北陸Ruby会議02 ／ 2026.11.14 ／ 福井県立恐竜博物館

diff の範囲だけ実行時トレースすると、コードレビューに役立つのでは？

笹田耕一
STORES, Inc. ／ Ruby インタプリタ開発



今日のテーマ

プログラムの状態の発掘

AI 時代に必要な、プログラムの観察手法とは？



自己紹介

笹田耕一

Ruby

Ruby インタプリタ開発

STORES, Inc.（2023年〜）

これまで手を入れてきたところ

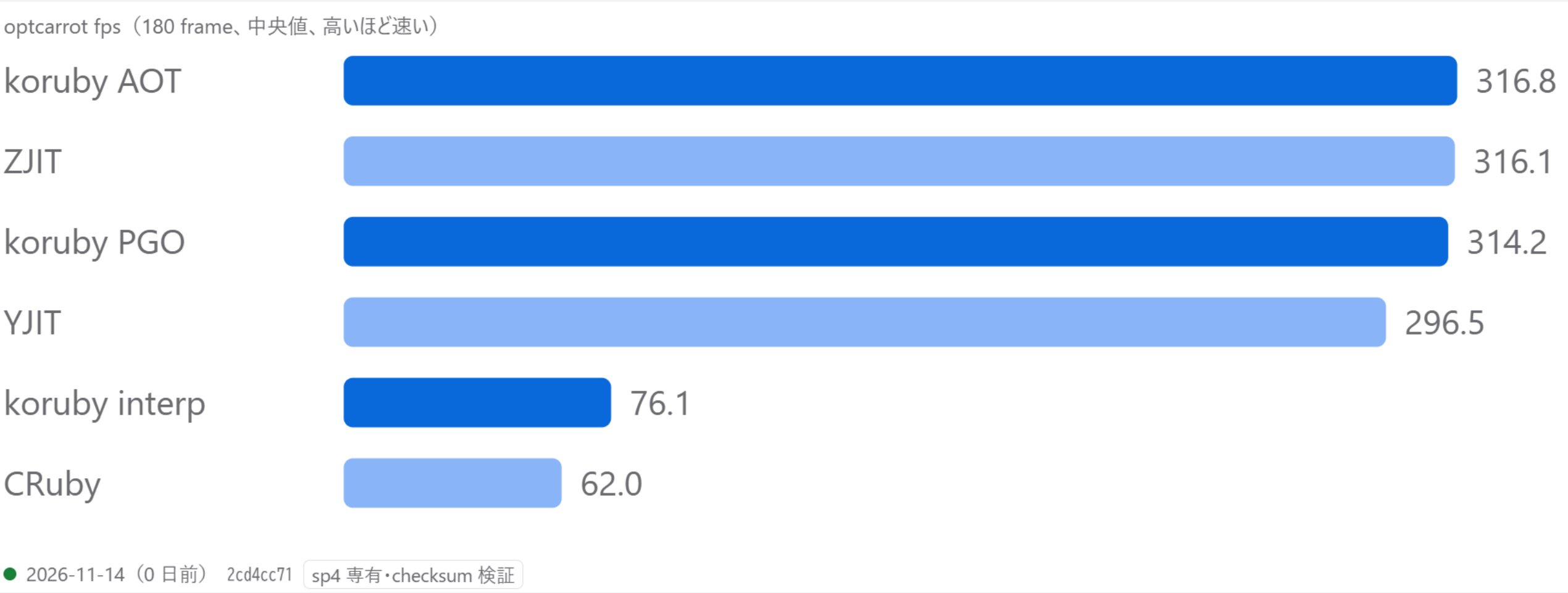
- VM（2004年〜）
- GC
- Ractor
- ・・・

本

本屋

Newly created Ruby interpreter with partial evaluation technique

ASTro/koruby



https://claude.ai/artifact/CX567HUfYvzC177ghpJzYi

SECTION 01

プログラムの観察

未来・現在・過去、どの時点を見るか

プログラムを観察する

未来

どう動くのか

静的解析 ／ 形式検証

現在

どう動いているのか

デバッガ ／ トレース

過去

どう動いたのか

ログ ／ プロファイル

未来 どう動くのか

静

静的解析

- 型検査
- 抽象解釈
- …

検

形式検証

