TypeEvalPy: Benchmark to Evaluate Type Inference Techniques of Python

***Abstract*—...**

***Index Terms*—Static analysis, Type inference, Python, Benchmark**

I. Introduction

- Begin by discussing the general concept of Python's type system.
- Highlight that the area of type inference has seen a surge in research interest in recent times.
- Emphasize the focus on generating type annotations, noting, however, that these are often seen as conservative estimations.
- Discuss the recent trend of applying machine learning-based techniques to address the challenge of type inference in Python.
- Point out the dependency of machine learning methods on vast datasets derived from public domain projects, and outline the issues related to this approach: the questionable validity of base type annotations as highlighted by several studies; the inability of these methods to handle user-defined types and rarely encountered types; and their frequent failure to capture the intricate features of Python due to the scarcity of these features in the foundational datasets.
- Criticize machine learning methods for neglecting to test their inferential outputs on smaller, more complex code snippets.
- Call attention to the absence of standard benchmarks available in the public domain that could facilitate a uniform evaluation of existing techniques.
- Considering the increasing popularity of Python type inference systems and the deficit of fair, standardized benchmarks for comparison, we present our comprehensive effort to construct a type benchmark designed to capture Python's diverse feature set.
- Inspired by CamBench [1] In order to scrutinize analysis capabilities against a variety of program analysis sensitivities, we build targeted snippets, also combining these sensitivities to evaluate more complex situations.
- Our TypeEvalPy will therefore serve as a fair, comprehensive benchmark for assessing type inference systems in practical applications.
- We've designed our benchmark to be modular and easily adaptable, allowing the community to conveniently test new type systems.
- Additionally, we apply our benchmark to evaluate ten existing tools from both academic and open-source communities, addressing three specific research questions in the process.

II. TypeEvalPy Benchmarking Framework

- Description of the design and structure of the benchmark
- Explanation of how the benchmark addresses Python's diverse feature set and program analysis sensitivities
- Description of the modularity and extensibility of the benchmark

III. Evaluation

- Detailed results from applying the benchmark to the selected tools
- Comparison and analysis of the performance of different tools
- Insights and findings from the evaluation

A. Research Questions

1) Are the current real-world benchmarks effectively capturing the complex and nuanced features of Python?
2) How well do the existing Python type inference tools perform when tested against TypeEvalPy?
3) To what extent do deep learning-based type inference tools support and handle the complexity of Python's feature set?
4) What insights can be derived from the evaluation of existing analysis tools using TypeEvalPy?

IV. Discussion

- Interpretation of the results and their implications
- Comparison of the results with the research questions
- Discussion on how the results fit into the larger context of Python type inference research

References

[1] M. Schlichtig, A.-K. Wickert, S. Krüger, E. Bodden, and M. Mezini, "CamBench – Cryptographic API Misuse Detection Tool Benchmark Suite," Apr. 2022.