HotSpot(TM) 64-Bit Server VM 25.201-b09
Runtime: Java(TM) SE Runtime Environment 1.8.0_201-b09
System: Mac OS X version 10.14.3 running on x86_64; UTF-8; en_GB (c1visualizer)
User directory: /Users/chrisseaton/Library/Application Support/c1visualizer/dev
Cache directory: /Users/chrisseaton/Library/Application Support/c1visualizer/dev/var/cache

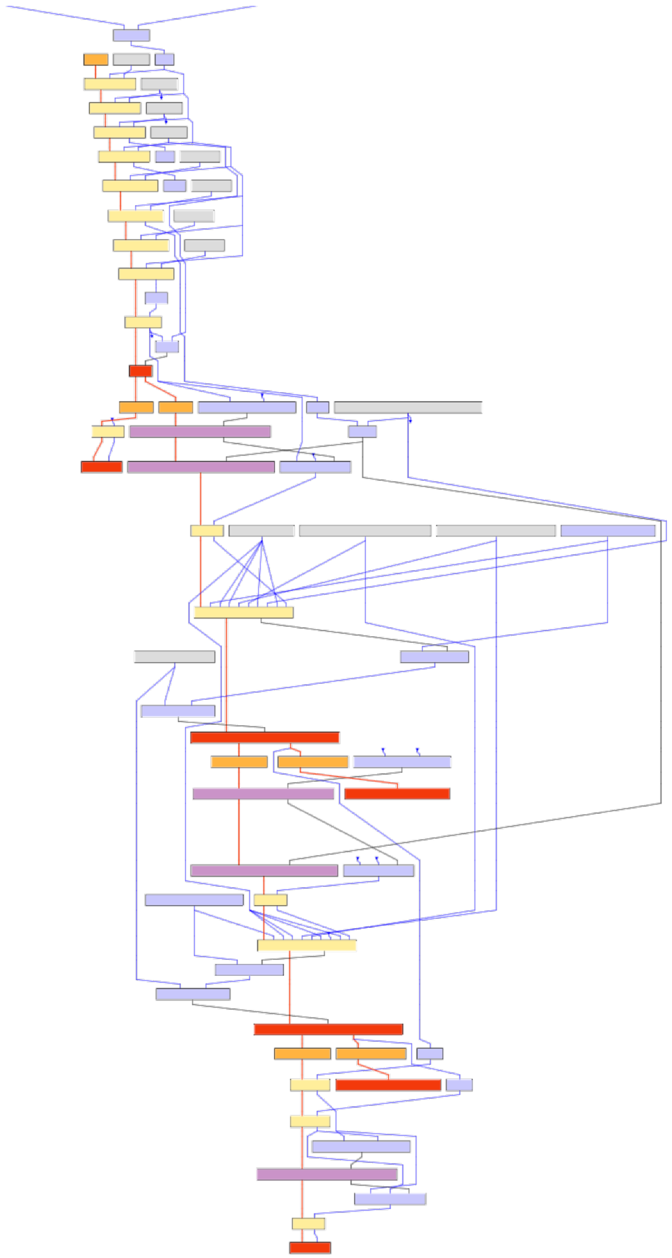
Close

ORACLE


0 RubyRoot
1 ExceptionTranslating
2 CatchForMethod
3 Sequence
4 CheckArity
5 WriteLocalVariable
9 WriteLocalVariable
13 IfElse
6 ProfileArgumentNodeGen
8 WriteFrameSlotNodeGen
10 ProfileArgumentNodeGen
12 WriteFrameSlotNodeGen
14 BooleanCastNodeGen
19 IntegerFixnumLiteral
20 InlinedAddNodeGen
7 ReadSelf
11 ReadPreArgument
15 InlinedLessOrEqualNodeGen
21 IntegerNodesFactory$AddNodeFactory$AddNodeGen
22 RubyCall
34 RubyCall
16 ReadLocalVariable
18 IntegerFixnumLiteral
23 Self
25 InlinedSubNodeGen
30 CallDispatchHead
35 Self
37 InlinedSubNodeGen
42 CallDispatchHead
17 ReadFrameSlotNodeGen
24 ReadFrameSlotNodeGen
26 IntegerNodesFactory$SubNodeFactory$SubNodeGen
27 ReadLocalVariable
29 IntegerFixnumLiteral
31 CachedSingletonDispatch
36 ReadFrameSlotNodeGen
38 IntegerNodesFactory$SubNodeFactory$SubNodeGen
39 ReadLocalVariable
41 IntegerFixnumLiteral
43 CachedSingletonDispatch
28 ReadFrameSlotNodeGen
32 UnresolvedDispatch
33 OptimizedDirectCall
40 ReadFrameSlotNodeGen
44 UnresolvedDispatch
45 OptimizedDirectCall


3 Sequence
13 IfElse
14 BooleanCastNodeGen
19 IntegerFixnumLiteral
20 InlinedAddNodeGen
15 InlinedLessOrEqualNodeGen
16 ReadLocalVariable
18 IntegerFixnumLiteral

