Static Analysis by Abstract Interpretation of Core Python Programs

Internship proposal, Master 2 MPRI, year 2016–2017

Supervisor:	Antoine Miné (`mine@di.ens.fr`)
Internship location:	Laboratoire d'informatique de Paris 6, APR team Université Pierre et Marie Curie, Paris, France
Relevant course:	M2–6: Abstract interpretation: application to verification and static analysis

Note: Other internships are possible on the topic of static analysis, and abstract interpretation in general. Please contact the internship supervisor for more information.

Motivation

The goal of the internship is to develop a sound static analysis by abstract interpretation for a core subset of Python.

Python has become a very popular general-purpose programming language. Its applications include: fast prototyping, application-embedded scripting, web applications, system administration, scientific computing. It is a popular language to learn programming and for use by non computer-scientists.

Python is a very dynamic language, which makes sound static analysis of even the simplest properties difficult. Difficult points include: the lack of static typing for variables, dynamic addition of object attributes, method updates, operator overloading, high-level control structures such as generators, etc. As a simple example, the complex dynamic semantics of Python can make determining whether a variable is initialized before use difficult. Nevertheless, inferring automatically and soundly even such a simple property would be very useful for the Python programmer. While formal methods and analyses have been considered on other popular dynamic languages (notably JavaScript [5]), there are very few works considering Python.

In this internship, we remain in the context of sound analyses. That is, the analysis always output properties that are true of all executions, as defined by a realistic concrete semantics modeling precisely the behaviors of programs. Nevertheless, to achieve this goal, it may be necessary to restrict ourselves to a well-chosen subset of Python.

Expected work

The expected work will consist in: choosing an adequate core subset of Python and its concrete semantics, choosing a class of properties of interest, designing a static analysis by abstract interpretation, implementing it in OCaml, and evaluating it on small Python programs.

We suggest basing the Python subset and its semantics on the work of [1, 2]. The language may need to be further restricted in order to achieve soundness, for instance by removing the `eval` construction (but consider also [4]).

We will focus mainly on one or more of the following simple properties:

- initialization analysis (is a variable initialized before use);
- flow-sensitive type analysis (which types of values a variable can contain at each program point);
- object attribute analysis (which attributes an object may contain);
- control-flow analysis (which code is executed by a method call);
- numeric invariants.

Generally, the analyses for these properties are not independent. For instance, in order to apply a classic numeric invariant analysis, it is first necessary to perform a type analysis able to infer that a variable is effectively numeric and an arithmetic operator such as `+` has its usual menaing (i.e., it is not overloaded). More generally, a possible approach is to design novel Python-specific analyses that enable soundly exploiting existing analyses designed with more static languages (such as C or Java) or different dynamic languages (such as Javascript) in mind.

Requested skills

In addition to a good knowledge of static analysis by abstract interpretation, the intern is expected to have some familiarity programming in Python and in OCaml.

Context of the Internship

The internship will take place in the APR team, in the LIP6 laboratory at Paris 6. It is proposed in the scope of the MOPSA EU research project. If the internship is successful, the project will provide opportunities for a funded PhD on a follow-up subject.

References

[1] Gideon Smeding. An Executable Operational Semantics for Python. Master's Thesis, Universiteit Utrecht, 2009.

[2] Joe Gibbs Politz, Alejandro Martinez, Matthew Milano, Sumner Warren, Daniel Patterson, Junsong Li, Anand Chitipothu, and Shriram Krishnamurthi. Python: The Full Monty: A Tested Semantics for the Python Programming Language. In *ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages & Applications (OOPSLA)*, 2013.

[3] Yin Wang. PySonar2 – A type inferencer and indexer for Python `https://github.com/yinwang0/pysonar2`

[4] Simon Holm Jensen and Peter A. Jonsson and Anders Møller. Remedying the Eval that Men Do. Proc. 21st International Symposium on Software Testing and Analysis (ISSTA), 2012.

[5] Simon Holm Jensen and Anders Møller and Peter Thiemann. Type Analysis for JavaScript. Proc. 16th International Static Analysis Symposium (SAS), vol. 5673 of LNCS, Springer, 2009.