ChangeGuard: Validating Code Changes via Pairwise *Learning-Guided Execution*

Lars Gröninger, **Beatriz Souza**, and Michael Pradel

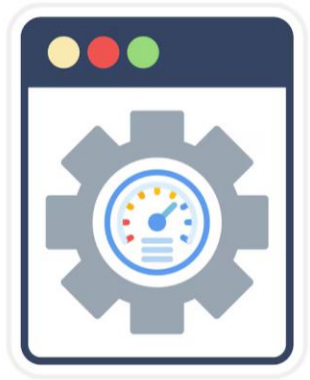
European Research Council
Established by the European Commission

Software is Continuously Evolving

New Features

Bug Fixes

Refactoring



Optimization

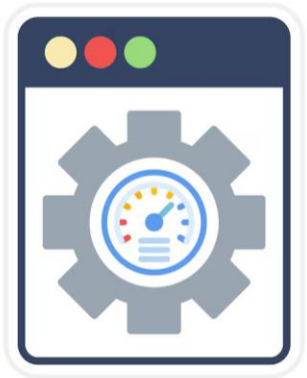
Software is Continuously Evolving

New Features

Bug Fixes

Refactoring

Should not change the semantics!



Optimization

Motivating Example

Simplify retry_times assignment statement

master (#2643) · 2.13.1 ••• 1.4.0 — 1 parent 0d57b5c commit 694c6d3

Filter files...

scrapy/downloadermiddle...

retry.py

1 file changed +1 -4 lines changed

Search within code

scrapy/downloadermiddlewares/retry.py +1 -4

```
@@ -63,10 +63,7 @@ def process_exception(self, request, exception, spider):
63  63      def _retry(self, request, reason, spider):
64  64          retries = request.meta.get('retry_times', 0) + 1
65  65
66      -       retry_times = self.max_retry_times
67      -
68      -       if 'max_retry_times' in request.meta:
69      -           retry_times = request.meta['max_retry_times']
    66  +       retry_times = request.meta.get('max_retry_times') or self.max_retry_times
70  67
71  68          stats = spider.crawler.stats
72  69          if retries <= retry_times:
```

https://github.com/scrapy/scrapy/commit/694c6d3d

Motivating Example

Simplify retry_times assignment statement

master (#2643) · 2.13.1 ••• 1.4.0 — 1 parent 0d57b5c commit 694c6d3

Filter files...

scrapy/downloadermiddle...

retry.py

1 file changed +1 -4 lines changed

Search within code

scrapy/downloadermiddlewares/retry.py +1 -4

```
@@ -63,10 +63,7 @@ def process_exception(self, request, exception, spider):
63  63      def _retry(self, request, reason, spider):
64  64          retries = request.meta.get('retry_times', 0) + 1
65  65
66      -       retry_times = self.max_retry_times
67      -
68      -       if 'max_retry_times' in request.meta:
69      -           retry_times = request.meta['max_retry_times']
    66  +       retry_times = request.meta.get('max_retry_times') or self.max_retry_times
70  67
71  68          stats = spider.crawler.stats
72  69          if retries <= retry_times:
```

Semantics changes if request.meta.get('max_retry_times') returns 0

https://github.com/scrapy/scrapy/commit/694c6d3d

Motivating Example

```
Fix bug involving OR condition
```

master (#2643) · 2.13.1 ••• 1.4.0 — 1 parent 9d97d78 commit 49c5afc

Filter files...

scrapy/downloadermiddle...

