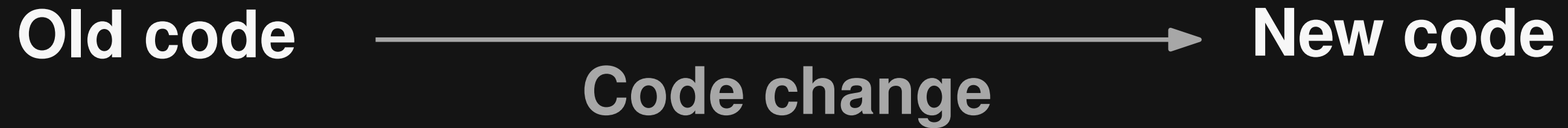


Natural Language as Specification: LLM-Based Validation of Software Evolution

Michael Pradel (CISPA, Germany)

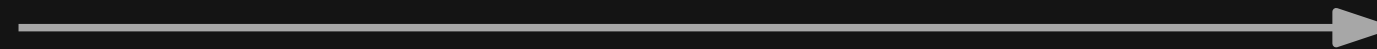


Software is Continuously Evolving



Software is Continuously Evolving

Old code



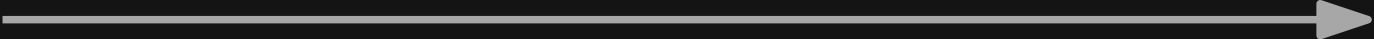
Code change

New code

The screenshot shows the GitHub interface for the SciPy repository. At the top left, the SciPy logo and name are displayed next to a 'Public' badge. To the right are interaction buttons: 'Sponsor', 'Watch' (341), 'Fork' (5.7k), and 'Star' (14.6k). Below this is a navigation bar with a 'main' branch selector, a search box labeled 'Go to file', and a '+ Code' button. A commit by 'andyfaff' is highlighted, with the message 'CI: win_arm64 removes LLVM...', a green checkmark, the hash '1c178d9', and the time '9 hours ago'. To the right, an 'About' section identifies it as the 'SciPy library main repository' and includes a link to 'scipy.org'.

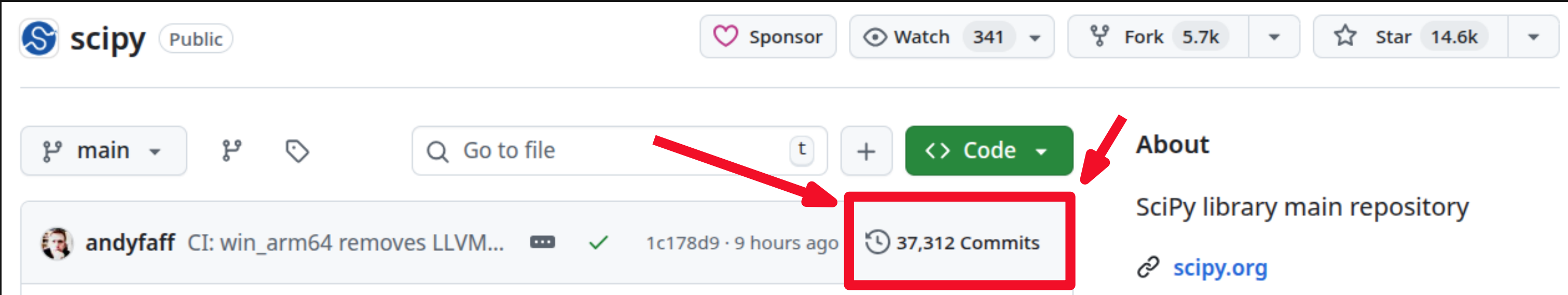
Software is Continuously Evolving

Old code



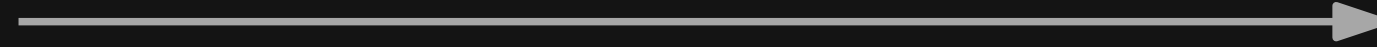
New code

Code change



Software is Continuously Evolving

Old code



New code

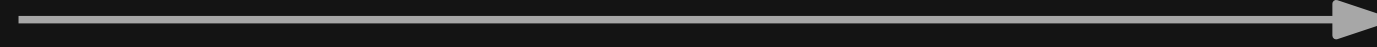
Code change

Intent



Software is Continuously Evolving

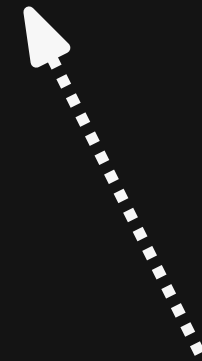
Old code



New code

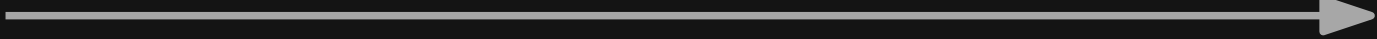
Code change

Intent



Software is Continuously Evolving

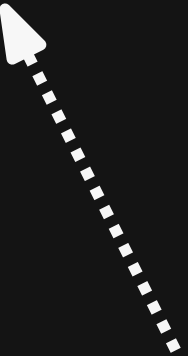
Old code



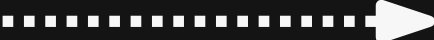
New code

Code change

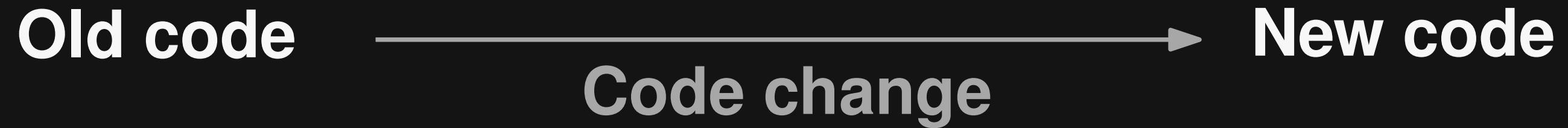
Intent



Intent

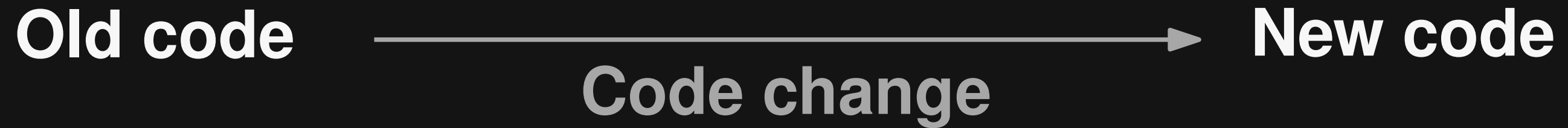


Software is Continuously Evolving



How to check correctness?

Software is Continuously Evolving

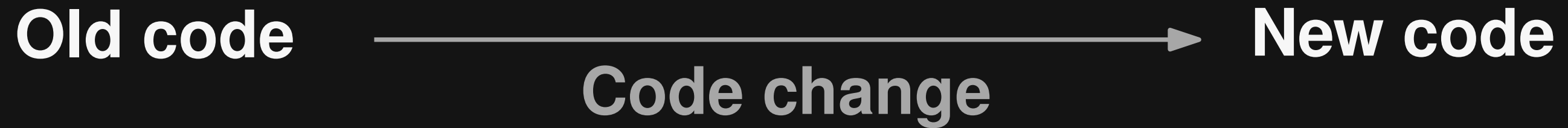


How to check correctness?

Run the regression tests!

**Checks that behavior stays as-is,
but cannot validate behavioral changes**

Software is Continuously Evolving

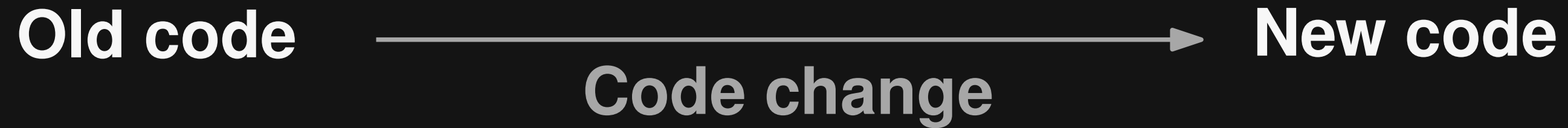


How to check correctness?

Add new tests!

**Typically limited to very few tests
per code change**

Software is Continuously Evolving

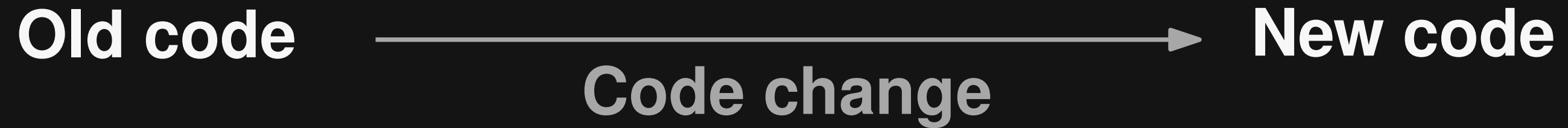


How to check correctness?

Static analysis and code reviewing!

**Useful, but can only approximate
runtime behavior**

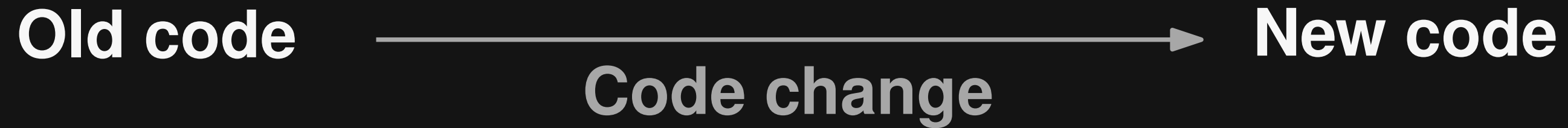
Software is Continuously Evolving



How to check correctness?

