Treefix: Enabling Execution with a Tree of Prefixes

Beatriz Souza and Michael Pradel

European Research Council
Established by the European Commission

SOLA SoftwareLab
University of Stuttgart
Germany

Motivating Example

```
try:
    register = get_register_func(self.user_type)
    if register is not None:
        self.email = register(self.name, self.alias)
        if '@' in self.email:
            result = 0
        else:
            result = -1
    else:
        result = -2
except SystemExit:
    result = 1
```

Motivating Example

Missing variable and attribute

```
try:
    register = get_register_func(self.user_type)
    if register is not None:
        self.email = register(self.name, self.alias)
        if '@' in self.email:
            result = 0
        else:
            result = -1
    else:
        result = -2
except SystemExit:
    result = 1
```

Motivating Example

Missing function

Missing variable and attribute

```
try:
    register = get_register_func(self.user_type)
    if register is not None:
        self.email = register(self.name, self.alias)
        if '@' in self.email:
            result = 0
        else:
            result = -1
    else:
        result = -2
except SystemExit:
    result = 1
```

Motivating Example

Missing function

Missing variable and attribute

```
try:
    register = get_register_func(self.user_type)
    if register is not None:
        self.email = register(self.name, self.alias)
        if '@' in self.email:
            result = 0
        else:
            result = -1
    else:
        result = -2
except SystemExit:
    result = 1
```

Missing variable and attribute

FSE'23

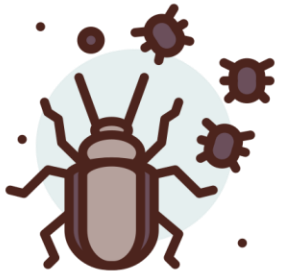
LExecutor: Learning-Guided Execution

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

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

Applications:

Detecting Runtime Type Erros [1]



Reproducing Bugs [2]

Validating Code Changes [3]

1. SelfPiCo: Self-Guided Partial Code Execution with LLMs, ISSTA'24 (Z. Xue, Z. Gao, S. Wang, X. Hu, X. Xia, and S. Li)
2. Feedback-Directed Partial Execution, ISSTA'24 (I. Hayet, A. Scott, and M. d'Amorim)
3. ChangeGuard: Validating Code Changes via Pairwise Learning-Guided Execution (L. Gröninger, B. Souza and M. Pradel)

LExecuting the Motivating Example

Callable: DummyObject class

Object: DummyObject instance

Non-empty string: "a"

DummyObject instance

```
try:
    register = get_register_func(self.user_type)
    if register is not None:
        self.email = register(self.name, self.alias)
        if '@' in self.email:
            result = 0
        else:
            result = -1
    else:
        result = -2
except SystemExit:
    result = 1
```

TypeError: DummyObject object is not iterable

Goal: Set **P** of Prefixes that Maximize Coverage

```
def get_register_func(user_type): return None
self = type('Mock', (object,), {
    'user_type': 'invalid', 'name': 'John Doe',
    'alias': 'jdoe', 'email': ''})()
```

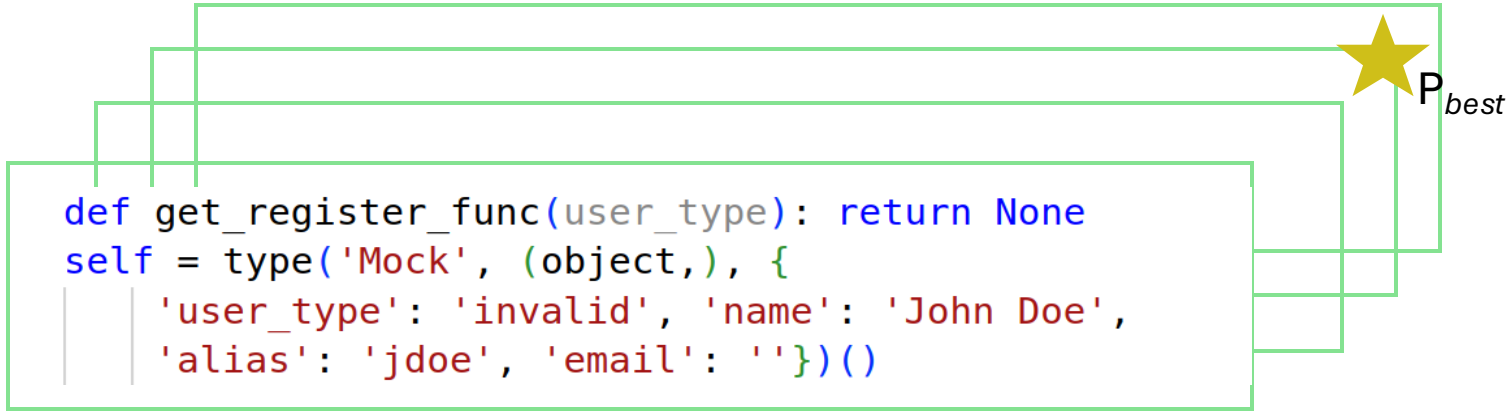
```
try:
    register = get_register_func(self.user_type)
    if register is not None:
        self.email = register(self.name, self.alias)
        if '@' in self.email:
            result = 0
        else:
            result = -1
    else:
        result = -2
except SystemExit:
    result = 1
```