- モデル検査
- 演繹的検証
- …

（難しいので今日はやりません）

現在 どう動いているのか

見

実行している様子を見る

止

実行を止めて様子を見る

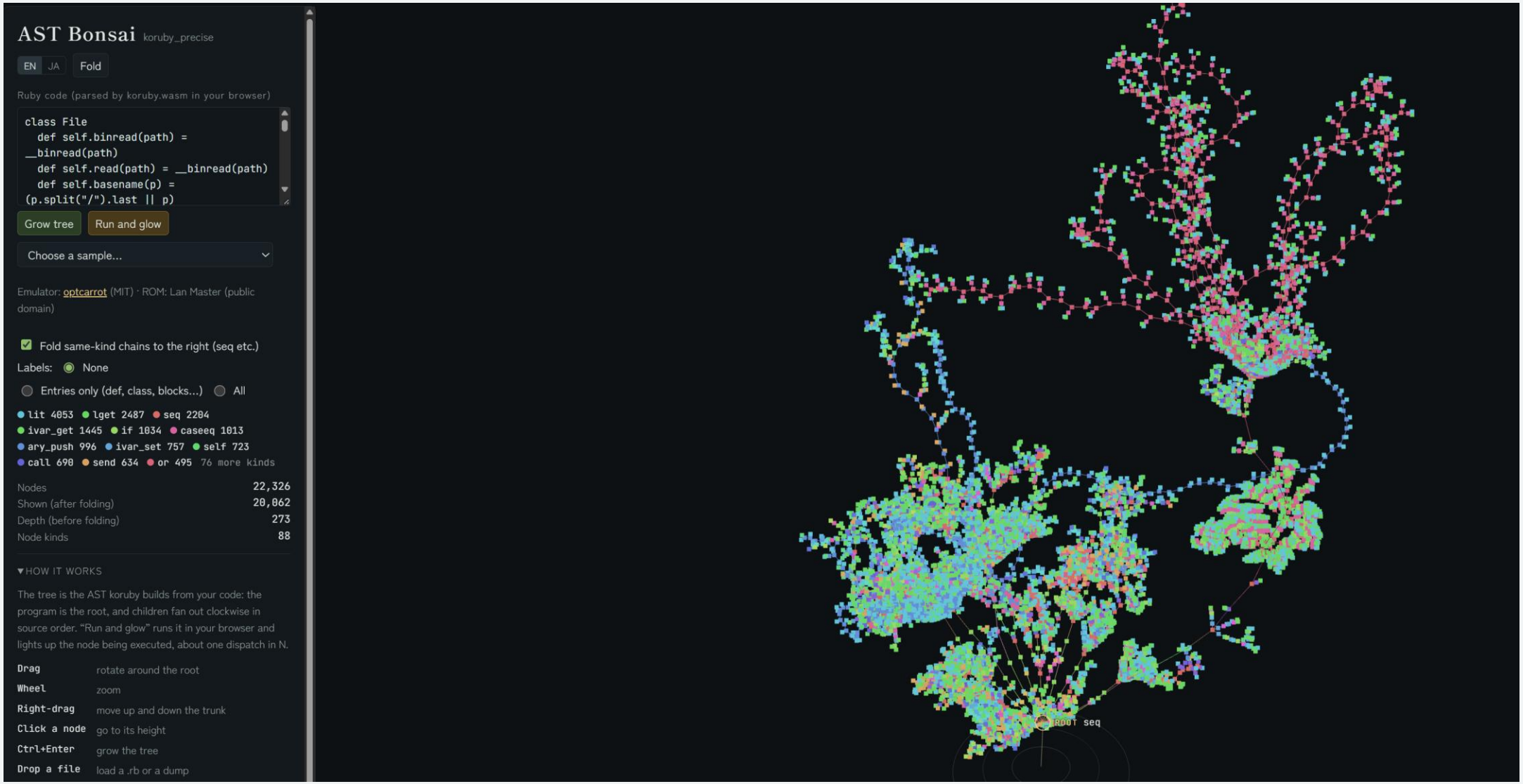
現在

dopairb



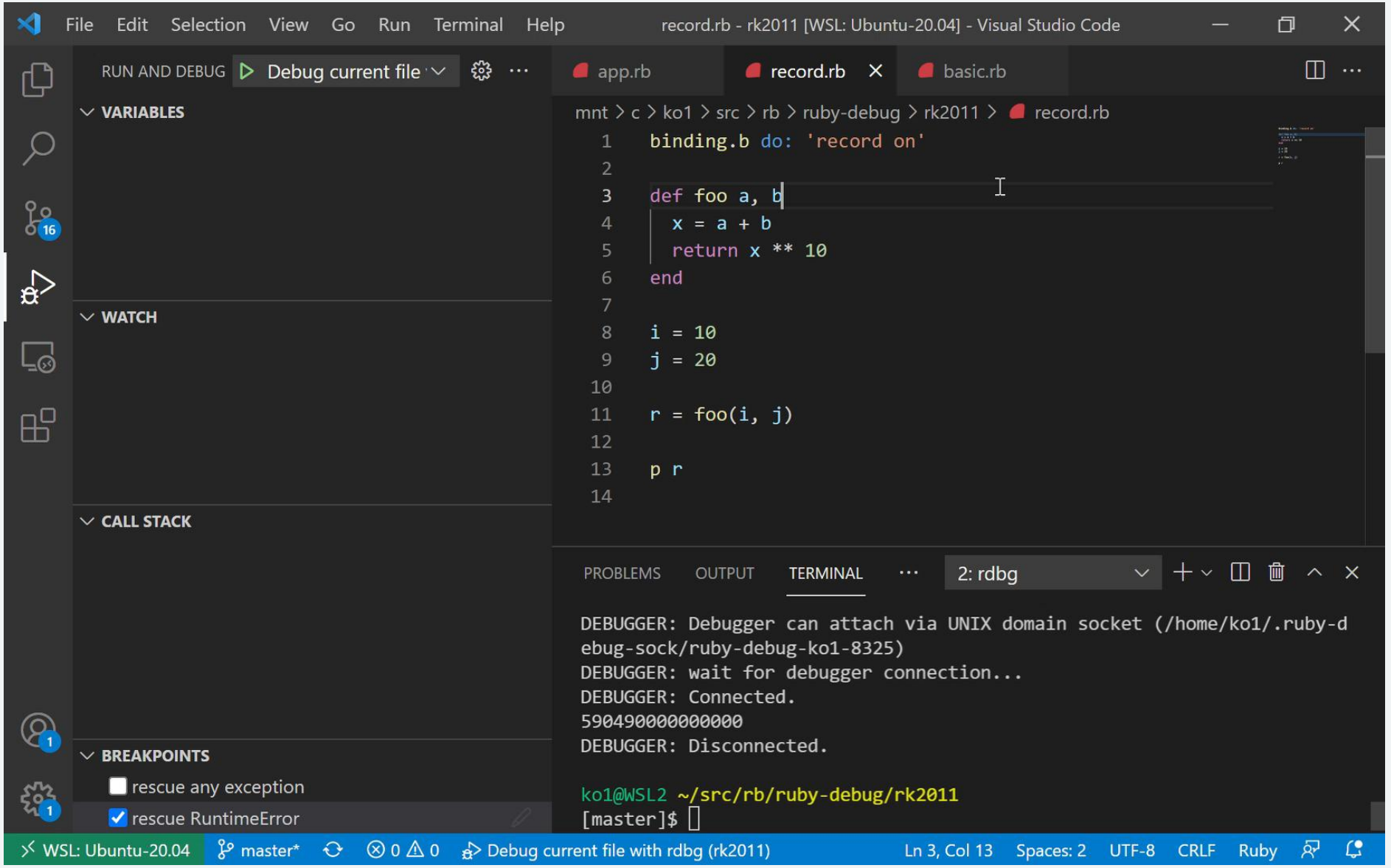
現在

AST盆栽


AST Bonsai koruby_precise
EN JA Fold
Ruby code (parsed by koruby.wasm in your browser)
class File
def self.binread(path) =
__binread(path)
def self.read(path) = __binread(path)
def self.basename(p) =
(p.split("/").last || p)
Grow tree
Run and glow
Choose a sample...
Emulator: optcarrot (MIT) · ROM: Lan Master (public domain)
Fold same-kind chains to the right (seq etc.)
Labels: None
Entries only (def, class, blocks...) All
lit 4053 lget 2487 seq 2204
ivar_get 1445 if 1034 caseeq 1013
ary_push 996 ivar_set 757 self 723
call 690 send 634 or 495 76 more kinds
Nodes 22,326
Shown (after folding) 20,062
Depth (before folding) 273
Node kinds 88
▼HOW IT WORKS
The tree is the AST koruby builds from your code: the program is the root, and children fan out clockwise in source order. "Run and glow" runs it in your browser and lights up the node being executed, about one dispatch in N.
Drag rotate around the root
Wheel zoom
Right-drag move up and down the trunk
Click a node go to its height
Ctrl+Enter grow the tree
Drop a file load a .rb or a dump
ROOT seq

