Automatic Test-Case Reduction in Proof Assistants

A Case Study in Coq

Jason Gross, Théo Zimmermann, Miraya Poddar-Agrawal, and Adam Chlipala

Category Theory Library Minimization in Coq

emacs@JasonGross-X1

File Edit Options Buffers Tools Coq Proof-General Outline YASnippet Holes Hide/Show Help

State Context Goal Retract Undo Next Use Goto Qed Home Find Info Command Prooftree Interrupt

```
Require Import JMeq FunctionalExtensionality ProofIrrelevance.
Require Export FunctorCategory SumCategory ProductCategory Functor.
Require Import Common NatCategory FEqualDep InitialTerminalCategory.
Require Import FunctorProduct ProductInducedFunctors FunctorialComposition.
Require Import SumInducedFunctors (*CanonicalStructureSimplification*).
Set Implicit Arguments.
Generalizable All Variables.
Set Asymmetric Patterns.
Set Universe Polymorphism.
Local Hint Immediate
      TerminalCategoryFunctorUnique InitialCategoryFunctorUnique
      InitialCategoryFunctor'Unique.
Local Hint Resolve Functor_eq Functor_JMeq NaturalTransformation_eq
      NaturalTransformation_JMeq eq_JMeq.
Section Law0.
  Context `(C : @SpecializedCategory objC).

  Definition ExponentialLaw0Functor : SpecializedFunctor (C ^ 0) 1
    := FunctorTo1 _.
  Definition ExponentialLaw0Functor_Inverse : SpecializedFunctor 1 (C ^ 0)
    := FunctorFrom1 _ (FunctorFrom0 _).

  Lemma ExponentialLaw0
  : ComposeFunctors ExponentialLaw0Functor ExponentialLaw0Functor_Inverse
    = IdentityFunctor _
    /\ ComposeFunctors ExponentialLaw0Functor_Inverse ExponentialLaw0Functor
       = IdentityFunctor _.
  Proof.
```

U:%%- *goals* All (1,0) (Coq Goals waka)

U:**- ExponentialLaws.v All (1,0) Git-bug-for-itp-2022 (Coq Script(0-)

U:%%- *response* All (1,0) (Coq Response waka Wrap)

Right click (or C-click) to show the refactoring menu.

emacs@JasonGross-X1

File Edit Options Buffers Tools Coq Proof-General Outline YASnippet Holes Hide/Show Help

State Context Goal Retract Undo Next Use Goto Qed Home Find Info Command Prooftree Interrupt

```
Require Import JMeq FunctionalExtensionality ProofIrrelevance.
Require Export FunctorCategory SumCategory ProductCategory Functor.
Require Import Common NatCategory FEqualDep InitialTerminalCategory.
Require Import FunctorProduct ProductInducedFunctors FunctorialComposition.
Require Import SumInducedFunctors (*CanonicalStructureSimplification*).
Set Implicit Arguments.
Generalizable All Variables.
Set Asymmetric Patterns.
Set Universe Polymorphism.
Local Hint Immediate
      TerminalCategoryFunctorUnique InitialCategoryFunctorUnique
      InitialCategoryFunctor'Unique.
Local Hint Resolve Functor_eq Functor_JMeq NaturalTransformation_eq
      NaturalTransformation_JMeq eq_JMeq.
Section Law0.
  Context `(C : @SpecializedCategory objC).

  Definition ExponentialLaw0Functor : SpecializedFunctor (C ^ 0) 1
    := FunctorTo1 _.
  Definition ExponentialLaw0Functor_Inverse : SpecializedFunctor 1 (C ^ 0)
    := FunctorFrom1 _ (FunctorFrom0 _).

  Lemma ExponentialLaw0
  : ComposeFunctors ExponentialLaw0Functor ExponentialLaw0Functor_Inverse
    = IdentityFunctor _
    /\ ComposeFunctors ExponentialLaw0Functor_Inverse ExponentialLaw0Functor
       = IdentityFunctor _.
  Proof.
```

U:%%- *goals* All (1,0) (Coq Goals waka)

U:**- ExponentialLaws.v All (1,0) Git-bug-for-itp-2022 (Coq Script(0-)

U:%%- *response* All (1,0) (Coq Response waka Wrap)

Right click (or C-click) to show the refactoring menu.

`split` fails with `Error: Refiner was given an argument "@SpecializedFunctor (* Set NaturalTransformation.283 Top.112 Top.113 *) (CardinalityRepresentative O) (NatCategory (* NaturalTransformation.283 *) O) objC C" of type "Type (* max(Set, Nat #7

Edit New issue

Closed JasonGross opened this issue on Jan 16, 2013 · 19 comments

JasonGross on Jan 16, 2013 Member

Jan 16, 2013, 7:01 PM EST

This is in the following environment. I'll try to get a cleaner one soon, but I figured that the error message and environment might be enough to point you to the error

```
  objC : Type (* Top.112 *)
  C : SpecializedCategory (* Top.112 Top.113 *) objC
  ============================
   and
     (@eq
       (@SpecializedFunctor (* Set Functor.290 Set Functor.290 *)
         (CardinalityRepresentative (S O))
         (NatCategory (* Functor.290 *) (S O))
         (CardinalityRepresentative (S O))
         (NatCategory (* Functor.290 *) (S O)))
       (@ComposeFunctors (* Set Functor.290 Set Functor.290 Set
         Functor.290 *) (CardinalityRepresentative (S O))
         (NatCategory (* Functor.290 *) (S O))
         (@SpecializedFunctor (* Set NaturalTransformation.283 Top.112
           Top.113 *) (CardinalityRepresentative O)
           (NatCategory (* NaturalTransformation.283 *) O) objC C)
         (@FunctorCategory (* Set NaturalTransformation.283 Top.112 Top.113
           Set Functor.290 *) (CardinalityRepresentative O)
           (NatCategory (* NaturalTransformation.283 *) O) objC C)
https://github.com/HoTT/coq/issues/7#issue-10041042 sentative (S O))
```

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Create a branch for this issue or link a pull request.