Goal: Set **P** of Prefixes that Maximize Coverage

```
def get_register_func(user_type): return None
self = type('Mock', (object,), {
    'user_type': 'invalid', 'name': 'John Doe',
    'alias': 'jdoe', 'email': ''})()
```

```
try:
    register = get_register_func(self.user_type)
    if register is not None:
        self.email = register(self.name, self.alias)
        if '@' in self.email:
            result = 0
        else:
            result = -1
    else:
        result = -2
except SystemExit:
    result = 1
```

Goal: Set **P** of Prefixes that Maximize Coverage



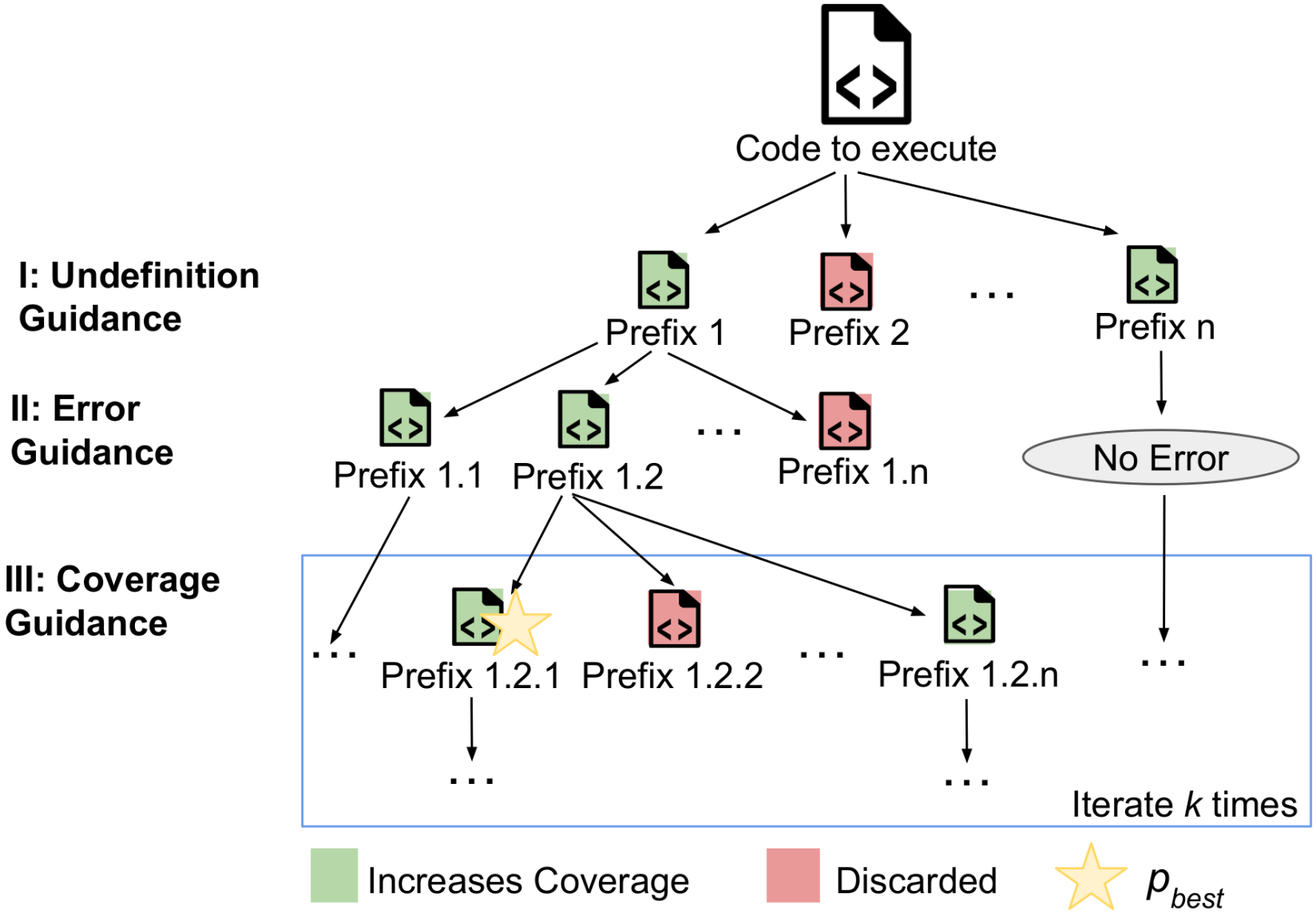

```
try:
    register = get_register_func(self.user_type)
    if register is not None:
        self.email = register(self.name, self.alias)
        if '@' in self.email:
            result = 0
        else:
            result = -1
    else:
        result = -2
except SystemExit:
    result = 1
```