現在

debug.gem


File Edit Selection View Go Run Terminal Help
record.rb - rk2011 [WSL: Ubuntu-20.04] - Visual Studio Code
RUN AND DEBUG
Debug current file
VARIABLES
WATCH
CALL STACK
BREAKPOINTS
rescue any exception
rescue RuntimeError
app.rb
record.rb
basic.rb
mnt > c > ko1 > src > rb > ruby-debug > rk2011 > record.rb
1 binding.b do: 'record on'
2
3 def foo a, b
4 x = a + b
5 return x ** 10
6 end
7
8 i = 10
9 j = 20
10
11 r = foo(i, j)
12
13 p r
14
PROBLEMS OUTPUT TERMINAL
2: rdbg
DEBUGGER: Debugger can attach via UNIX domain socket (/home/ko1/.ruby-debug-sock/ruby-debug-ko1-8325)
DEBUGGER: wait for debugger connection...
DEBUGGER: Connected.
590490000000000
DEBUGGER: Disconnected.
ko1@WSL2 ~/src/rb/ruby-debug/rk2011
[master]$
WSL: Ubuntu-20.04
master*
0 0
Debug current file with rdbg (rk2011)
Ln 3, Col 13
Spaces: 2
UTF-8
CRLF
Ruby

過去 どう動いたのか

実行ログ

設

アプリ用に
設計したもの

自 **自動的に集計したもの**

統計情報

- プロファイル情報
- カバレッジ情報
- …

詳細な情報

- トレース情報
- リバースデバッグ
- …

TracePoint とかが便利なやつ

過去

rperf / prperf

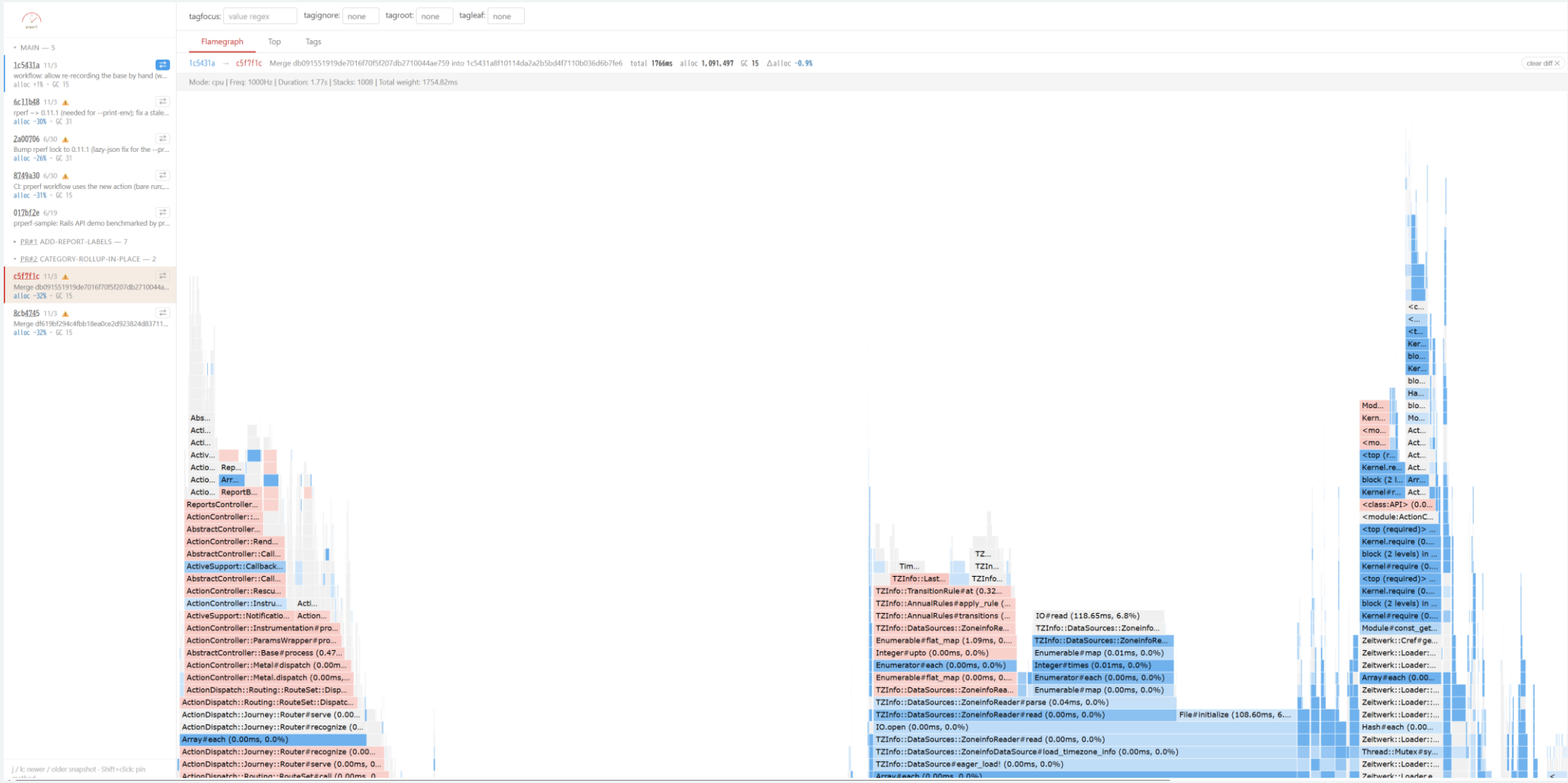
rperf

rperf

Know where your Ruby spends its time — accurately.
A sampling profiler that corrects safepoint bias using real time deltas.

prperf

Diff every Ruby PR's performance



過去

debug.gem line trace

“Introduction of new features for VS Code debugging” by Naoto Ono at RubyKaigi 2023

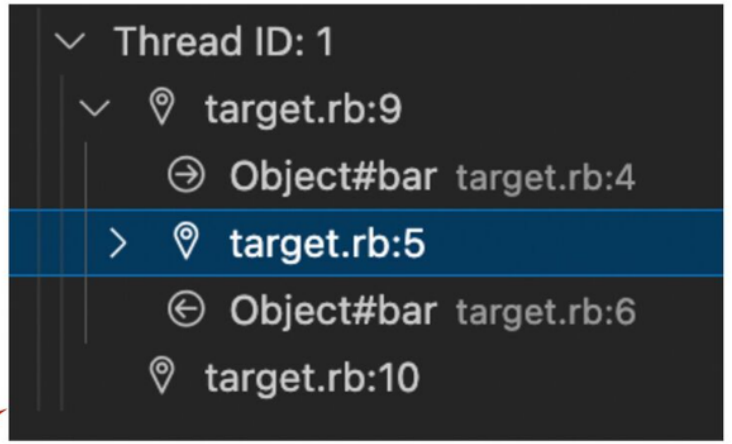
https://rubykaigi.org/2023/presentations/ono-max.html

The way of displaying trace logs

rdbg(CLI)

```
#th:1 #depth:1  at target.rb:9
#th:1 #depth:2 >  Object#bar at target.rb:4
#th:1 #depth:2  at target.rb:5
#th:1 #depth:3 >   Object#foo at target.rb:1
#th:1 #depth:3  at target.rb:2
#th:1 #depth:3 <   Object#foo #=> [1, 2] at target.rb:3
#th:1 #depth:2 <  Object#bar #=> [1, 2] at target.rb:6
#th:1 #depth:1  at target.rb:10
```