ORACLE

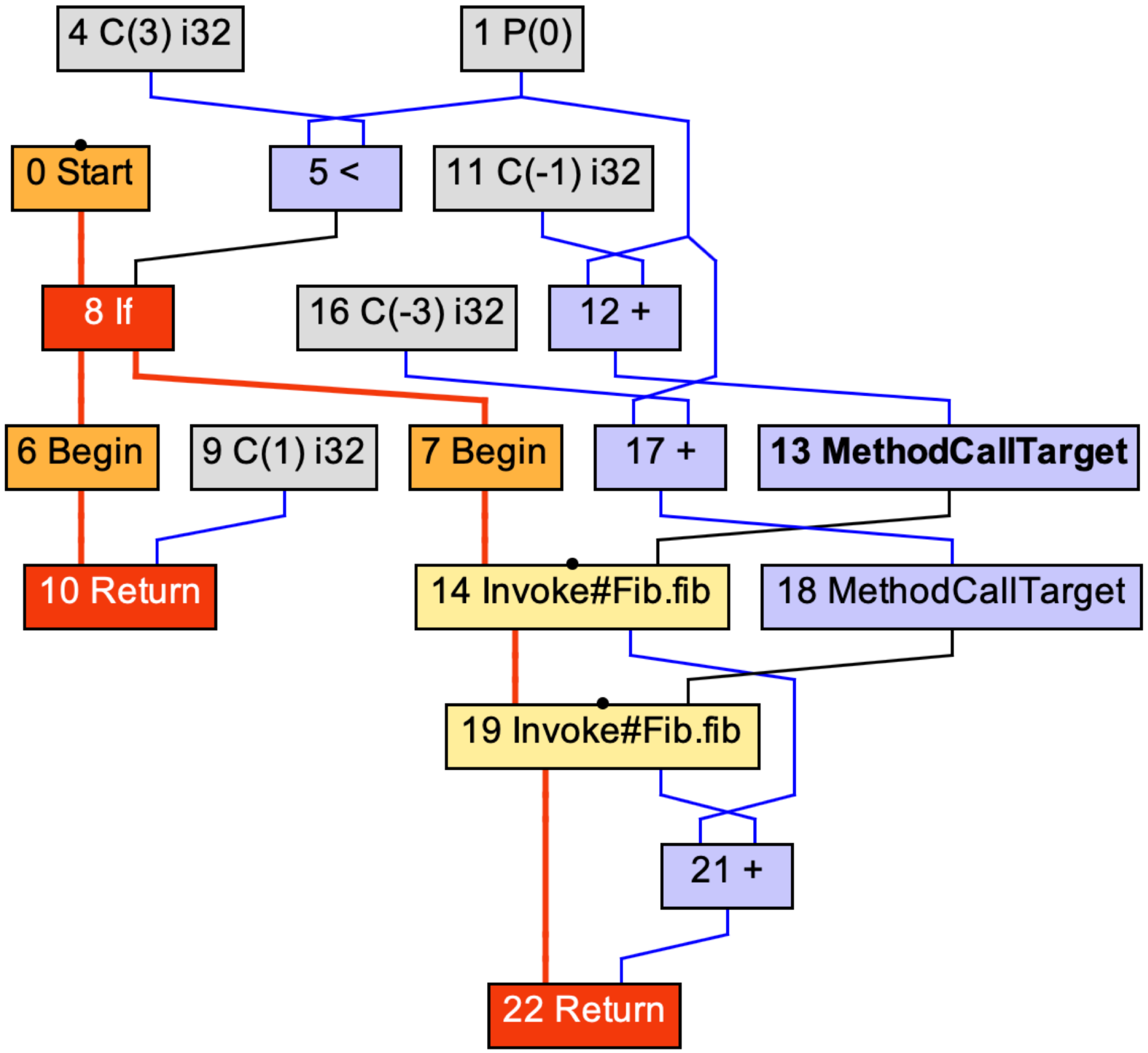


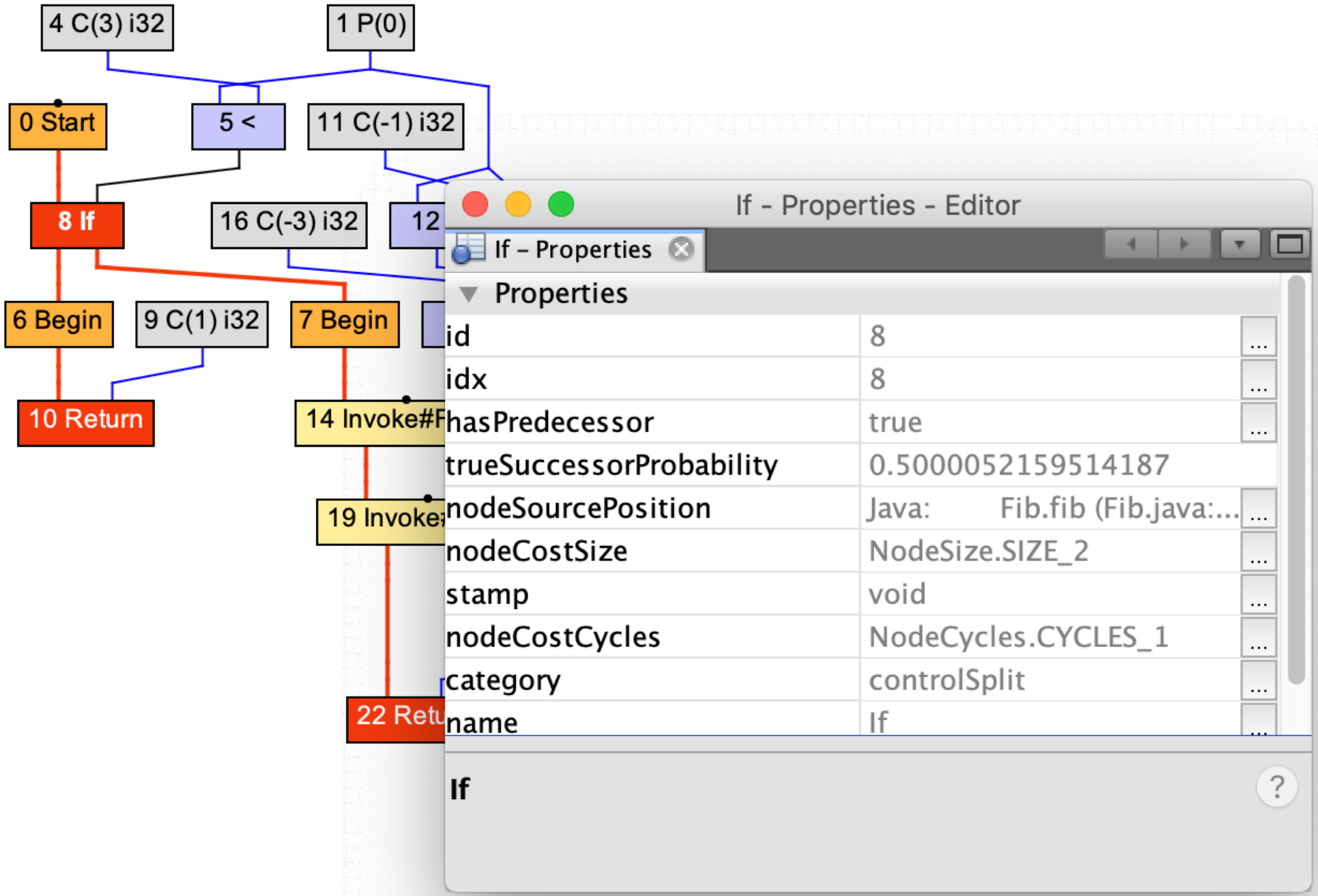
ORACLE

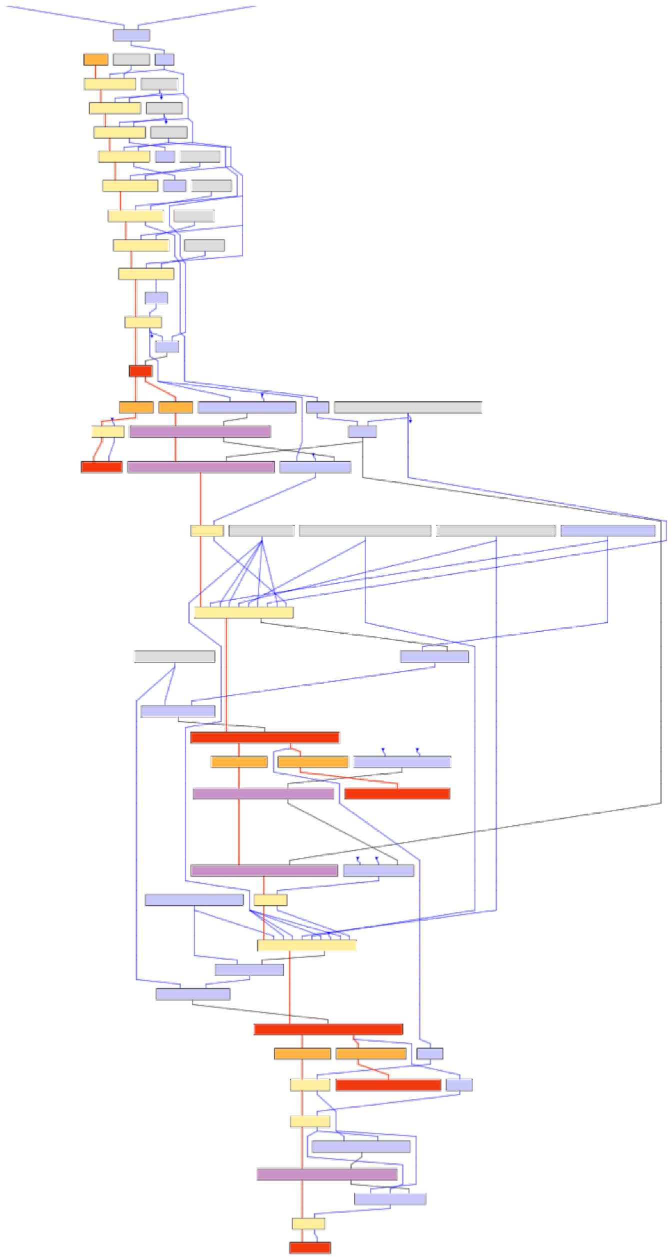
Fib.java

```java
class Fib {

  private static int fib(int n) {
    if (n <= 2) {
      return 1;
    } else {
      return fib(n - 1) + fib(n - 3);
    }
  }

  public static void main(String[] args) {
    while (true) {
      System.out.println(fib(30));
    }
  }

}
```