Treefix

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

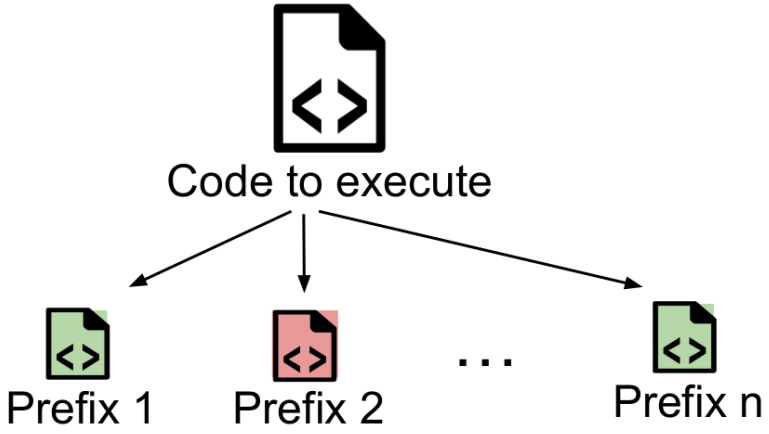
- Key idea is to combine LLMs and feedback to create prefixes that maximize line coverage!

Overview of **Treefix**



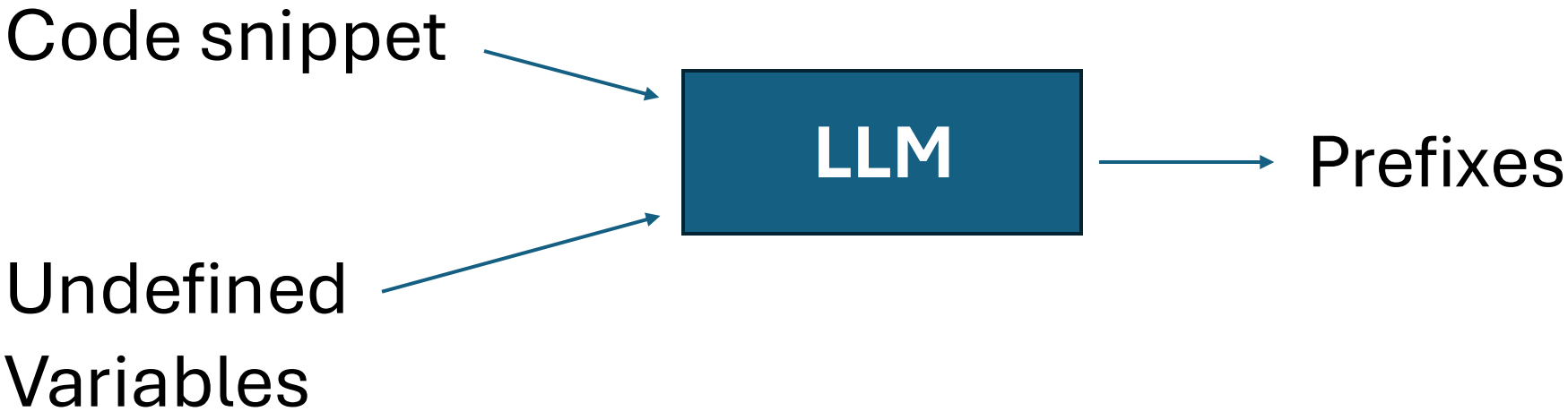
I: Undefinition Guidance

I: Undefinition Guidance



Increases Coverage Discarded

I: Undefinition Guidance



I: Undefinition Guidance

```
def dummy_register_func(name, alias):
    return f'{name}@example.com'
get_register_func = dummy_register_func
self = type('Mock', (object,), {
    'user_type': 'standard', 'name': 'JohnDoe',
    'alias': 'jdoe', 'email': None})()
```

```
try:
    register = get_register_func(self.user_type)
    if register is not None:
        self.email = register(self.name, self.alias)
        if '@' in self.email:
            result = 0
        else:
            result = -1
    else:
        result = -2
except SystemExit:
    result = 1
```