Notifications Customize

Unsubscribe

You're receiving notifications because you're watching

emacs@JasonGross-X1

File Edit Options Buffers Tools Coq Proof-General Outline YASnippet Holes Hide/Show Help

State Context Goal Retract Undo Next Use Goto Qed Home Find Info Command Prooftree Interrupt

```
Require Import JMeq FunctionalExtensionality ProofIrrelevance.
Require Export FunctorCategory SumCategory ProductCategory Functor.
Require Import Common NatCategory FEqualDep InitialTerminalCategory.
Require Import FunctorProduct ProductInducedFunctors FunctorialComposition.
Require Import SumInducedFunctors (*CanonicalStructureSimplification*).
Set Implicit Arguments.
Generalizable All Variables.
Set Asymmetric Patterns.
Set Universe Polymorphism.
Local Hint Immediate
      TerminalCategoryFunctorUnique InitialCategoryFunctorUnique
      InitialCategoryFunctor'Unique.
Local Hint Resolve Functor_eq Functor_JMeq NaturalTransformation_eq
      NaturalTransformation_JMeq eq_JMeq.
Section Law0.
  Context `(C : @SpecializedCategory objC).

  Definition ExponentialLaw0Functor : SpecializedFunctor (C ^ 0) 1
    := FunctorTo1 _.
  Definition ExponentialLaw0Functor_Inverse : SpecializedFunctor 1 (C ^ 0)
    := FunctorFrom1 _ (FunctorFrom0 _).

  Lemma ExponentialLaw0
  : ComposeFunctors ExponentialLaw0Functor ExponentialLaw0Functor_Inverse
    = IdentityFunctor _
    /\ ComposeFunctors ExponentialLaw0Functor_Inverse ExponentialLaw0Functor
       = IdentityFunctor _.
  Proof.
    split.
```

U:**- ExponentialLaws.v All (29,4) Git-bug-for-itp-2022 (Coq Script(0-)

U:--- *goals* All (1,0) (Coq Goals waka)

```
Error: (diff) Refiner was given an argument "SpecializedFunctor 0 C" of type
 "Type" instead of "Set".
```

U:%*- *response* All (1,0) (Coq Response waka Wrap)

Closed

`split` fails with `Error: Refiner was given an argument "@SpecializedFunctor (* Set NaturalTransformation.283 Top.112 Top.113 *) (CardinalityRepresentative O) (NatCat... #7

JasonGross opened this issue on Jan 16, 2013 · 19 comments

JasonGross on Jan 16, 2013 Member Author

Jan 16, 2013, 7:04 PM EST

There's something odd going on here. When I have something as a goal, it fails:

```
    Set Printing All.
    Goal (@eq
      (@SpecializedFunctor (CardinalityRepresentative (S O))
        (NatCategory (S O)) (CardinalityRepresentative (S O))
        (NatCategory (S O)))
      (@ComposeFunctors (CardinalityRepresentative (S O))
        (NatCategory (S O))
        (@SpecializedFunctor (CardinalityRepresentative O)
          (NatCategory O) objC C)
        (@FunctorCategory (CardinalityRepresentative O)
          (NatCategory O) objC C) (CardinalityRepresentative (S O))
        (NatCategory (S O)) ExponentialLaw0Functor
        ExponentialLaw0Functor_Inverse)
      (@IdentityFunctor (CardinalityRepresentative (S O)) (NatCategory (S O)))).
    generalize ExponentialLaw0Functor. (* Toplevel input, characters 0-33:
Error: Illegal application (Type Error):
The term "SpecializedFunctor" of type
 "forall (objC : Type) (_ : SpecializedCategory objC)
    (objD : Set) (_ : SpecializedCategory objD), Type"
cannot be applied to the terms
```

But if instead I do

```
    Set Printing All.
    Goal True.
    assert (@eq
      (@SpecializedFunctor (CardinalityRepresentative (S O))
        (NatCategory (S O)) (CardinalityRepresentative (S O))
        (NatCategory (S O)))
epresentative (S O))
```

2 participants

Lock conversation

Pin issue

Transfer issue

Delete issue

emacs@JasonGross-X1

File Edit Options Buffers Tools Coq Proof-General Outline YASnippet Holes Hide/Show Help

State Context Goal Retract Undo Next Use Goto Qed Home Find Info Command Prooftree Interrupt

```
Require Import JMeq FunctionalExtensionality ProofIrrelevance.
Require Export FunctorCategory SumCategory ProductCategory Functor.
Require Import Common NatCategory FEqualDep InitialTerminalCategory.
Require Import FunctorProduct ProductInducedFunctors FunctorialComposition.
Require Import SumInducedFunctors (*CanonicalStructureSimplification*).
Set Implicit Arguments.
Generalizable All Variables.
Set Asymmetric Patterns.
Set Universe Polymorphism.
Local Hint Immediate
      TerminalCategoryFunctorUnique InitialCategoryFunctorUnique
      InitialCategoryFunctor'Unique.
Local Hint Resolve Functor_eq Functor_JMeq NaturalTransformation_eq
      NaturalTransformation_JMeq eq_JMeq.
Section Law0.
  Context `(C : @SpecializedCategory objC).

  Definition ExponentialLaw0Functor : SpecializedFunctor (C ^ 0) 1
    := FunctorTo1 _.
  Definition ExponentialLaw0Functor_Inverse : SpecializedFunctor 1 (C ^ 0)
    := FunctorFrom1 _ (FunctorFrom0 _).

  Lemma ExponentialLaw0
  : ComposeFunctors ExponentialLaw0Functor ExponentialLaw0Functor_Inverse
    = IdentityFunctor _
    /\ ComposeFunctors ExponentialLaw0Functor_Inverse ExponentialLaw0Functor
       = IdentityFunctor _.
  Proof.
    split.
```

U:**- ExponentialLaws.v All (29,4) Git-bug-for-itp-2022 (Coq Script(1-)

```
1 subgoals, subgoal 1 (ID 36)

- objC : Type
- C : SpecializedCategory objC
  ____________________________________________________________
   ComposeFunctors ExponentialLaw0Functor ExponentialLaw0Functor_Inverse =
   IdentityFunctor 1 /\
   ComposeFunctors ExponentialLaw0Functor_Inverse ExponentialLaw0Functor =
   IdentityFunctor (C ^ 0)
(dependent evars:)
```

U:%%- *goals* All (10,0) (Coq Goals waka)

```
Error: (diff) Refiner was given an argument "SpecializedFunctor 0 C" of type
 "Type" instead of "Set".
```

U:%*- *response* All (1,0) (Coq Response waka Wrap)

emacs@JasonGross-X1

File Edit Options Buffers Tools Coq Proof-General Outline YASnippet Holes Hide/Show Help

State Context Goal Retract Undo Next Use Goto Qed Home Find Info Command Prooftree Interrupt

```
Require Import JMeq FunctionalExtensionality ProofIrrelevance.
Require Export FunctorCategory SumCategory ProductCategory Functor.
Require Import Common NatCategory FEqualDep InitialTerminalCategory.
Require Import FunctorProduct ProductInducedFunctors FunctorialComposition.
Require Import SumInducedFunctors (*CanonicalStructureSimplification*).
Set Implicit Arguments.
Generalizable All Variables.
Set Asymmetric Patterns.
Set Universe Polymorphism.
Local Hint Immediate
      TerminalCategoryFunctorUnique InitialCategoryFunctorUnique
      InitialCategoryFunctor'Unique.
Local Hint Resolve Functor_eq Functor_JMeq NaturalTransformation_eq
      NaturalTransformation_JMeq eq_JMeq.
Section Law0.
  Context `(C : @SpecializedCategory objC).

  Definition ExponentialLaw0Functor : SpecializedFunctor (C ^ 0) 1
    := FunctorTo1 _.
  Definition ExponentialLaw0Functor_Inverse : SpecializedFunctor 1 (C ^ 0)
    := FunctorFrom1 _ (FunctorFrom0 _).

  Lemma ExponentialLaw0
  : ComposeFunctors ExponentialLaw0Functor ExponentialLaw0Functor_Inverse
    = IdentityFunctor _
    /\ ComposeFunctors ExponentialLaw0Functor_Inverse ExponentialLaw0Functor
       = IdentityFunctor _.
  Proof.
    split.
```

