MANCHESTER 1824
The University of Manchester

ESBMC-Python: A Bounded Model Checker for Python Programs

Bruno Farias, Rafael Menezes, Eddie Lima Filho, Youcheng Sun, Lucas C. Cordeiro
bruno.farias@manchester.ac.uk

University of Manchester
18th September 2024

MANCHESTER 1824
The University of Manchester

Introduction

Research Problem

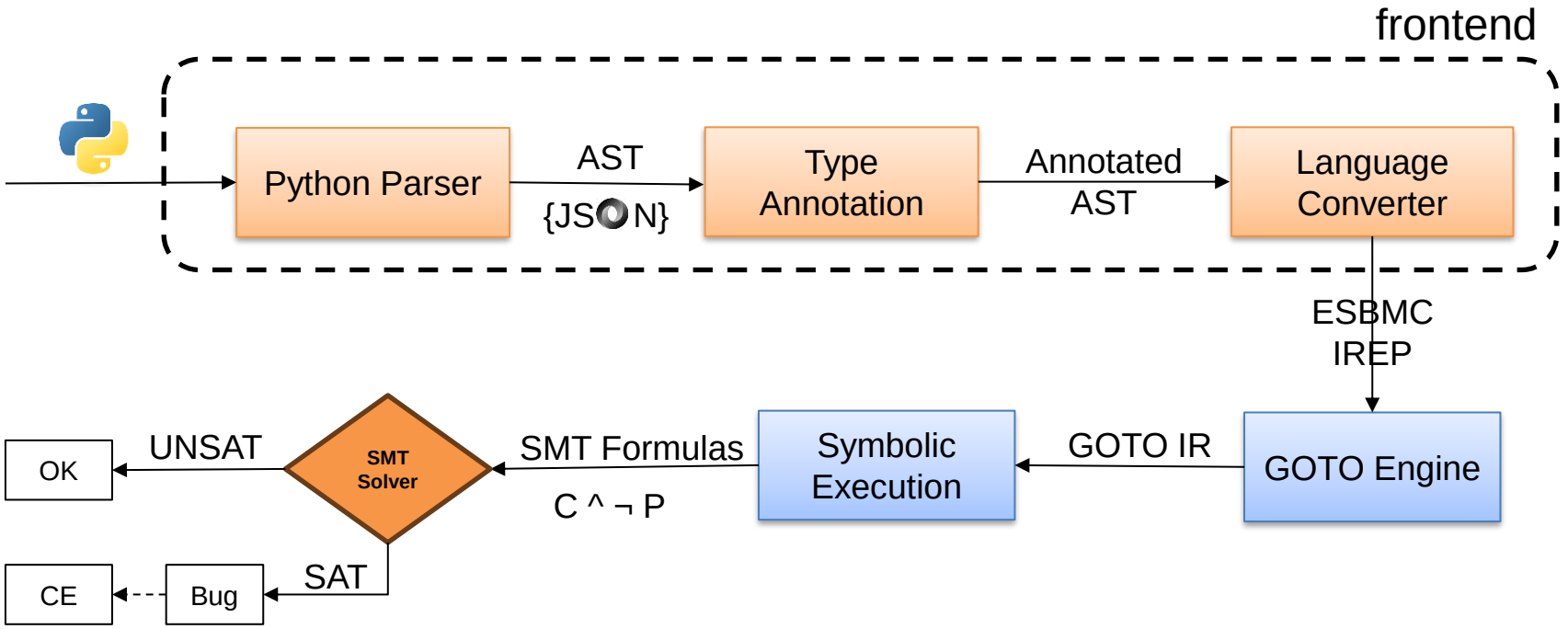
Python is widely used in critical systems such as AI, automotive, and computer vision. However, there is a **lack of formal tools** for verifying Python program correctness, primarily due to the **dynamic nature** of the language and the **absence of type information**.

Approach

Develop a **frontend** for an SMT-based **Bounded Model Checker** that can infer and **add type information**, enabling exhaustive exploration of program paths to identify issues.

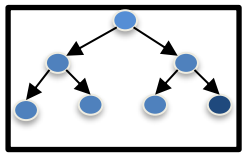
MANCHESTER 1824
The University of Manchester

ESBMC-Python: Overview



Verification properties: Division-by-zero, indexing errors, arithmetic overflow, and user-defined assertions.

Language Translations



{JSON}

ESBMC IREP

```python
def add(a:int, b:int) -> int:
  return a+b

n1 = 1
n2 = 2
result = add(n1,n2)
assert result == 3
```

```
{
    "_type": "Assert",
    "test": {
        "_type": "Compare",
        "comparators": [
            {
                "_type": "Constant",
                "value": 3
            }
        ],
        "left": {
            "id": "result",
            "lineno": 7
        },
        "ops": [
            {
                "_type": "Eq"
            }
        ]
    }
}
```

```
code
  * type: code
  * operands:
    0: =
        * type: bool
        * operands:
          0: symbol
              * type: signedbv
                  * width: 32
              * name: result
              * identifier: py:test.py@result
          1: constant
              * type: signedbv
                  * width: 32
              * value: 00000000000000000000000000000011
  * statement: assert
```

MANCHESTER 1824
The University of Manchester

JSON-Based Type Annotation

Constant Values

x = 10 → x:**int** = 10

Referred Variables

*y = x → y:**int** = x*

Class Instances

z = MyClass()