I: Undefinition Guidance

```
def dummy_register_func(name, alias):
    return f'{name}@example.com'
get_register_func = dummy_register_func
self = type('Mock', (object,), {
    'user_type': 'standard', 'name': 'JohnDoe',
    'alias': 'jdoe', 'email': None})()
```

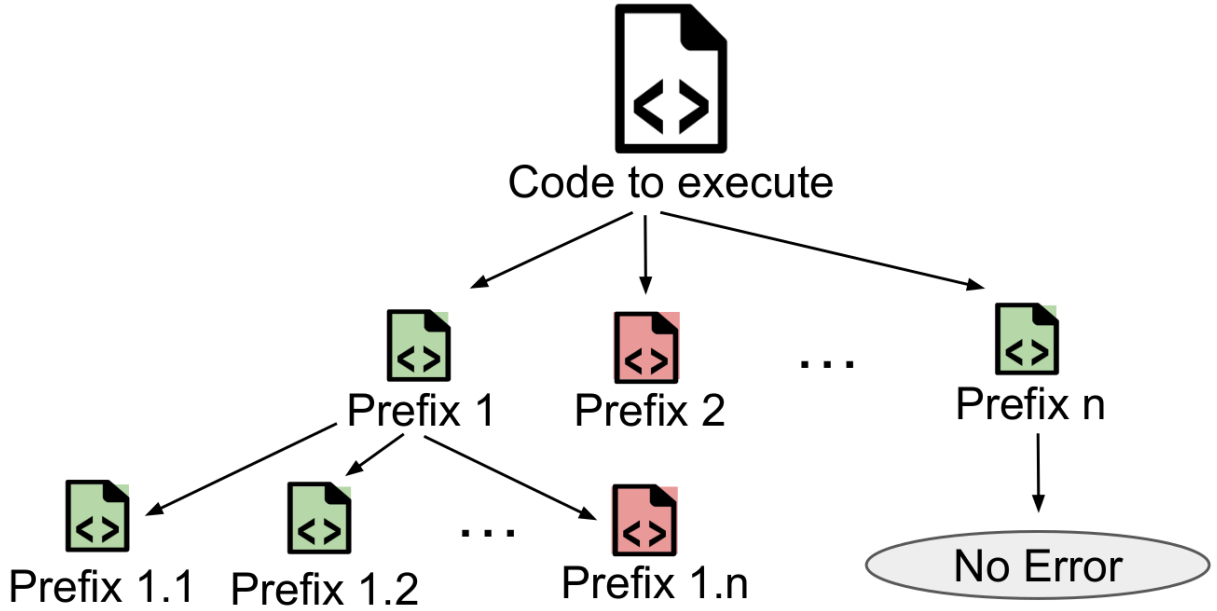
```
try:
    register = get_register_func(self.user_type)
    if register is not None:
        self.email = register(self.name, self.alias)
        if '@' in self.email:
            result = 0
        else:
            result = -1
    else:
        result = -2
except SystemExit:
    result = 1
```

TypeError: dummy register func() missing 1 required positional argument: 'alias'