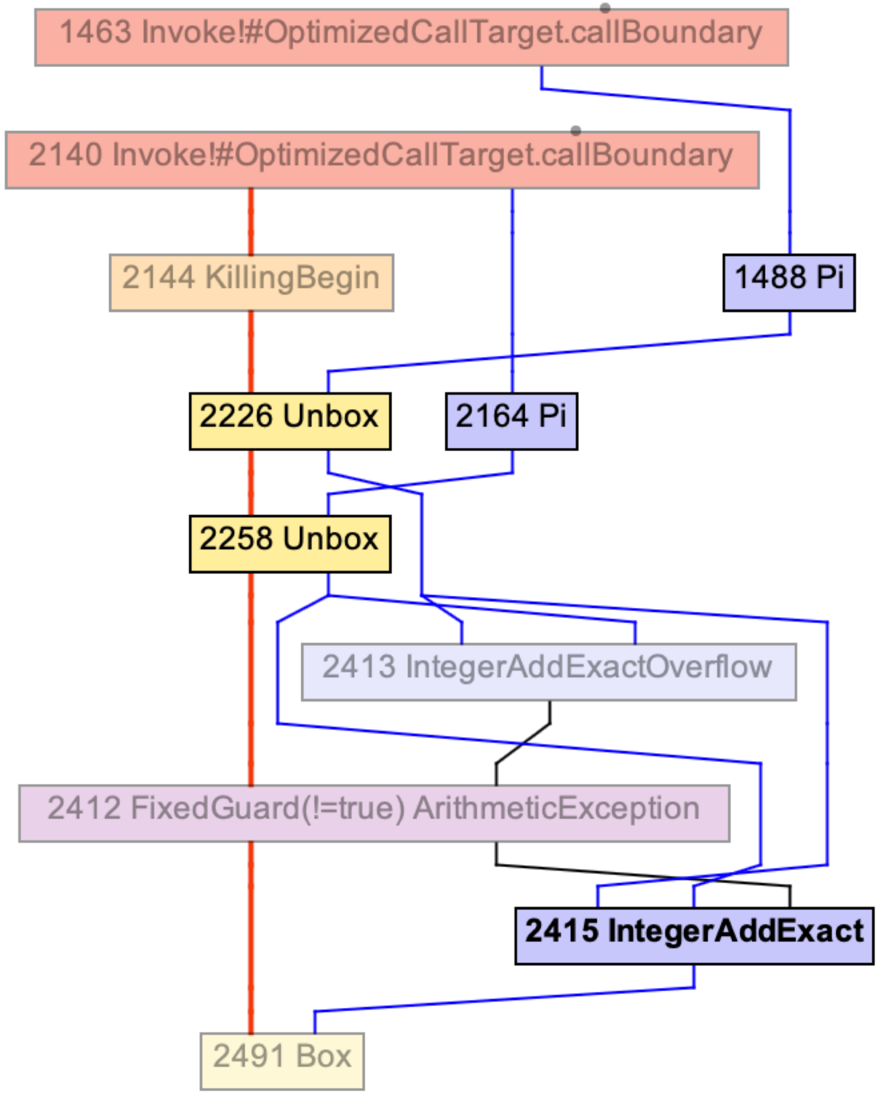
Line 1, Column 1 master 9 Spaces: 2 Java


4 C(3) i32
1 P(0)
0 Start
5 <
11 C(-1) i32
8 If
16 C(-3) i32
12 +
6 Begin
9 C(1) i32
7 Begin
17 +
13 MethodCallTarget
10 Return
14 Invoke#Fib.fib
18 MethodCallTarget
19 Invoke#Fib.fib
21 +
22 Return

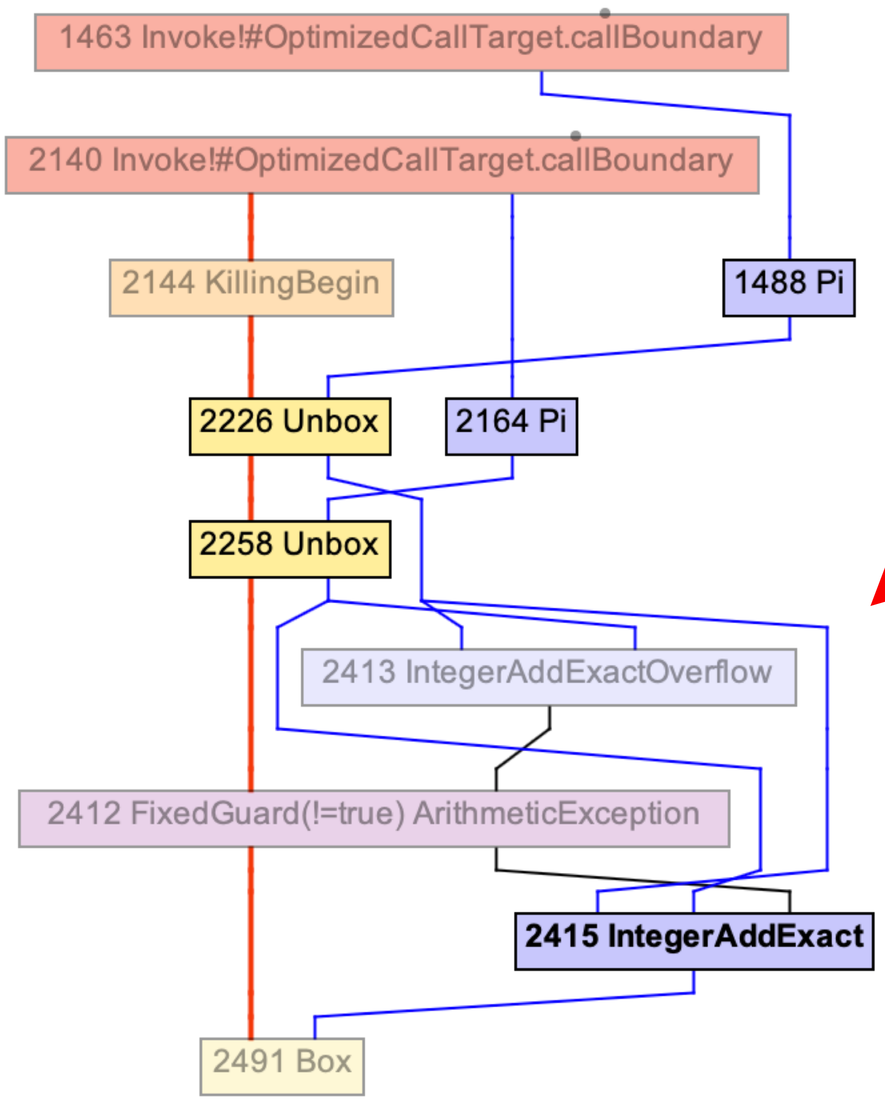

4 C(3) i32
1 P(0)
0 Start
5 <
11 C(-1) i32
8 If
16 C(-3) i32
12
6 Begin
9 C(1) i32
7 Begin
10 Return
14 Invoke#F
19 Invoke
22 Retu
If - Properties - Editor
If – Properties
Properties
id
8
idx
8
hasPredecessor
true
trueSuccessorProbability
0.5000052159514187
nodeSourcePosition
Java: Fib.fib (Fib.java:...
nodeCostSize
NodeSize.SIZE_2
stamp
void
nodeCostCycles
NodeCycles.CYCLES_1
category
controlSplit
name
If
If



Back to Ruby…



1463 Invoke!#OptimizedCallTarget.callBoundary
2140 Invoke!#OptimizedCallTarget.callBoundary
2144 KillingBegin
1488 Pi
2226 Unbox
2164 Pi
2258 Unbox
2413 IntegerAddExactOverflow
2412 FixedGuard(!=true) ArithmeticException
2415 IntegerAddExact
2491 Box



