ETH Zürich
Department of Computer Science
ETH Zurich

GUC German University in Cairo
Department of Computer Science and Engineering
German University in Cairo

Bachelor Thesis

A Static Type Inference for Python 3

August, 2017

Author: Mostafa Hassan
Supervisors: Marco Eilers
Dr. Caterina Urban
Prof. Dr. Peter Muller

I confirm that this bachelor thesis is my own work and I have documented all sources and material used.

Zurich, August, 2017 Mostafa Hassan

Acknowledgments

Abstract

Contents

1 Introduction

"The cost to fix an error found after product release was four to five times as much as one uncovered during design, and up to 100 times more than one identified in the maintenance phase.", reported by the System Science Institute at IBM. This fact justifies the increasing investments in software analysis, software verification and the need to make programs more reliable and safe.

In Python, being a dynamically-typed language, the variables are bound to their types during the execution time. This may look appealing because programs have more type flexibility, and they do not need to contain the writing overhead for the type system, leading to shorter and quicker to write code. However, this comes at the cost of losing many static guarantees of program correctness. Dynamically-typed languages perform type checking at runtime, while statically typed languages perform type checking at compile time. Therefore, some type errors that can be detected at compile time in a statically-typed system, may lead the system to crash at runtime in a dynamically-typed one, incurring high costs and a harder debugging experience.

In this thesis, we are presenting a tool for static type inference and static type checking for a subset of Python 3. The aim of the tool is to gain the benefits of static typing while maintaining some (yet not all) dynamic features of Python. We discuss later the details of the dynamic limitations imposed on the inferable Python programs.

The type inference is based on a nominal static type system that we define in later chapters. The type inference is intended to be integrated into Lyra and VerifySCION, two ongoing projects at the Chair of Programming Methodology at ETH Zurich, which aim to develop a static analyzer and a program verifier for Python programs.

1.1 Related Work

Many attempts have been made to infer types for Python, each of which had its own goal and limitations. We discuss here some work that we have studied, and we present some of their limitations and how different they are from our tool.

1.1.1 MyPy

MyPy is a static type checker for Python. It depends on defining type annotations for almost all the constructs in the Python program to be checked. In addition, it performs local type inference. However, this type inference cannot be extended beyond local scopes. It requires that function definitions to be fully type-annotated and cannot infer function calls whose return type annotation is not specified. For example, MyPy will fail to infer the type of variable x in the following program:

```
def f():
        return "string"

x = f() # Infer type Any for x
```

What MyPy intends to provide is closely related to the goal of our tool, that is to provide static type checking for the program. However, we try to reduce the writing overhead in defining the type annotations for program constructs.

2 Background Information

2.1 Dynamic Typing and Static Typing

3 Type System

4 Type Inference

5 Evaluation

6 Future Work

7 Conclusion

List of Figures

List of Tables

Bibliography

[Lam94] L. Lamport. *LaTeX : A Documentation Preparation System User's Guide and Reference Manual*. Addison-Wesley Professional, 1994.