Ideally: Specification of intended behavior and intent of the change

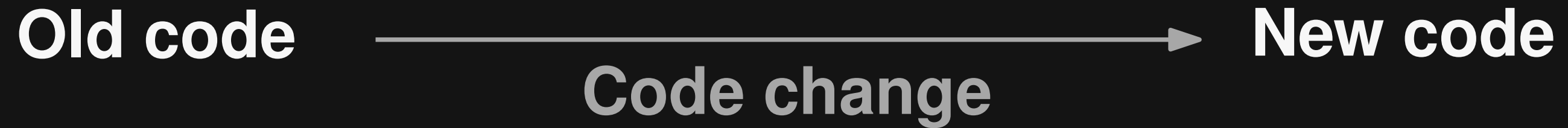
Software is Continuously Evolving



How to check correctness?

~~Ideally: Specification of intended behavior and intent of the change~~

Software is Continuously Evolving



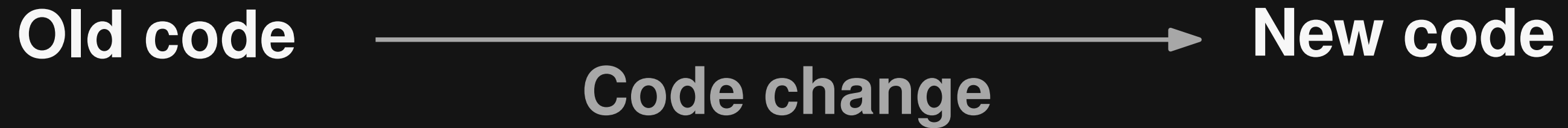
How to check correctness?

Reality:

Natural language* \approx Specification

* Issues, pull requests, prompts, etc. 2 - 13

Software is Continuously Evolving



**Idea: Natural language as oracle
to validate code changes**

This Talk

- **Testora**: Using Natural Language Intent to Detect Behavioral Regressions

[Pradel, ICSE'26]

- **PatchGuru**: Patch Oracle Inference from Natural Language Artifacts with Large Language Models

[Joint work with Le-Cong, Le, Murray, Cadar; under submission]

This Talk

- **Testora**: Using Natural Language Intent to Detect Behavioral Regressions

[Pradel, ICSE'26]

- **PatchGuru**: Patch Oracle Inference from Natural Language Artifacts with Large Language Models

[Joint work with Le-Cong, Le, Murray, Cadar; under submission]

Motivating Example

ENH: `stats.differential_entropy`: add array API support #21076 <> Code

Merged

j-bowhay merged 5 commits into `scipy:main` from `mdhaber:xp_differential_entropy` on Jul 1, 2024

Conversation 5 | Commits 5 | Checks 34 | Files changed 2 | +101 -93

mdhaber commented on Jun 28, 2024 Contributor

Reference issue

Toward [gh-20544](#)

What does this implement/fix?

Adds array API support to `differential_entropy`.

Reviewers

j-bowhay ✓

Assignees

No one assigned

Labels

Pull request in *scipy* intended to support different kinds of array types

Motivating Example

Shouldn't change API behavior ... but:

```
import numpy as np
from scipy.stats import differential_entropy

values = np.array([1, 1, 2, 3, 3, 4, 5, 5, 6, 7, 8, 9, 10, 11])
result = differential_entropy(values)
print(result)
```

Motivating Example

Shouldn't change API behavior ... but:

```
import numpy as np
```

```
from scipy.stats import differential_entropy
```

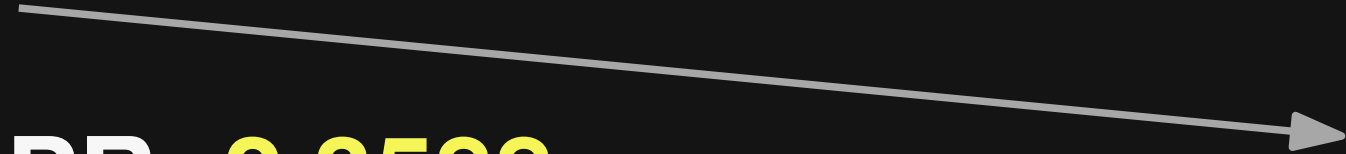
```
values = np.array([1, 1, 2, 3, 3, 4, 5, 5, 6, 7, 8, 9, 10, 11])
```

```
result = differential_entropy(values)
```

```
print(result)
```



Before PR: **2.3588**



After PR: **2.5285**

Motivating Example

Shouldn't change API behavior ... but:

```
import numpy as np
```

```
from scipy.stats import differential_entropy
```

```
values = np.array([1, 1, 2, 3, 3, 4, 5, 5, 6, 7, 8, 9, 10, 11])
```

```
result = differential_entropy(values)
```

```
print(result)
```

↓
Before PR: **2.3588**

↘
After PR: **2.5285**

↘
Unintended regression

Goal & Challenges

Goal: Check code changes for unintended regressions

- Given: Pull request
- Want: Automated detection of unintended behavioral changes

Goal & Challenges

Goal: Check code changes for unintended regressions

- Given: Pull request
- Want: Automated detection of unintended **behavioral changes**

Challenge 1: How to find behavioral changes?

→ **Insight: Target test generation at the changed code**

Goal & Challenges

Goal: Check code changes for unintended regressions

- Given: Pull request
- Want: Automated detection of unintended behavioral changes

Challenge 1: How to find behavioral changes?

→ Insight: **Target test generation** at the changed code

Challenge 2: How to know what's intended?

→ Insight: **Use natural language intent as a specification**

Testora: Overview

Pull request (PR)

Targeted test generation

Test cases

Differential testing

Behavioral difference

Classification

**No behavioral
difference**

“Intended”

“Unintended” + test + explanation

Selecting PRs to Analyze

All PRs of a project



Modifies only test code, README, configuration files, etc.

Modifies only comments

PRs to analyze in detail

Targeted Test Generation

Expose behavioral differences between old and new code

- 1) Gather **relevant information** about the code change
- 2) Query LLM to **generate tests**
- 3) Find and fix **undefined references**

Targeted Test Generation

Expose behavioral differences between old and new code

- 1) Gather **relevant information** about the code change
- 2) Query LLM to **generate tests**
- 3) Find and fix **undefined references**

Raw diff

```
diff --git a/scipy/stats/_entropy.py b/sci
index 4d033bae4752..5c575fff11d0 100644
--- a/scipy/stats/_entropy.py
+++ b/scipy/stats/_entropy.py
@@ -9,7 +9,7 @@
 import numpy as np
 from scipy import special
 from ._axis_nan_policy import _axis_nan_p
-from scipy._lib._array_api import array_n
+from scipy._lib._array_api import array_n

all = ['entropy', 'differential entro
```

Hunks of the diff

```
335 336
336 - sorted_data = np.sort(values, axis=-1)
337 + sorted_data = xp.sort(values, axis=-1)
337 338

358 359      if base is not None:
359 - res /= np.log(base)
360 + res /= math.log(base)
360 361
```

**Names of
surrounding
functions**

Targeted Test Generation

Expose behavioral differences between old and new code

- 1) Gather **relevant information** about the code change
- 2) Query LLM to **generate tests**
- 3) Find and fix **undefined references**

“Provide self-contained usage examples of `<lib>` that show differences between old and new code”



Targeted Test Generation

Expose behavioral differences between old and new code

- 1) Gather **relevant information** about the code change
- 2) Query LLM to **generate tests**
- 3) Find and fix **undefined references**

- Statically detect **undefined references**

- Ask LLM to refine test case

```
import numpy as np
```

```
values = np.array([1, 1, 2, 3, 3, 4])
```

```
result = differential_entropy(values)
```

```
print(result)
```

Targeted Test Generation

Expose behavioral differences between old and new code

- 1) Gather **relevant information** about the code change
- 2) Query LLM to **generate tests**
- 3) Find and fix **undefined references**

- Statically detect

undefined references

- Ask LLM to
refine test case

```
import numpy as np
```

```
values = np.array([1, 1, 2, 3, 3, 4])
```

```
result = differential_entropy(values)
```

```
print(result)
```

Targeted Test Generation

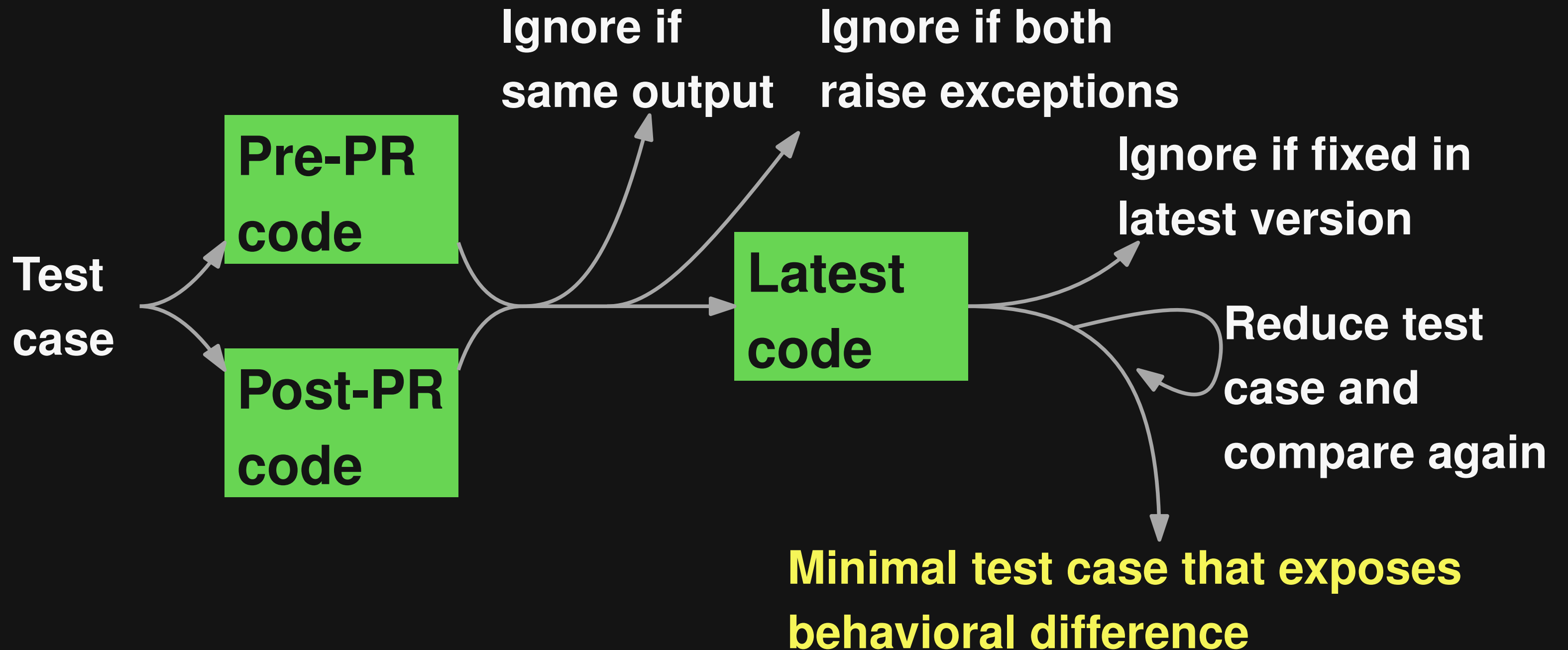
Expose behavioral differences between old and new code