Become registers

Become machine instructions

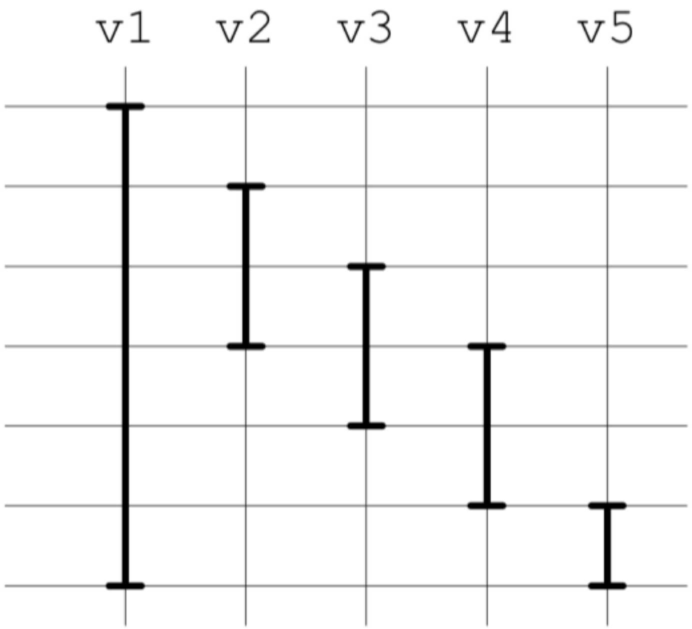
```
B0 -> B1,B2 [-1, -1]
_f__tid__d__instruction______org______________________________________________ (HIR)
 #  v0    d  StartNode stateAfter: - #next: v8
 ~  i1    d  Parameter
 ~  i4    d  Constant
 ~  v5    d  <@               .graalvm.compiler.nodes.calc.IntegerLessThanNode>@ x: i1 y: i4
 *  v8    d  If condition: v5 #trueSuccessor: v6 #falseSuccessor: v7

B1 <- B0 [-1, -1]
_f__tid__d__instruction______org______________________________________________ (HIR)
 #  v6    d  Begin #next: v10
 ~  i9    d  Constant
 *  v10   d  Return result: i9 memoryMap: -

B2 <- B0 [-1, -1]
_f__tid__d__instruction______org______________________________________________ (HIR)
 #  v7    d  Begin #next: i14
 ~  i11   d  Constant
 ~  i12   d  + x: i1 y: i11
    v23   d  HotSpotDirectCallTarget arguments: i12
    ?25   d  FrameState outerFrameState: - values: i1
 #  i14   d  Invoke stateAfter: - classInit: - callTarget: v23 stateDuring: ?25 #next: i19
 ~  i16   d  Constant
 ~  i17   d  + x: i1 y: i16
    v24   d  HotSpotDirectCallTarget arguments: i17
    ?26   d  FrameState outerFrameState: - values: - i14
 #  i19   d  Invoke stateAfter: - classInit: - callTarget: v24 stateDuring: ?26 #next: v22
 ~  i21   d  + x: i14 y: i19
 *  v22   d  Return result: i21 memoryMap: -
```

```java
@Override
public void generate(NodeLIRBuilderTool generator) {
    generator.setResult(this, generateArithmetic(generator));
    generator.emitOverflowCheckBranch(getOverflowSuccessor(), getNext(), stamp, probability(getOverflowSuccessor()));
}
```

```java
public final void jcc(ConditionFlag cc, Label l) {
    assert (0 <= cc.getValue()) && (cc.getValue() < 16) : "illegal cc";
    if (l.isBound()) {
        jcc(cc, l.position(), false);
    } else {
        // Note: could eliminate cond. jumps to this jump if condition
        // is the same however, seems to be rather unlikely case.
        // Note: use jccb() if label to be bound is very close to get
        // an 8-bit displacement
        l.addPatchAt(position());
        emitByte(0x0F);
        emitByte(0x80 | cc.getValue());
        emitInt(0);
    }

}
```

```
0x000000011a63b6e4: cmp    r10d,0x3
0x000000011a63b6e8: jl     0x000000011a63ba3f
0x000000011a63b703: sub    r13d,0x1
0x000000011a63b707: jo     0x000000011a63be54
0x000000011a63b897: call   0x000000011950bde0
0x000000011a63b8a8: sub    ebx,0x2
0x000000011a63b8ab: jo     0x000000011a63bda2
0x000000011a63b9af: call   0x000000011950bde0
0x000000011a63b9c5: add    r13d,eax
0x000000011a63b9c8: jo     0x000000011a63bd7b
0x000000011a63ba35: test   DWORD PTR [rip+0xfffffffff0e5e5cb],eax
0x000000011a63ba3e: ret
0x000000011a63ba3f: movabs rax,0x6c012dd20
0x000000011a63ba58: test   DWORD PTR [rip+0xfffffffff0e5e5a8],eax
0x000000011a63ba61: ret
```

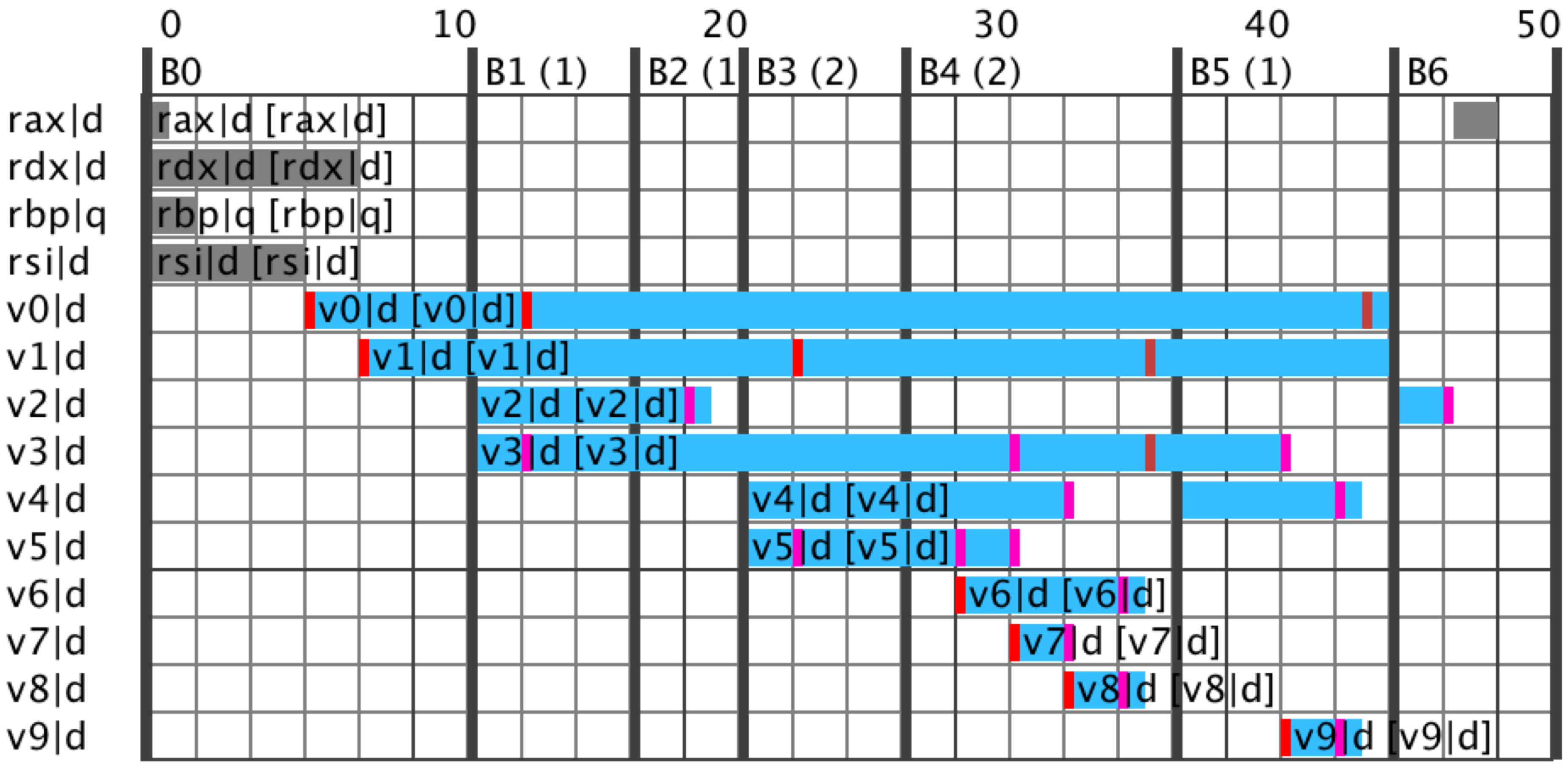
```
(1)   v1 = 10
(2)   v2 = 20
(3)   v3 = v1 + v2
(4)   v4 = v2 + v3
(5)   v1 = v3 + v4
(6)   v5 = v4 + v1
(7)   return v1 + v5
```

