Neural Software Analysis:

Recent Advances on Types, Bugs, and Executions

Michael Pradel (University of Stuttgart)

Joint work with Beatriz Souza, Islem Bouzenia, Luca Di Grazia, and Wai Chow

SOLA SoftwareLab

University of Stuttgart Germany

erc European Research Council

Program Analysis

How to Create a Program Analysis?

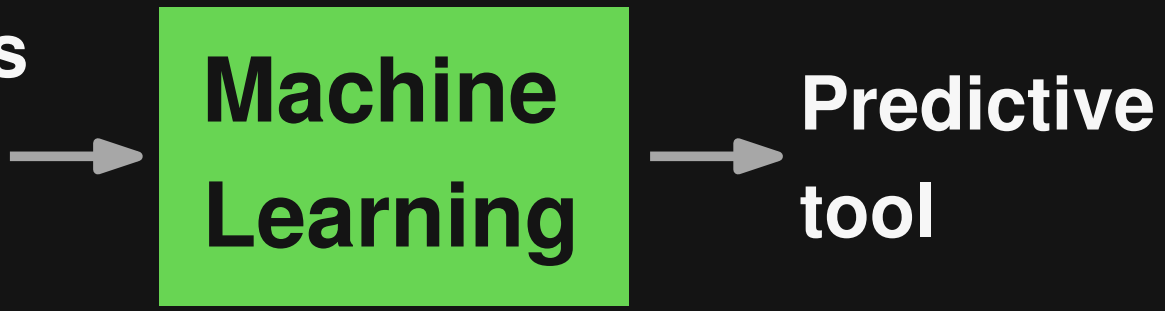
Traditional program analysis

- Manually crafted
- Years of work
- Precise, logical reasoning
- Heuristics to handle undecidability
- Challenged by large code bases

Neural Software Analysis

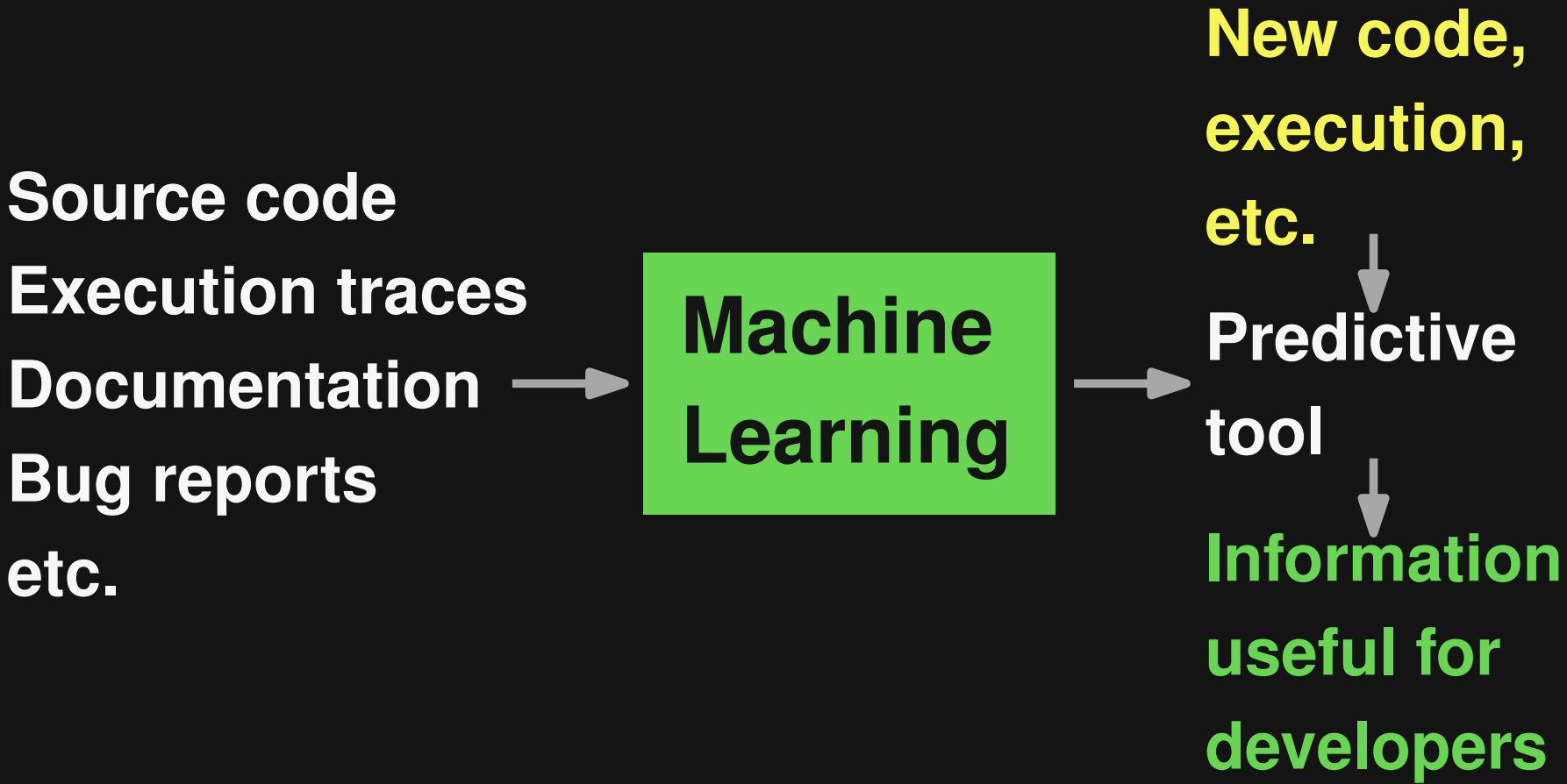
Insight: Lots of data about software development to learn from

Source code
Execution traces
Documentation
Bug reports
etc.



Neural Software Analysis

Insight: Lots of data about software development to learn from



Traditional program analysis

- Manually crafted
- Years of work
- Precise, logical reasoning
- Heuristics to handle undecidability
- Challenged by large code bases

Traditional program analysis

- Manually crafted
- Years of work
- Precise, logical reasoning
- Heuristics to handle undecidability
- Challenged by large code bases

Neural software analysis

- Automatically learned within hours or days
- Data-driven prediction
- Learn instead of hard-code heuristics
- Use big code to our benefit

This Talk

- **Fixing type errors with PyTy**

  PyTy: Repairing Static Type Errors in Python, FSE'23 (major rev.)

- **Neural bug detection with CMI-Finder**

  When to Say What: Learning to Find Condition-Message Inconsistencies, ICSE'23

- **Enabling execution with LExecutor**

  LExecutor: Learning-Guided Execution, FSE'23 (major rev.)

Types in Python

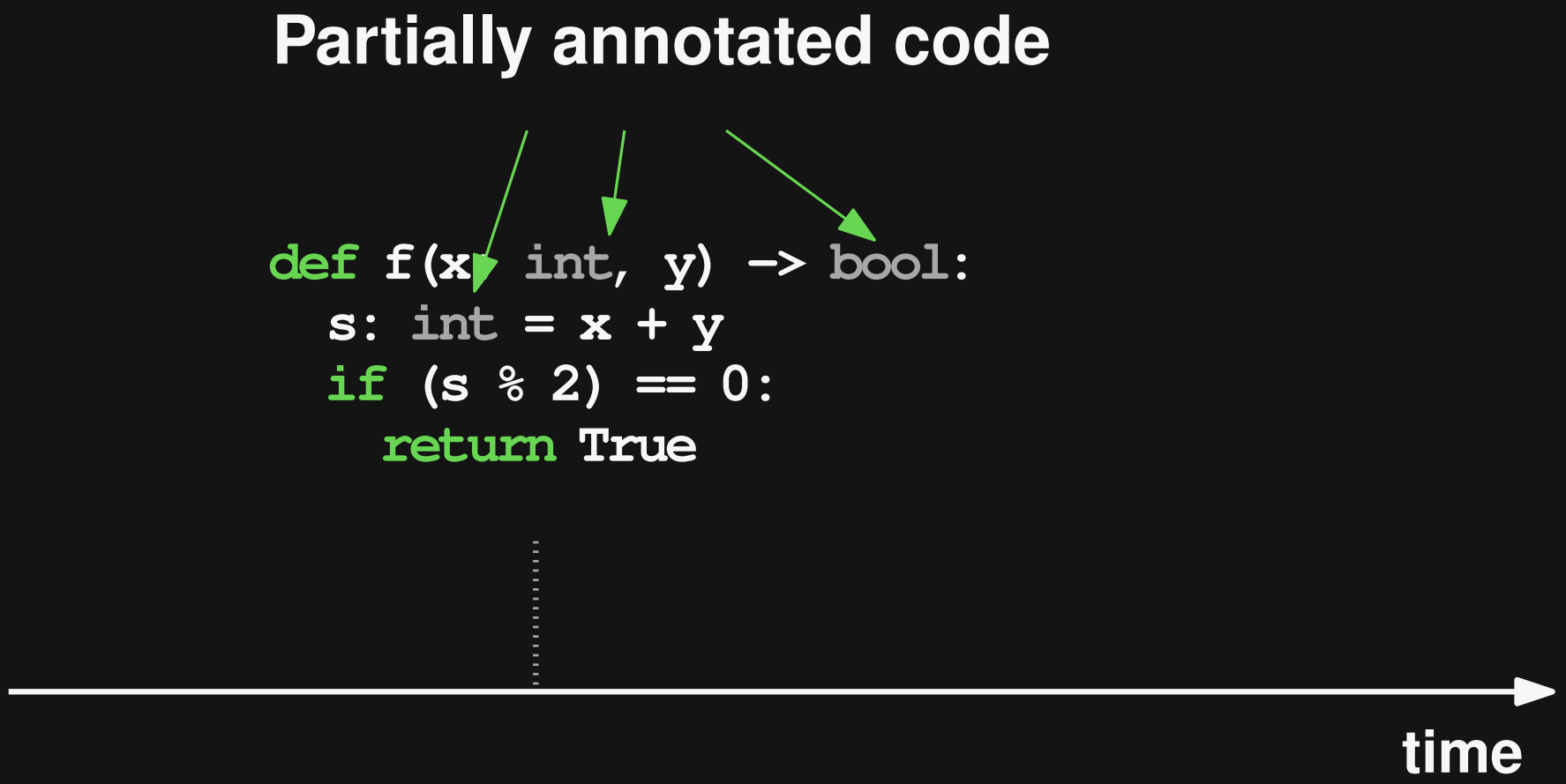
Typical evolution of a Python project:

Code without
type annotations

```
def f(x, y):
  s = x + y
  if (s % 2) == 0:
    return True
```

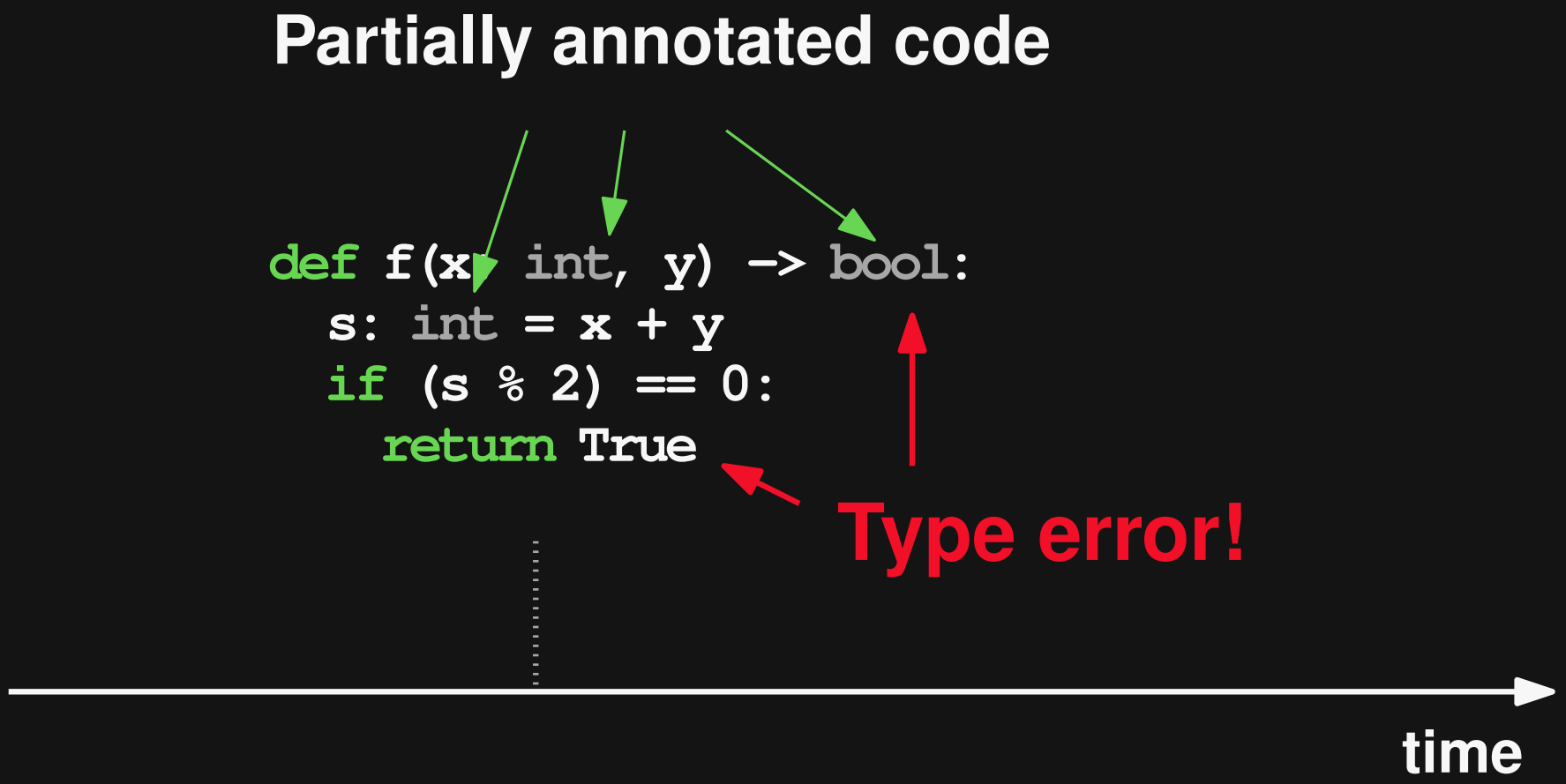
Types in Python

Typical evolution of a Python project:



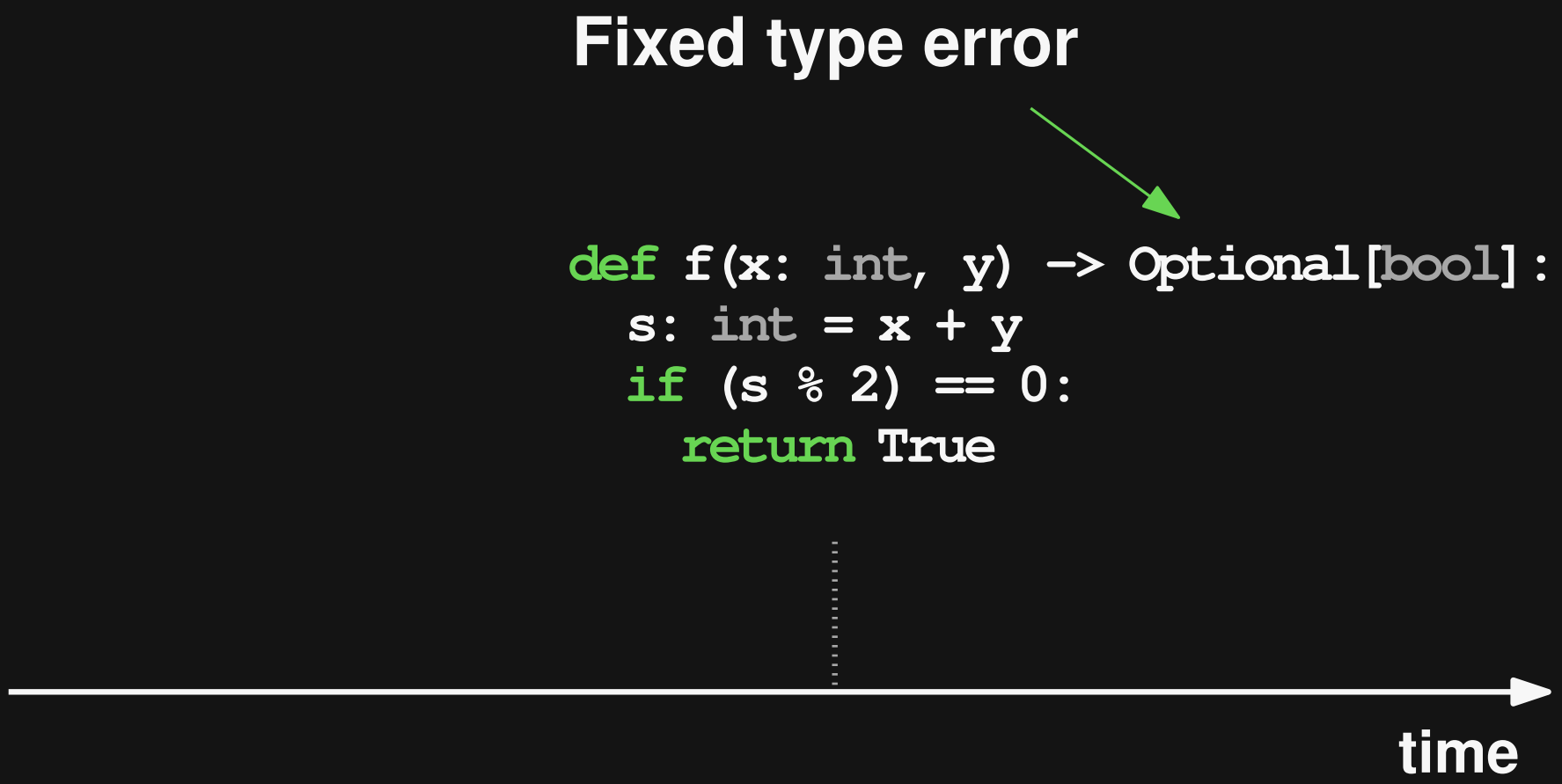
Types in Python

Typical evolution of a Python project:



Types in Python

Typical evolution of a Python project:



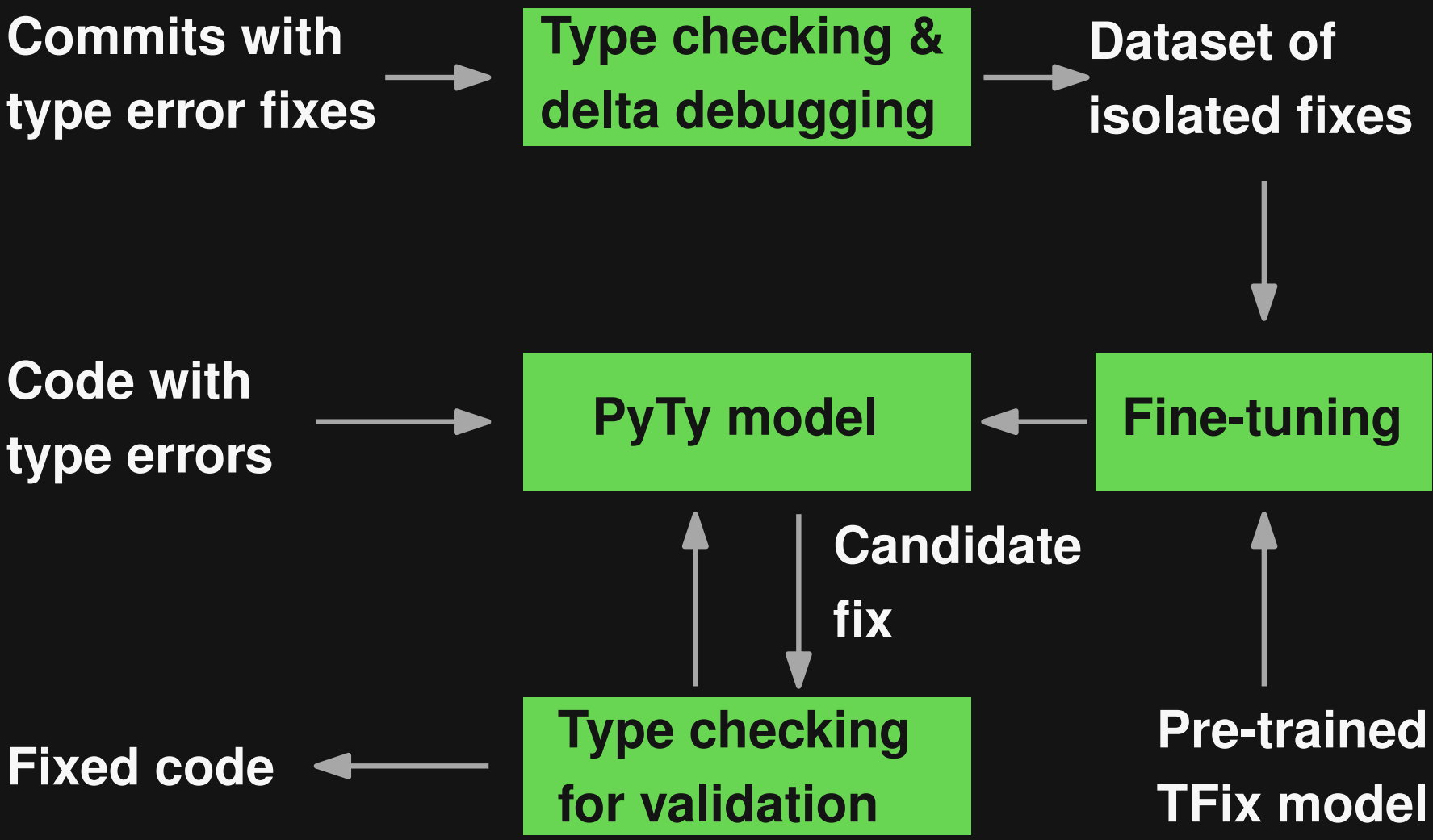
Too Many Type Errors

- Most existing Python code bases: Plenty of static type errors
- Easy to detect by gradual type checker
- But: No time to fix them all

Preliminary Study

- **Many recurring fix patterns**
  - But: No unambiguous repair rules
- **Most fixes are local, e.g., single-line**
- **Type checker helps localize fix location**

PyTy: Approach



Data Gathering

1) Keyword-based search for commits

2) Type check old and new code

32k type errors removed in 4.5k commits

3) Isolate fixes of exactly one type error

2.8k isolated type error fixes

Data Gathering: Example

Error: Unbound name `basestring`

Old code:

```
# Hunk H1
class CacheKey(basestring):

# Hunk H2
  pass



# Hunk H3
  if isinstance(key, CacheKey):
    key = CacheKey(smart_str(key))

# Hunk H4
  if timeout == 0:
```

New code:

```
# Hunk H1
class CacheKey(object):

# Hunk H2
  def __init__(self, key):
    self._key = key
    ...

# Hunk H3
  if not isinstance(key, CacheKey):
    key = CacheKey(key)

# Hunk H4
  if timeout is None:
    ...
```

Data Gathering: Example

Error: Unbound name `basestring`

Old code:

```
# Hunk H1
class CacheKey(basestring):

# Hunk H2
  pass



# Hunk H3
  if isinstance(key, CacheKey):
    key = CacheKey(smart_str(key))

# Hunk H4
  if timeout == 0:
```

New code:

```
# Hunk H1
class CacheKey(object):

# Hunk H2
  def __init__(self, key):
    self._key = key
    ...

# Hunk H3
  if not isinstance(key, CacheKey):
    key = CacheKey(key)

# Hunk H4
  if timeout is None:
    ...
```

Data Gathering: Example

Error: Unbound name `basestring`

Old code:

```
# Hunk H1
class CacheKey(basestring):

# Hunk H2
  pass

# Hunk H3
  if isinstance(key, CacheKey):
    key = CacheKey(smart_str(key))

# Hunk H4
  if timeout == 0:
```

New code:

```
# Hunk H1
class CacheKey(object):

# Hunk H2
  def __init__(self, key):
    self._key = key
    ...

# Hunk H3
  if not isinstance(key, CacheKey):
    key = CacheKey(key)

# Hunk H4
  if timeout is None:
    ...
```

Data Gathering: Example

Error: Unbound name `basestring`

Old code:

```
# Hunk H1
class CacheKey(basestring):

# Hunk H2
  pass
```

New code:

```
# Hunk H1
class CacheKey(object):

# Hunk H2
  def __init__(self, key):
    self._key = key
    ...
```

```
# Hunk H3
  if isinstance(key, CacheKey):
    key = CacheKey(smart_str(key))

# Hunk H4
  if timeout == 0:
```

```
# Hunk H3
  if not isinstance(key, CacheKey):
    key = CacheKey(key)

# Hunk H4
  if timeout is None:
    ...
```

Data Gathering: Example

Error: Unbound name `basestring`

Old code:

```
# Hunk H1
class CacheKey(basestring):

# Hunk H2
  pass

# Hunk H3
  if isinstance(key, CacheKey):
    key = CacheKey(smart_str(key))

# Hunk H4
  if timeout == 0:
```

New code:

```
# Hunk H1
class CacheKey(object):

# Hunk H2
  def __init__(self, key):
    self._key = key
    ...

# Hunk H3
  if not isinstance(key, CacheKey):
    key = CacheKey(key)

# Hunk H4
  if timeout is None:
    ...
```

Data Gathering: Example

Error: Unbound name `basestring`

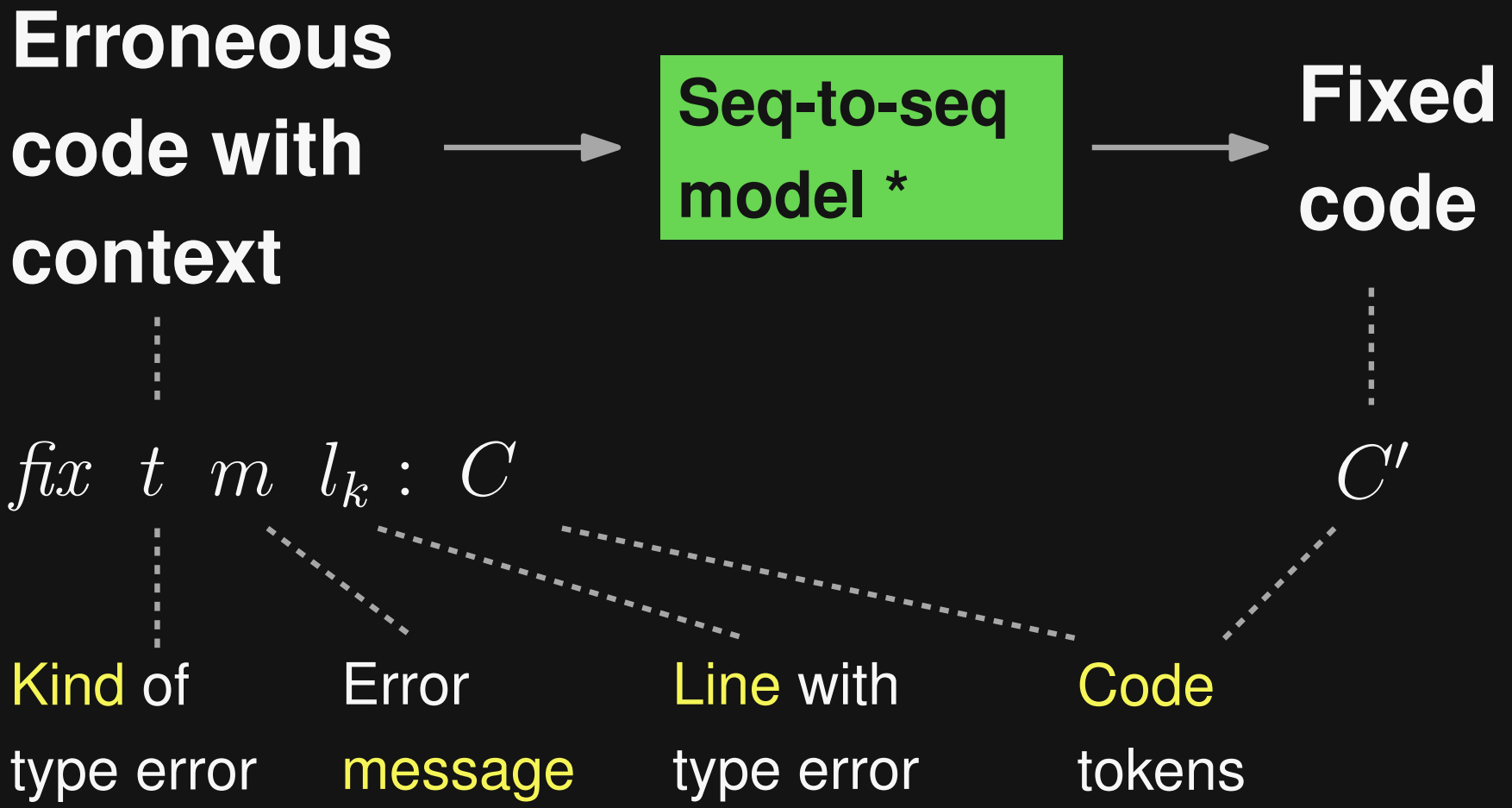
Old code:

```
# Hunk H1
class CacheKey(basestring):

# Hunk H2
  pass


# Hunk H3
  if isinstance(key, CacheKey):
    key = CacheKey(smart_str(key))

# Hunk H4
  if timeout == 0:
```