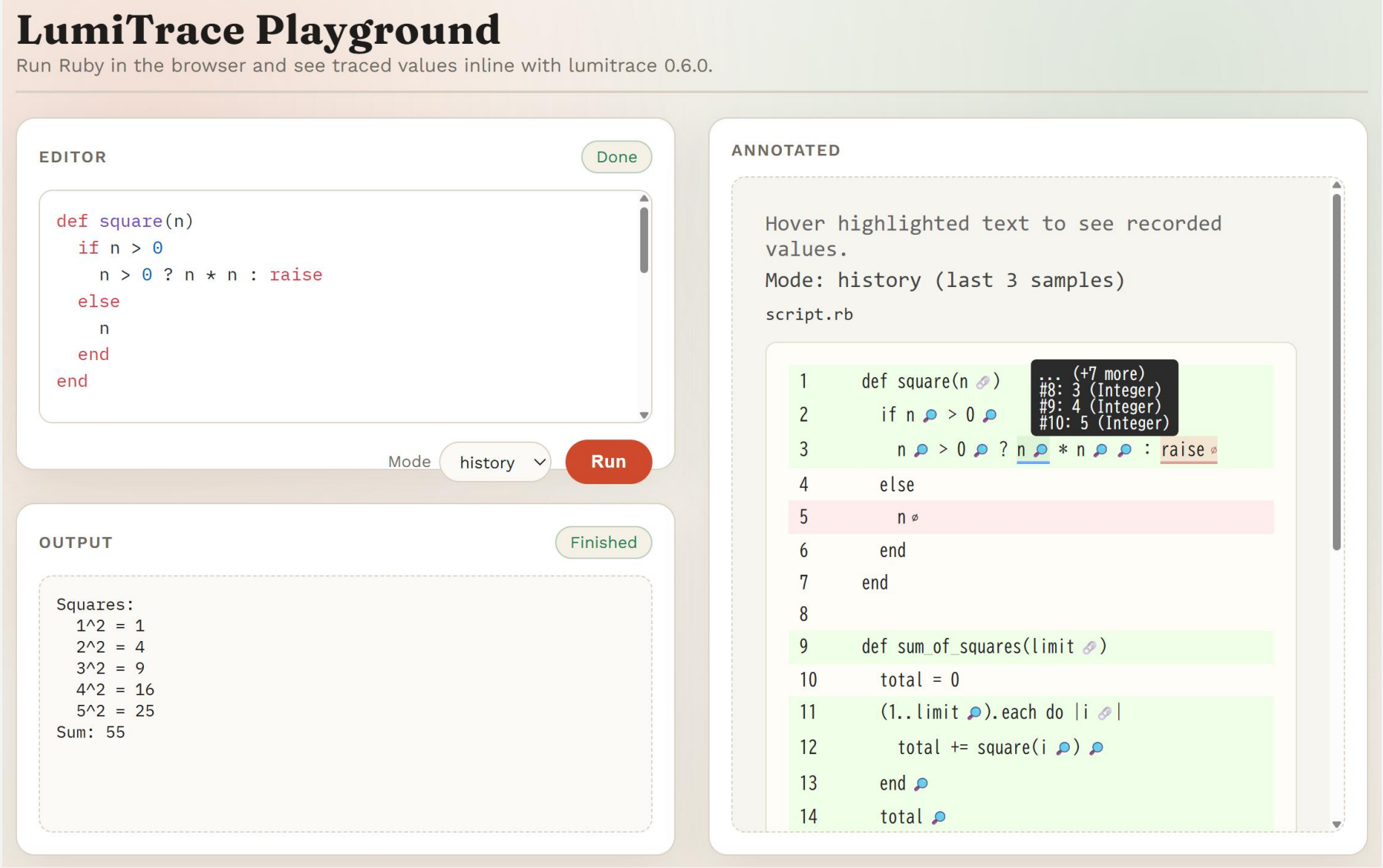
Trace Inspector



You can see only what you want to see by clicking!