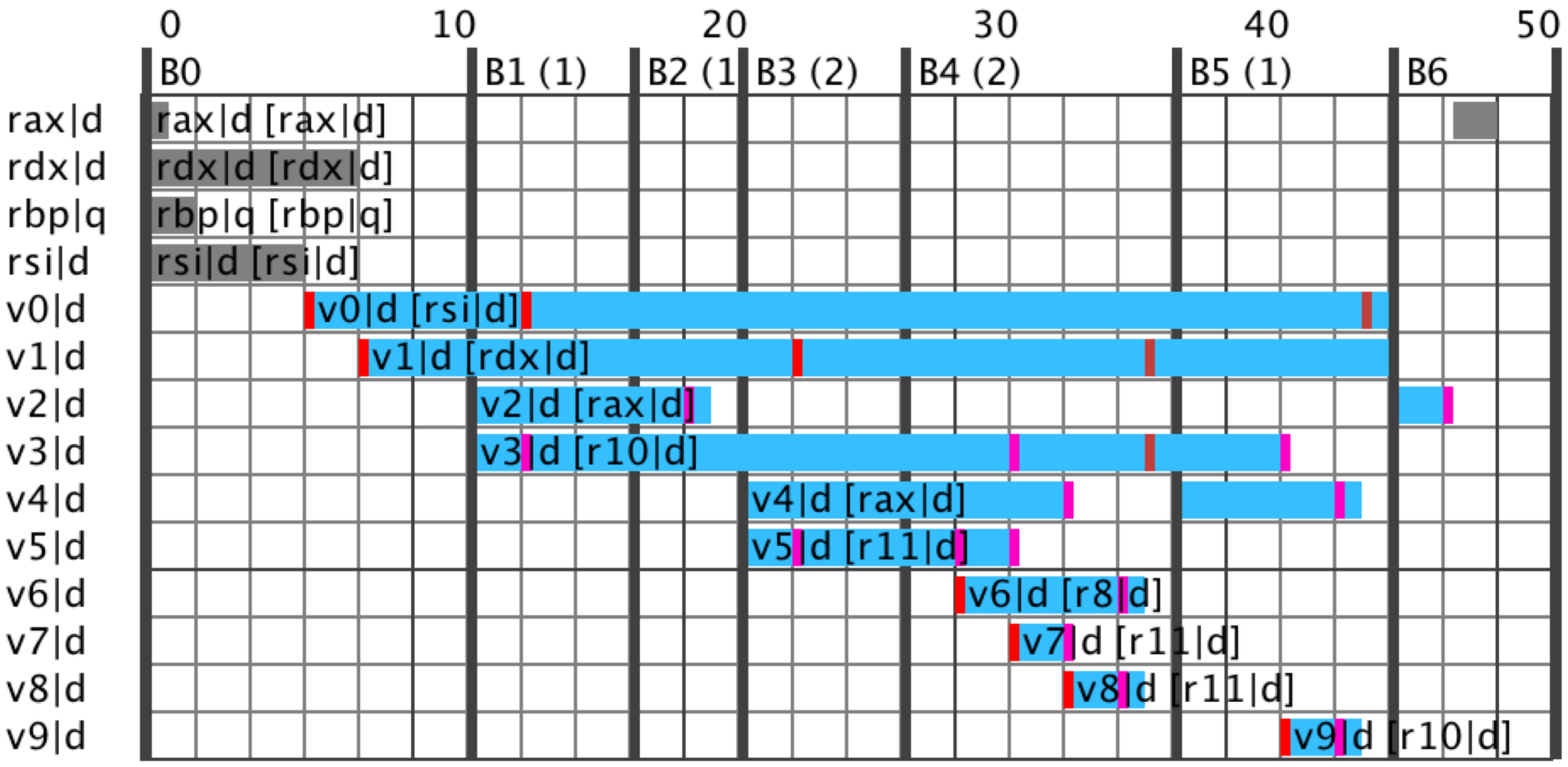


Linear Scan Register Allocation for the Java HotSpot™ Client Compiler

Christian Wimmer

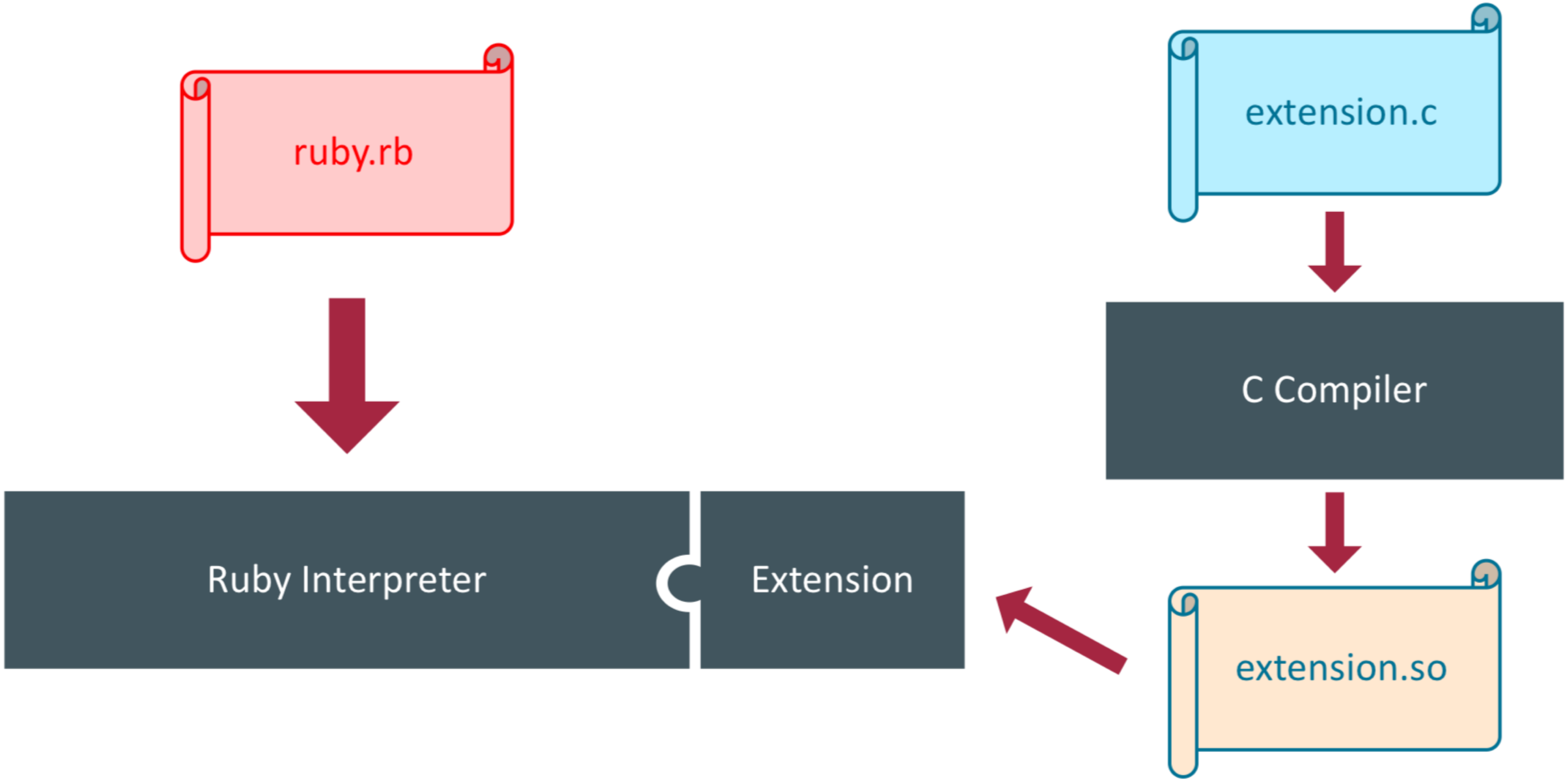
ORACLE®

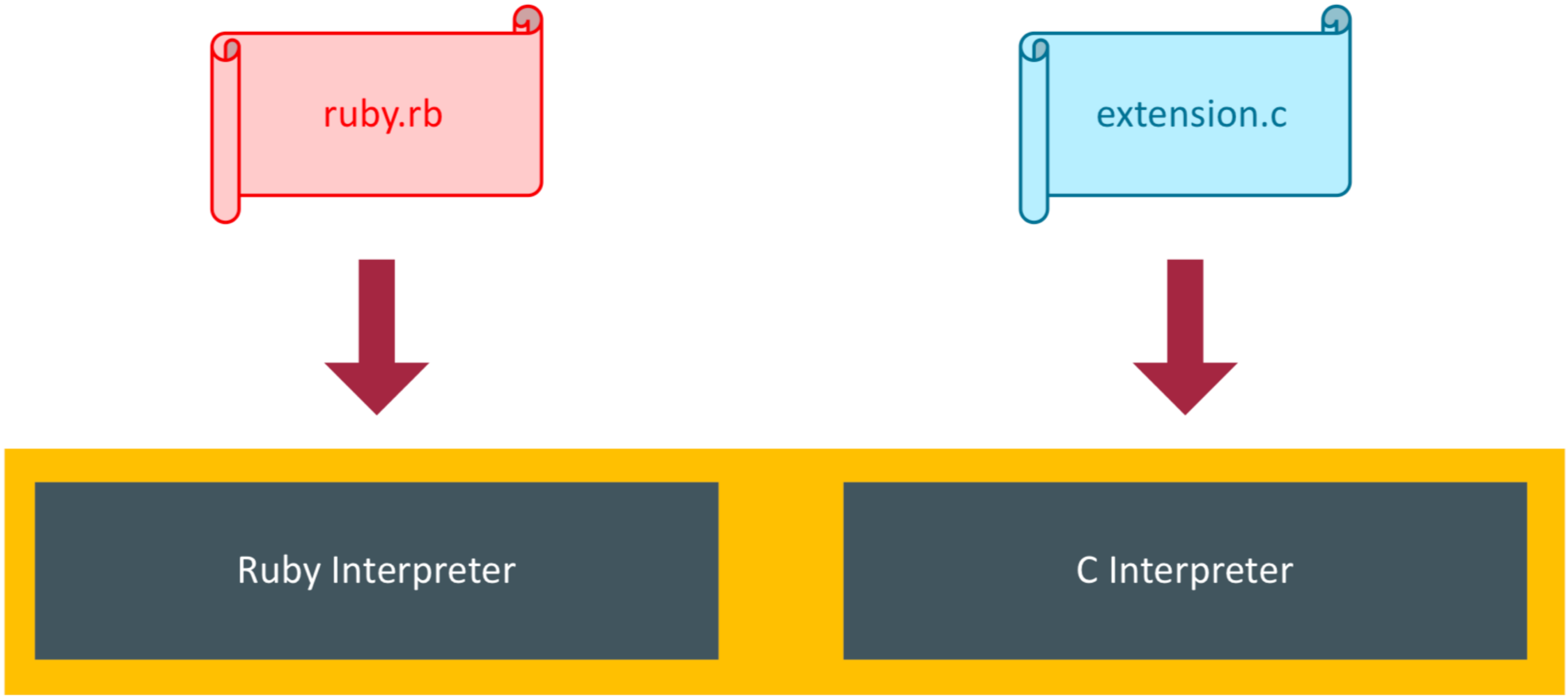

0
10
20
30
40
50
B0
B1 (1)
B2 (1
B3 (2)
B4 (2)
B5 (1)
B6
rax|d
rdx|d
rbp|q
rsi|d
v0|d
v1|d
v2|d
v3|d
v4|d
v5|d
v6|d
v7|d
v8|d
v9|d
rax|d [rax|d]
rdx|d [rdx|d]
rbp|q [rbp|q]
rsi|d [rsi|d]
v0|d [v0|d]
v1|d [v1|d]
v2|d [v2|d]