- 1) Gather **relevant information** about the code change
- 2) Query LLM to **generate tests**
- 3) Find and fix **undefined references**

- Statically detect undefined references
- Ask LLM to refine test case

```
import numpy as np
from scipy.stats import differential_entropy
values = np.array([1, 1, 2, 3, 3, 4])
result = differential_entropy(values)
print(result)
```

Differential Testing



Each runs the test in a separate Docker container with the built project

Classifying Behavioral Differences

Is the behavioral difference intended?

Example:

ENH: `special.logsumexp`: improve precision when one element is much bigger than the rest #21597

Merged lucascolley merged 6 commits into `scipy:main` from `mdhaber:gh18295` on Sep 21, 2024

Conversation 43 Commits 6 Checks 0 Files changed 3



mdhaber commented on Sep 20, 2024 • edited

Contributor

Reviewers

jakevdp

lucascolley

stefanv

Reference issue

Closes [gh-18295](#)

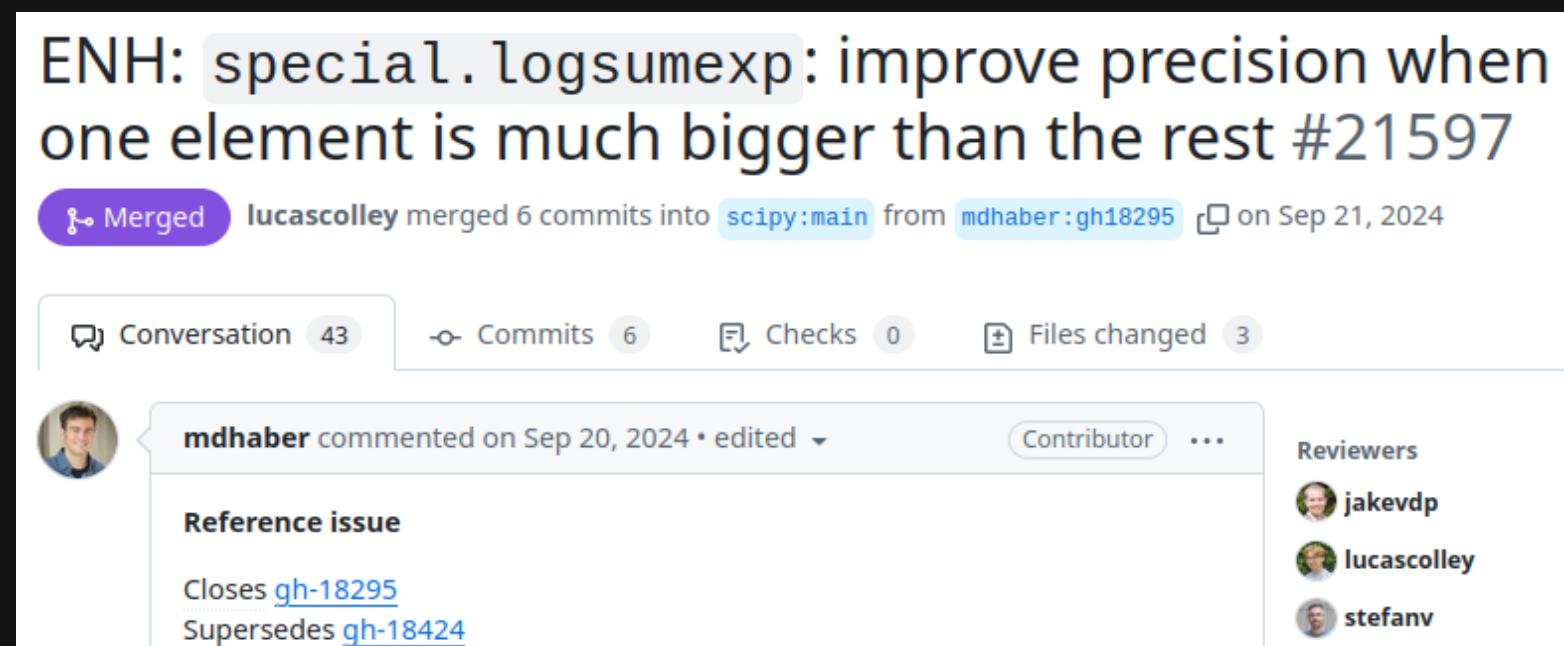
Supersedes [gh-18424](#)

```
import numpy as np
from scipy.special import logsumexp
a = np.array([1.0, 2.0, 3.0])
result = logsumexp(a)
print(result)
```

Classifying Behavioral Differences

Is the behavioral difference intended?

Example:



ENH: `special.logsumexp`: improve precision when one element is much bigger than the rest #21597

Merged lucascolley merged 6 commits into `scipy:main` from `mdhaber:gh18295` on Sep 21, 2024

Conversation 43 Commits 6 Checks 0 Files changed 3

mdhaber commented on Sep 20, 2024 • edited

Reference issue

Closes [gh-18295](#)

Supersedes [gh-18424](#)

Contributor: mdhaber

Reviewers: jakevdp, lucascolley, stefanv

```
import numpy as np
from scipy.special import logsumexp
a = np.array([1.0, 2.0, 3.0])
result = logsumexp(a)
print(result)
```

Before PR:

3.4076059644443806

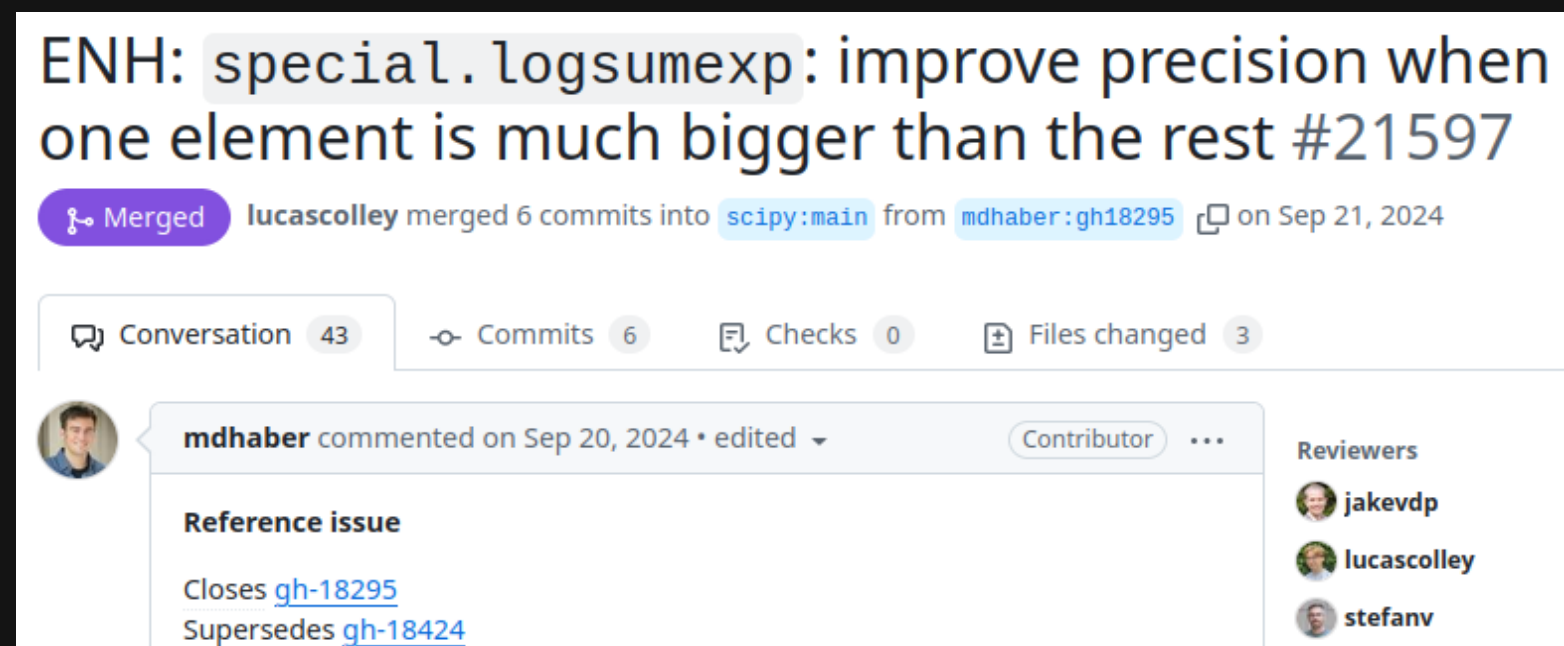
After PR:

3.40760596444438

Classifying Behavioral Differences

Is the behavioral difference intended?