retry.py

1 file changed +4 -1 lines changed

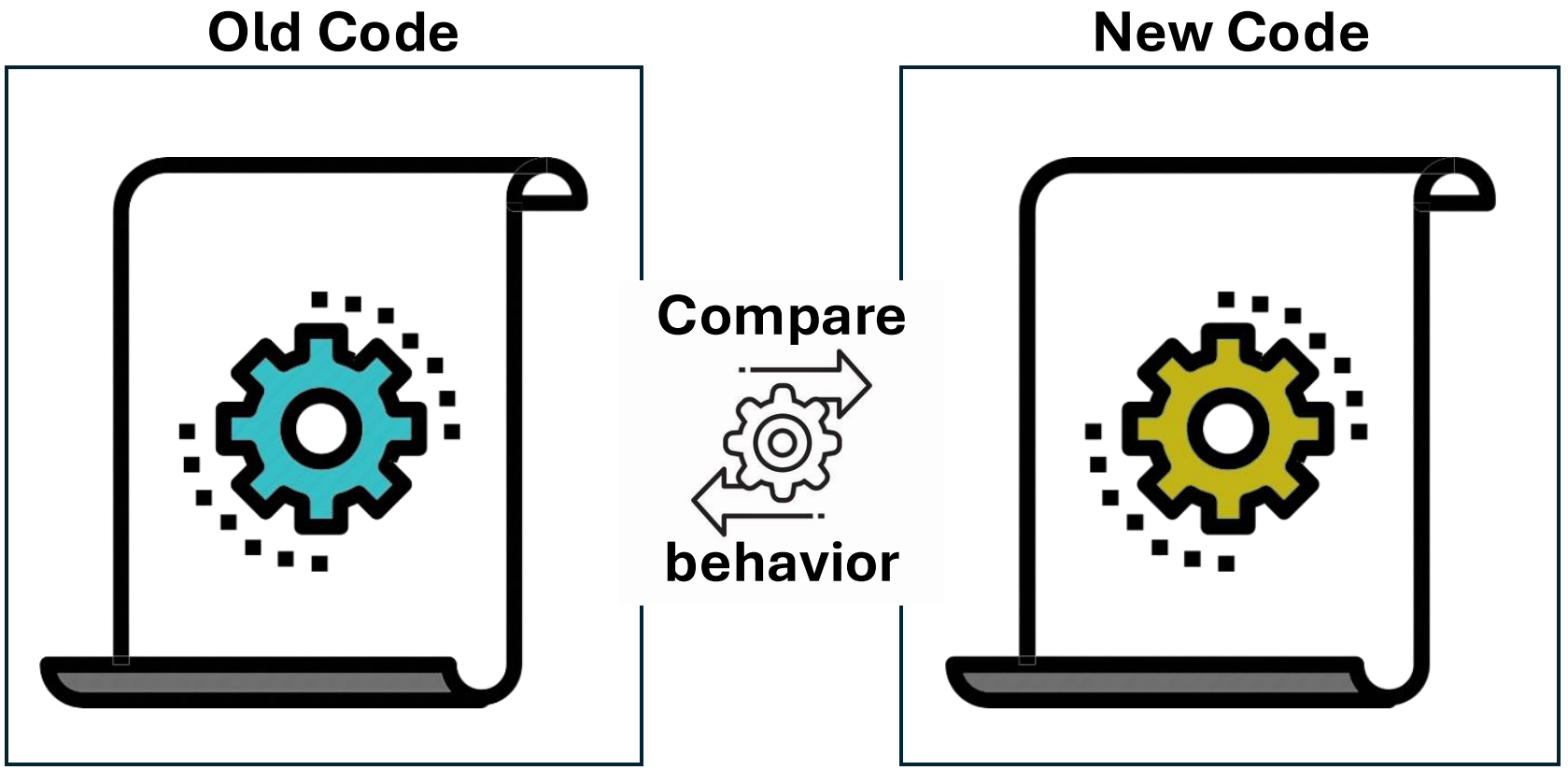
Search within code

scrapy/downloadermiddlewares/retry.py +4 -1

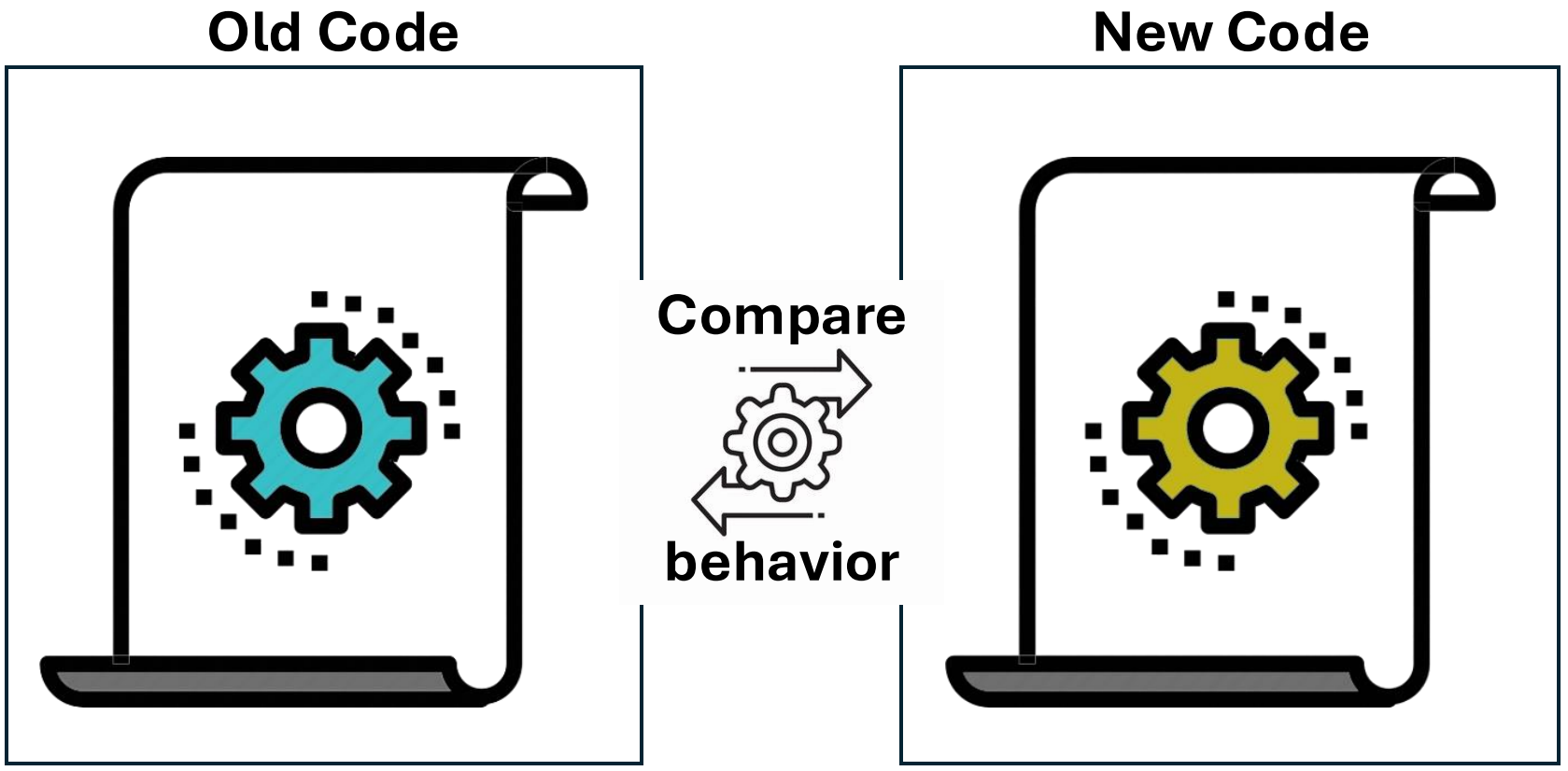
```
@@ -63,7 +63,10 @@ def process_exception(self, request, exception, spider):
63  63      def _retry(self, request, reason, spider):
64  64          retries = request.meta.get('retry_times', 0) + 1
65  65
66      -       retry_times = request.meta.get('max_retry_times') or self.max_retry_times
        66  +   retry_times = self.max_retry_times
        67  +
        68  +   if 'max_retry_times' in request.meta:
        69  +       retry_times = request.meta['max_retry_times']
67  70
68  71          stats = spider.crawler.stats
69  72          if retries <= retry_times:
```

https://github.com/scrapy/scrapy/commit/49c5afc5

How to Find Semantics-Breaking Changes?



How to Find Semantics-Breaking Changes?