U:%%- *goals* All (1,0) (Coq Goals waka)

U:**- ExponentialLaws.v All (6,0) Git-bug-for-itp-2022 (Coq Script(0-)

U:%%- *response* All (1,0) (Coq Response waka Wrap)

Undo

emacs@JasonGross-X1

File Edit Options Buffers Tools Coq Proof-General Outline YASnippet Holes Hide/Show Help

State Context Goal Retract Undo Next Use Goto Qed Home Find Info Command Prooftree Interrupt

```
Require Export FunctorCategory Functor.
Require Import Common NatCategory InitialTerminalCategory.


Generalizable All Variables.
Section Law0.
  Context `(C : @SpecializedCategory objC).

  Definition ExponentialLaw0Functor : SpecializedFunctor (C ^ 0) 1
    := FunctorTo1 _.
  Definition ExponentialLaw0Functor_Inverse : SpecializedFunctor 1 (C ^ 0)
    := FunctorFrom1 _ (FunctorFrom0 _).

  Lemma ExponentialLaw0
  : ComposeFunctors ExponentialLaw0Functor ExponentialLaw0Functor_Inverse
    = IdentityFunctor _
    /\ ComposeFunctors ExponentialLaw0Functor_Inverse ExponentialLaw0Functor
       = IdentityFunctor _.
  Proof.
    split.
```

U:**- ExponentialLaws.v All (2,57) Git-bug-for-itp-2022 (Coq Script(0-)

U:%%- *goals* All (1,0) (Coq Goals waka)

```
Error: (diff) Refiner was given an argument "SpecializedFunctor 0 C" of type
 "Type" instead of "Set".
```

U:%*- *response* All (1,0) (Coq Response waka Wrap)

Closed `split` fails with `Error: Refiner was given an argument "@SpecializedFunctor (* Set NaturalTransformation.283 Top.112 Top.113 *) (CardinalityRepresentative O) (NatCat... #7

JasonGross opened this issue on Jan 16, 2013 · 19 comments

JasonGross on Jan 16, 2013 Member Author

Jan 16, 2013, 8:01 PM EST

Ok, here's a self-contained version. I'm really surprised that `intro` can fail with `Illegal application (Type Error)` I'll try to trim this down sometime soon, but, if I don't get to it, I hope it's enough to work off of:

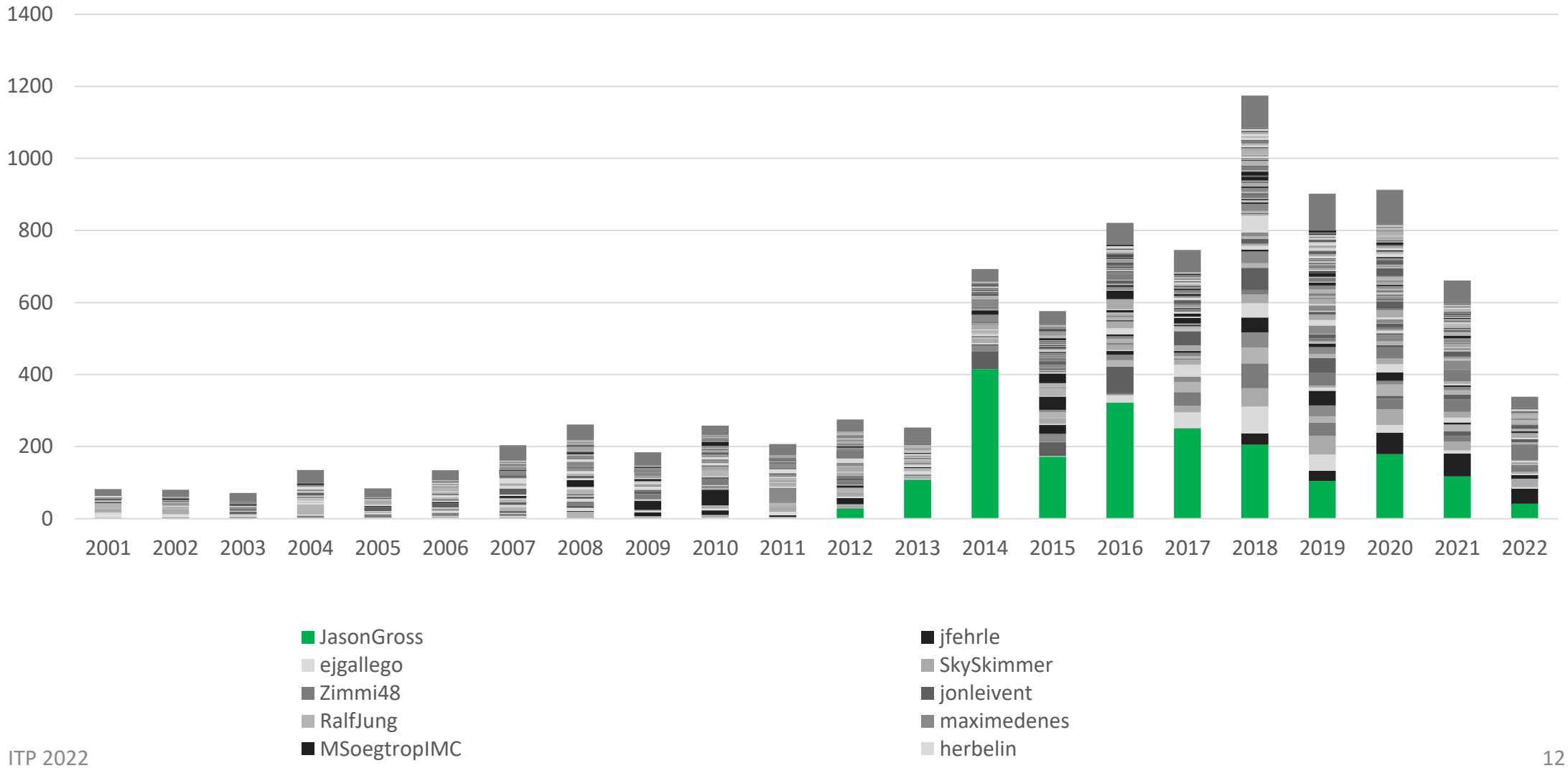
```
Delimit Scope object_scope with object.
Delimit Scope morphism_scope with morphism.
Delimit Scope category_scope with category.
Delimit Scope functor_scope with functor.

Set Implicit Arguments.

Generalizable All Variables.

Polymorphic Record SpecializedCategory (obj : Type) := Build_SpecializedCategory' {
  Object :> _ := obj;
  Morphism' : obj -> obj -> Type;

  Identity' : forall o, Morphism' o o;
  Compose' : forall s d d', Morphism' d d' -> Morphism' s d -> Morphism' s d';

  Associativity' : forall o1 o2 o3 o4 (m1 : Morphism' o1 o2) (m2 : Morphism' o2 o3) (m3 : Morphism' o3 o4),
    Compose' (Compose' m3 m2) m1 = Compose' m3 (Compose' m2 m1);
  (* ask for [eq_sym (Associativity' ...)], so that C^{op}^{op} is convertible with C *)
```

It has something to do with the `intros; autorewrite with functor; reflexivity` line; if I change that to `admit`, it all goes through. (Same thing if I do the rewriting manually.)