Example:



ENH: `special.logsumexp`: improve precision when one element is much bigger than the rest #21597

Merged lucascolley merged 6 commits into `scipy:main` from `mdhaber:gh18295` on Sep 21, 2024

Conversation 43 Commits 6 Checks 0 Files changed 3

mdhaber commented on Sep 20, 2024 • edited

Reference issue

Closes [gh-18295](#)

Supersedes [gh-18424](#)

Contributor: mdhaber

Reviewers: jakevdp, lucascolley, stefanv

```
import numpy as np
from scipy.special import logsumexp
a = np.array([1.0, 2.0, 3.0])
result = logsumexp(a)
print(result)
```

Before PR:

3.4076059644443806

After PR:

3.40760596444438

Intended behavioral change

Classifying Behavioral Differences

Context about the code change:

- PR title & description
- Diff
- Test case
- Old & new output
- Docstrings of called APIs

Five questions:

- Noteworthy vs. minor change?
- Deterministic?
- Uses public APIs only?
- Legal API usage?
- Matches PR intent?



Classifying Behavioral Differences

Context about the code change:

- PR title & description
- Diff
- Test case
- Old & new output
- Docstrings of called APIs

Five questions:

- Noteworthy vs. minor change?
- Deterministic?
- Uses public APIs only?
- Legal API usage?
- Matches PR intent?

- Noteworthy
- Yes
- Yes
- Yes
- No

LLM

Raise warning if **answers align**

Experimental Setup

- PRs from popular **Python projects**:
keras, marshmallow, pandas, scipy
- **Three LLMs**: GPT-4o-mini, GPT-4o, DeepSeek-R1
- **Research questions**
 - RQ1: **Real-world problems**
 - RQ2: Effectiveness of **test generation**
 - RQ3: Accuracy of **classifier**
 - RQ4: **Costs**

RQ1: Real-World Problems

Does it find unintended changes?

30 PRs with unintended behavioral changes

- 19 regressions
 - 13 reported (others were fixed independently)
 - 12 confirmed, 11 fixed
- 11 coincidental fixes

Example of Regression

Faster in_top_k implementation for Jax backend #19814

Merged fchollet merged 2 commits into keras-team:master from Hilly12:master on Jun 7, 2024

Conversation 3 Commits 2 Checks 6 Files changed 1

Hilly12 commented on Jun 7, 2024

Microbenchmarks are here [1]. Seems to be faster on CPU, GPU, TPU. Also XLA is sometimes able to dedup everything before k is referenced when it's called multiple times with different ks. This results in nice gains for retrieval applications with multiple top k metrics.

Reviewers: fchollet

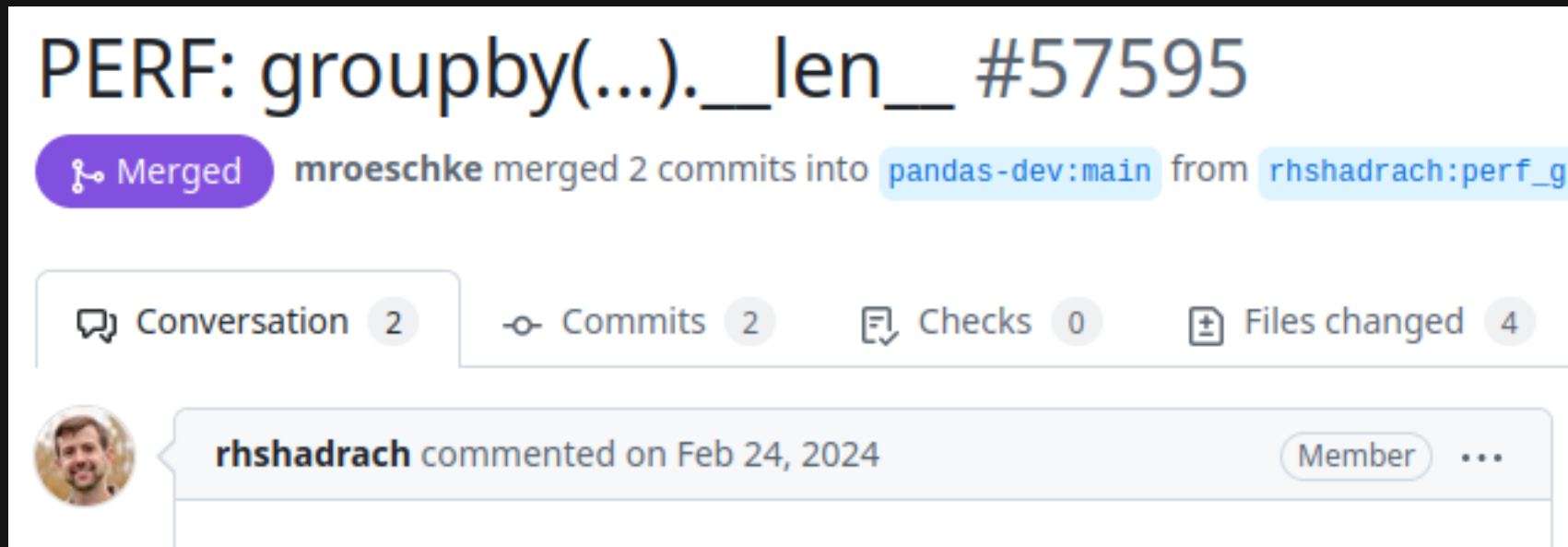
Assignees: gbaned

```
import jax.numpy as jnp
from numpy import nan
from keras.src.backend.jax.math import in_top_k
r = in_top_k(targets=jnp.array([1, 0]),
              predictions=jnp.array([[.1, nan, .5],
                                     [.3, .2, .5]]), k=2)
print(r)
```

Before PR:
[False True]

After PR:
[True True]

Example of Coincidental Fix



```
import pandas as pd
```

```
df = pd.DataFrame({'A': ['foo', 'bar', None],  
                  'B': [1, 2, 3]})
```

```
grouped = df.groupby('A', dropna=False)
```

```
print(len(grouped))
```

Before PR:

Exception

After PR:

3

RQ2: Effectiveness of Test Generation

Does it find behavioral differences?

Project	Pull requests			
	Total	Ignored	Checked	Behavioral difference found
keras	271	111	160	0
marshmallow	138	73	65	8
pandas	439	296	143	35
scipy	426	269	157	57
Total	1,274	749	525	100

RQ3: Accuracy of Classifier

**Dataset of 164 labeled behavioral differences
(139 intended, 25 unintended)**

LLM	Single-question classifier			Multi-question classifier		
	Precision	Recall	F1	Precision	Recall	F1
GPT-4o-mini	49%	80%	61%	55%	67%	60%
GPT-4o	80%	64%	71%	71%	42%	53%
DeepSeek-R1	69%	36%	47%	83%	42%	56%

RQ4: Costs

- **Tokens (per PR)**

- 5,818 input tokens, 3,622 output tokens
- Equivalent to **USD 0.003**

- **Time (per PR)**

- **12.3 minutes**
- Largest contributors: Build & test execution

This Talk

- **Testora**: Using Natural Language Intent to Detect Behavioral Regressions

[Pradel, ICSE'26]

- **PatchGuru**: Patch Oracle Inference from Natural Language Artifacts with Large Language Models

[Joint work with Le-Cong, Le, Murray, Cadar; under submission]

This Talk

- **Testora**: Using Natural Language Intent to Detect Behavioral Regressions

[Pradel, ICSE'26]

- **PatchGuru**: Patch Oracle Inference from Natural Language Artifacts with Large Language Models

[Joint work with Le-Cong, Le, Murray, Cadar; under submission]

Limitations of Testora

Powerful technique, but two key limitations:

- 1) Misses bugs when **behavior should change but doesn't**
- 2) Specification remains **implicit**