II: Error Guidance

I: Undefinition Guidance

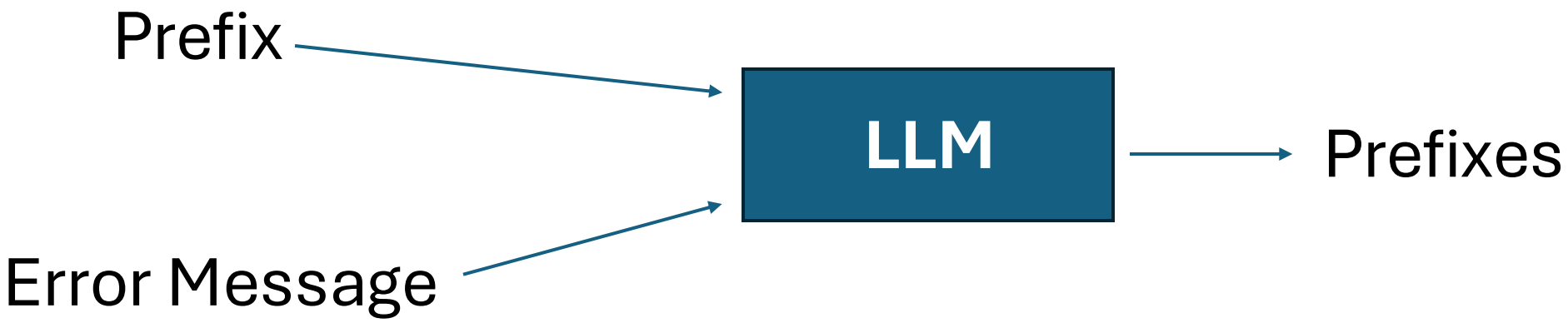
II: Error Guidance



Increases Coverage

Discarded

II: Error Guidance

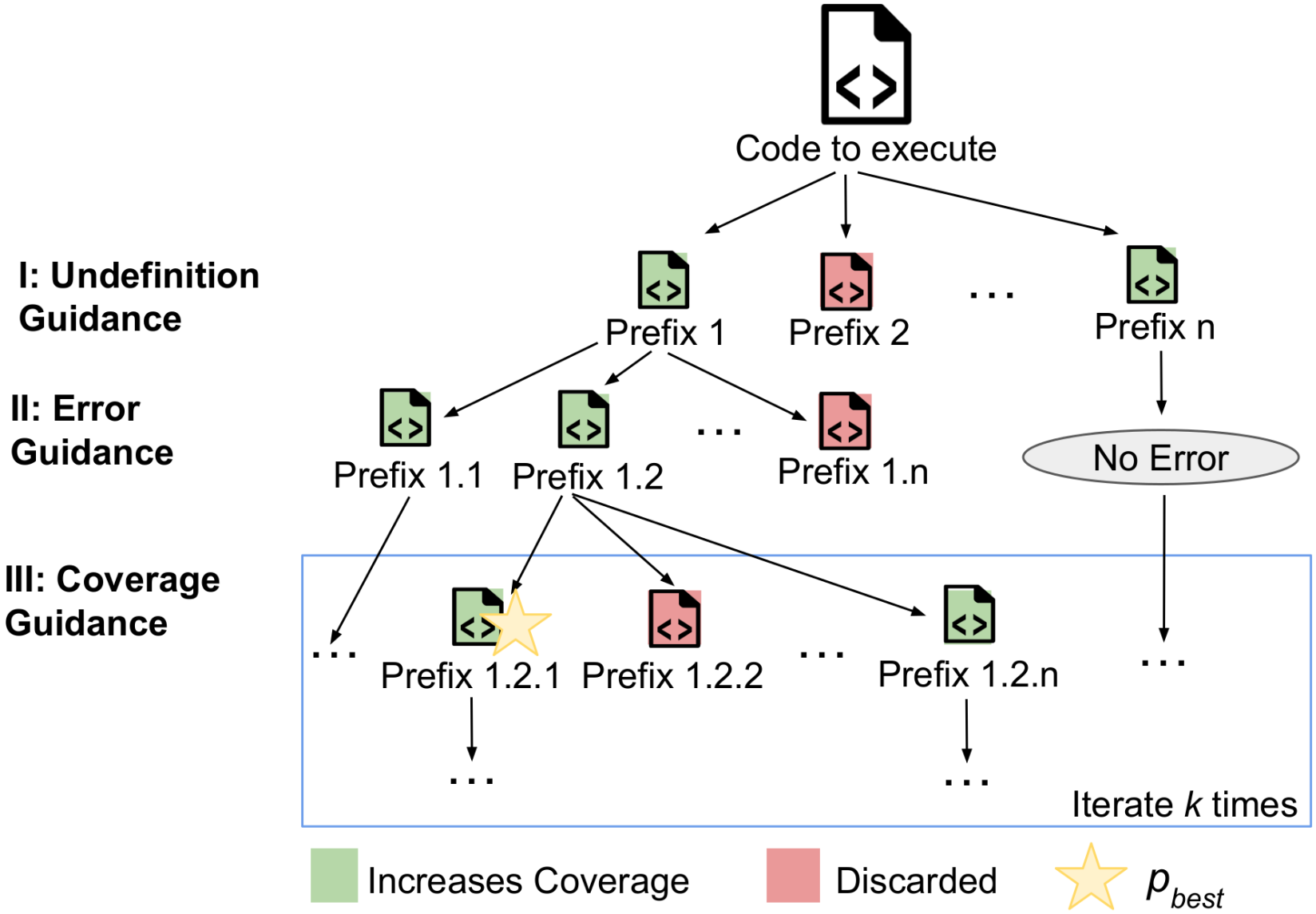


II: Error Guidance

```python
def dummy_register_func(user_type):
    def register(name, alias):
        return f'{name}@example.com'
    return register
get_register_func = dummy_register_func
self = type('Mock', (object,), {
    'user_type': 'standard', 'name': 'JohnDoe',
    'alias': 'jdoe', 'email': None})()
```

```python
try:
    register = get_register_func(self.user_type)
    if register is not None:
        self.email = register(self.name, self.alias)
        if '@' in self.email:
            result = 0
        else:
            result = -1
    else:
        result = -2
except SystemExit:
    result = 1
```