Member Author

https://github.com/HoTT/coq/issues/7#issuecomment-12349484

Origin Story: Coq Bug Reports By Year



Goal: Automate This!

Windows Pov | jgross@Jasor | jgross@Jasor | jgross@Jasor | jgross@Jasor | jgross@Jasor | jgross@Jasoı

```
Every 0.1s: ./watch.sh head -70                                  JasonGross-X1: Wed Jul 27 23:06:10 2022

cat: bug_01.v: No such file or directory
```

jgross@JasonGross-X1: ~/catd

```
jgross@JasonGross-X1:~/catdb$ command time -f "(real: %e, user: %U, sys:
 %S, mem: %M ko)" find-bug.py ExponentialLaws.v bug_01.v -R . ""
  --no-deps --coqbin=../HoTT-coq/bin/
```

Debugging Regressions: coqdev experience

Concentrate the Clenv code in Equality. #15501

Merged coqbot-app merged 12 commits into `coq:master` from `ppedrot:equality-limit-clenv` on Jan 26 `V8.16+rc1`

Conversation 21 | Commits 12 | Checks 4 | Files changed 10

ppedrot on Jan 18 • edited — Member

This PR makes it easier to track the use of clausenvs in the Equality module. The various parts using them have been brought closer together and some invariants are now clearer from the code.

coqbot-app bot commented on Jan 18 — Contributor

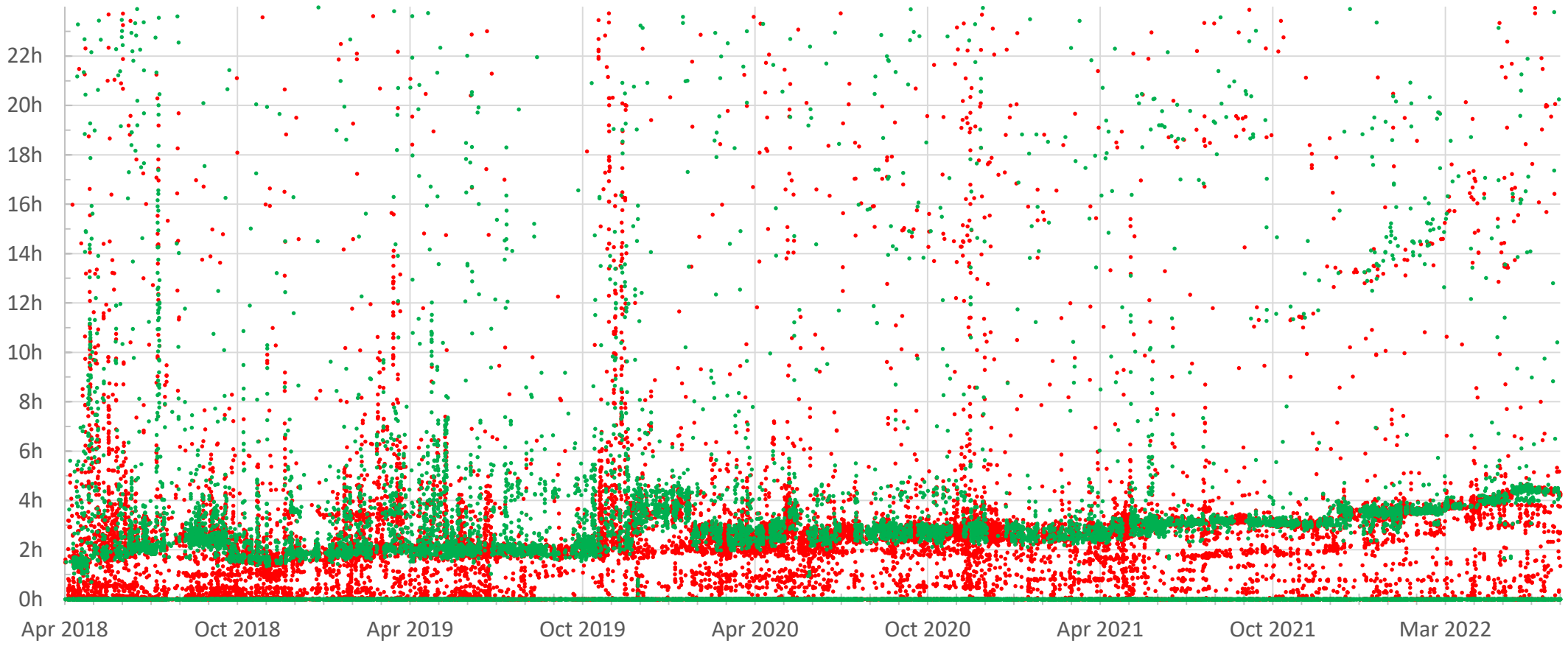
🔴 CI failures at commit `3238a7a` without any failure in the test-suite

✅ Corresponding jobs for the base commit `d5e61b2` succeeded

❓ Ask me to try to extract minimal test cases that can be added to the test-suite

▶ 🏃 `@coqbot ci minimize` will minimize the following targets: ci-bbv, ci-bedrock2, ci-color, ci-fiat_crypto_legacy, ci-gappa, ci-itauto, ci-rewriter, ci-verdi_raft, ci-vst

CI Running Time



Concentrate the Clenv code in Equality. #15501

Merged coqbot-app merged 12 commits into `coq:master` from `ppedrot:equality-limit-clenv` on Jan 26 `V8.16+rc1`

Conversation 21 | Commits 12 | Checks 4 | Files changed 10

ppedrot on Jan 18 • edited — Member

This PR makes it easier to track the use of clausenvs in the Equality module. The various parts using them have been brought closer together and some invariants are now clearer from the code.

coqbot-app bot commented on Jan 18 — Contributor

🔴 CI failures at commit `3238a7a` without any failure in the test-suite

✅ Corresponding jobs for the base commit `d5e61b2` succeeded

❓ Ask me to try to extract minimal test cases that can be added to the test-suite

▶ 🏃 `@coqbot ci minimize` will minimize the following targets: ci-bbv, ci-bedrock2, ci-color, ci-fiat_crypto_legacy, ci-gappa, ci-itauto, ci-rewriter, ci-verdi_raft, ci-vst

ppedrot on Jan 18 — Member Author

@coqbot ci minimize

ppedrot on Jan 18 Member Author

@coqbot ci minimize

coqbot-app bot commented on Jan 18 Contributor

I have initiated minimization at commit `3238a7a` for the suggested targets ci-bbv, ci-bedrock2, ci-color, ci-fiat_crypto_legacy, ci-gappa, ci-itauto, ci-rewriter, ci-verdi_raft, ci-vst as requested.

coqbot-app bot commented on Jan 18 Contributor

Minimized File /github/workspace/builds/coq/coq-failing/_build_ci/bbv/src/bbv/Word.v (from ci-bbv) (full log on GitHub Actions)

We are collecting data on the user experience of the Coq Bug Minimizer.
If you haven't already filled the survey *for this PR*, please fill out our short survey!