lumitrace

https://ko1.github.io/lumitrace/runv/



SECTION 02

lumitrace

式ごとの実行時情報を、必要な範囲だけ集める



実装技術

1

Prism でプログラムのノードを生成

2

各ノード（＝式）に情報収集処理を挿入

ある種のプログラム変換

3

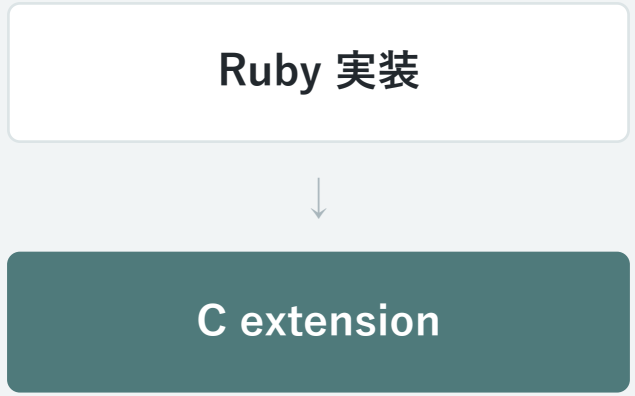
実行時に情報を収集

最適化技術

手 手作業で頑張る

- AI 生成したコードに対して
- 最終的に C extension に

最終形



とれる情報

式ごとに取得

各ノード（＝式）単位で記録する

値

型

保持する履歴

最後の n 回分

全部は残さない

PR ごとに動かすといいのでは…？

PR

PR をレビューするときに実行時情報が取れたら、嬉しいのでは？

- AI 生成コードをレビューするときに、ちょっと嬉しいかも？
- ついでにテストカバレッジくらい見ておきたい

課題

全部取ると遅いんでは…

提案

diff の範囲だけ実行時トレースすると、コードレビューに役立つのでは？



lumitrace の範囲指定

1 lumitrace gem

- コマンドおよびライブラリ
- 式単位で実行結果を収集
- 実行結果を織り込んで出力（text, html）
- **diff 指定機能**
  - git diff と連携