III: Coverage Guidance



III: Coverage Guidance

III: Coverage Guidance

```
class MockRegisterFunction:
    def __call__(self, name, alias): return ''
def get_register_func(user_type):
    return MockRegisterFunction()
self = type('MockSelf', (object,), {
    'user_type': 'mock', 'name': 'test_name',
    'alias': 'test_alias'})()
```

```
try:
    register = get_register_func(self.user_type)
    if register is not None:
        self.email = register(self.name, self.alias)
        if '@' in self.email:
            result = 0
        else:
            result = -1
    else:
        result = -2
except SystemExit:
    result = 1
```

III: Coverage Guidance

```
def get_register_func(user_type): return None
self = type('Mock', (object,), {
    'user_type': 'invalid', 'name': 'John Doe',
    'alias': 'jdoe', 'email': ''})()
```

```
try:
    register = get_register_func(self.user_type)
    if register is not None:
        self.email = register(self.name, self.alias)
        if '@' in self.email:
            result = 0
        else:
            result = -1
    else:
        result = -2
except SystemExit:
    result = 1
```

III: Coverage Guidance

```
class MockRegisterFunction:
    def __call__(self, name, alias): raise SystemExit
def get_register_func(user_type):
    return MockRegisterFunction()
self = type('MockSelf', (object,), {
    'user_type': 'system_exit', 'name': 'test_name',
    'alias': 'test_alias'})()
```

```
try:
    register = get_register_func(self.user_type)
    if register is not None:
        self.email = register(self.name, self.alias)
        if '@' in self.email:
            result = 0
        else:
            result = -1
    else:
        result = -2
except SystemExit:
    result = 1
```

Treefix Output

Set **P** of Prefixes that Maximize Coverage

```
def dummy_register_func(user_type):
    def register(name, alias):
        return f'{name}@example.com'
    return register
get_register_func = dummy_register_func
self = type('Mock', (object,), {
    'user_type': 'standard', 'name': 'JohnDoe',
    'alias': 'jdoe', 'email': None})()
```

```
class MockRegisterFunction:
    def __call__(self, name, alias): return ''
def get_register_func(user_type):
    return MockRegisterFunction()
self = type('MockSelf', (object,), {
    'user_type': 'mock', 'name': 'test_name',
    'alias': 'test_alias'})()
```

```
def get_register_func(user_type): return None
self = type('Mock', (object,), {
    'user_type': 'invalid', 'name': 'John Doe',
    'alias': 'jdoe', 'email': ''})()
```

```
class MockRegisterFunction:
    def __call__(self, name, alias): raise SystemExit
def get_register_func(user_type):
    return MockRegisterFunction()
self = type('MockSelf', (object,), {
    'user_type': 'system_exit', 'name': 'test_name',
    'alias': 'test_alias'})()
```

```
try:
    register = get_register_func(self.user_type)
    if register is not None:
        self.email = register(self.name, self.alias)
        if '@' in self.email:
            result = 0
        else:
            result = -1
    else:
        result = -2
except SystemExit:
    result = 1
```

Evaluation

- **RQ1: Effectiveness at Covering Code**
- **RQ2: Design Choices**
- RQ3: Case Studies
- RQ4: Diversity of Values
- RQ5: Efficiency and Costs

Evaluation

Baselines:

- As Is
- Pynguin Tests [1]
- Type4Py [2]
- **LExecutor** [3]
- **Imcompleter** [4]
- **SelfPiCo** [5]

1. Automated Unit Test Generation for Python, SSBSE'20 (S. Lukasczyk, F. Kroiß, and G. Fraser)
2. Type4Py: Practical deep similarity learning-based type inference for Python, ICSE'22 (A. M Mir, E. Latoškinas, S. Proksch, and G. Gousios)
3. LExecutor: Learning-guided Execution, FSE'23 (B. Souza and M. Pradel)
4. Feedback-Directed Partial Execution, ISSTA'24 (I. Hayet, A. Scott, and M. d'Amorim)
5. SelfPiCo: Self-Guided Partial Code Execution with LLMs, ISSTA'24 (Z. Xue, Z. Gao, S. Wang, X. Hu, X. Xia, and S. Li)

Evaluation

Datasets[1]:

Functions

Project	Description	Functions	LoC
Black	Code formatting	200	2,961
Flask	Web applications	200	1,354
Pandas	Data analysis	200	2,015
Scrapy	Web scraping	200	1,198
TensorFlow	Deep learning	200	2,125
Total		1,000	9,653

stackoverflow Snippets

462 syntactically correct code snippets in answers to 1,000 Python-related questions

RQ1: Effectiveness at Covering Code

Approach	Coverage			
	Open-source functions		Stack Overflow snippets	
Treefix (GPT4o)	P = **0.84**	p_{best} = **0.76**	P = **0.82**	p_{best} = 0.72
Treefix (GPT4o-mini)	P = 0.79	p_{best} = 0.73	P = 0.79	p_{best} = **0.78**
SelfPiCo		0.59		0.75
Incompleter		0.51		0.69
LExecutor		0.51		0.65
Type4Py		0.13		0.46
Pynguin tests		0.04		-
As Is		0.04		0.43

RQ2: Design Choices

Model	Dataset	Coverage		
		I	II	III
GPT4o	Open-source functions	0.72	0.78	**0.84**
	Stack Overflow snippets	0.73	0.77	**0.82**
GPT4o (mini)	Open-source functions	0.64	0.74	0.79
	Stack Overflow snippets	0.70	0.72	0.79

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

Applications:

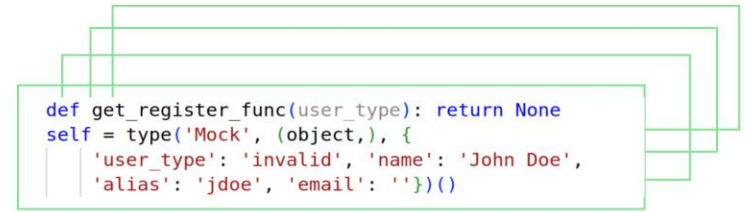
Detecting Runtime Type Erros [1]

Reproducing Bugs [2]

Validating Code Changes [3]

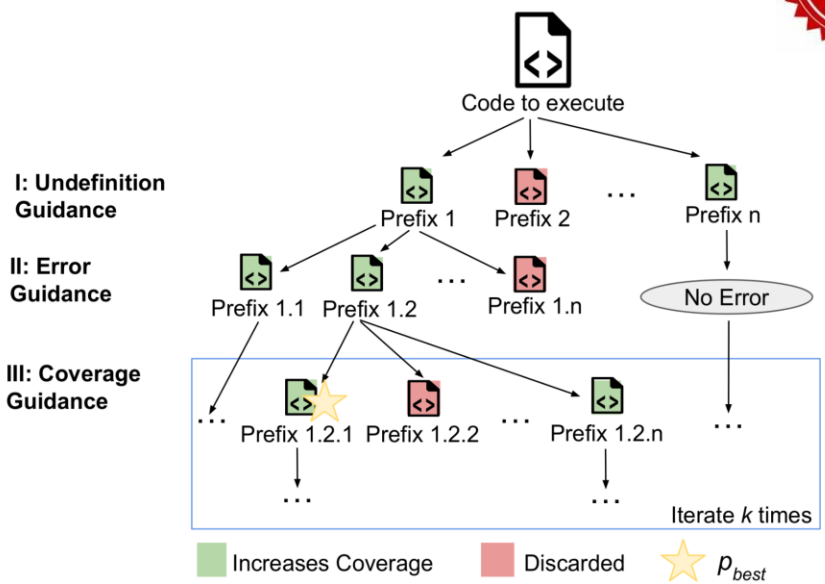
1. SelfPiCo: Self-Guided Partial Code Execution with LLMs, ISSTA'24 (Z. Xue, Z. Gao, S. Wang, X. Hu, X. Xia, and S. Li)
2. Feedback-Directed Partial Execution, ISSTA'24 (I. Hayet, A. Scott, and M. d'Amorim)
3. ChangeGuard: Validating Code Changes via Pairwise Learning-Guided Execution (L. Gröninger, B. Souza and M. Pradel)

Goal: Set **P** of Prefixes that Maximize Coverage




```
try:
    register = get_register_func(self.user_type)
    if register is not None:
        self.email = register(self.name, self.alias)
        if '@' in self.email:
            result = 0
        else:
            result = -1
    else:
        result = -2
except SystemExit:
    result = 1
```