▶ Minimized Coq File (consider adding this file to the test-suite)

▶ Intermediate Coq File (useful for debugging if minimization did not go as far as you wanted)

▶ Build Log (contains the Coq error message) (truncated to last 8.0KiB; full 2.7MiB file on GitHub Actions Artifacts under `build.log`)

▶ Minimization Log (truncated to last 8.0KiB; full 82KiB file on GitHub Actions Artifacts under `bug.log`)

If you have any comments on your experience of the minimizer, please share them in a reply (possibly tagging `@JasonGross`).
If you believe there's a bug in the bug minimizer, please report it on the bug minimizer issue tracker.

coqbot-app bot commented on Jan 18 Contributor

Minimized File /github/workspace/builds/coq/coq-failing/_build_ci/bbv/src/bbv/Word.v (from ci-bbv) (full log on GitHub Actions)

We are collecting data on the user experience of the Coq Bug Minimizer.
If you haven't already filled the survey *for this PR*, please fill out our short survey!

▼ Minimized Coq File (consider adding this file to the test-suite)

```
(* -*- mode: coq; coq-prog-args: ("-emacs" "-q" "-w" "-deprecated-native-compiler-option" "-Q" "/github/workspace/cwd"
(* File reduced by coq-bug-minimizer from original input, then from 7591 lines to 50 lines, then from 55 lines to 49 l
(* coqc version 8.16+alpha compiled with OCaml 4.05.0
···coqtop version runner-zxwgkjap-project-6138686-concurrent-0:/builds/coq/coq/_build/default,(HEAD detached at e1c1bc
···Expected coqc runtime on this file: 0.095 sec *)
Set Implicit Arguments.

Inductive word : nat -> Set :=
| WO : word O
| WS : bool -> forall n, word n -> word (S n).
Fixpoint wones (sz : nat) : word sz.
Admitted.
Definition whd sz (w : word (S sz)) : bool.
Admitted.
Definition wtl sz (w : word (S sz)) : word sz.
Admitted.

Fixpoint combine (sz1 : nat) (w : word sz1) : forall sz2, word sz2 -> word (sz1 + sz2) :=
··match w in word sz1 return forall sz2, word sz2 -> word (sz1 + sz2) with
····| WO => fun _ w' => w'
····| WS b w' => fun _ w'' => WS b (combine w' w'')
··end.
Fixpoint split1 (sz1 sz2 : nat) : word (sz1 + sz2) -> word sz1.
Admitted.

Fixpoint wnot sz (w : word sz) : word sz :=
··match w with
····| WO => WO
····| WS b w' => WS (negb b) (wnot w')
```

```
  match w with
    | WO => WO
    | WS b w' => WS (negb b) (wnot w')
  end.

Fixpoint bitwp (f : bool -> bool -> bool) sz (w1 : word sz) : word sz -> word sz :=
  match w1 with
    | WO => fun _ => WO
    | WS b w1' => fun w2 => WS (f b (whd w2)) (bitwp f w1' (wtl w2))
  end.

Definition wnot' sz := bitwp xorb (wones sz).

Theorem split1_combine : forall sz1 sz2 (w : word sz1) (z : word sz2),
  split1 sz1 sz2 (combine w z) = w.
Admitted.

Theorem wnot_wnot'_equiv : forall sz (w : word sz), wnot w = wnot' w.
Admitted.

Theorem wnot_split1 : forall sz1 sz2 w, wnot (split1 sz1 sz2 w) = split1 sz1 sz2 (wnot w).
Proof.
  intros.
  repeat rewrite wnot_wnot'_equiv.
  unfold wnot'.
  erewrite <- split1_combine with (w := wones _).
```

▶ Intermediate Coq File (useful for debugging if minimization did not go as far as you wanted)

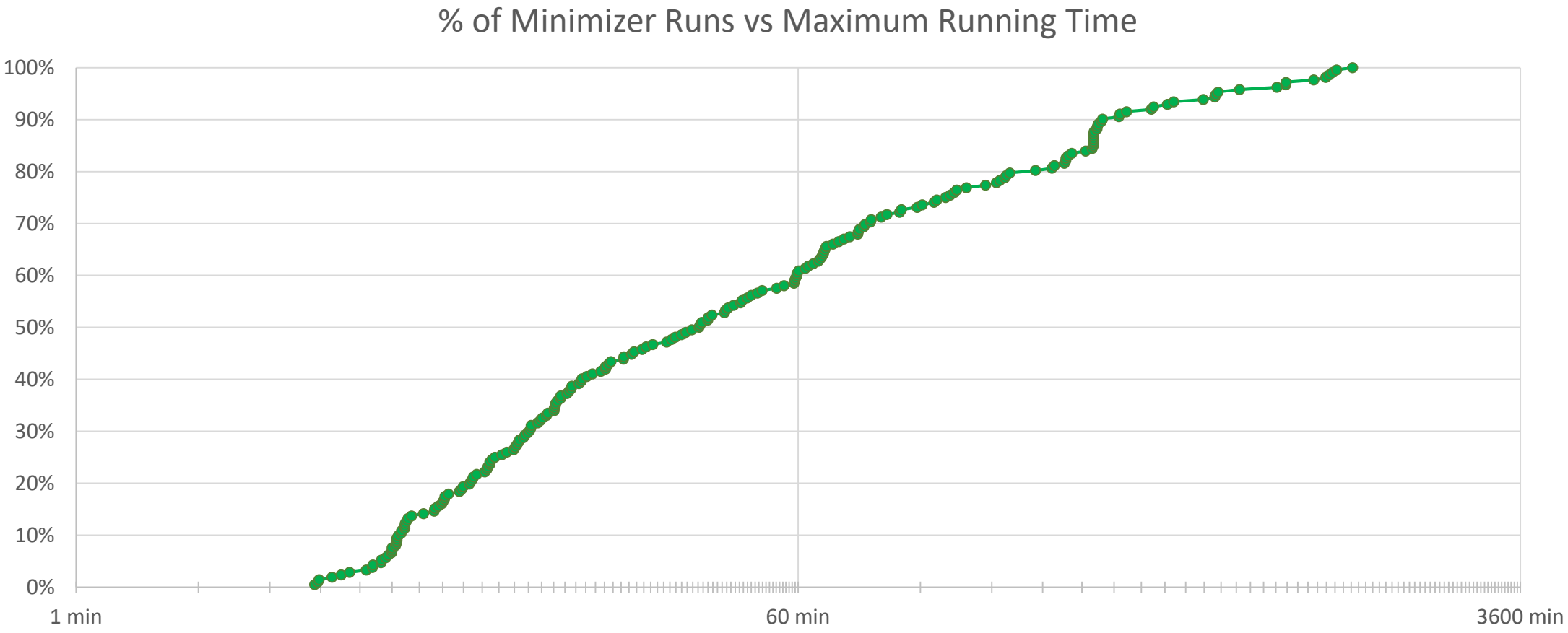
▶ Build Log (contains the Coq error message) (truncated to last 8.0KiB; full 2.7MiB file on GitHub Actions Artifacts under `build.log`)