New code:

```
# Hunk H1
class CacheKey(object):

# Hunk H2
  def __init__(self, key):
    self._key = key
    ...

# Hunk H3
  if not isinstance(key, CacheKey):
    key = CacheKey(key)

# Hunk H4
  if timeout is None:
    ...
```

Model



*** Fine-tuned model, based on pre-trained TFix** [Berabi'21]**, which is based on pre-trained T5** [Raffel'20]

PyTy: Effectiveness

Classes of type errors	Samples (test set)		Effectiveness of PyTy: Error removal	Effectiveness of PyTy: Exact match
Incompatible variable type	821	(83)	90.4%	65.1%
Incompatible parameter type	600	(60)	80.0%	36.7%
Incompatible return type	296	(30)	73.3%	43.3%
Invalid type	291	(30)	100.0%	83.3%
Unbound name	258	(26)	76.9%	42.3%
Incompatible attribute type	258	(26)	92.3%	73.1%
Unsupported operand	124	(13)	76.9%	38.5%
Strengthened precondition	59	(6)	83.3%	50.0%
Weakened postcondition	51	(6)	50.0%	0.0%
Call error	8	(1)	100.0%	100.0%
Total	2,766	(281)	85.4%	54.4%

PyTy: Repairing Static Type Errors in Python, FSE'23 (major rev.)

Examples

Code with type error:

```
vprint(f"{prefix} {lineno}: {action_name}
  Constrain Mouse: {'yes' if constraint > 0
  else ('no' if constrained == 0 else 'check stack')}")
```

Unbound name

PyTy finds exactly the developer fix:

```
vprint(f"{prefix} {lineno}: {action_name}
  Constrain Mouse: {'yes' if constraint > 0
  else ('no' if constraint == 0 else 'check stack')}")
```

Examples

Code with type error:

Declared to have type `str`
but used as `bytes`

```
string = _fmt(string)
return lib.TCOD_console_get_height_rect_fmt(
  self.console_c, x, y, width, height, string
)
```

PyTy finds a valid fix:

```
byte_string = _fmt(string)
return lib.TCOD_console_get_height_rect_fmt(
  self.console_c, x, y, width, height, byte_string
)
```

Developer fix (semantically equivalent):

```
return lib.TCOD_console_get_height_rect_fmt(
  self.console_c, x, y, width, height, _fmt(string)
)
```

This Talk

- **Fixing type errors with PyTy**

  PyTy: Repairing Static Type Errors in Python, FSE'23 (major rev.)

- **Neural bug detection with CMI-Finder**

  When to Say What: Learning to Find Condition-Message Inconsistencies, ICSE'23

- **Enabling execution with LExecutor**

  LExecutor: Learning-Guided Execution, FSE'23 (major rev.)

Motivation

Example 1:

```
if len(bits) != 4 or len(bits) != 6:
  raise template.TemplateSyntaxError("%r takes
    exactly four or six arguments (second argument
    must be 'as')" % str(bits[0]))
```

Motivation

Example 1:

Always `True`

Doesn't match the message

```
if len(bits) != 4 or len(bits) != 6:
  raise template.TemplateSyntaxError("%r takes
    exactly four or six arguments (second argument
    must be 'as')" % str(bits[0]))
```

Motivation

Example 1:

Always `True`

Doesn't match the message

```
if len(bits) != 4 or len(bits) != 6:
  raise template.TemplateSyntaxError("%r takes
    exactly four or six arguments (second argument
    must be 'as')" % str(bits[0]))
```

Example 2:

```
if n2 > n1 :
  raise ValueError('Total internal reflection
    impossible for n1 > n2')
```

Motivation

Example 1:

Always `True`

Doesn't match the message

```
if len(bits) != 4 or len(bits) != 6:
  raise template.TemplateSyntaxError("%r takes
    exactly four or six arguments (second argument
    must be 'as')" % str(bits[0]))
```

Example 2:

Condition and message are inconsistent

```
if n2 > n1 :
  raise ValueError('Total internal reflection
    impossible for n1 > n2')
```

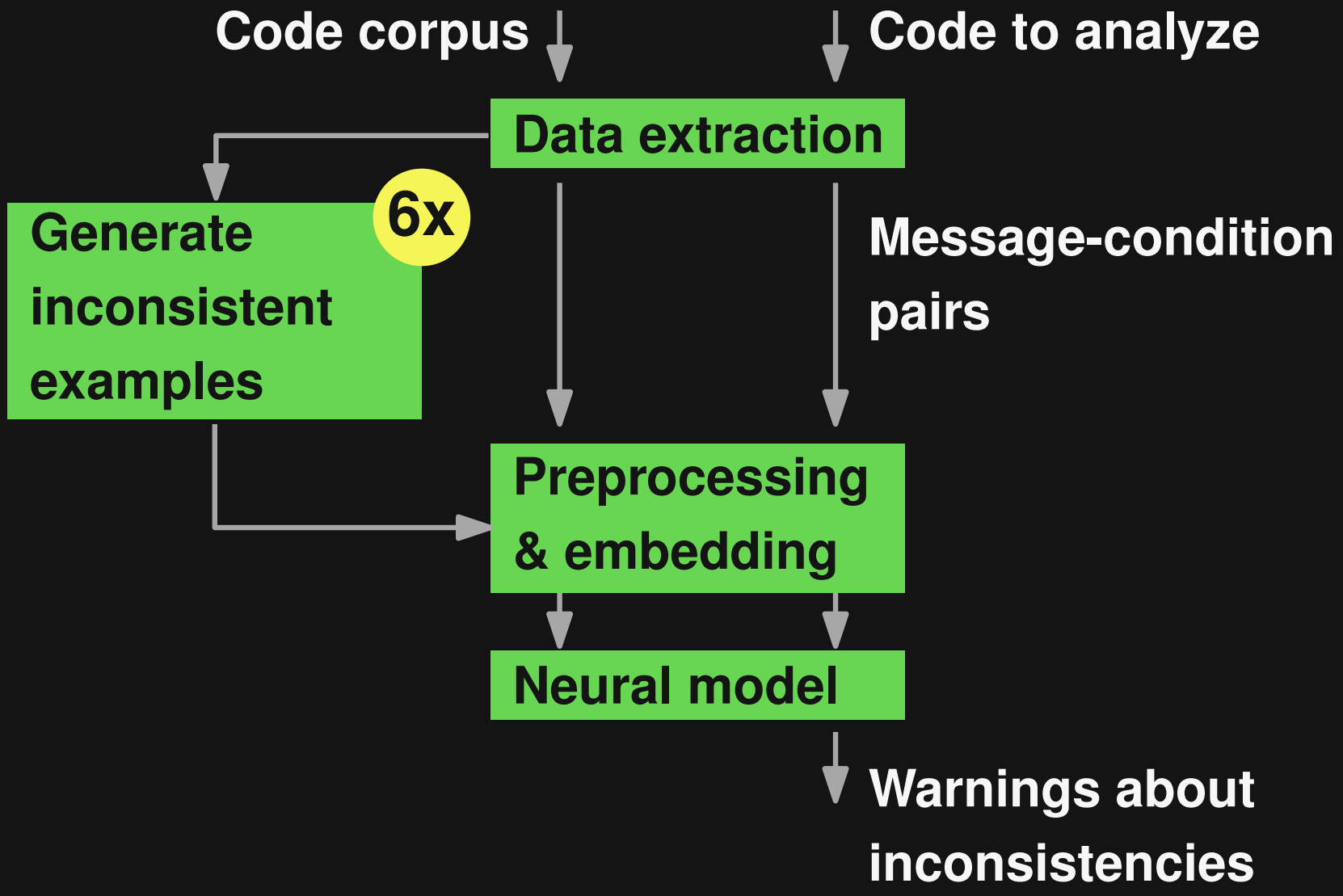
CMI-Finder

Goal:

Detect condition-message inconsistencies

- Why?
  - Incorrect conditions may raise unnecessary warnings or suppress expected warnings
  - Incorrect messages make debugging unnecessarily hard
- Hard problem!
  - Must understand both NL and PL

Overview of CMI-Finder



Generating Inconsistent Examples

Six generation strategies

- Mutation of operators
- Mutation of error messages
- Random re-combination
- Pattern-based mutation
- Embedding-based token replacement
- Language model-based generation of error message

Generating Inconsistent Examples

Six generation strategies

- Mutation of operators
- Mutation of error messages
- Random re-combination
- Pattern-based mutation
- Embedding-based token
- Language model-based g
  message

Example:

```
if result.status in (0, 3):
  log.warning("Invalid status")
```

⇓

```
if result.status in (0, 3):
  log.warning("Valid status")
```

Generating Inconsistent Examples

Six generation strategies

- Mutation of operators
- Mutation of error mess
- Random re-combinatio
- Pattern-based mutatior
- Embedding-based token replacement
- Language model-based generation of error message

Example:

```
if not isinstance(config, (tuple, list)):
  raise TypeError('Unable to decode
    config: {}'.format(config))
```

⇓

```
if not isinstance(config, (tuple, list)):
  raise ValueError('Unable to decode
    config: {}'.format(config))
```

Generating Inconsistent Examples

Six generation strategies

- Mutation of operators
- Mutation of error mes
- Random re-combinati
- Pattern-based mutatic
- Embedding-based token replacement
- Language model-based generation of error message

Example:

```
if x == 0:
  raise ValueError('x must not be zero')
```

⇓

```
if x != 0:
  raise ValueError('x cannot be lower than 0')
```

Train & Predict

Fine-tuned CodeT5 model

Also tried, but less effective:

- Binary classifier
- Contrastive learning

Evaluation

- **Training data**
  - 300k pairs from 40k Python projects
    + 300k inconsistent pairs
- **Real-world test data**
  - 66 pairs from 33 historic fixes of condition-message inconsistencies
  - Seven previously unseen Python projects

Results

- **AUC of 0.91 (synthetic data) and 0.82 (real-world data)**
  - E.g., 0.78 precision and 0.72 recall on historic fixes
- **50 new inconsistencies in held-out projects**
- **Complements flake8 and outperforms a GPT-3 baseline**

This Talk

- **Fixing type errors with PyTy**

  PyTy: Repairing Static Type Errors in Python, FSE'23 (major rev.)