Overview of **Treefix**

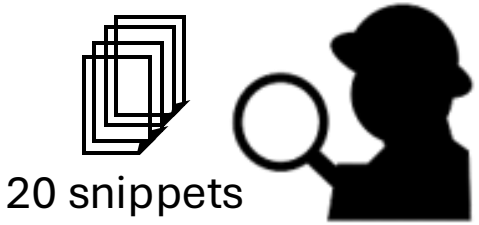


RQ1: Effectiveness at Covering Code

Approach	Coverage			
	Open-source functions		Stack Overflow snippets	
Treefix (GPT4o)	P = **0.84**	p_{best} = **0.76**	P = **0.82**	p_{best} = 0.72
Treefix (GPT4o-mini)	P = 0.79	p_{best} = 0.73	P = 0.79	p_{best} = **0.78**
SelfPiCo		0.59		0.75
Incompleter		0.51		0.69
LExecutor		0.51		0.65
Type4Py		0.13		0.46
Pynguin tests		0.04		-
As Is		0.04		0.43

RQ3: Case Studies (Comparisson w/ LExecutor)

- Treefix achieves higher coverage than LExecutor in 707/1462 snippets



Reason for Higher Coverage	# Snippets
Adequate imports and usage of dependencies	15
Complex objects	13
Diverse primitive values	11
Multiple paths covered	1

Example of *adequate imports* and *complex values* predicted by **Treefix**

```
import pandas as pd
import numpy as np

index = pd.MultiIndex.from_product(
    [['foo', 'bar'], ['one', 'two']],
    names=['first', 'second'])
data = np.random.rand(4, 4)
multiindex_dataframe_random_data = pd.DataFrame(
    data, index=index, columns=index)
tm = type('Mock', (object,),
          {'assert_almost_equal': lambda x, y: None})
```

```
df = multiindex_dataframe_random_data.T
expected = df.values[:, 0]
result = df["foo", "one"].values
tm.assert_almost_equal(result, expected)
```

RQ4: Diversity of Values

Type	Unique values
str	**2962**
list	**2567**
Mock	**1383**
type	**1083**
dict	**936**
object	**919**
set	657
MockSelf	406
ContextVar	342
tuple	322
SimpleNamespace	320
ndarray	298
bytes	256
Pattern	196
module	181
Line	148
int	110
...	
Total	16528

RQ5: Efficiency and Costs

Model	Dataset	Time (seconds)				Price (USD)						
		Step 1	Step 2	Step 3	All	Step1		Step 2		Step 3		All
						Input	Output	Input	Output	Input	Output	
GPT4o	Open-source functions	13.7	2.7	2.2	**18.6**	0.002	0.019	0.056	0.106	0.132	0.109	**0.425**
	Stack Overflow snippets	5.2	0.7	1.9	**7.8**	0.001	0.006	0.012	0.013	0.119	0.059	**0.212**
GPT4o-mini	Open-source functions	14.2	3.8	3.2	**21.2**	6.67x10-5	0.0007	0.0018	0.0045	0.004	0.004	**0.016**
	Stack Overflow snippets	4.4	0.4	2.2	**7.0**	3.62x10-5	0.0003	0.0005	0.0008	0.004	0.003	**0.008**

I: Undefinition Guidance

Provide self-contained and concrete Python values to initialize the undefined variables in the code snippet. *1*

```
# begin code snippet
(see Figure 1a)
# end code snippet
```
2

```
# begin undefined variables
self
get_register_func
# end undefined variables
```
3

```
# begin undefined attributes and methods
self.user_type
self.name
self.alias
# end undefined attributes and methods
```
4

Respond strictly with JSON. The JSON should be compatible with the TypeScript type "Response":

```ts
interface Response {
  // Python import statements, one string per import
  imports: string[];

  // Python code to initialize undefined variables,
       one string per variable
  initialization: string[];
}
```

II: Error Guidance

When trying to execute the code snippet with the provided imports and initialization, the following error happens: *1*

```
# begin error message
Execution error at line 14:
    register = get_register_func(self.user_type)
TypeError: dummy_register_func() missing 1 required
     positional argument: 'alias'
# end error message
```
2

Provide a fixed version of the imports and initialization to solve the error and make the code snippet executable. *3*

Respond strictly with JSON. The JSON should be compatible with the TypeScript type "Response":

```ts
interface Response {
  // Python import statements, one string per import
  imports: string[];

  // Python code to initialize undefined variables,
       one string per variable
  initialization: string[];
}
```

III: Coverage Guidance

1

When trying to execute the code snippet with the provided imports and initialization, the lines commented with "uncovered" are not executed.

2

```
# begin code snippet
try:
    register = get_register_func(
        self.user_type)
    if register is not None:
        self.email = register(
            self.name, self.alias)
        if '@' in self.email:
            result = 0
        else:
            result = -1 # uncovered
    else:
        result = -2 # uncovered
 except SystemExit: # uncovered
    result = 1 # uncovered
# end code snippet
```

3

Provide a modified version of the imports and initialization to execute one of the uncovered paths in the code snippet.