▶ Minimization Log (truncated to last 8.0KiB; full 82KiB file on GitHub Actions Artifacts under `bug.log`)

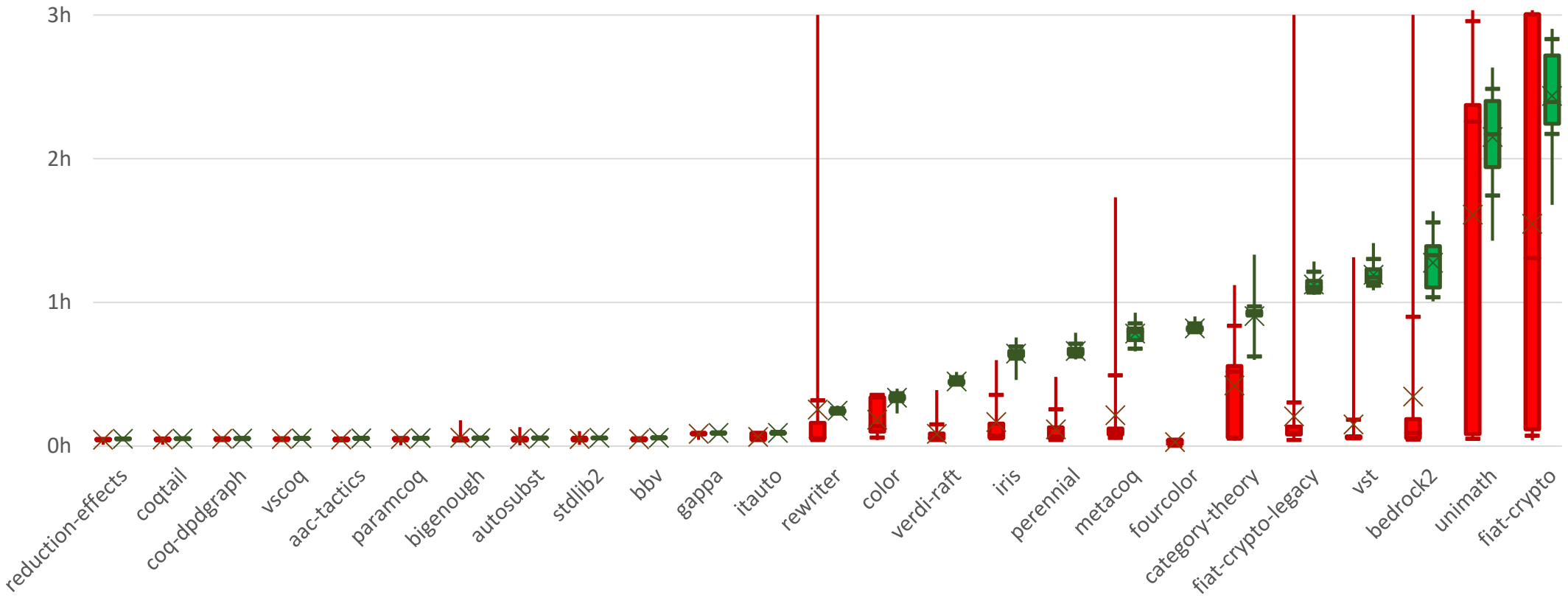
If you have any comments on your experience of the minimizer, please share them in a reply (possibly tagging `@JasonGross`).
If you believe there's a bug in the bug minimizer, please report it on the bug minimizer issue tracker.

How Long Does It Take?

Minimizer Running Time CDF

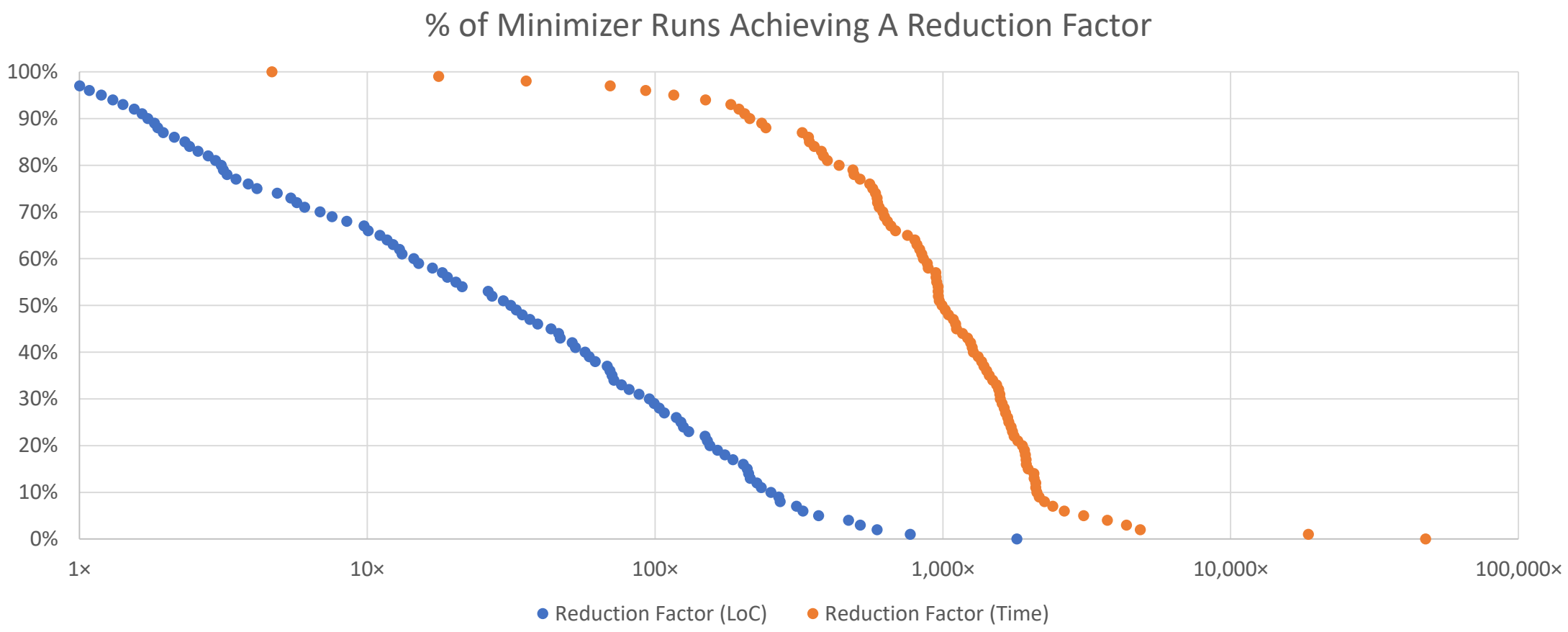


How Long Do Projects Take On CI?