FSE'23

LExecutor: Learning-Guided Execution

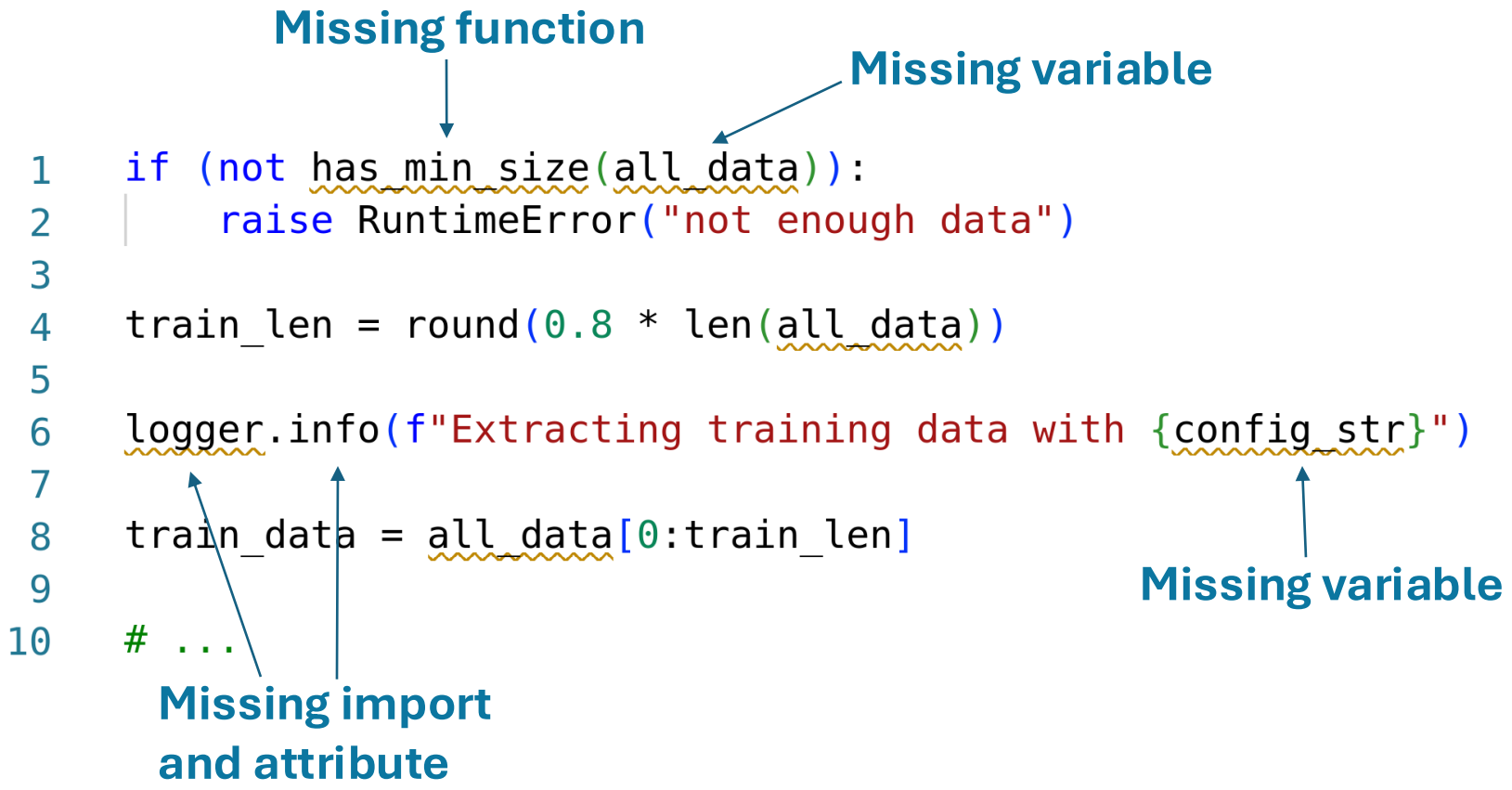
Beatriz Souza
University of Stuttgart
Stuttgart, Germany
beatrizbzsouza@gmail.com

Michael Pradel
University of Stuttgart
Stuttgart, Germany
michael@binaervarianz.de

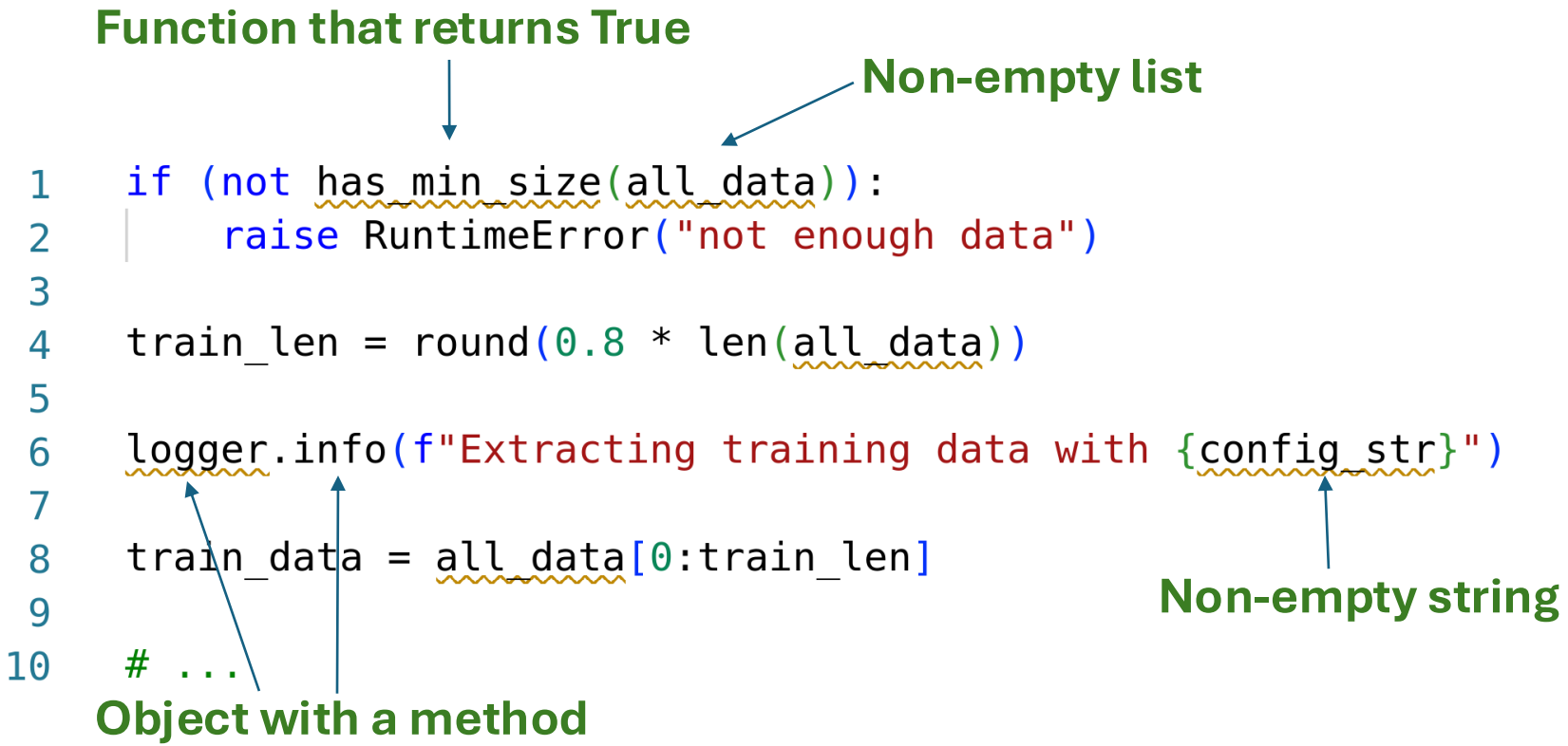
Learning-guided approach for **executing arbitrary code snippets**