- **Neural bug detection with CMI-Finder**

  When to Say What: Learning to Find Condition-Message Inconsistencies, ICSE'23

- **Enabling execution with LExecutor** ←

  LExecutor: Learning-Guided Execution, FSE'23 (major rev.)

Motivation

Imagine you want to execute this code:

```
if (not has_min_size(all_data)):
  raise RuntimeError("not enough data")

train_len = round(0.8 * len(all_data))
logger.info(f"Extracting data with {config_str}")
train_data = all_data[0:train_len]

# ...
```

Motivation

Imagine you want to execute this code:

Missing variable

```
if (not has_min_size(all_data)):
  raise RuntimeError("not enough data")

train_len = round(0.8 * len(all_data))
logger.info(f"Extracting data with {config_str}")
train_data = all_data[0:train_len]

# ...
```

Motivation

Imagine you want to execute this code:

Missing function

Missing variable

```
if (not has_min_size(all_data)):
  raise RuntimeError("not enough data")

train_len = round(0.8 * len(all_data))
logger.info(f"Extracting data with {config_str}")
train_data = all_data[0:train_len]

# ...
```

Motivation

Imagine you want to execute this code:

Missing function

Missing variable

```
if (not has_min_size(all_data)):
  raise RuntimeError("not enough data")

train_len = round(0.8 * len(all_data))
logger.info(f"Extracting data with {config_str}")
train_data = all_data[0:train_len]

# ...
```

Missing variable

Motivation

Imagine you want to **execute this code**:

Missing function

Missing variable

```
if (not has_min_size(all_data)):
  raise RuntimeError("not enough data")

train_len = round(0.8 * len(all_data))
logger.info(f"Extracting data with {config_str}")
train_data = all_data[0:train_len]

# ...
```

Missing import and attribute

Missing variable

Why Execute Incomplete Code?

Enables various dynamic analyses

- Check for exceptions and assertion violations
- Compare two code snippets for semantic equivalence
- Validate static analysis warnings
- Validate and filter LLM-predicted code
- $\langle Your\ favorite\ application\ here \rangle$

LExecutor: Learning-Guided Execution, FSE'23 (major rev.)

Executing Ain't Easy

Lots of incomplete code:

- Code snippets from Stack Overflow
- Code generated by language models
- Code extracted from deep inside complex projects

Executing Ain't Easy

Lots of incomplete code:

- Code snippets from Stack Overflow
- Code generated by language models
- Code extracted from deep inside complex projects

Can we automatically fill in the missing information?

LExecutor: Learning-Guided Execution, FSE'23 (major rev.)

LExecutor

Learning-guided approach for executing arbitrary code snippets

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

Underconstrained execution:
No guarantee that values are realistic

LExecutor: Learning-Guided Execution, FSE'23 (major rev.)

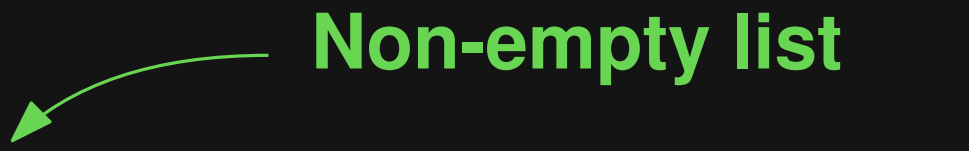
Example

Let's "lexecute" the motivating example:

```
if (not has_min_size(all_data)):
  raise RuntimeError("not enough data")

train_len = round(0.8 * len(all_data))
logger.info(f"Extracting data with {config_str}")
train_data = all_data[0:train_len]

# ...
```

Example

Let's "lexecute" the motivating example:




```
if (not has_min_size(all_data)):
  raise RuntimeError("not enough data")

train_len = round(0.8 * len(all_data))
logger.info(f"Extracting data with {config_str}")
train_data = all_data[0:train_len]

# ...
```

Example

Let's "lexecute" the motivating example:

Function that returns `True`

Non-empty list

```
if (not has_min_size(all_data)):
  raise RuntimeError("not enough data")

train_len = round(0.8 * len(all_data))
logger.info(f"Extracting data with {config_str}")
train_data = all_data[0:train_len]

# ...
```

Example

Let's "lexecute" the motivating example:

Function that returns `True`

Non-empty list

```
if (not has_min_size(all_data)):
  raise RuntimeError("not enough data")

train_len = round(0.8 * len(all_data))
logger.info(f"Extracting data with {config_str}")
train_data = all_data[0:train_len]

# ...
```

Non-empty string

Example

Let's "lexecute" the motivating example:

Function that returns `True`

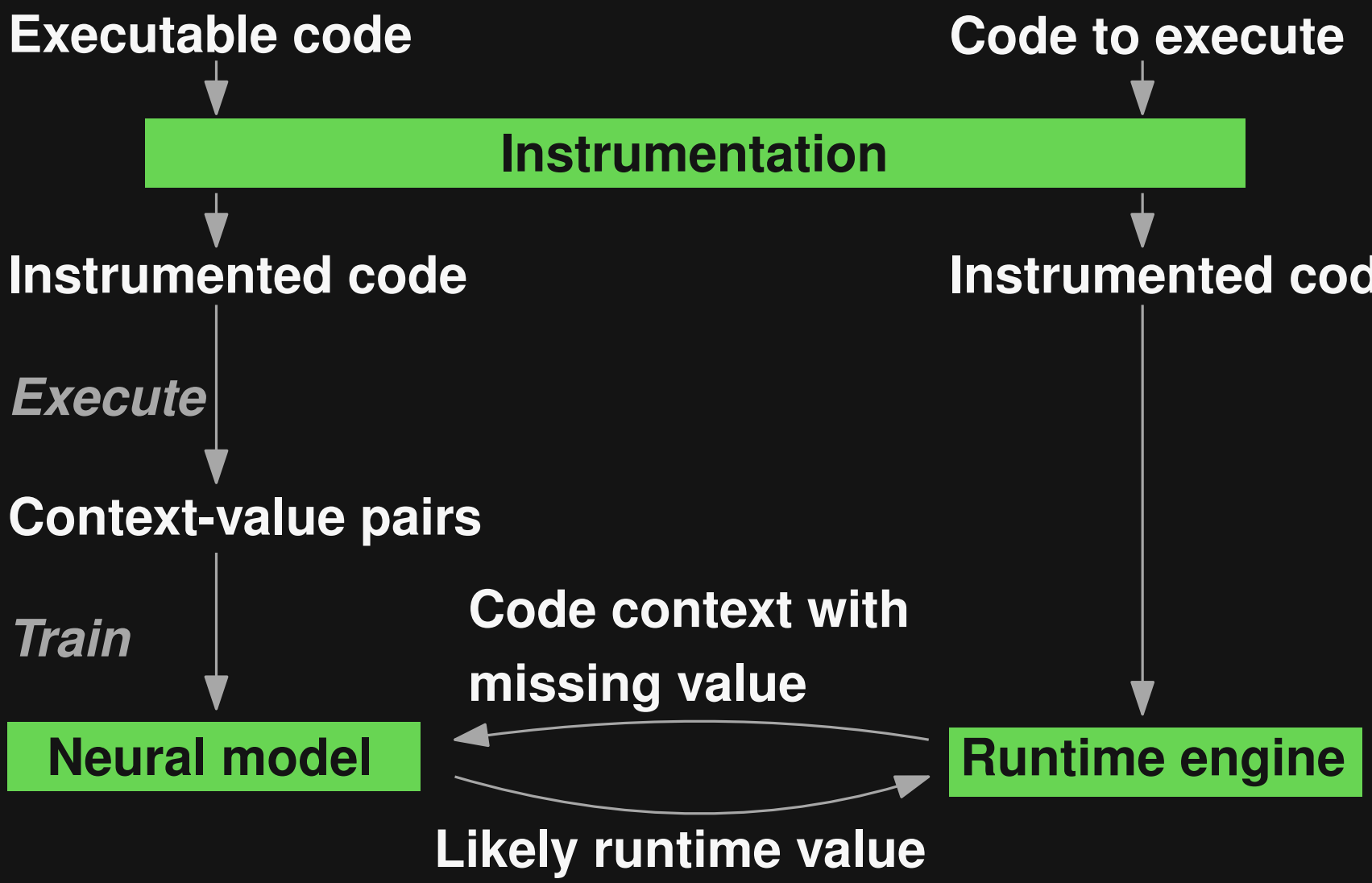
Non-empty list

```
if (not has_min_size(all_data)):
  raise RuntimeError("not enough data")

train_len = round(0.8 * len(all_data))
logger.info(f"Extracting data with {config_str}")
train_data = all_data[0:train_len]

# ...
```