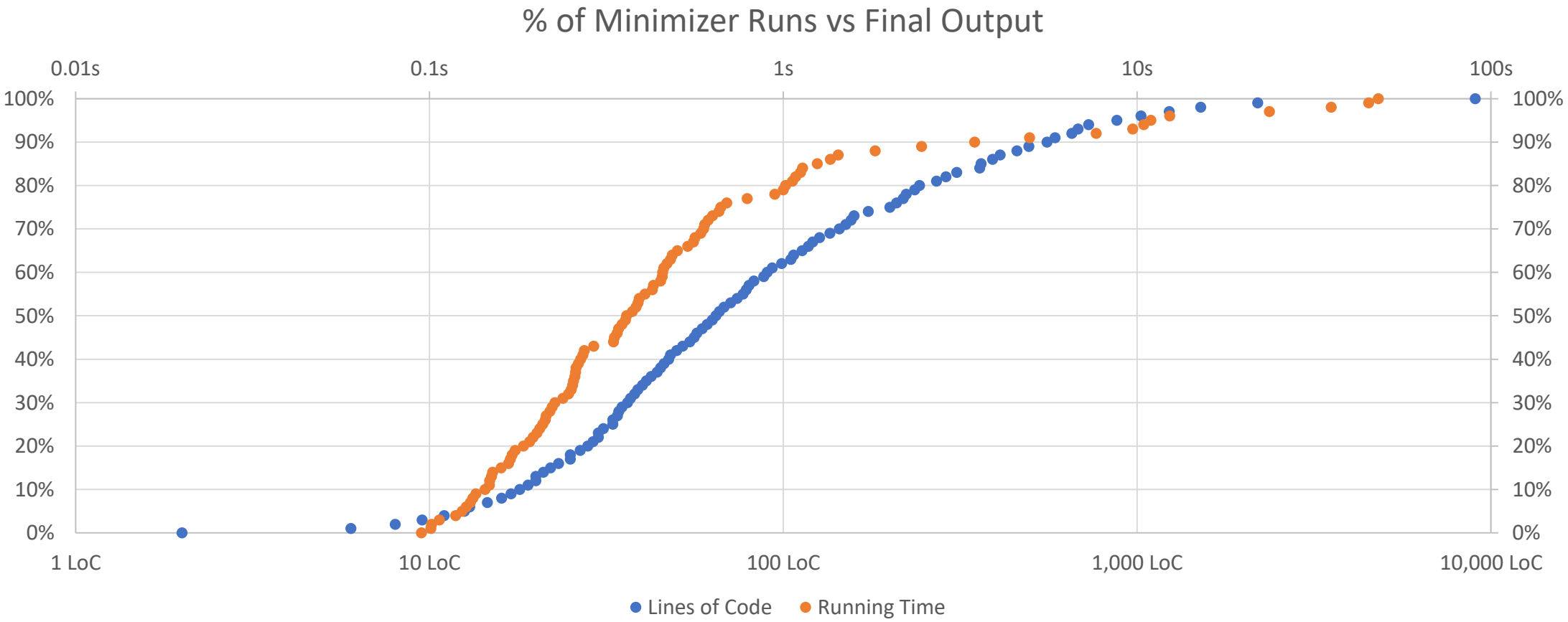
Selected CI Jobs, January through July 2022
Modified box plot for seven-number summary + mean

How Much Do We Minimize?



Better

How Minimal is Minimal in Absolute Terms?



Better

Specification Summary

- Input: a development displaying a potential problem with Coq
- Output: a single (shorter, faster) file displaying the same behavior
- Desires:
  - File should be standalone, as short as possible, as fast as possible
  - Fully automated, no interaction beyond pointing it at the issue initially
  - Interruptible and resumable: user should be able to do a manual pass over the file at any point and resume automatic minimization from there

Specification Details

What does “the same behavior” mean?

- In bug reporting case: “same” error message, without changing the final bit of code
  - “print ‘exact error message’” is not a valid minimization
- In CI case: should also succeed on the master version of Coq

What needs to be automated?

- Workflow automation (responding to user’s comment, replying as a comment)
- Removing code to shorten example (in compile time and LoC)
- Adjustments to minimize compile time
- Inlining and linearization (to reduce complexity)

How do we do this?

- Interfacing with Coq externally
- Delta debugging / brute force
- Tricks for inlining

To Achieve Adequate Performance:

- Inline files one-by-one
- Remove proof scripts as early as possible
- Linear pass to remove code, don’t consider all subsets

Delta Debugging / Brute Force Removal

- “Trust me, it’s true” primitives
- Not line by line, but blocks
- Minimal coupling, lack of forward references

Coupling

Coupling in Agda:

```
data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ
data Even : ℕ → Set
data Odd : ℕ → Set
data Even where
  even-zero : Even zero
  even-succ : ∀ {n} → Odd n → Even (succ n)
data Odd where
  odd-succ : ∀ {n} → Even n → Odd (succ n)
```

Coupling

Coupling in Agda:

```
data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ
data Even : ℕ → Set
data Odd : ℕ → Set
data Even where
  even-zero : Even zero
  even-succ : ∀ {n} → Odd n → Even (succ n)
data Odd where
  odd-succ : ∀ {n} → Even n → Odd (succ n)
```

Missing type signature for data definition Even
when scope checking the declaration
Odd

Missing type signature for data definition Odd
when scope checking the declaration
Even

The following names are declared but not accompanied by a definition: Even
when scope checking Odd

Coupling

Coupling in Agda:

```
data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ
```

Coupling

Coupled definitions are almost always single statements in Coq

```
Inductive ℕ : Set :=
| zero : ℕ
| succ : ℕ -> ℕ.
Inductive Even : ℕ -> Set :=
| even_zero : Even zero
| even_succ : forall {n}, Odd n -> Even (succ n)
with Odd : ℕ -> Set :=
| odd_succ : forall {n}, Even n -> Odd (succ n).
```

How do we do this?