Example

`fields.Url` does not accept `file` URLs without host #2249

Closed

Bug

#2800

**Bug: File URLs (e.g., “file:///dir/file.zip”)
cause `ValidationError`**

Example

`fields.Url` does not accept `file` URLs without host #2249

Closed

Bug

#2800

Bug: File URLs (e.g., “file:///dir/file.zip”) cause `ValidationError`

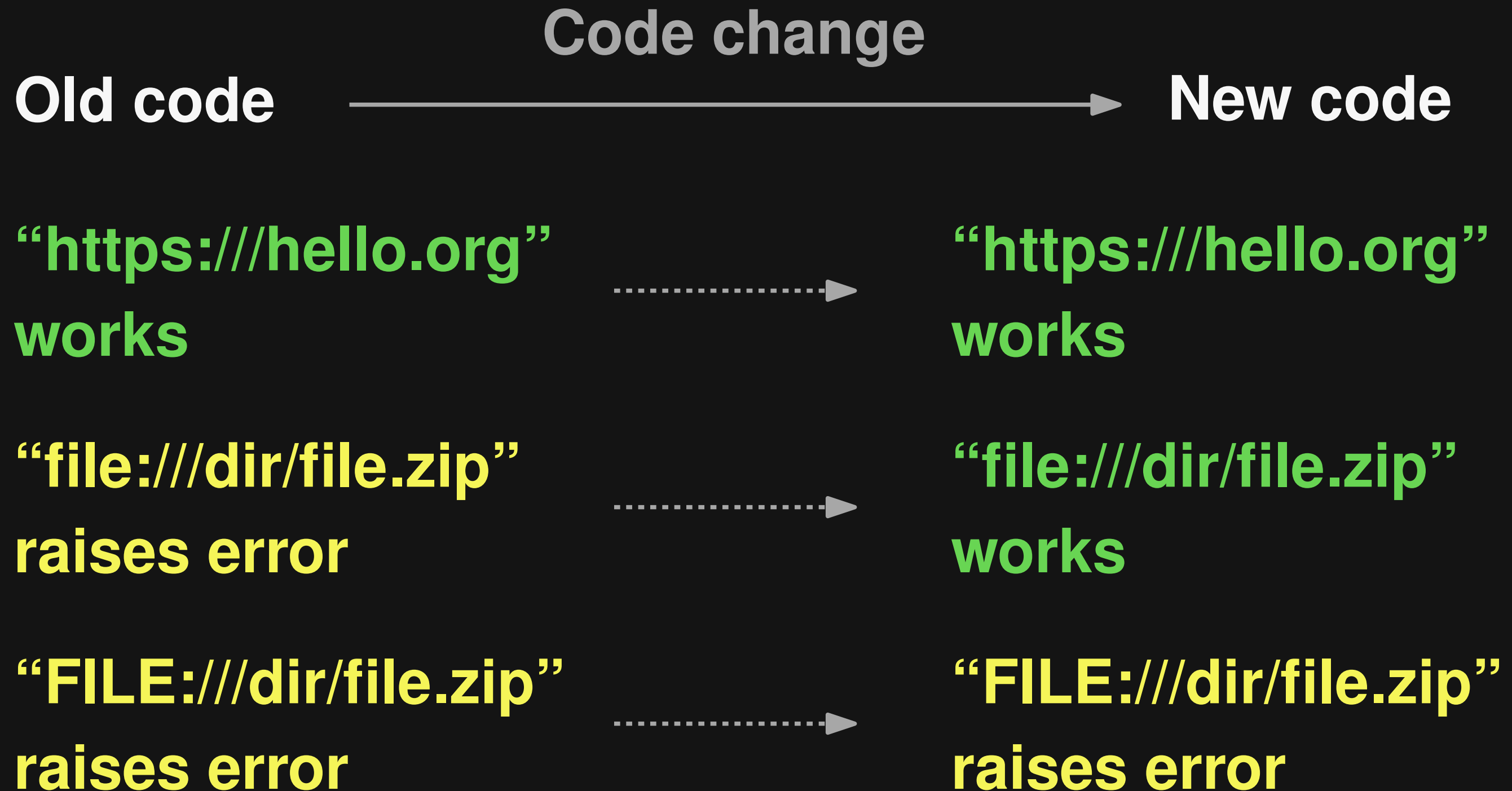
**Fix attempt
in PR**

fix: add `file` handling to URL fields #2800

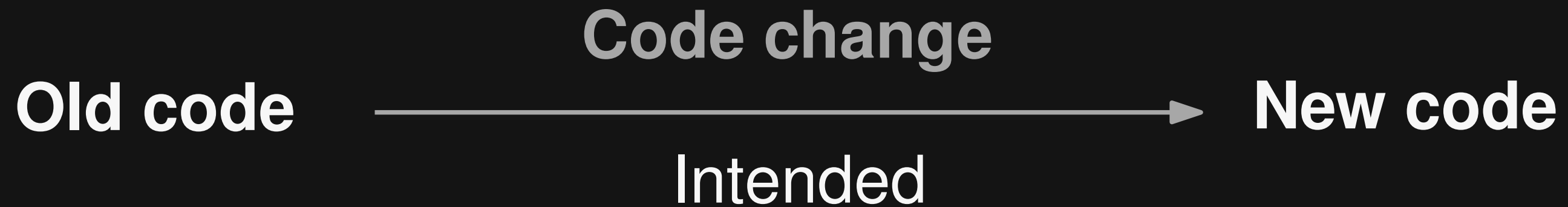
Merged

[sloria](#) merged 2 commits into [marshmallow-code:3.x-line](#) from [0xDEC0DE:issue/2249](#) on Jan 23, 2025

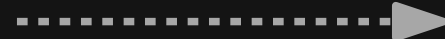
Example (cont.)



Example (cont.)

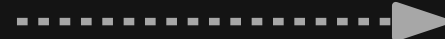


“https:///hello.org”
works



“https:///hello.org”
works

“file:///dir/file.zip”
raises error



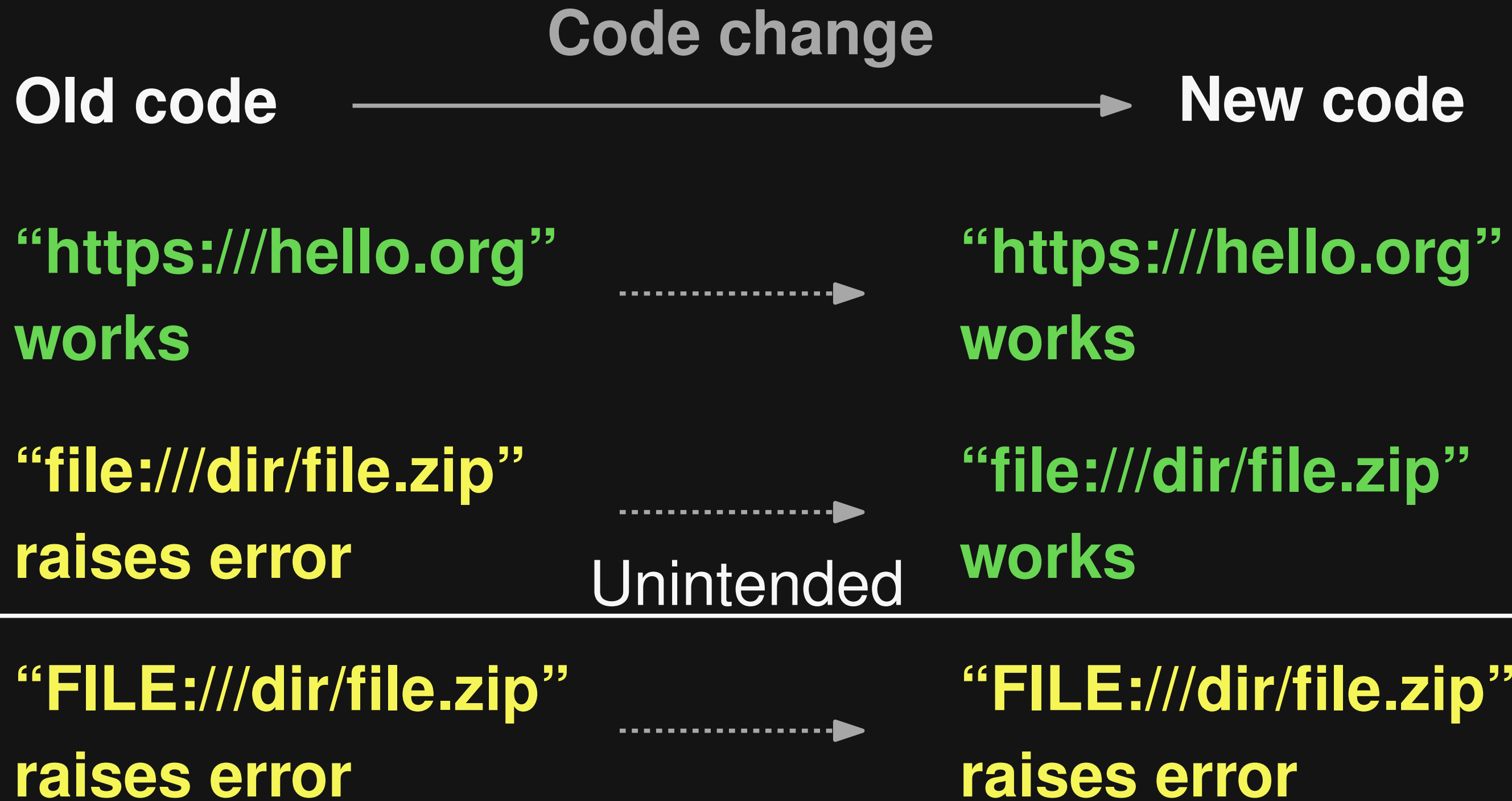
“file:///dir/file.zip”
works

“FILE:///dir/file.zip”
raises error



“FILE:///dir/file.zip”
raises error

Example (cont.)



PatchGuru: Idea

NL intent



Patch oracle

PatchGuru: Idea

- **PR** description,
related issues, etc.

NL intent



Patch oracle

PatchGuru: Idea

- **PR** description,
related issues, etc.

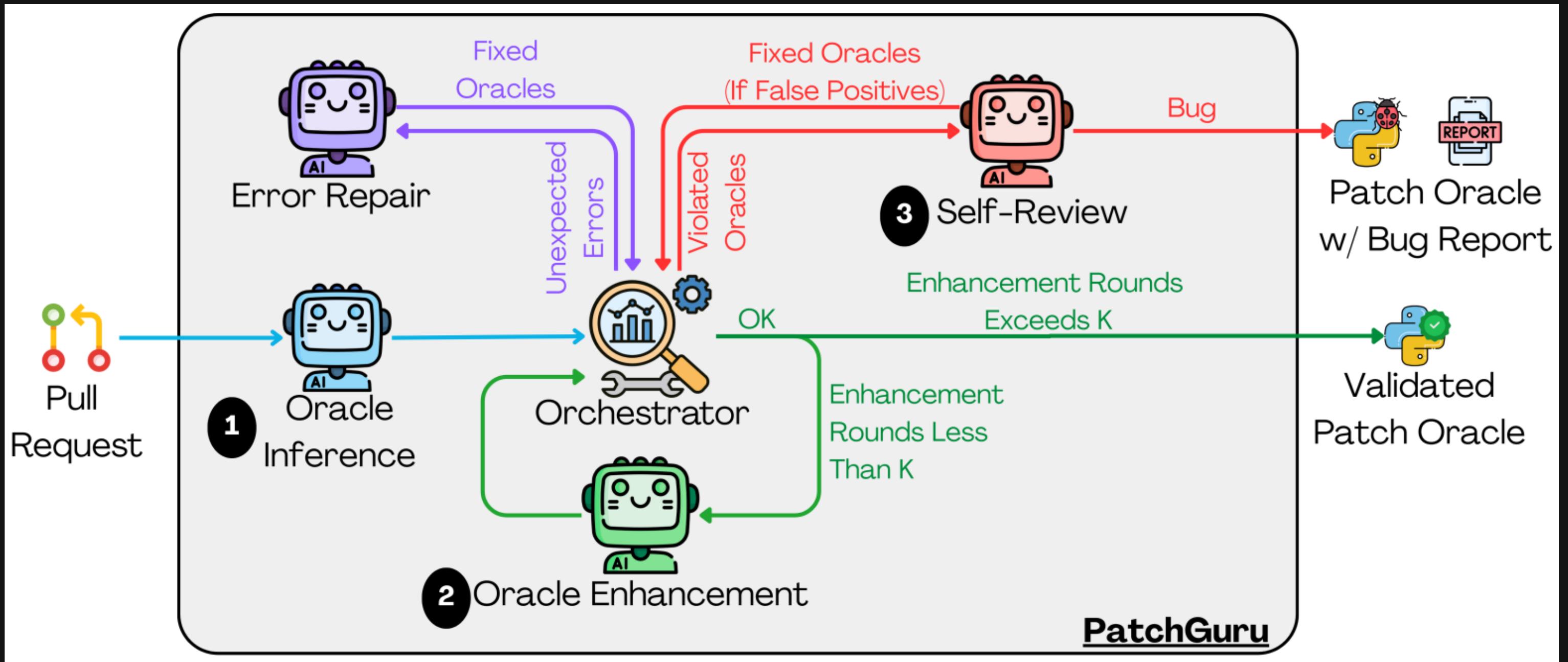
NL intent



Patch oracle

- Runtime **assertions** that **old and new behavior**
- Checked in “**comparison program**”
(runs old and new code side by side)

Overview of PatchGuru



Oracle Inference

Goal: **Formalize** intended **pre-PR** and **post-PR** **behavior** into assertions

PR (description,
diff, etc.)