Object with a method

Non-empty string

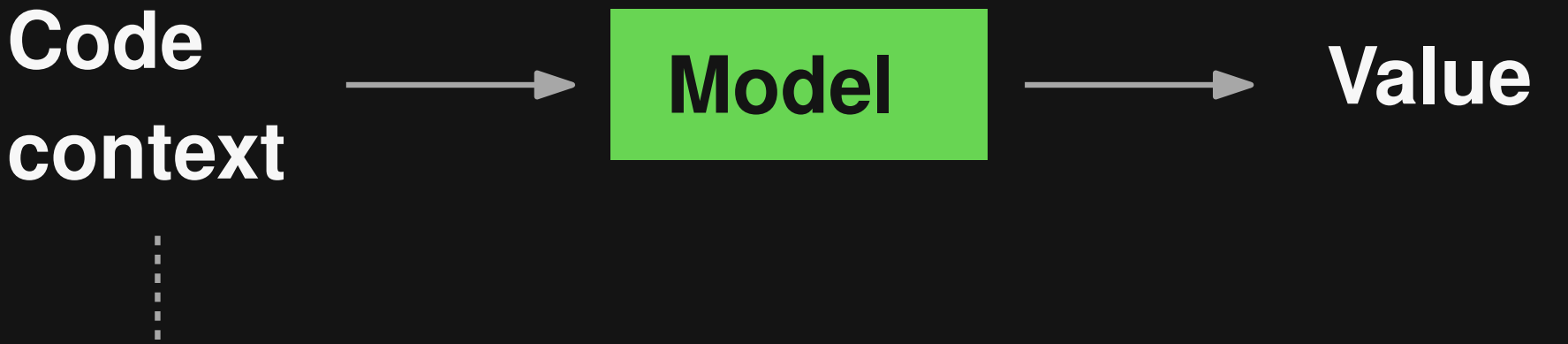
Overview of LExecutor



Neural Model: Data Representation

LExecutor: Learning-Guided Execution, FSE'23 (major rev.)

Neural Model: Data Representation



$$n \ \langle sep \rangle \ k \ \langle sep \rangle \ c_{pre} \ \langle mask \rangle \ c_{post}$$

Name used to refer to a value

Kind of value (variable, attribute, or return value)

Code before/after the reference to the value

LExecutor: Learning-Guided Execution, FSE'23 (major rev.)

Neural Model: Data Representation

Concrete values abstracted into 23 classes, e.g.,

- `None`, `True`, `False`
- Negative/zero/positive integer
- Empty/non-empty list
- Callable

Train & Predict

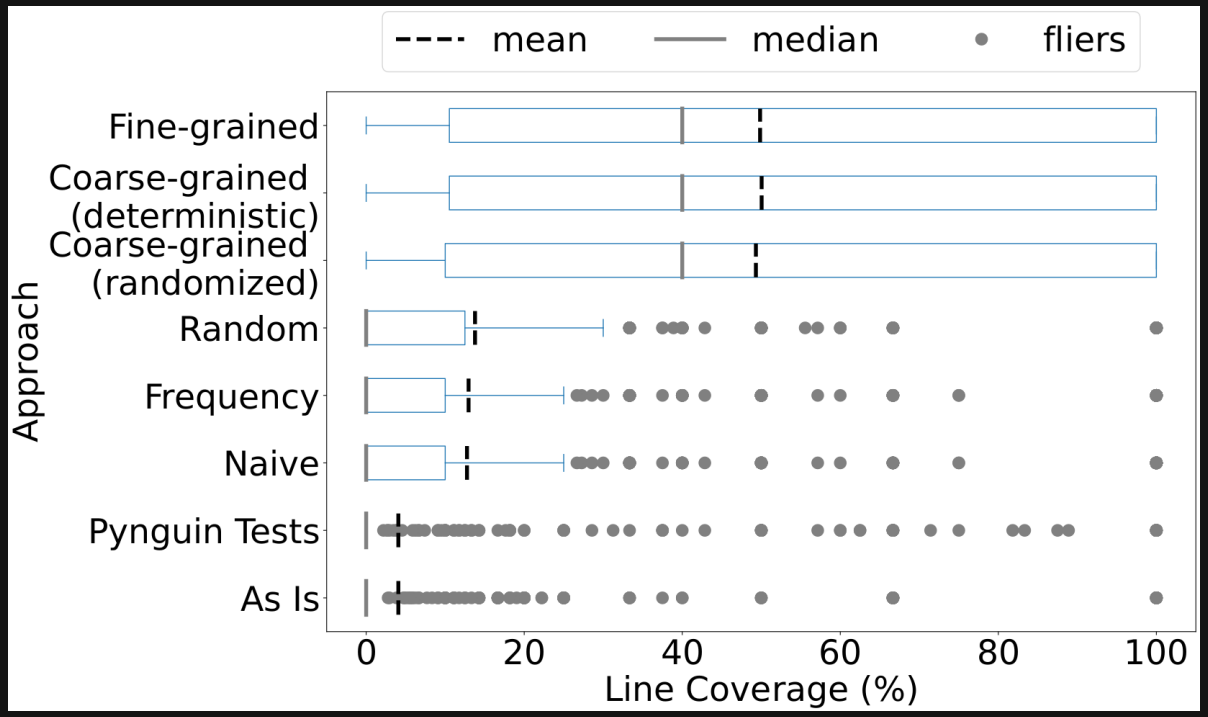
- **Fine-tune a pre-trained CodeT5 model**
- **During prediction:**
  For each use of a value
  - Read value and, if it exists, return it
  - If undefined, query the model and return its prediction

Evaluation

- **Training data**
  - 226k unique value-use events from five projects
- **Code snippets to execute**
  - Open-source functions: 1,000 extracted from five projects
  - Stack Overflow snippets: 462 syntactically correct code snippets in answers to 1,000 Python-related questions

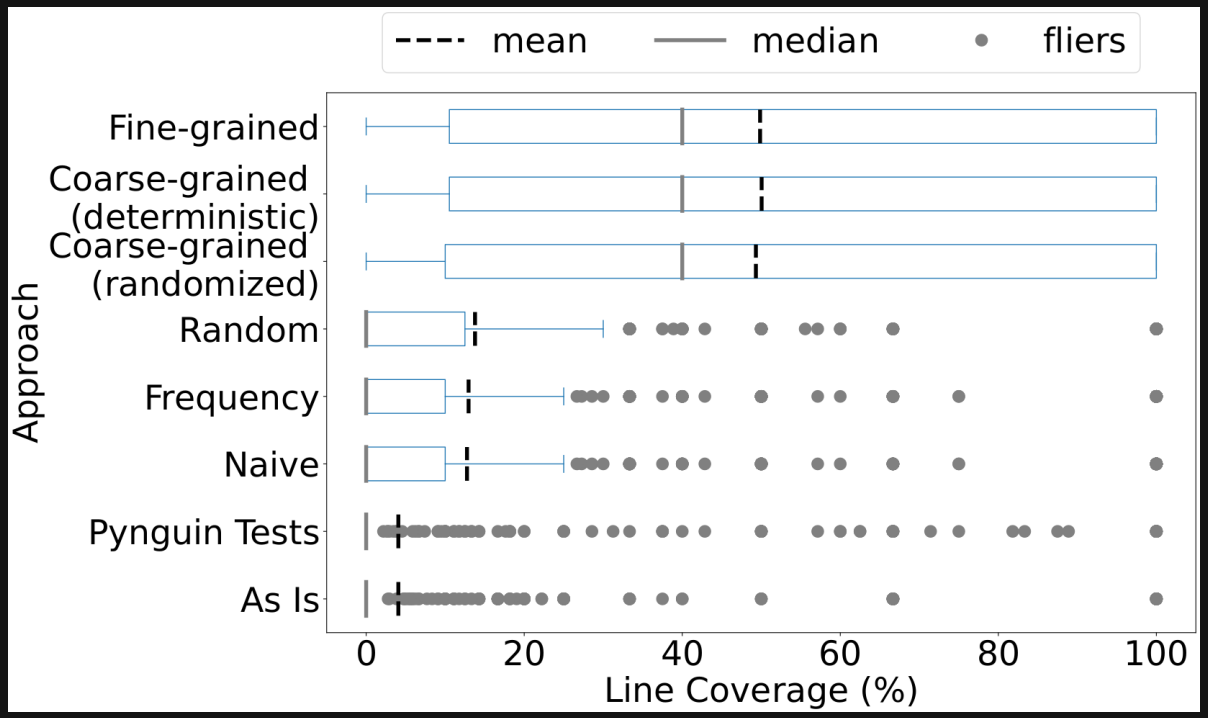
Results

- **Accuracy** of neural model:
  80.1% (top-1) – 94.2% (top-5)
- **Successfully executed lines**:



Results

- **Accuracy** of neural model:
  80.1% (top-1) – 94.2% (top-5)
- **Successfully executed lines**:



Variants of LExecutor

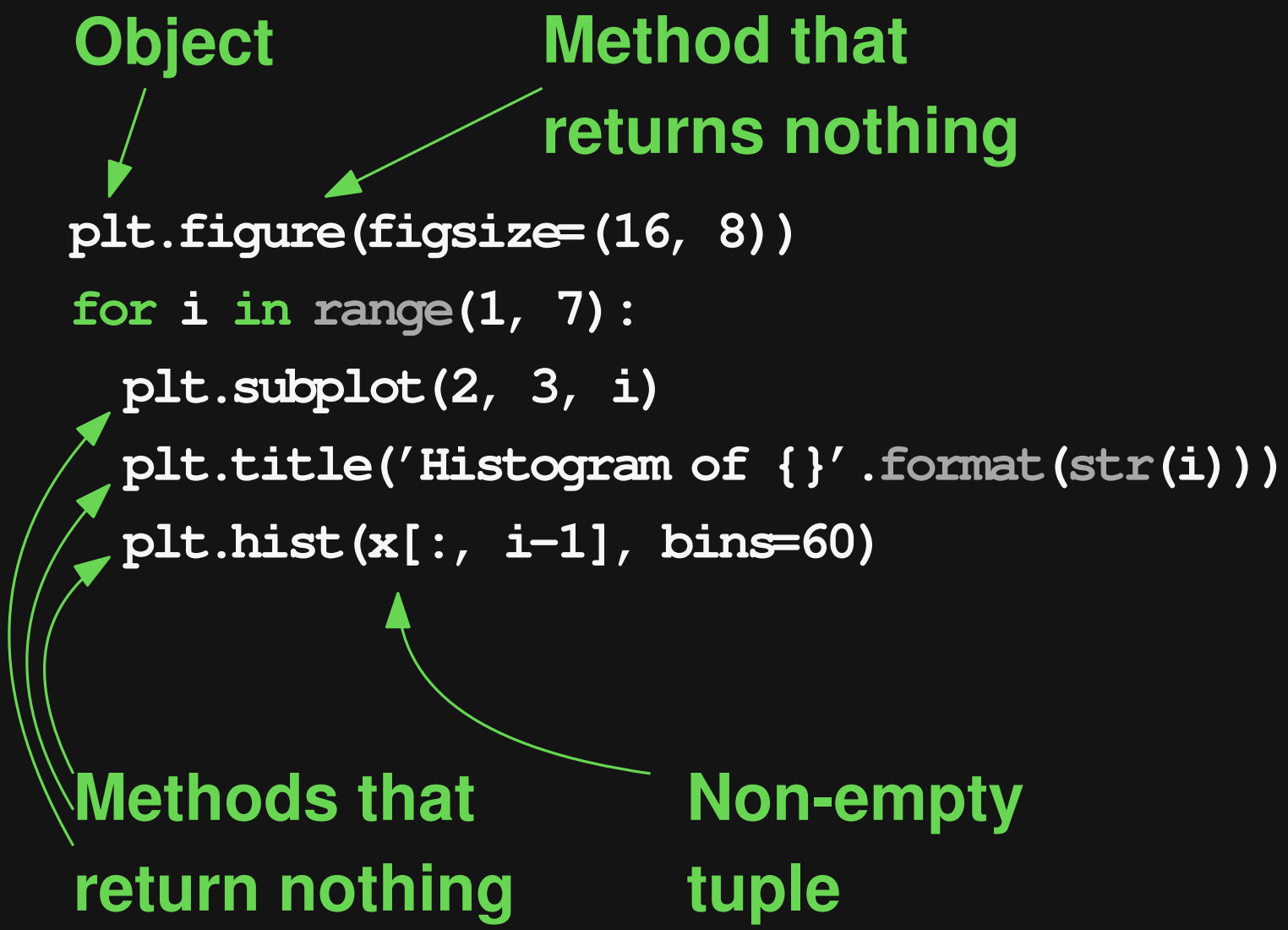
LExecutor without model

State of the art

Example: Stack Overflow Snippet

```
plt.figure(figsize=(16, 8))
for i in range(1, 7):
  plt.subplot(2, 3, i)
  plt.title('Histogram of {}'.format(str(i)))
  plt.hist(x[:, i-1], bins=60)
```

Example: Stack Overflow Snippet



Example: Stack Overflow Snippet

Object

Method that returns nothing

```
plt.figure(figsize=(16, 8))
for i in range(1, 7):
  plt.subplot(2, 3, i)
  plt.title('Histogram of {}'.format(str(i)))
  plt.hist(x[:, i-1], bins=60)
```