2 lumitrace-action

- GitHub Actions で、PR の diff 部分だけ収集
- PR 画面にカバレッジ情報を追加
- AI 向けに JSON 出力
- 結果を収集

https://lumitrace.atdot.net/

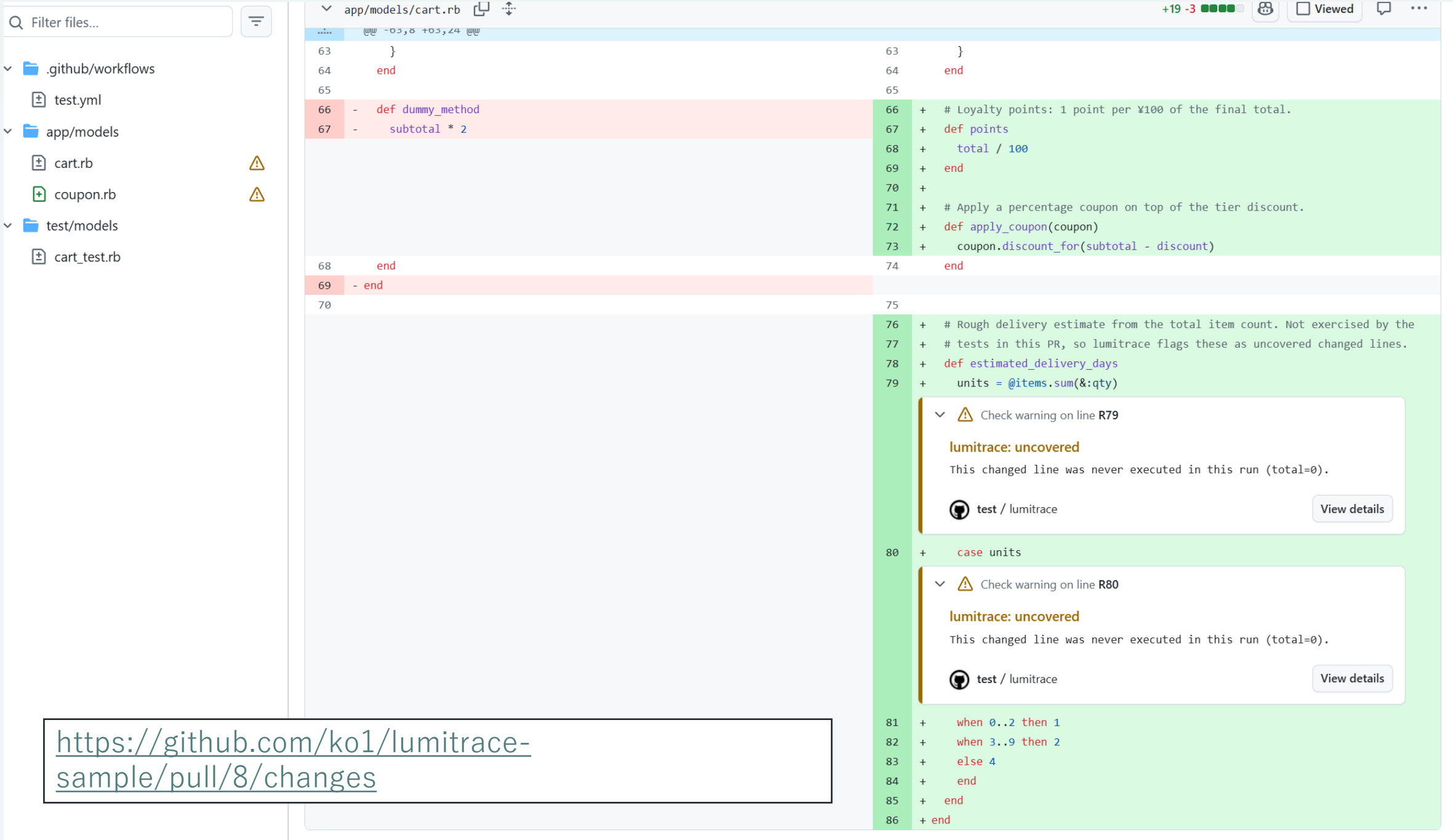
ADD IT TO A WORKFLOW

Take your existing test workflow and add the highlighted lines — your test command stays exactly the same:

```yaml
name: test
on: pull_request

permissions:
  checks: write
  contents: read
  id-token: write                          # upload the report (omit = PR check only)

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: ruby/setup-ruby@v1
        with: { bundler-cache: true }
      - uses: ko1/lumitrace-action/setup@v1
      - run: bundle exec rake test          # unchanged
      - uses: ko1/lumitrace-action/report@v1
        if: always()
```



https://github.com/ko1/lumitrace-sample/pull/8/changes

Conversation 0 · Commits 1 · Checks 2 · Files changed 4 +49 -

Demo: coupon + loyalty points (recorded values) 9711e2d

test
on: pull_request
4

test

lumitrace

GitHub Actions / lumitrace
completed on Jun 28 in 0s

5 uncovered · 54 traced

Lumitrace traced 54 expression(s) in the changed range.

File	Covered	%
`app/models/cart.rb`	3/5	60.0%
`app/models/coupon.rb`	6/7	85.7%
`test/models/cart_test.rb`	13/13	100.0%

⚠️ 5 changed line(s) never executed.

Highlights

- `app/models/cart.rb:68` → Integer 62
- `app/models/cart.rb:72` `coupon` → Coupon #<Coupon:0x00007f5fee50f2f8 **@code**="SAVE10", @percent_off=10>
- `app/models/cart.rb:73` → Integer 570
- `app/models/coupon.rb:5` `code` → String "SAVE10"
- `app/models/coupon.rb:6` → String "SAVE10"
- `app/models/coupon.rb:7` → Integer 10