v3|d [v3|d]
v4|d [v4|d]
v5|d [v5|d]
v6|d [v6|d]
v7|d [v7|d]
v8|d [v8|d]
v9|d [v9|d]

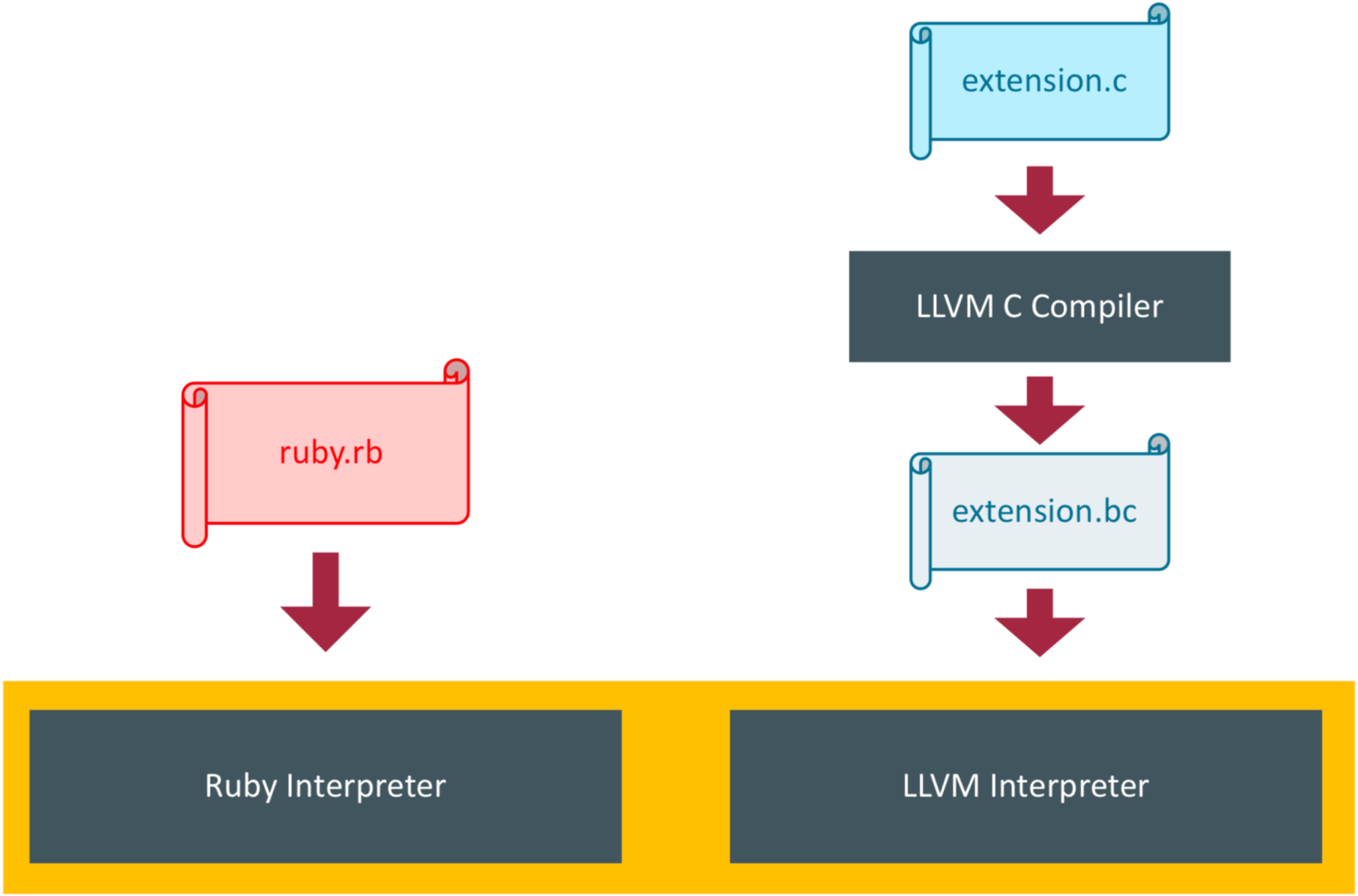

0
10
20
30
40
50
B0
B1 (1)
B2 (1
B3 (2)
B4 (2)
B5 (1)
B6
rax|d
rdx|d
rbp|q
rsi|d
v0|d
v1|d
v2|d
v3|d
v4|d
v5|d
v6|d
v7|d
v8|d
v9|d
rax|d [rax|d]
rdx|d [rdx|d]
rbp|q [rbp|q]
rsi|d [rsi|d]
v0|d [rsi|d]
v1|d [rdx|d]
v2|d [rax|d]
v3|d [r10|d]
v4|d [rax|d]
v5|d [r11|d]
v6|d [r8|d]
v7|d [r11|d]
v8|d [r11|d]
v9|d [r10|d]