Function Calls

def foo() : return 1
*x = foo() → x:**int** = foo()*

x = 10 → *x:**int** = 10*


```
{
    "_type": "Assign",
    "col_offset": 0,
    "end_col_offset": 6,
    "end_lineno": 1,
    "lineno": 1,
    "targets": [
        {
            "_type": "Name",
            "col_offset": 0,
            "ctx": {
                "_type": "Store"
            },
            "end_col_offset": 1,
            "end_lineno": 1,
            "id": "x",
            "lineno": 1
        }
    ],
    "type_comment": null,
    "value": {
        "_type": "Constant",
        "col_offset": 4,
        "end_col_offset": 6,
        "end_lineno": 1,
        "kind": null,
        "lineno": 1,
        "n": 10,
        "s": 10,
        "value": 10
    }
}
```

```
{
    "_type": "AnnAssign",
    "annotation": {
        "_type": "Name",
        "col_offset": 2,
        "ctx": {
            "_type": "Load"
        },
        "end_col_offset": 5,
        "end_lineno": 1,
        "id": "int",
        "lineno": 1
    },
    "col_offset": 0,
    "end_col_offset": 10,
    "end_lineno": 1,
    "lineno": 1,
    "simple": 1,
    "target": {
        "_type": "Name",
        "col_offset": 0,
        "ctx": {
            "_type": "Store"
        },
        "end_col_offset": 1,
        "end_lineno": 1,
        "id": "x",
        "lineno": 1
    },
    "value": {
        "_type": "Constant",
        "col_offset": 8,
        "end_col_offset": 10,
        "end_lineno": 1,
        "kind": null,
        "lineno": 1,
        "n": 10,
        "s": 10,
        "value": 10
    }
}
```

MANCHESTER 1824
The University of Manchester

ESBMC usage

$ esbmc main.py --multi-property

```
def div(a:int, b:int) -> int:
  return a/b

x: int = nondet_int()
y:int = nondet_int()
res = div(x,y)

l1 = [1,2,3]
i = 0
sum = 0
while i <= len(l1):
  sum += l1[i]
  i += 1

assert sum == 6
```

```
[Counterexample]


State 1 file main.py line 5 column 0 thread 0
------------------------------------------------
  y = 0 (00000000 00000000 00000000 00000000)

State 2 file main.py line 2 column 4 function div thread 0
------------------------------------------------
Violated property:
  file main.py line 2 column 4 function div
  division by zero
  b != 0
```

```
[Counterexample]


State 1 file main.py line 12 column 4 thread 0
------------------------------------------------
Violated property:
  file main.py line 12 column 4
  array bounds violated: array `l1' upper bound
  (signed long int)i < 3
```

ESBMC usage

Benchmark: **Ethereum Blockchain Consensus Specification**
https://github.com/ethereum/consensus-specs

- Infrastructure for converting Markdown documents into a Python library.

- Non-determinism and BMC revealed an arithmetic overflow when calling *integer_square_root* below with INT_MAX as a parameter.

```python
def integer_squareroot(n: uint64) -> uint64:
    """
    Return the largest integer ``x`` such that ``x**2 <= n``.
    """
    x = n
    y = (x + 1) // 2
    while y < x:
        x = y
        y = (x + n // x) // 2
    return x
```

Python program

```
[Counterexample]


State 1 line 1486 column 4 function integer_squareroot thread 0
------------------------------------------------------
  x = 0xFFFFFFFFFFFFFFFF (11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111)

State 2 line 1487 column 4 function integer_squareroot thread 0
------------------------------------------------------
Violated property:
  line 1487 column 4 function integer_squareroot
  arithmetic overflow on add
  !overflow("+", x, 1)


VERIFICATION FAILED
```

ESBMC-Python output

MANCHESTER 1824
The University of Manchester

Experimental Results

Category	Test Cases	Memory Usage	Execution Time
Arith operations	2	26.4 MB	33.5 ms
Assignments	5	18.5 MB	38 ms
Assume	4	16.5 MB	28.2 ms
Binary operations	2	20.5 MB	29.5 ms
Binary types	4	20.4 MB	28.5 ms
Built-in functions	7	19.9 MB	28.1 ms
Classes	9	19 MB	27.1 ms
Conditionals	4	17.8 MB	25.5 ms
Functions	11	21.8 MB	30 ms
Imports	8	15.3 MB	49.1 ms
Logical operations	6	20.4 MB	24.5 ms
Loops	10	20.7 MB	35.4 ms
Non-determinism	4	21.4 MB	29.2 ms
Numeric types	6	20.9 MB	29.1 ms
Type annotation	3	14.5 MB	27.3 ms

- Benchmark suite consisting of 85 programs, categorized into 15 groups.
- Tests with both failling and passing assertions to evaluate reasoning on different Python features.
- The verification time (24.5 to 49.1 ms) is satisfactory compared to other BMC tools.
- Memory consumption (14.5 to 26.4 MB) is also usual and considered low for modern computers.

MANCHESTER 1824
The University of Manchester

Conclusion and Future Work

- ESBMC-Python demonstrates the feasibility of using BMC for the formal verification of Python programs.
- Verification of AI Libraries, such as TensorFlow and Pytorch.
- Support for concurrency issues, unhandled exceptions and unbounded integers handling.
- Enhance and extend type annotation algorithm.

ESBMC-Python: A Bounded Model Checker for Python programs

Thank you

Bruno Farias

bruno.farias@manchester.ac.uk

University of Manchester

18th September 2024