- Interfacing with Coq externally
- Delta debugging / brute force
- Tricks for inlining

Tricks for Inlining and Linearization

Name Changes on Inlining

A.v:
```
Definition foo
: nat := 0.
```

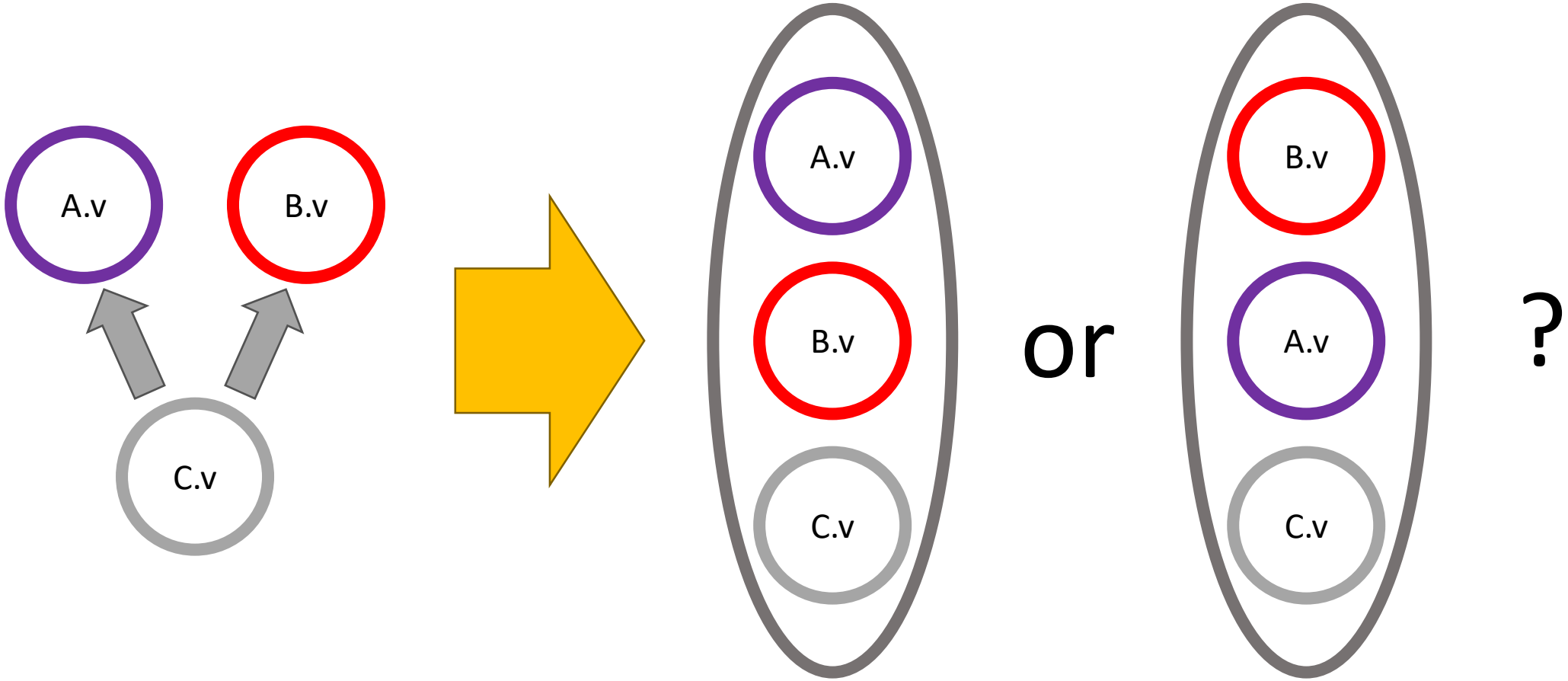
B.v:
```
Require Import A.
Definition foo
: nat := 1.
Print foo.
(*foo = 1 : nat*)
Print A.foo.
(*A.foo = 0 : nat*)
```

```
Definition foo
: nat := 0.
Definition foo
: nat := 1.
(* Error: foo
already exists. *)
Print foo.
(*foo = 1 : nat*)
Print A.foo.
```

VS

```
Module A.
Definition foo
: nat := 0.
End A.
Import A.
Definition foo
: nat := 1.
Print foo.
(*foo = 1 : nat*)
Print A.foo.
(*A.foo = 0 : nat*)
```

Linearization



How to make your proof assistant amenable to minimization

- Make it possible to have a single file with the same semantics as code in multiple files
- Primitive for “trust me, it’s true”
- Lack of forward references is nice

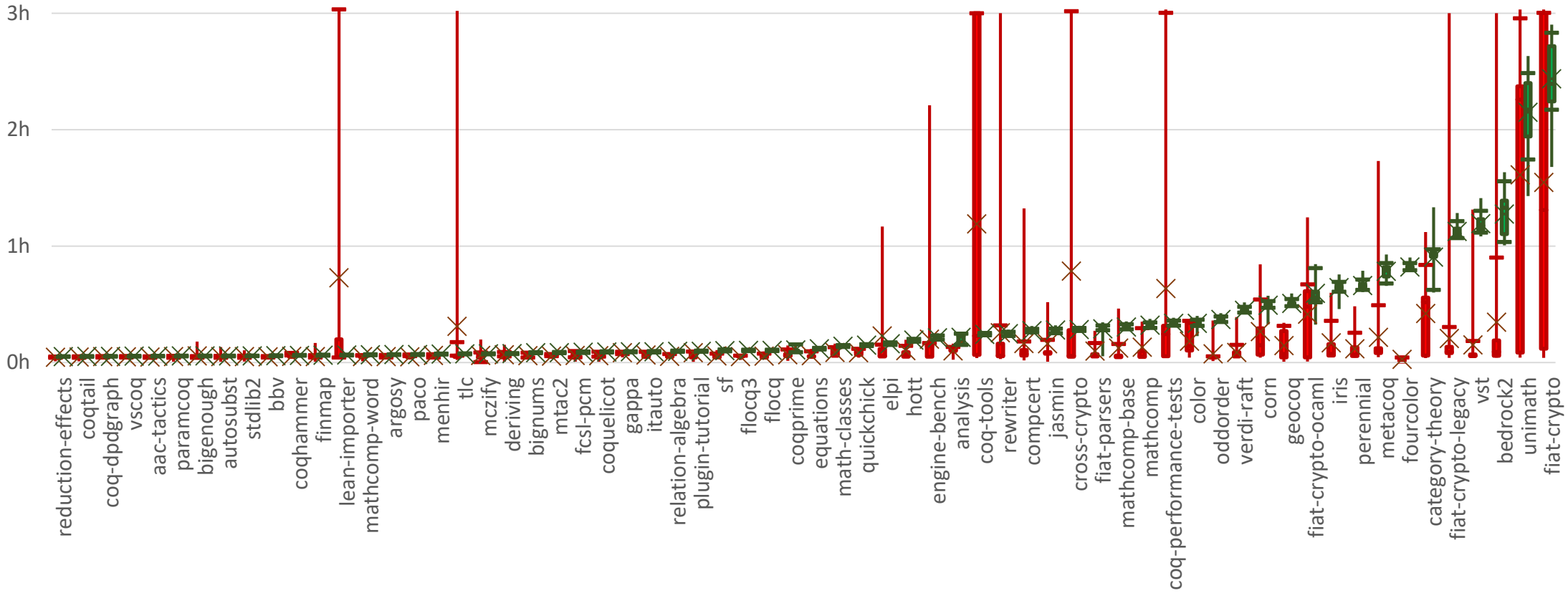
- Don’t forget the last mile of automating integration with CI!

Future Work

- More automation of bug reporter work flow
- Better inlining
- More minimization
- Other proof assistants???

Extra Content

How Long Do Projects Take On CI?



Selected CI Jobs, January through July 2022
Modified box plot for seven-number summary + mean