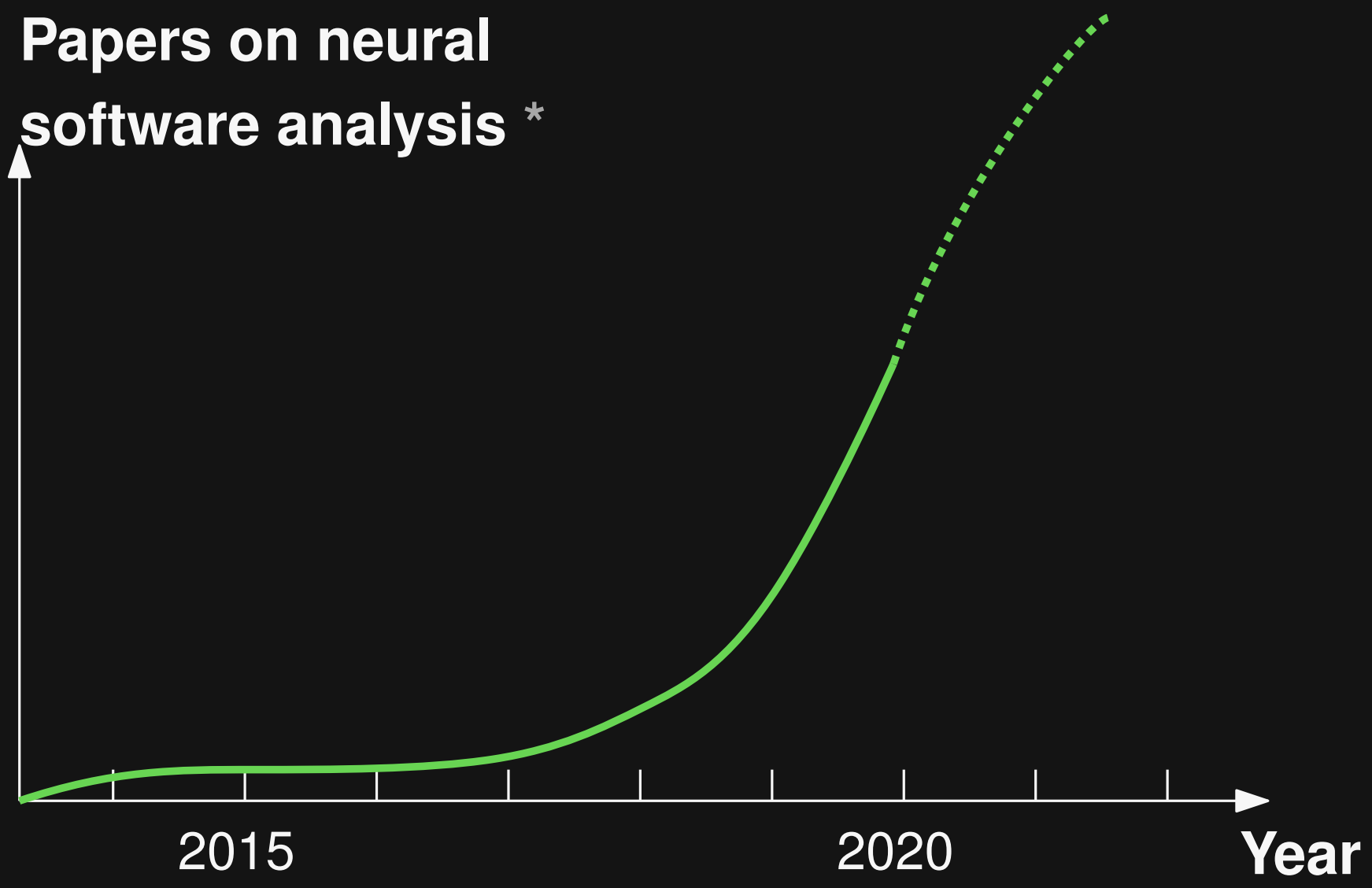
Crash

TupleError: tuple indices must be integers or slices, not tuple

Methods that return nothing

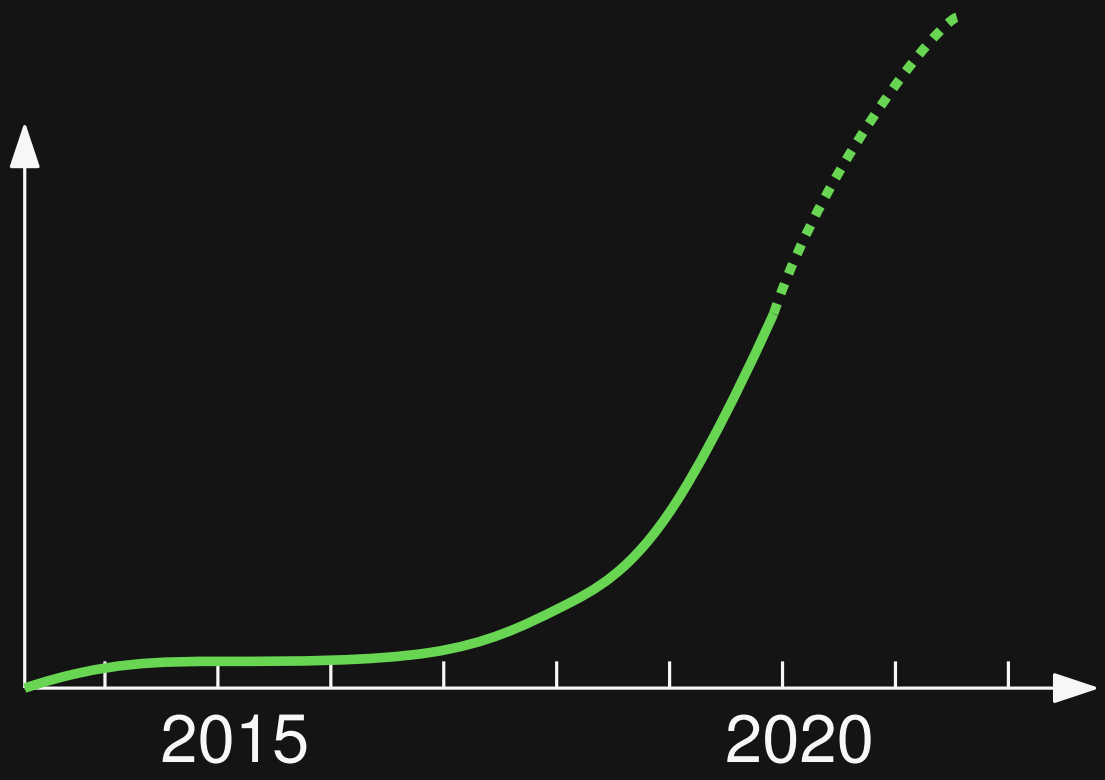
Non-empty tuple

The Road Ahead



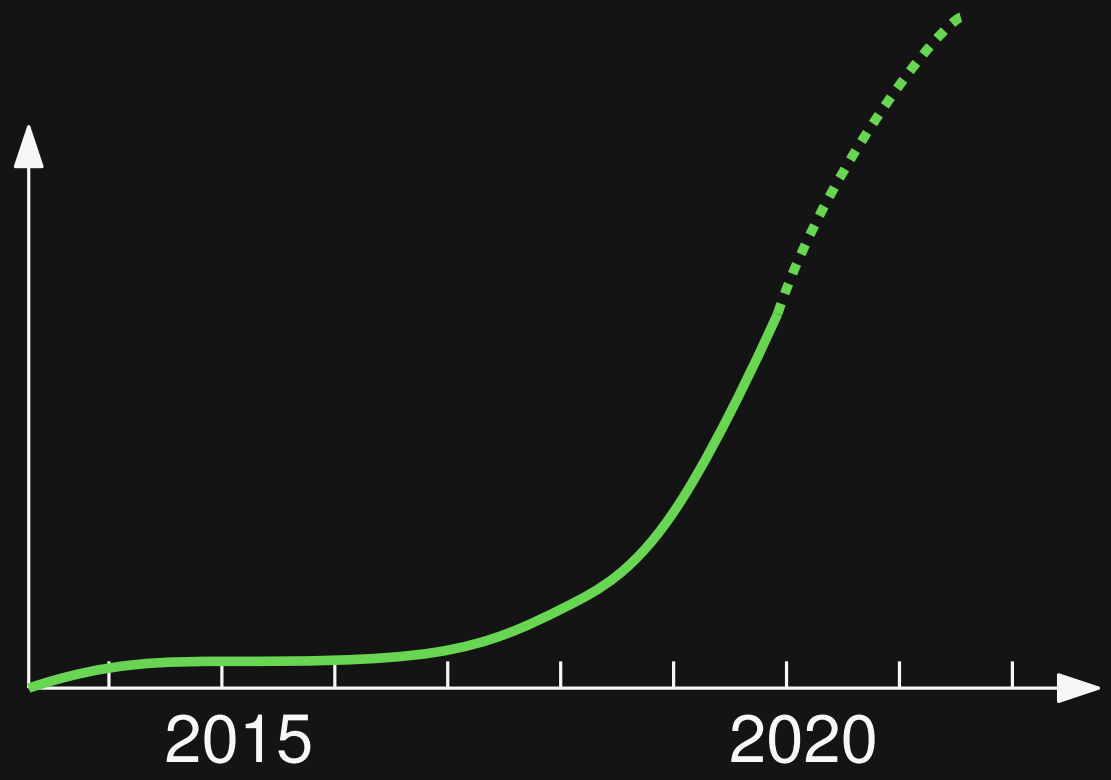
* Estimate based on *Neural Software Analysis*, CACM'22

The Road Ahead



The Road Ahead

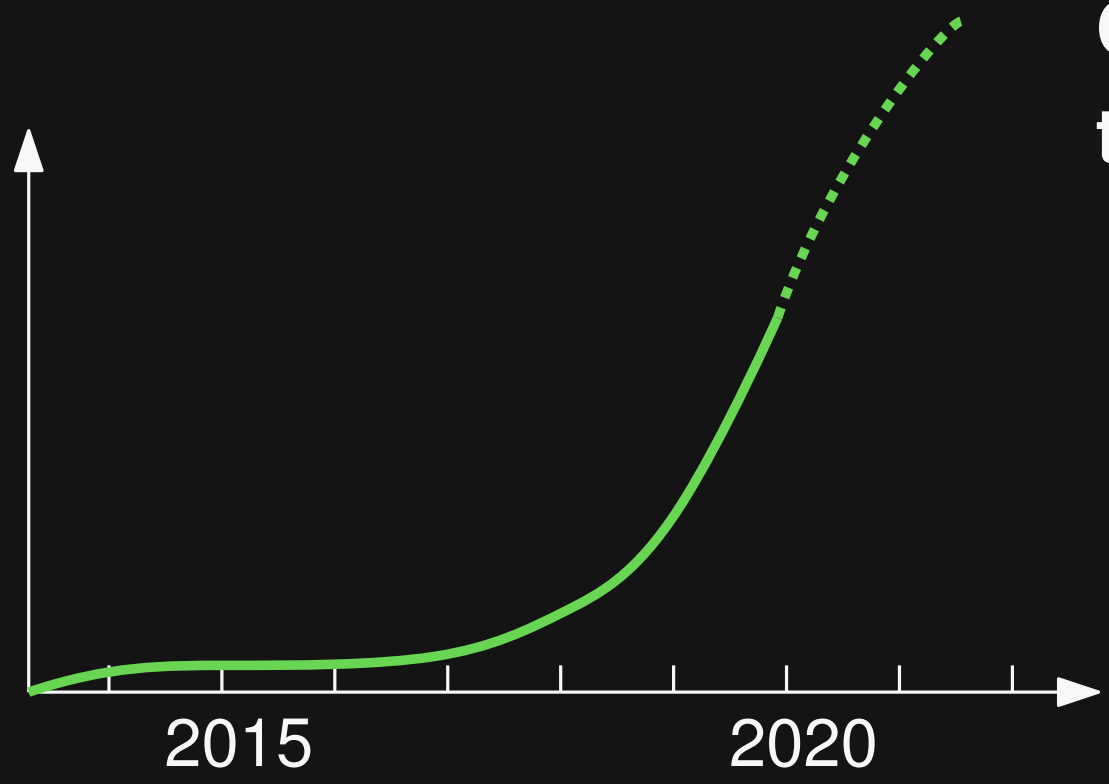
General-purpose language models



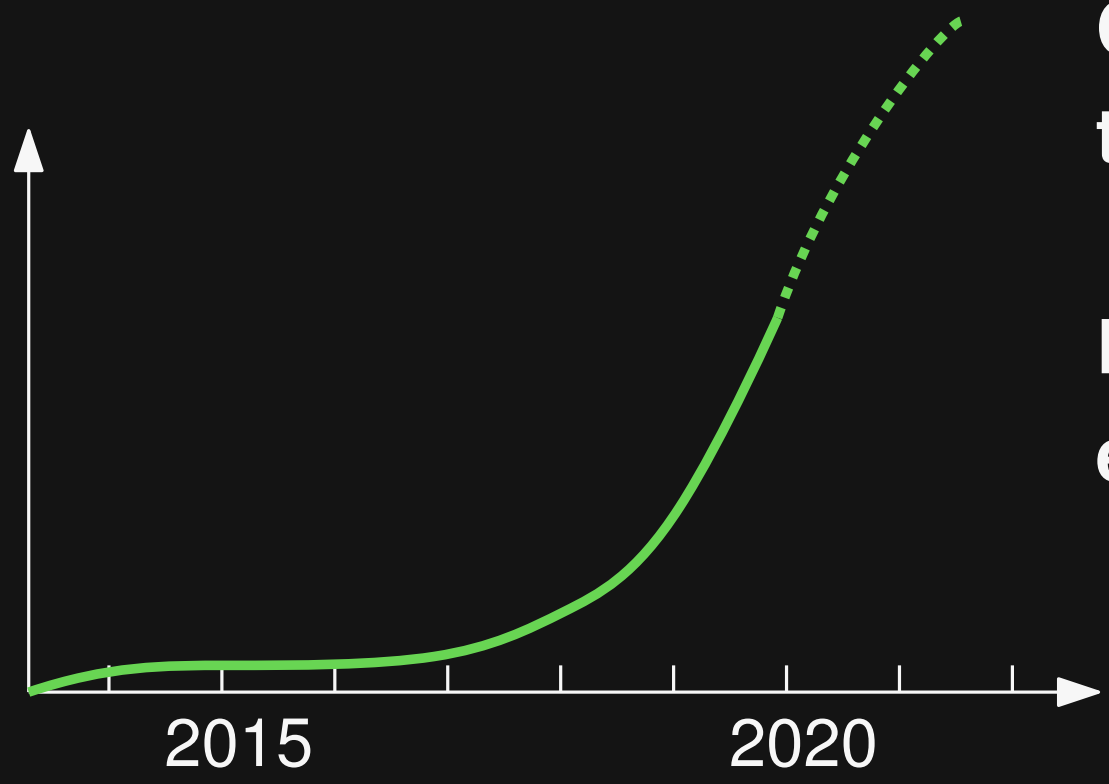
The Road Ahead

General-purpose language models

Combining neural & traditional analysis



The Road Ahead



General-purpose language models

Combining neural & traditional analysis

Reasoning about executions

Summary

- **Fixing type errors with PyTy**

  PyTy: Repairing Static Type Errors in Python, FSE'23 (major rev.)

- **Neural bug detection with CMI-Finder**

  When to Say What: Learning to Find Condition-Message Inconsistencies, ICSE'23

- **Enabling execution with LExecutor**

  LExecutor: Learning-Guided Execution, FSE'23 (major rev.)

Thanks!