Related issues

Code context

(imported files,
class definitions, etc.)

LLM

Patch
oracle

Oracle Inference

Goal: **Formalize** intended **pre-PR** and **post-PR** **behavior** into assertions

PR (description,
diff, etc.)

Related issues

Code context

(imported files,
class definitions, etc.)

LLM-based
distillation

LLM

Patch
oracle

Patch Oracle: Example

```
# Preserved behavior: Valid URLs
valid_url = "http://example.com/path"
pre_res, pre_exc, post_res, post_exc = \
    call_impl(pre_validation, post_validation, valid_url)
assert pre_exc is None and \
    post_exc is None and \
    pre_res == valid_url and \
    post_res == valid_url
```

Patch Oracle: Example

```
# Preserved behavior: Valid URLs
```

```
valid_url = "http://example.com/path"
```

Input to test the
modified code

```
pre_res, pre_exc, post_res, post_exc = \
```

```
    call_impl(pre_validation, post_validation, valid_url)
```

```
assert pre_exc is None and \
```

```
    post_exc is None and \
```

```
    pre_res == valid_url and \
```

```
    post_res == valid_url
```

Patch Oracle: Example

```
# Preserved behavior: Valid URLs
```

```
valid_url = "http://example.com/path"
```

```
pre_res, pre_exc, post_res, post_exc = \
```

```
    call_impl(pre_validation, post_validation, valid_url)
```

```
assert pre_exc is None and \
```

```
    post_exc is None and \
```

```
    pre_res == valid_url and \
```

```
    post_res == valid_url
```

**Pre-PR and post-PR
versions of the
validate function**

Patch Oracle: Example

```
# Preserved behavior: Valid URLs
```

```
valid_url = "http://example.com/path"
```

```
pre_res, pre_exc, post_res, post_exc = \  
    call_impl(pre_validation, post_validation, valid_url)
```

```
assert pre_exc is None and \  
    post_exc is None and \  
    pre_res == valid_url and \  
    post_res == valid_url
```

Invokes both versions and stores their results (return values, raised exceptions)

Patch Oracle: Example

```
# Preserved behavior: Valid URLs
valid_url = "http://example.com/path"
pre_res, pre_exc, post_res, post_exc = \
    call_impl(pre_validation, post_validation, valid_url)
```

```
assert pre_exc is None and \
    post_exc is None and \
    pre_res == valid_url and \
    post_res == valid_url
```

**Check and compare
behavior of old/new code**

Patch Oracle: Example (cont.)

```
# Changed behavior: File URLs
file_url = "file:///etc/passwd"
pre_res, pre_exc, post_res, post_exc = \
    call_impl(pre_validation, post_validation, file_url)
assert isinstance(pre_exc, ValidationError) \
    and post_exc is None \
    and post_res == file_url
```

Behavior should change for file URLs

Oracle Enhancement

Goal: Create **additional assertions** to **trigger bugs**

“Generalize assertions,
explore edge cases,
diversify inputs”

Current
patch oracle



Enhanced
patch oracle

Patch Oracle: Augmented Example

```
# Special case: Upper-cases file URLs
upper_file = "FILE:///etc/passwd"
pre_res, pre_exc, post_res, post_exc = \
    call_impl(pre_validation, post_validation, upper_file)
assert isinstance(pre_exc, ValidationError) and \
    post_exc is None and \
    post_res == upper_file
```

Fix should also consider URLs starting with “FILE”

Patch Oracle: Augmented Example

```
# Special case: Upper-cases file URLs
upper_file = "FILE:///etc/passwd"
pre_res, pre_exc, post_res, post_exc = \
    call_impl(pre_validation, post_validation, upper_file)
assert isinstance(pre_exc, ValidationError) and \
    post_exc is None and \
    post_res == upper_file
```

**Assertion fails:
Post-PR code raises
ValidationError**

Fix should also consider URLs starting with "FILE"

Self-Review

Goal: Identify **spurious assertion failures** to **eliminate false positives**

Patch oracle

Log of runtime
behavior

NL information
(PR, etc.)

LLM

```
graph LR; A[Patch oracle] --> LLM[LLM]; B[Log of runtime behavior] --> LLM; C[NL information (PR, etc.)] --> LLM; LLM --> D[True positive OR False positive]
```

True positive
OR
False positive

Example (cont.)

```
# Special case: Upper-cases file URLs
upper_file = "FILE:///etc/passwd"
pre_res, pre_exc, post_res, post_exc = \
    call_impl(pre_validation, post_validation, upper_file)
assert isinstance(pre_exc, ValidationError) and \
    post_exc is None and \
    post_res == upper_file
```

**Assertion fails:
Post-PR code raises
ValidationError**

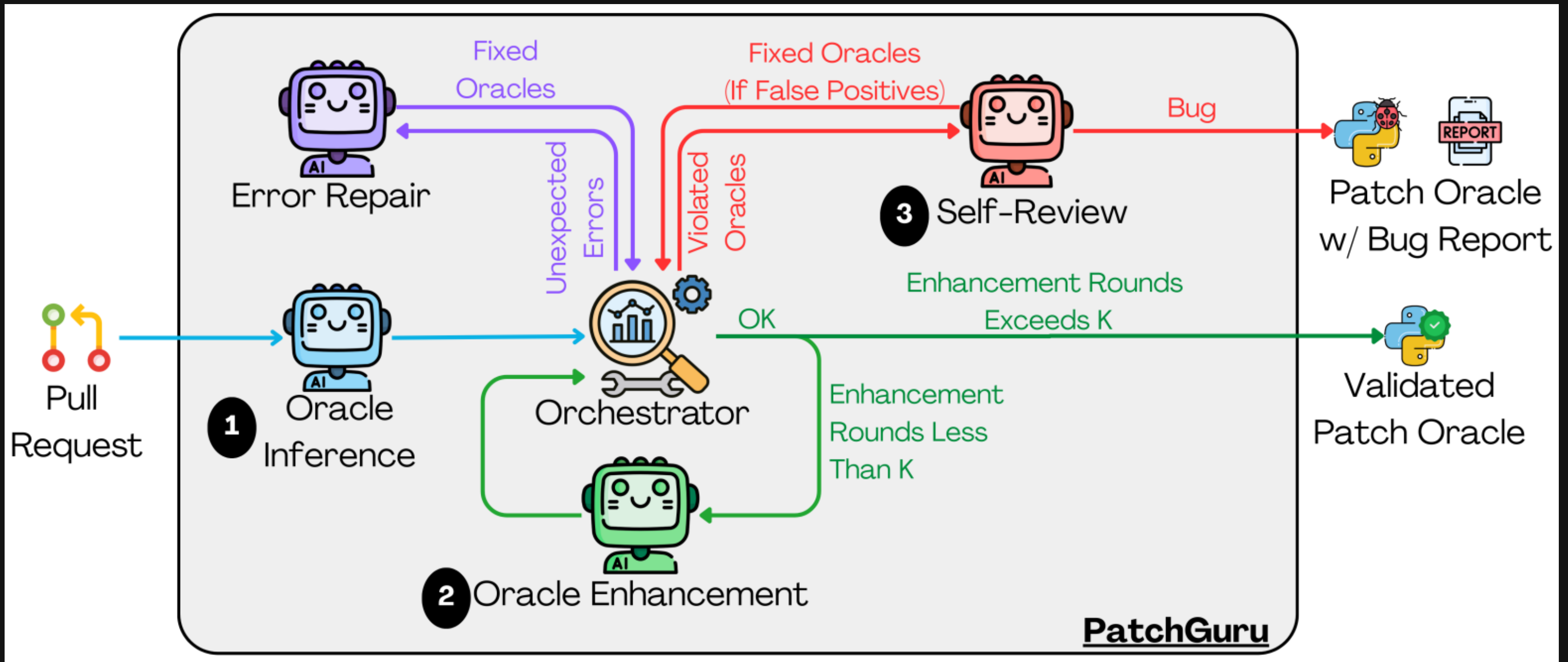
Example (cont.)

```
# Special case: Upper-cases file URLs
upper_file = "FILE:///etc/passwd"
pre_res, pre_exc, post_res, post_exc = \
    call_impl(pre_validation, post_validation, upper_file)
assert isinstance(pre_exc, ValidationError) and \
    post_exc is None and \
    post_res == upper_file
```