- Predict missing values with neural model
- Inject values into the execution

Example of Incomplete Code



LExecuting the Incomplete Code

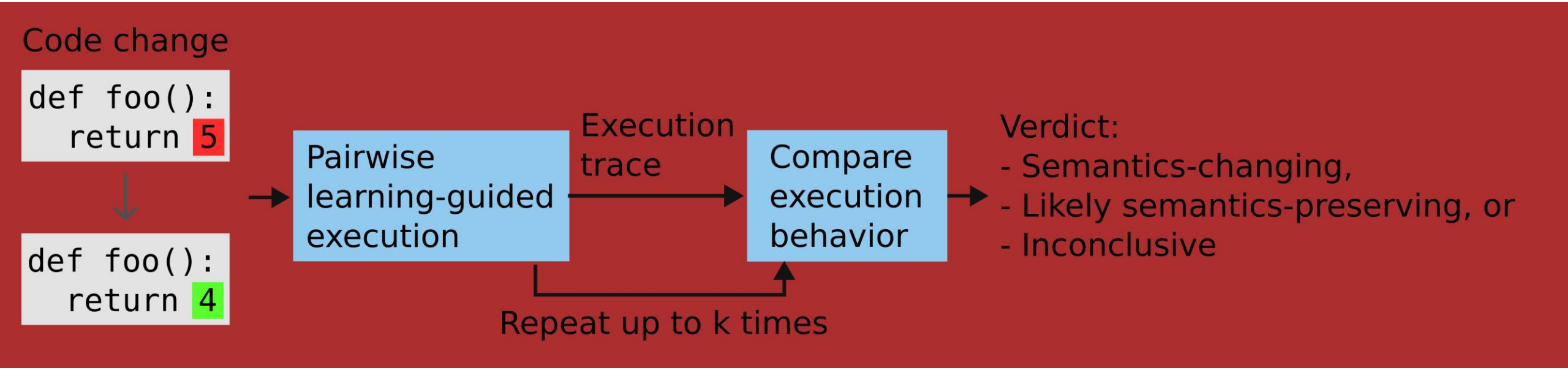


ChangeGuard

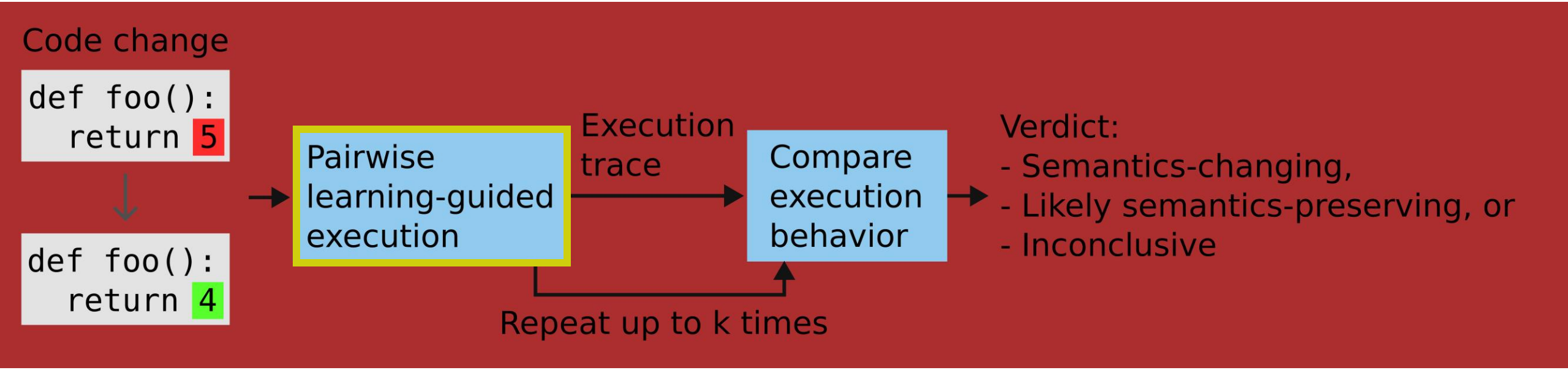
Pairwise Learning-guided approach for **validating code changes**

- Key idea is to execute two versions of a code snippet side-by-side, while predicting and injecting any missing values into the execution.

Overview of **ChangeGuard**



Overview of **ChangeGuard**



Pairwise Learning-Guided Execution

- Merge `f_old` and `f_new` into *Comparison Program*

```
def f_old():
  # old code

def f_new():
  # new code

f_old()
f_new()

# Code to compare behavior
```

Pairwise Learning-Guided Execution

- Predict diverse and project-specific values (!= LExecutor)

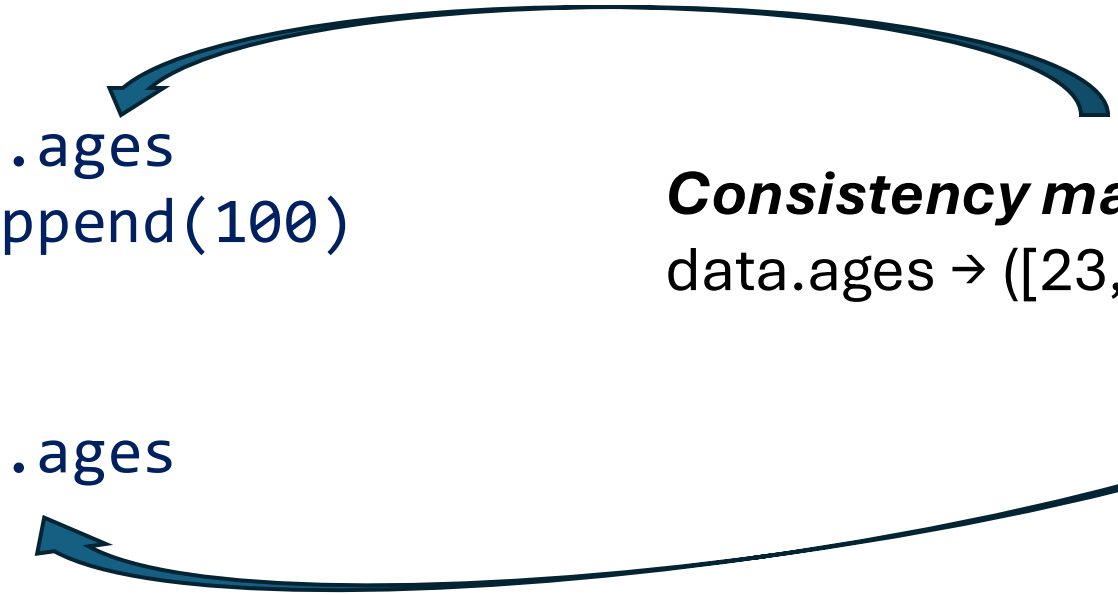
Abstract value	Concrete value(s)
None	`None`
Boolean	`True`, `False`
Integer	`-100`, `-10`, `-1`, `0`, `1`, `10`, `100`, and all integer literals in the code
Float	`-100.0`, `-10.0`, `-1.0`, `0.0`, `1.0`, `10.0`, `100.0`, and all float literals in the code
String	`""`, `"a"`, and all string literals in the code
List	List with random size and elements of a randomly selected type
Tuple	Tuple with random size and elements of a randomly selected type
Dictionary	Dictionary with random size that maps strings to values of a random type
Set	Set with random size and elements of a randomly selected type
Callable	Class of a versatile object with various default methods
Resource	Versatile object with various default methods
Object	Versatile object with various default methods