- `app/models/coupon.rb:10` `amount` → Integer 5700
- `app/models/coupon.rb:11` → FalseClass false
- `app/models/coupon.rb:12` → Integer 570
- `test/models/cart_test.rb:14` → Cart #<Cart:0x00007f5fee50a118 **@items**=[#<struct Cart::Item name="book", price=1500, qty=4>]>
- ...and 12 more

Full report ↗ · JSON for tooling/AI

https://github.com/ko1/lumitrace-sample/pull/8/checks?check_run_id=83910391941

Hover highlighted text to see recorded values.
Mode: last (last value)

Files (3)

Overview

- app
  - models
    - cart.rb (2/4)
    - coupon.rb (4/5)
- test
  - models
    - cart_test.rb (13/13)

app/models/cart.rb

```
...
63        }
64      end
65
66      # Loyalty points: 1 point per ¥100 of the final total.
67      def points
68        total / 100
69      end
70
71      # Apply a percentage coupon on top of the tier discount.
72      def apply_coupon(coupon)
73        coupon.discount_for(subtotal - discount)
74      end
75
76      # Rough delivery estimate from the total item count. Not exercised by the
77      # tests in this PR, so lumitrace flags these as uncovered changed lines.
78      def estimated_delivery_days
79        units = @items.sum(&:qty)
80        case units
81        when 0..2 then 1
82        when 3..9 then 2
83        else 4
84        end
85      end
86    end
```


https://lumitrace.atdot.net/r/HoSkGeRUHxo5z1YF64VzkQ

AI 時代のレビュー

見る人への手助け

人 人間が見る

AI AI が見る

print デバッグの、ちょっと先へ

まとめ

AI 時代に必要な、プログラム発掘手法とは？

1

観察するツール

未来・現在・過去

2

**PR ごとに情報を
もっとリッチに**

レビューのときにこそ
実行時情報がほしい

3

**全部じゃなくて
diff 範囲だけ**

とすると賢いのでは…？

北陸Ruby会議02 ／ #hokurikurk02

ありがとうございました



笹田耕一　STORES, Inc.

https://lumitrace.atdot.net/