**Assertion fails:
Post-PR code raises
ValidationError**

**Verdict: True positive. Patch should
also handle "FILE" URLs**

Overall PatchGuru Algorithm



Experimental Setup

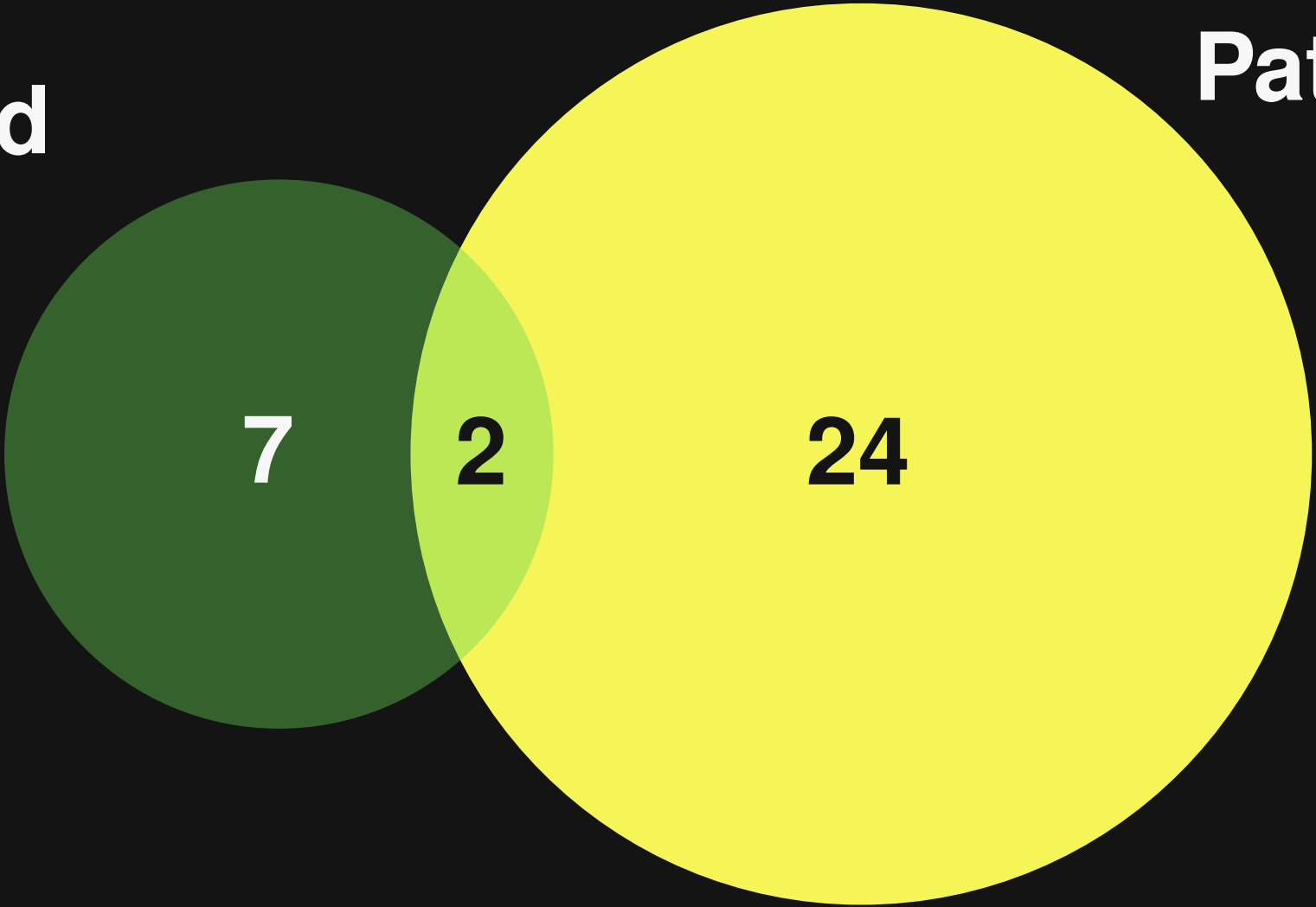
- 400 PRs from popular **Python projects**:
keras, marshmallow, pandas, scipy
- **LLM**: GPT-5-mini
- **Research questions**
 - RQ1: **Real-world problems**
 - RQ2: Adequacy of **patch oracles**
 - RQ3: **Costs**
 - RQ4: **Ablation study**

RQ1: Effectiveness

Project	#PRs	#Oracles	PatchGuru			
			#Warn	#TP	#FP	Precision
Keras	100	80	9	5	4	0.56
Marshmallow	100	83	7	4	3	0.57
Pandas	100	83	11	8	3	0.73
SciPy	100	90	12	7	5	0.58
Overall	400	336	39	24	15	0.62

Comparison with Testora

Bugs found
by Testora



Bugs found by
PatchGuru

Example: Incomplete Fix

Bug in Pandas:

String categories printed without quotes

↳ **Patch oracle to check fix:**

```
assert pre_out == post_out[1:-1]
```

↳ Works for `str`, but not for `string`

Example: Newly Introduced Bug

PR in Keras:

Fix bug in the `TimeDistributed` layer

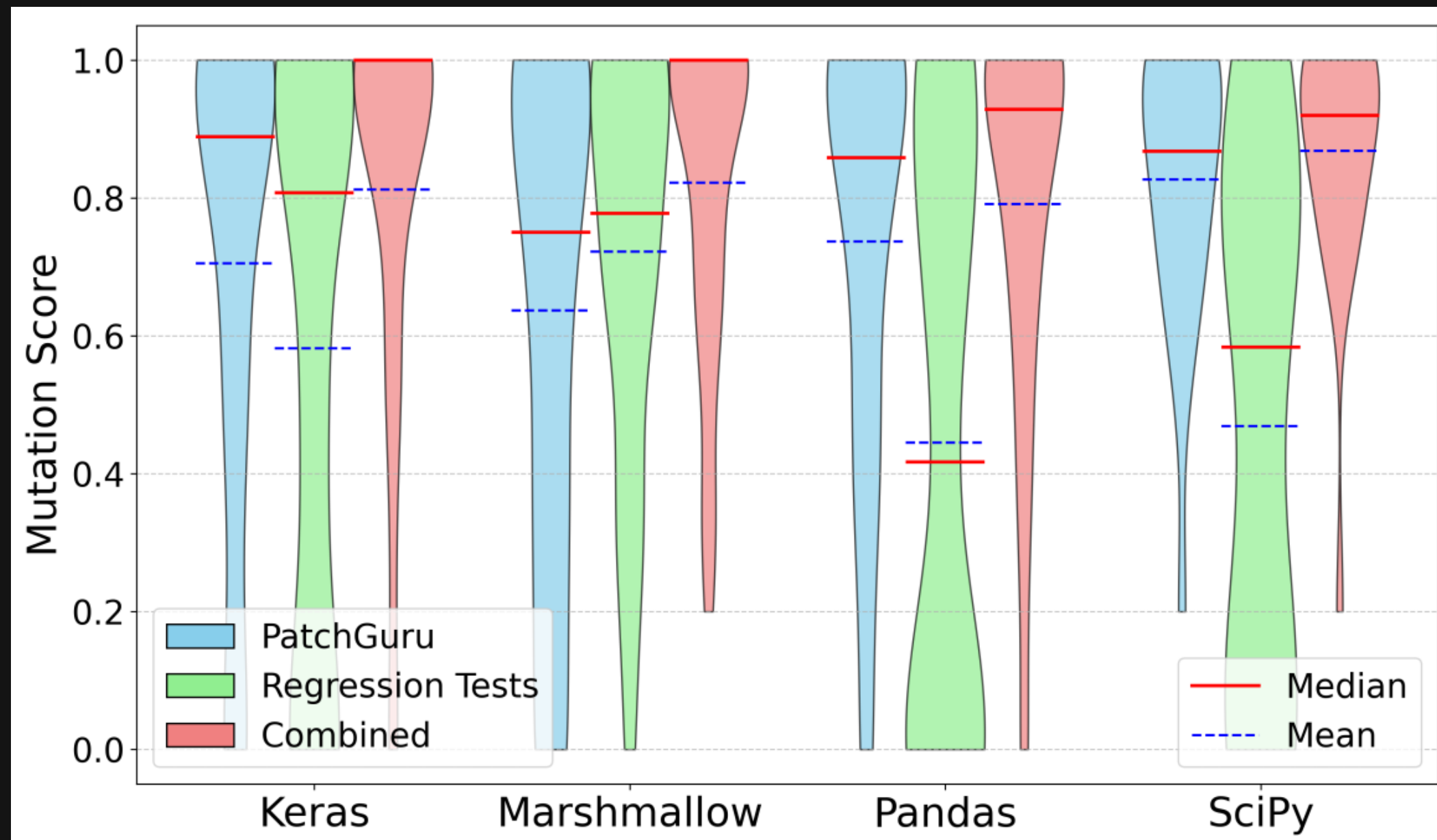
Patch oracle:

Both versions should fail when masks have mismatched timesteps

But: **Invariant broken** in new version

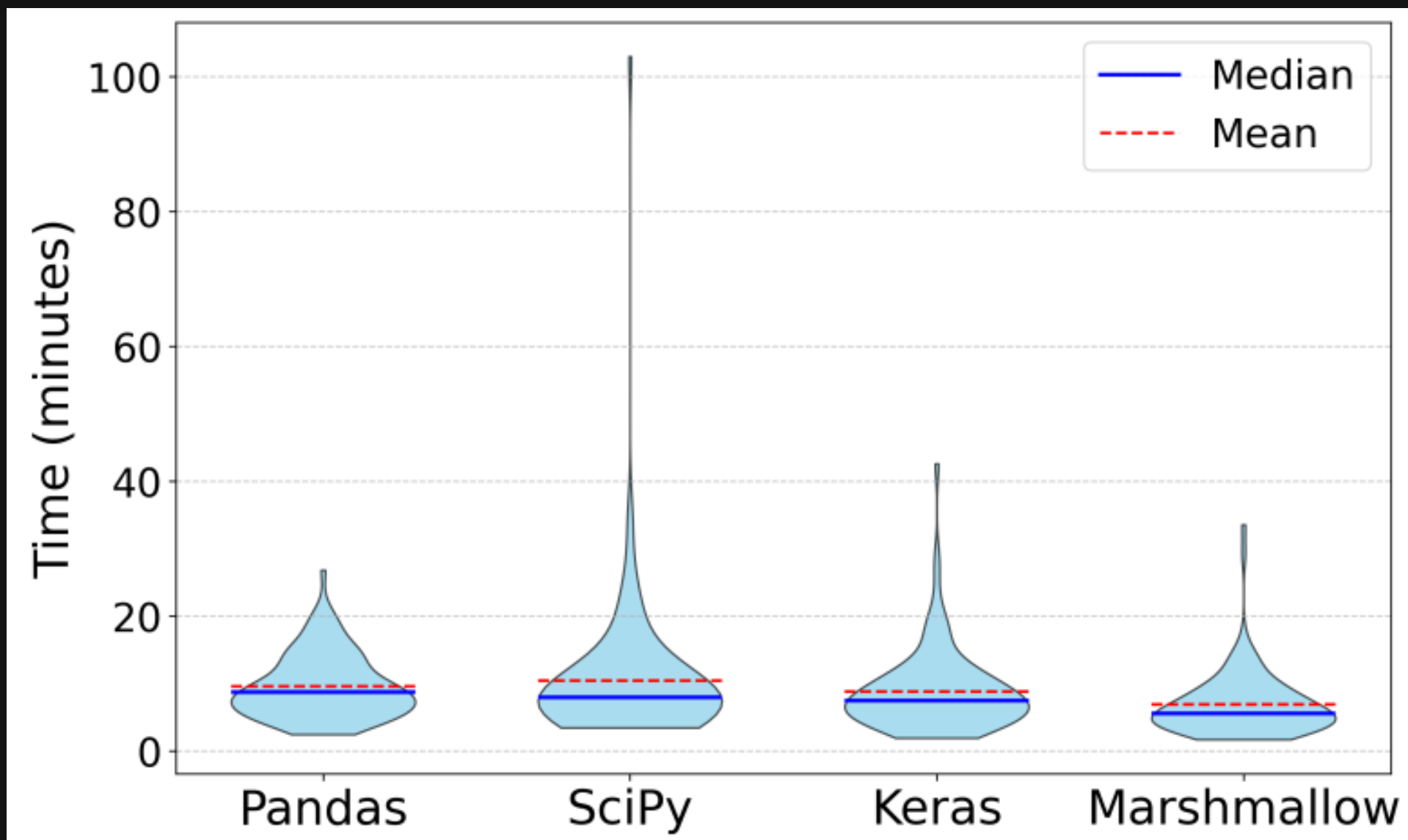
RQ2: Adequacy of Patch Oracles

Do patch oracles **kill mutants** in the patched functions?



RQ3: Costs

7–10 minutes and **USD 0.07** per PR



RQ4: Ablation Study

Method	#Warnings	#Bugs	Precision	Mut. score
PatchGuru	39	24	0.62	0.70
W/o oracle enh.	18	12	0.69	0.64
W/o self-review	195	25*	0.13*	N/A

* estimated based on random sample

RQ4: Ablation Study

Method	#Warnings	#Bugs	Precision	Mut. score
PatchGuru	39	24	0.62	0.70
W/o oracle enh.	18	12	0.69	0.64
W/o self-review	195	25*	0.13*	N/A

Oracle enhancement helps find more bugs

* estimated based on random sample

RQ4: Ablation Study

Method	#Warnings	#Bugs	Precision	Mut. score
PatchGuru	39	24	0.62	0.70
W/o oracle enh.	18	12	0.69	0.64
W/o self-review	195	25*	0.13*	N/A

Self-review reduces false positives

* estimated based on random sample

Bigger Picture

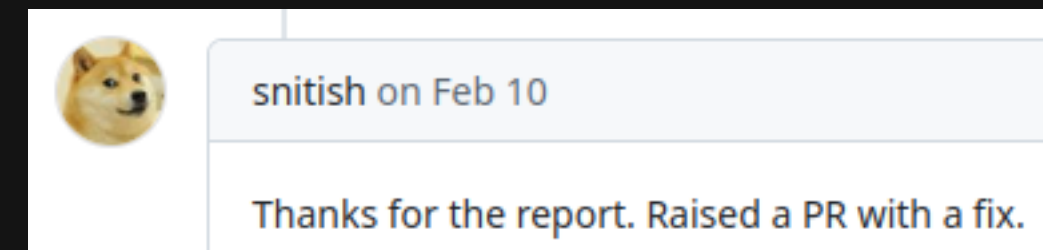
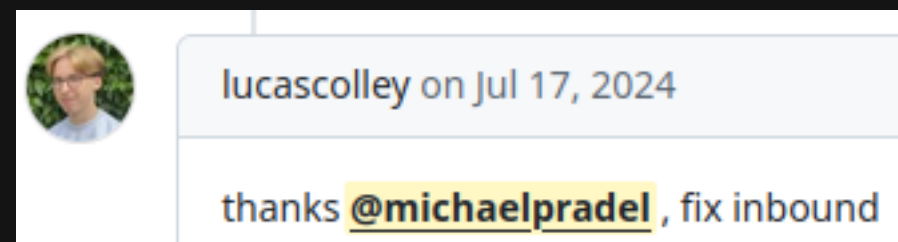
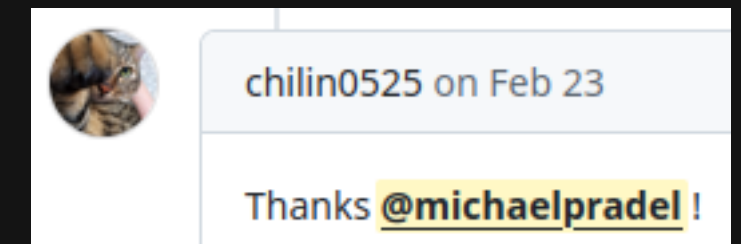
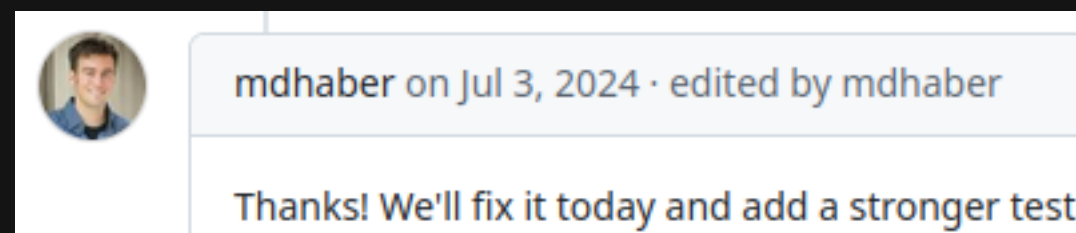
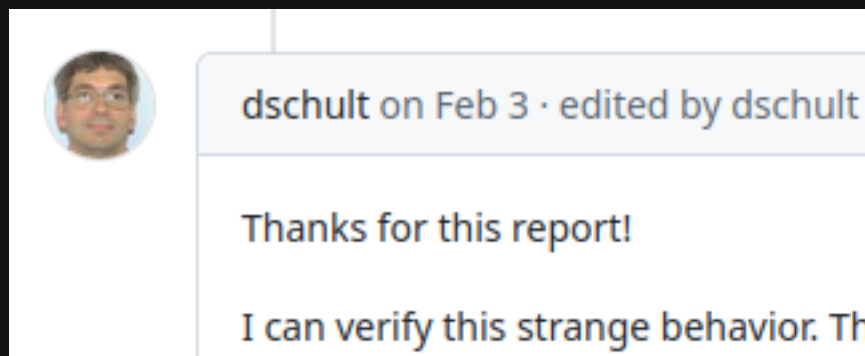
- Making **code changes** has never been this easy
- Need strong **validation techniques**
- Relying on “vague” **NL specifications** may be our best option

Envisioned Deployment

- **Run on each PR before merging**
- **If unintended change detected:
Bot adds comment and test case to PR discussion**

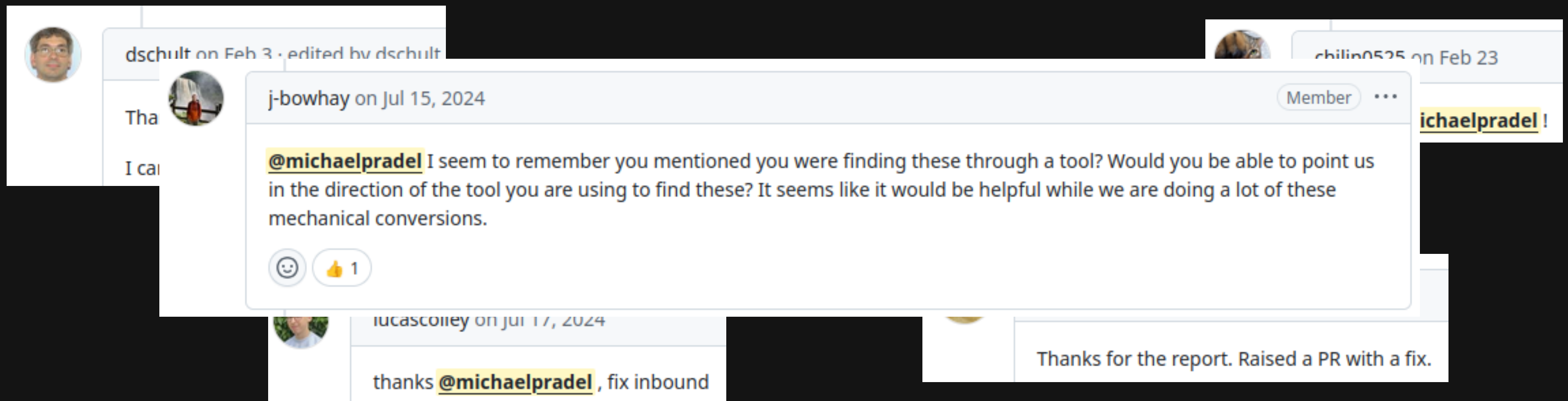
Envisioned Deployment

- **Run on each PR before merging**
- **If unintended change detected:
Bot adds comment and test case to PR discussion**



Envisioned Deployment

- Run on each PR before merging
- If unintended change detected:
Bot adds comment and test case to PR discussion



Conclusion



**Natural language as oracle
to validate code changes**

1) **Testora** [Pradel, ICSE'26]

- Using natural language intent to detect regressions

2) **PatchGuru** [Le-Cong, Le, Murray, Pradel, Cadar (under submission)]

- Patch Oracle Inference from Natural Language Artifacts