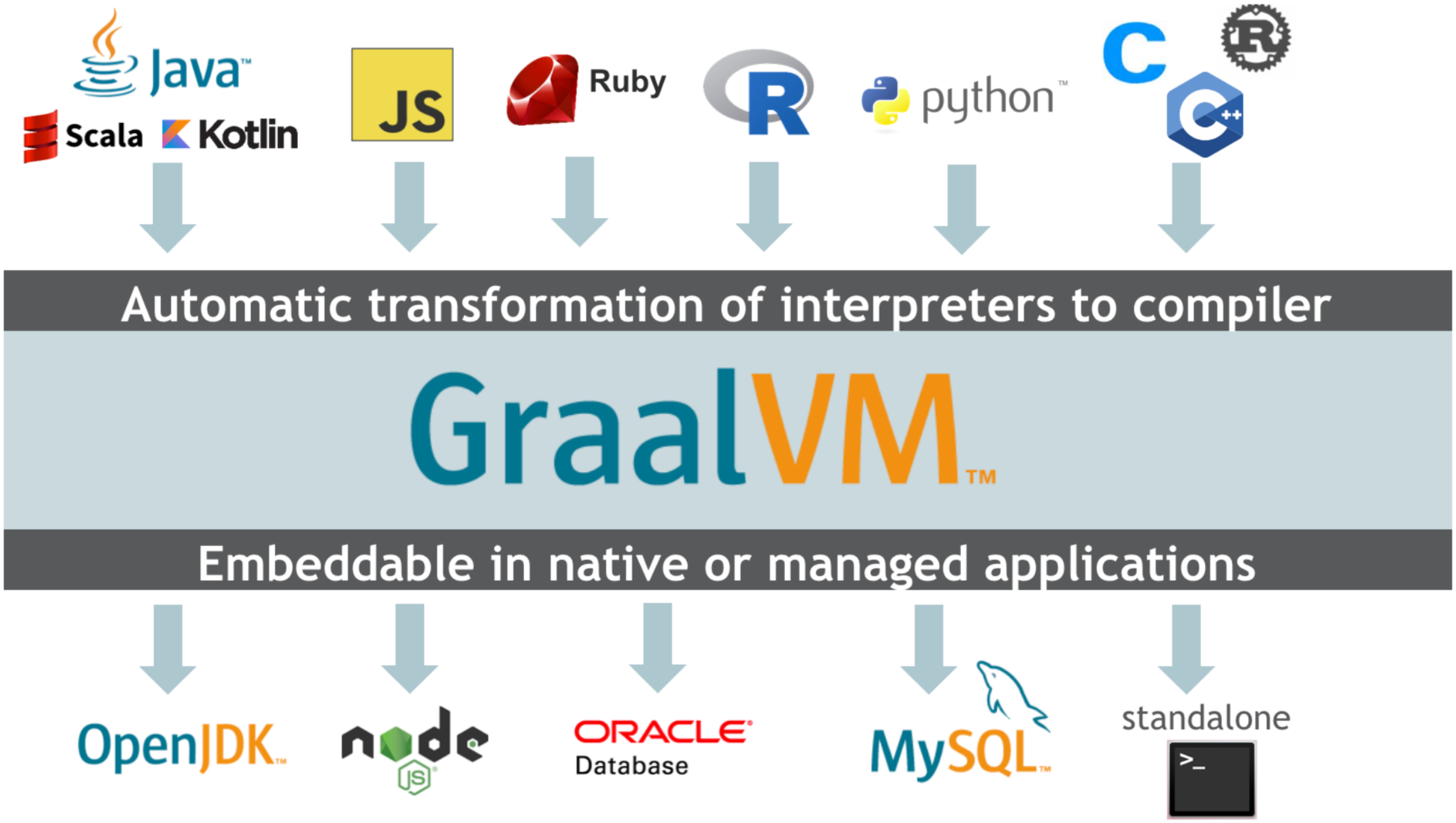
```
0x000000011a63b6e4: cmp    r10d,0x3
0x000000011a63b6e8: jl     0x000000011a63ba3f
0x000000011a63b703: sub    r13d,0x1
0x000000011a63b707: jo     0x000000011a63be54
0x000000011a63b897: call   0x000000011950bde0
0x000000011a63b8a8: sub    ebx,0x2
0x000000011a63b8ab: jo     0x000000011a63bda2
0x000000011a63b9af: call   0x000000011950bde0
0x000000011a63b9c5: add    r13d,eax
0x000000011a63b9c8: jo     0x000000011a63bd7b
0x000000011a63ba35: test   DWORD PTR [rip+0xffffffff0e5e5cb],eax
0x000000011a63ba3e: ret
0x000000011a63ba3f: movabs rax,0x6c012dd20
0x000000011a63ba58: test   DWORD PTR [rip+0xffffffff0e5e5a8],eax
0x000000011a63ba61: ret
```

There’s more to TruffleRuby


ruby.rb
extension.c
C Compiler
Ruby Interpreter
Extension
extension.so


ruby.rb
extension.c
Ruby Interpreter
C Interpreter


extension.c
LLVM C Compiler
ruby.rb
extension.bc
Ruby Interpreter
LLVM Interpreter

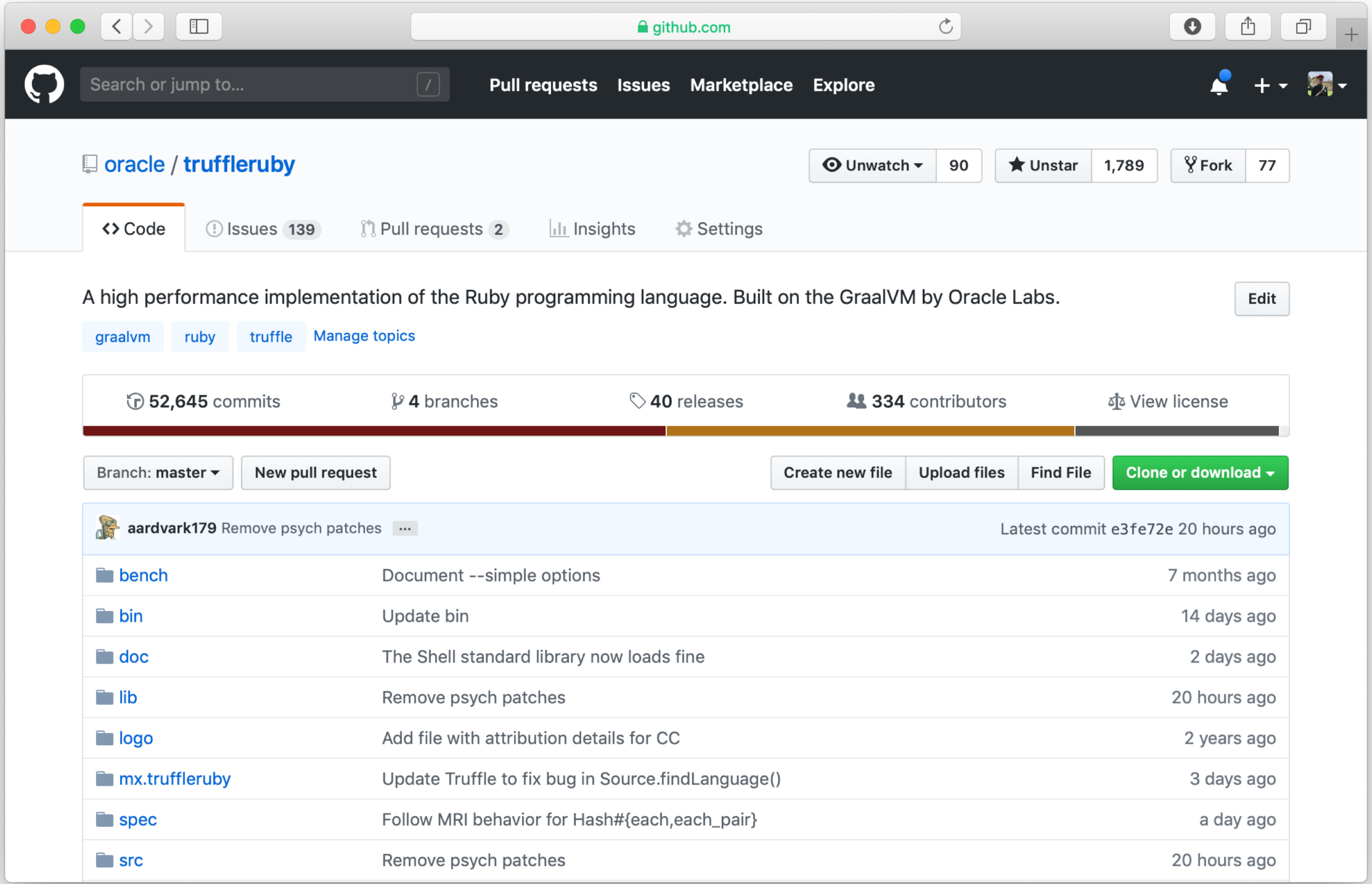

Java
Scala
Kotlin
JS
Ruby
R
python
C
C++
Automatic transformation of interpreters to compiler
GraalVM
Embeddable in native or managed applications
OpenJDK
node JS
ORACLE Database
MySQL
standalone
>_