Pairwise Learning-Guided Execution

- Ensure consistency and non-interference

```
def f_old():
  ages = data.ages
  data.ages.append(100)
  …
def f_new():
  ages = data.ages
  …
f_old()
f_new()

# Code to compare behavior
```



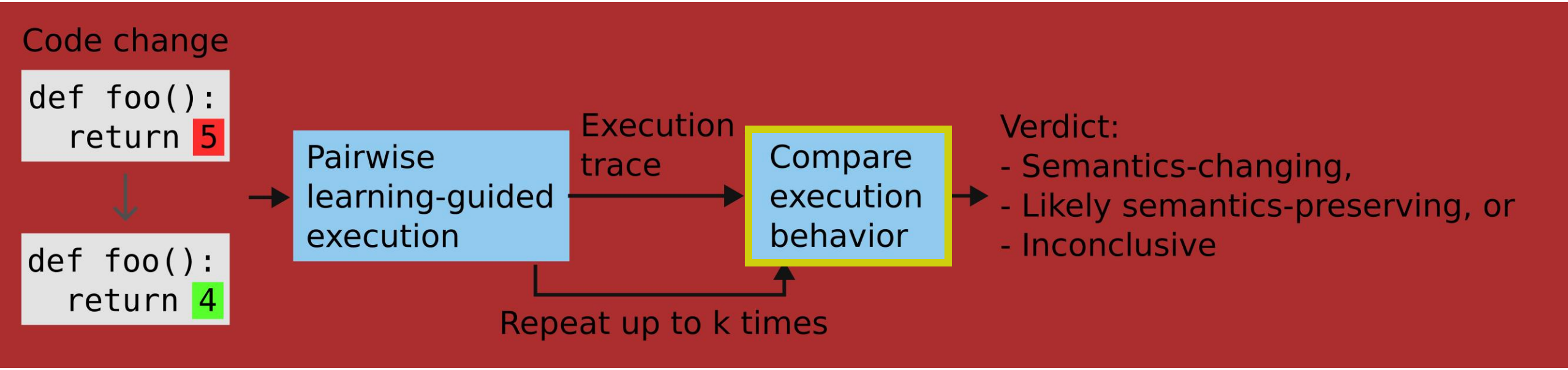
Consistency map with deep-copies:
data.ages → ([23, 42], deepcopy([23, 42]))

Pairwise Learning-Guided Execution

More details in the paper:

- *Handling Calls to External Functions*
- *Handling Indexing Operations*

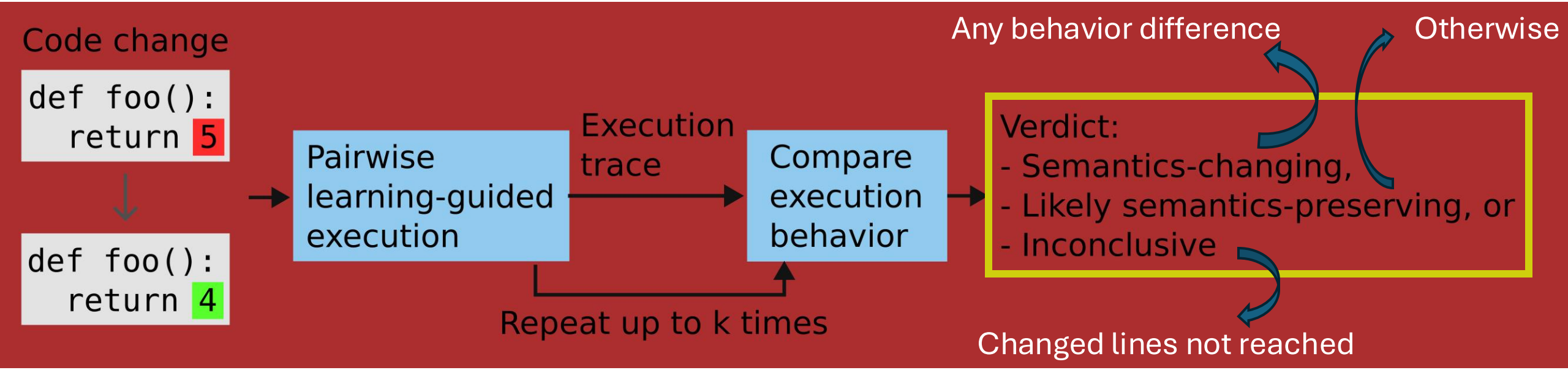
Overview of **ChangeGuard**



Compare Execution Behavior

- Repeat pairwise learning-guided execution (k = 300)
- After each execution of `f_old` and `f_new` **compare**:
  - Argument and return values
  - Output written to console
  - Called functions
  - Raised exceptions

Overview of **ChangeGuard**

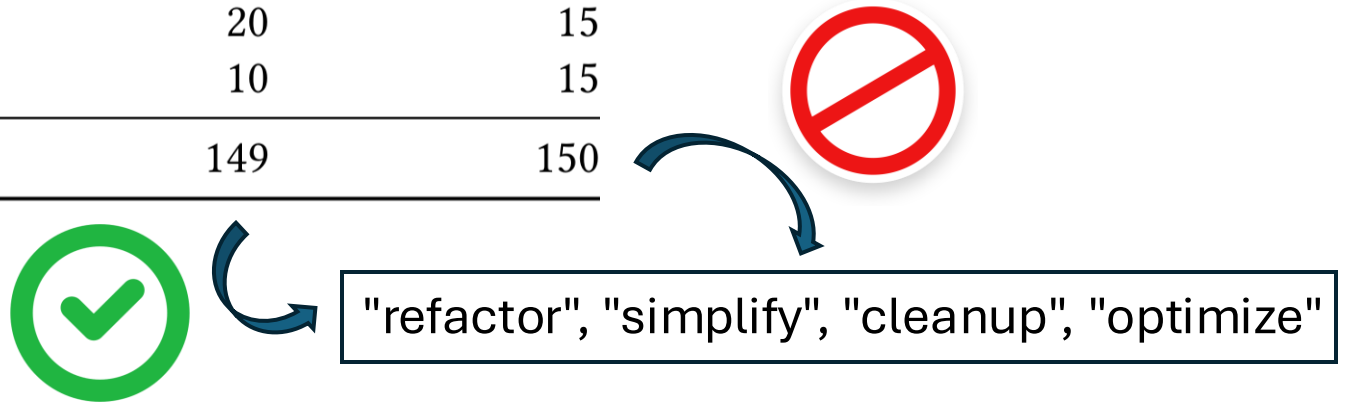


Evaluation

- **RQ1: Effectiveness**
- **RQ2: Comparison with Regression Testing**
- RQ3: Accuracy of the Neural Model
- RQ4: Robustness and Coverage
- **RQ5: Efficiency**

Evaluation

Project	By commit message	
	Sem.-preserving	Sem.-changing
Airflow	31	15
Black	3	15
FastAPI	9	15
Flask	3	15
HTTPie	6	15
Pandas	27	15
Poetry	3	15
Scikit-Learn	37	15
Scrapy	20	15
TheAlgorithms	10	15
Total	149	150



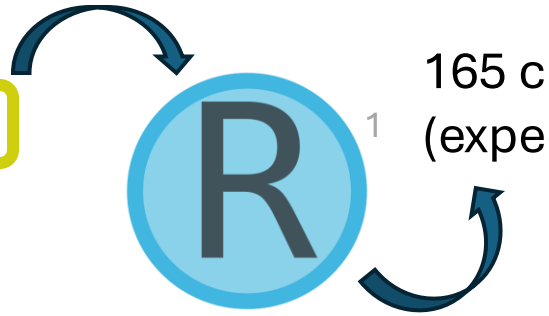
Evaluation

Project	Code changes				
	By commit message		Manually annotated		
	Sem.-preserving	Sem.-changing	Sem.-preserving	Sem.-changing	Unclear
Airflow	31	15	15	25	6
Black	3	15	4	13	1
FastAPI	9	15	8	12	4
Flask	3	15	5	10	3
HTTPie	6	15	7	14	0
Pandas	27	15	5	13	24
Poetry	3	15	2	11	5
Scikit-Learn	37	15	23	16	13
Scrapy	20	15	15	12	8
TheAlgorithms	10	15	9	5	11
Total	149	150	93	131	75

224 single function changes

Evaluation

Project	By commit message	
	Sem.-preserving	Sem.-changing
Airflow	31	15
Black	3	15
FastAPI	9	15
Flask	3	15
HTTPie	6	15
Pandas	27	15
Poetry	3	15
Scikit-Learn	37	15
Scrapy	20	15
TheAlgorithms	10	15
Total	149	150

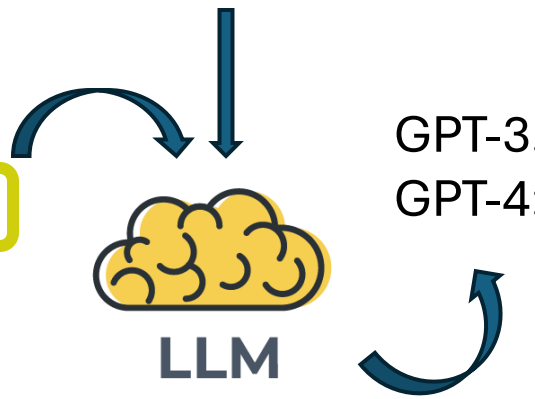


165 changes
(expected to be semantics-preserving)

1. RIdiom: Making Python Code Idiomatic by Automatic Refactoring Non-Idiomatic Python Code with Pythonic Idioms (FSE'22)

Evaluation

Project	By commit message	
	Sem.-preserving	Sem.-changing
Airflow	31	15
Black	3	15
FastAPI	9	15
Flask	3	15
HTTPie	6	15
Pandas	27	15
Poetry	3	15
Scikit-Learn	37	15
Scrapy	20	15
TheAlgorithms	10	15
Total	149	150



GPT-3.5: 187 changes
GPT-4: 258 changes

RQ1: Effectiveness on Manually Annotated Changes

Ground truth		ChangeGuard		
	Total	Changing	Preserving	Inconclusive
Changing	131	91	12	28
Preserving	93	27	48	18
Total	224	118	60	46

Recall: 69.5%

Precision: 77.1%

Accuracy: 78.1%

RQ1: Effectiveness on Manually Annotated Changes

Ground truth		ChangeGuard		
	Total	Changing	Preserving	Inconclusive
Changing	131	91	12	28
Preserving	93	27	48	18
Total	224	118	60	46

Recall: 69.5%

Precision: 77.1%

Accuracy: 78.1%

-> ChangeGuard's predictions are correct for the majority of the code changes!

RQ1: Effectiveness on Changes by  and 

Refactoring tool	Prediction		
	Inconclusive	Changing	Preserving
RIdiom	38	5	122
GPT-3.5	31	87	69
GPT-4	38	143	77

RQ1: Effectiveness on Changes by  and 

Refactoring tool	Prediction		
	Inconclusive	Changing	Preserving
RIdiom	38	5	122
GPT-3.5	31	87	69
GPT-4	38	143	77

Bug in RIdiom

RQ1: Effectiveness on Changes by R and LLM

Refactoring tool	Prediction		
	Inconclusive	Changing	Preserving
RIdiom	38	5	122
GPT-3.5	31	87	69
GPT-4	38	143	77

Bug in RIdiom

30 random cases
(per model)

GPT-3.5: 21 changing
GPT-4: 16 changing

RQ1: Effectiveness on Changes by R and LLM

Refactoring tool	Prediction		
	Inconclusive	Changing	Preserving
RIdiom	38	5	122
GPT-3.5	31	87	69
GPT-4	38	143	77

Bug in RIdiom

30 random cases
(per model)

GPT-3.5: 21 changing
GPT-4: 16 changing

-> LLMs often fail to improve the code while preserving its semantics;

-> ChangeGuard is effective to identify such semantics-breaking code changes!

Example

Code Refactored by GPT-4

```
    def param_allowed(stat_name, include, exclude):
      if not include and not exclude: return True
-     for p in exclude:
-         if p in stat_name: return False
-     if exclude and not include: return True
-     for p in include:
-         if p in stat_name: return True
-     return False
+     if any(p in stat_name for p in exclude): return False
+     if include: return any(p in stat_name for p in include)
+     return not exclude
```

Example

Code Refactored by GPT-4

Behavior changes for args = [], [], [...]

```
    def param_allowed(stat_name, include, exclude):
      if not include and not exclude: return True
-     for p in exclude:
-         if p in stat_name: return False
-     if exclude and not include: return True
-     for p in include:
-         if p in stat_name: return True
-     return False
+     if any(p in stat_name for p in exclude): return False
+     if include: return any(p in stat_name for p in include)
+     return not exclude
```

RQ2: Comparison with Existing Regression Tests

Ground truth		ChangeGuard			Regression testing		
	Total	Changing	Preserving	Inconclusive	Changing	Preserving	Inconclusive
Changing	131	91	12	28	10	29	92
Preserving	93	27	48	18	2	18	73
Total	224	118	60	46	12	47	165