Summing up

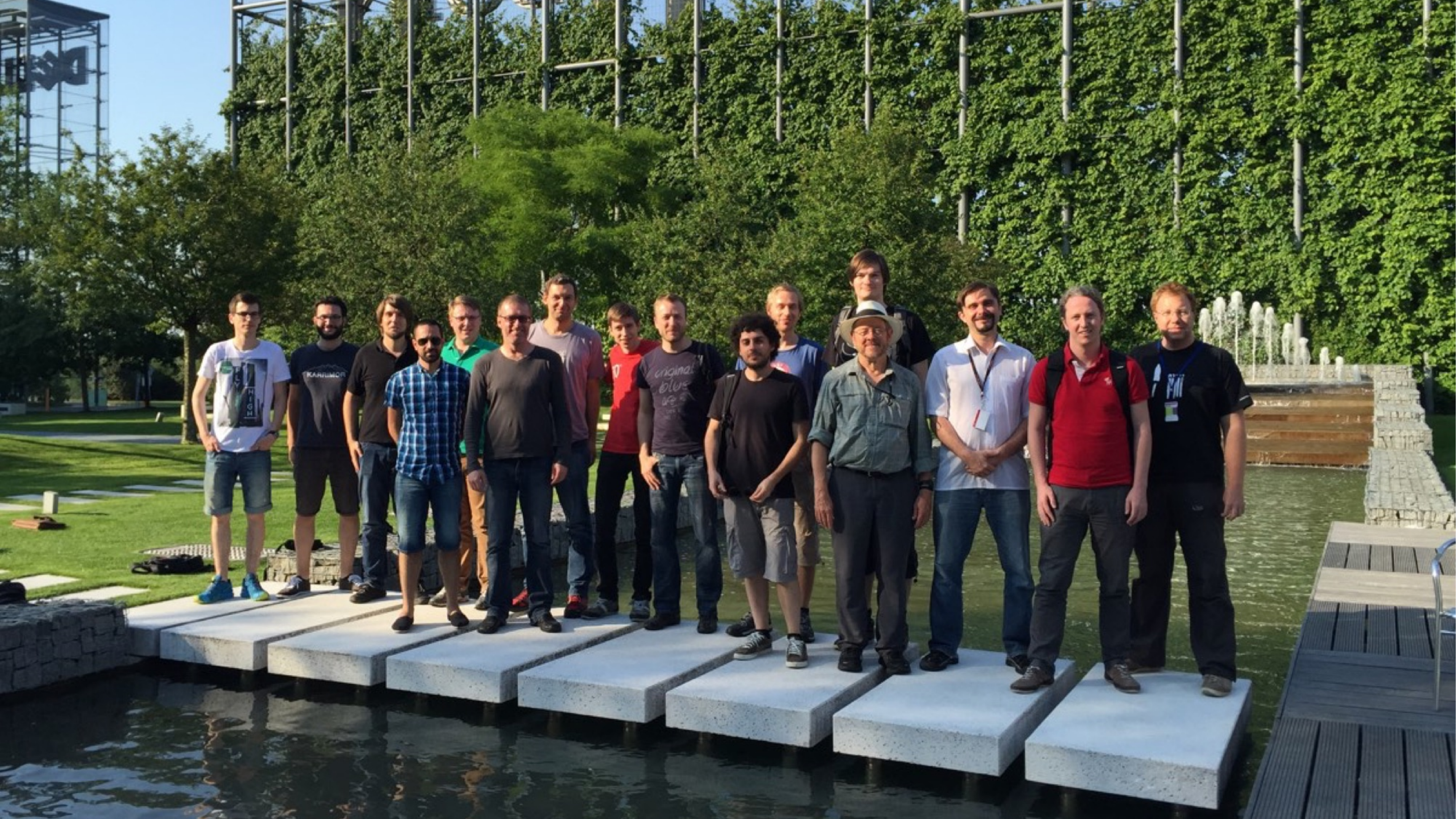
Documents — -bash — 84×15

```
$ rbenv install truffleruby-1.0.0-rc14
$ rbenv shell truffleruby-1.0.0-rc14
$ ruby -v
truffleruby 1.0.0-rc14, like ruby 2.6.2, GraalVM CE Native [x86_64-darwin]
$
```

ORACLE®


github.com
Search or jump to...
Pull requests
Issues
Marketplace
Explore
oracle / truffleruby
Unwatch 90
Unstar 1,789
Fork 77
Code
Issues 139
Pull requests 2
Insights
Settings
A high performance implementation of the Ruby programming language. Built on the GraalVM by Oracle Labs.
Edit
graalvm
ruby
truffle
Manage topics
52,645 commits
4 branches
40 releases
334 contributors
View license
Branch: master
New pull request
Create new file
Upload files
Find File
Clone or download
aardvark179 Remove psych patches
Latest commit e3fe72e 20 hours ago
bench
Document --simple options
7 months ago
bin
Update bin
14 days ago
doc
The Shell standard library now loads fine
2 days ago
lib
Remove psych patches
20 hours ago
logo
Add file with attribution details for CC
2 years ago
mx.truffleruby
Update Truffle to fix bug in Source.findLanguage()
3 days ago
spec
Follow MRI behavior for Hash#{each,each_pair}
a day ago
src
Remove psych patches
20 hours ago


graalvm.org
Home
Docs
Downloads
Community
Star
8,222
GraalVM™
Run Programs Faster Anywhere
WHY GRAALVM
GET STARTED
JAVA
Standalone
MySQL
GraalVM
OpenJDK
Oracle Database
Node
High-performance polyglot VM
GraalVM is a universal virtual machine for running applications written in JavaScript, Python, Ruby, R, JVM-based languages like Java, Scala, Kotlin, Clojure, and LLVM-based languages such as C and C++.
GraalVM removes the isolation between programming languages and enables interoperability in a shared


FLY
HIGH
KARRIMOR
original
blue

Team

Oracle
Florian Angerer
Danilo Ansaloni
Stefan Anzinger
Martin Balin
Cosmin Basca
Daniele Bonetta
Dušan Bálek
Matthias Brantner
Lucas Braun
Petr Chalupa
Jürgen Christ
Laurent Daynès
Gilles Duboscq
Svatopluk Dědic
Martin Entlicher
Pit Fender
Francois Farquet
Brandon Fish
Matthias Grimmer
Christian Häubl
Peter Hofer
Bastian Hossbach
Christian Humer
Tomáš Hůrka
Mick Jordan

Oracle (continued)
Vojin Jovanovic
Anantha Kandukuri
Harshad Kasture
Cansu Kaynak
Peter Kessler
Duncan MacGregor
Jiří Maršík
Kevin Menard
Miloslav Metelka
Tomáš Myšík
Petr Pišl
Oleg Pliss
Jakub Podlešák
Aleksandar Prokopec
Tom Rodriguez
Roland Schatz
Benjamin Schlegel
Chris Seaton
Jiří Sedláček
Doug Simon
Štěpán Šindelář
Zbyněk Šlajchrt
Boris Spasojevic
Lukas Stadler
Codrut Stancu

Oracle (continued)
Jan Štola
Tomáš Stupka
Farhan Tauheed
Jaroslav Tulach
Alexander Ulrich
Michael Van De Vanter
Aleksandar Vitorovic
Christian Wimmer
Christian Wirth
Paul Wögerer
Mario Wolczko
Andreas Wöß
Thomas Würthinger
Tomáš Zezula
Yudi Zheng

Red Hat
Andrew Dinn
Andrew Haley

Intel
Michael Berg

Twitter
Chris Thalinger

Oracle Interns
Brian Belleville
Ondrej Douda
Juan Fumero
Miguel Garcia
Hugo Guiroux
Shams Imam
Berkin Ilbeyi
Hugo Kapp
Alexey Karyakin
Stephen Kell
Andreas Kunft
Volker Lanting
Gero Leinemann
Julian Lettner
Joe Nash
Tristan Overney
Aleksandar Pejovic
David Piorkowski
Philipp Riedmann
Gregor Richards
Robert Seilbeck
Rifat Shariyar

Oracle Alumni
Erik Eckstein
Michael Haupt
Christos Kotselidis
David Leibs
Adam Welc
Till Westmann

JKU Linz
Hanspeter Mössenböck
Benoit Daloze
Josef Eisl
Thomas Feichtinger
Josef Haider
Christian Huber
David Leopoldseder
Stefan Marr
Manuel Rigger
Stefan Rumzucker
Bernhard Urban

TU Berlin:
Volker Markl
Andreas Kunft
Jens Meiners
Tilmann Rabl

University of Edinburgh
Christophe Dubach
Juan José Fumero Alfonso
Ranjeet Singh
Toomas Remmelg

LaBRI
Floréal Morandat

University of California, Irvine
Michael Franz
Yeoul Na
Mohaned Qunaibit
Gulfem Savrun Yeniceri
Wei Zhang

Purdue University
Jan Vitek
Tomas Kalibera
Petr Maj
Lei Zhao

T. U. Dortmund
Peter Marwedel
Helena Kotthaus
Ingo Korb

University of California, Davis
Duncan Temple Lang
Nicholas Ulle

University of Lugano, Switzerland
Walter Binder
Sun Haiyang

ORACLE

Questions

Safe Harbor Statement

The preceding is intended to provide some insight into a line of research in Oracle Labs. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. Oracle reserves the right to alter its development plans and practices at any time, and the development, release, and timing of any features or functionality described in connection with any Oracle product or service remains at the sole discretion of Oracle. Any views expressed in this presentation are my own and do not necessarily reflect the views of Oracle.

Integrated Cloud

Applications & Platform Services

ORACLE

ORACLE®