Recall: 7.6%

Precision: 83.3%

Accuracy: 47.5%

No test results

RQ2: Comparison with Existing Regression Tests

Ground truth		ChangeGuard			Regression testing		
	Total	Changing	Preserving	Inconclusive	Changing	Preserving	Inconclusive
Changing	131	91	12	28	10	29	92
Preserving	93	27	48	18	2	18	73
Total	224	118	60	46	12	47	165

Recall: 7.6%

Precision: 83.3%

Accuracy: 47.5%

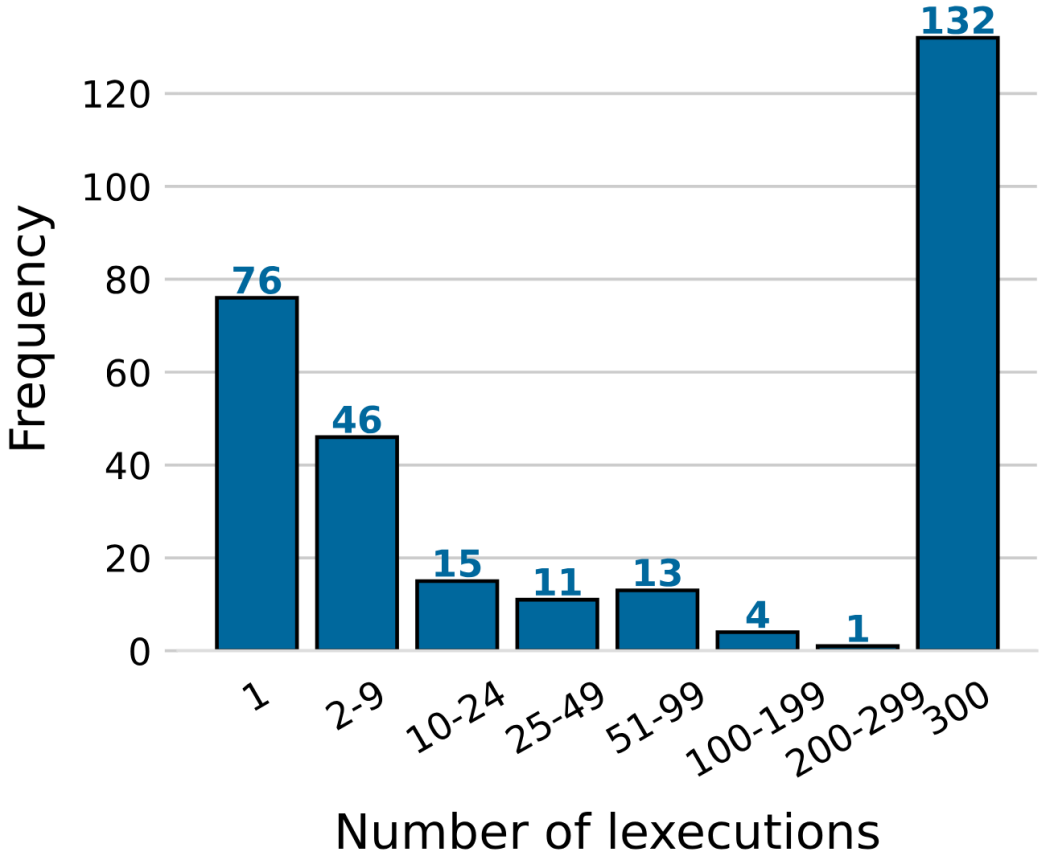
No test results

-> ChangeGuard is more effective than the existing regression tests!

RQ5: Efficiency

- Instrumentation: 1.15 seconds
- First execution: 1.67 seconds
- Extra executions: 1.01 seconds

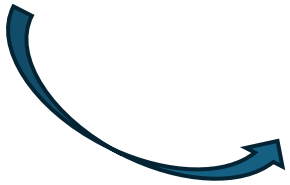
Executions to reach a conclusion



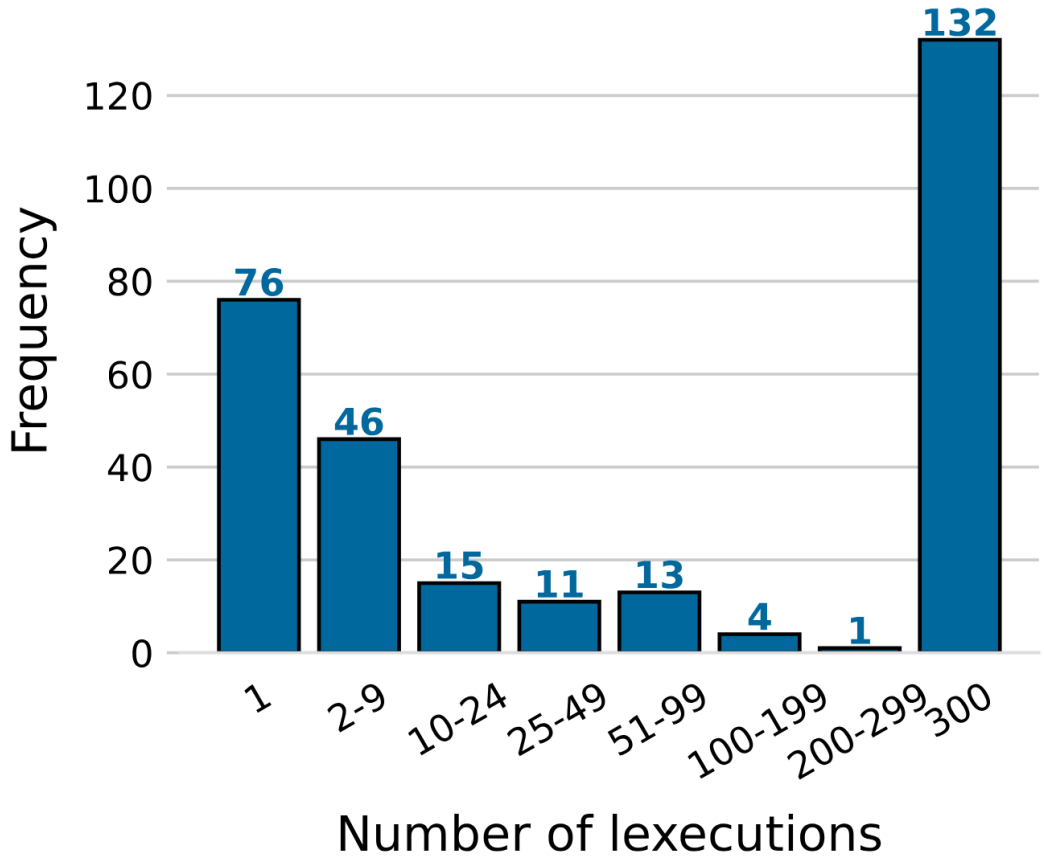
RQ5: Efficiency

- Instrumentation: 1.15 seconds
- First execution: 1.67 seconds
- Extra executions: 1.01 seconds

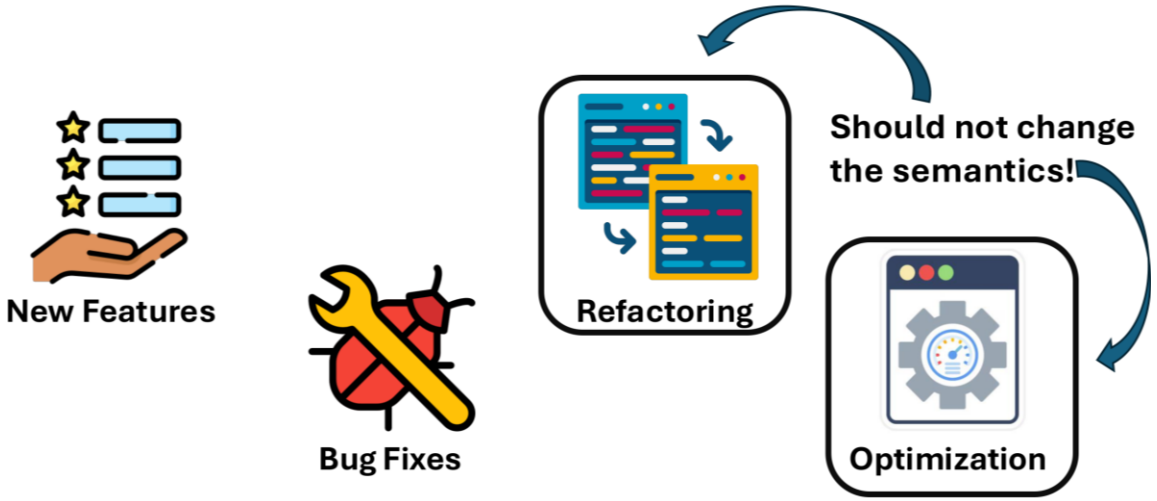
Executions to reach a conclusion



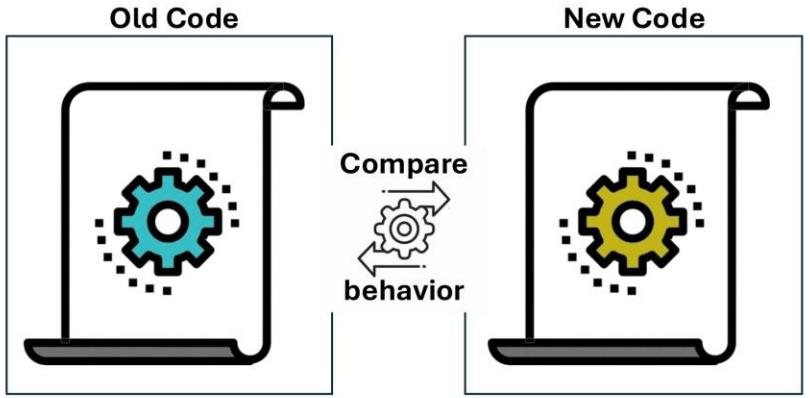
-> Reducing the number of executions, ChangeGuard becomes more efficient for a small reduction in recall!



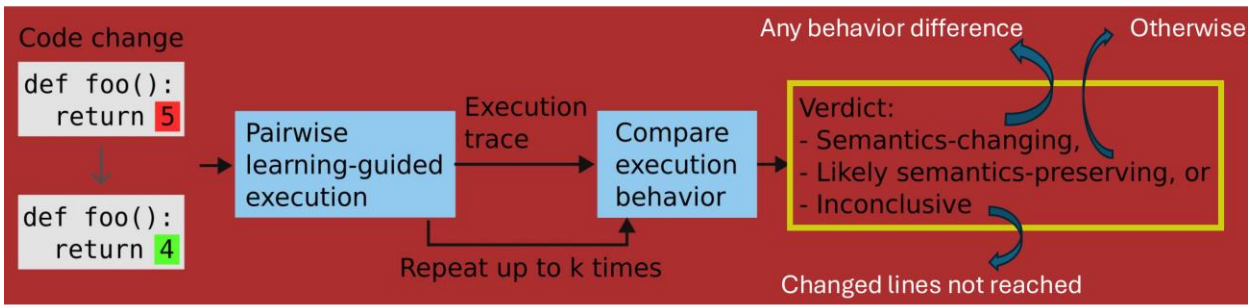
Software is Continuously Evolving



How to Find Semantics-Breaking Changes?



Overview of **ChangeGuard**



RQ1: Effectiveness on Changes by R and LLM

Refactoring tool	Prediction		
	Inconclusive	Changing	Preserving
RIdiom	38	5	122
GPT-3.5	31	87	69
GPT-4	38	143	77

Bug in RIdiom

-> LLMs often fail to improve the code while preserving its semantics;

-> ChangeGuard is effective to identify such semantics-breaking code changes!

30 random cases (per model)

GPT-3.5: 21 changing
GPT-4: 16 changing

Example

Code Refactored by RIdiom

```
- provider = Provider(self._pkg, self._pool, self._io)
- locked = {}
- for package in self._locked.packages: locked[package.name] = package
+ provider, locked = Provider(self._pkg, self._pool, self._io), {}
```

Example

Code Refactored by RIdiom

```
- provider = Provider(self._pkg, self._pool, self._io)
- locked = {}
- for package in self._locked.packages: locked[package.name] = package
+ provider, locked = Provider(self._pkg, self._pool, self._io), {}
```

Line removed!

Example

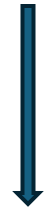
Code Refactored by GPT-3.5

```
- if isinstance(arr_or_dtype, ExtensionType): return arr_or_dtype.name == "category"
- if arr_or_dtype is None: return False
- return CategoricalDtype.is_dtype(arr_or_dtype)
+ if isinstance(arr_or_dtype, ExtensionType) and arr_or_dtype.name == "category": return True
+ elif arr_or_dtype is None: return False
+ else: return CategoricalDtype.is_dtype(arr_or_dtype)
```

Example

Code Refactored by GPT-3.5

```
- if isinstance(arr_or_dtype, ExtensionType): return arr_or_dtype.name == "category"
- if arr_or_dtype is None: return False
- return CategoricalDtype.is_dtype(arr_or_dtype)
+ if isinstance(arr_or_dtype, ExtensionType) and arr_or_dtype.name == "category": return True
+ elif arr_or_dtype is None: return False
+ else: return CategoricalDtype.is_dtype(arr_or_dtype)
```



Expression moved to condition!

RQ3: Accuracy of the Neural Model

Fine-tuned CodeT5 with the following differences from LExecutor:

- Predicts values for indexing operations;
- Larger and diverse training dataset.

	Accuracy
Top-1	95.13%
Top-3	96.17%
Top-5	97.52%

LExecutor Accuracy (coarse-grained): 88.1%

RQ4: Robustness and Coverage