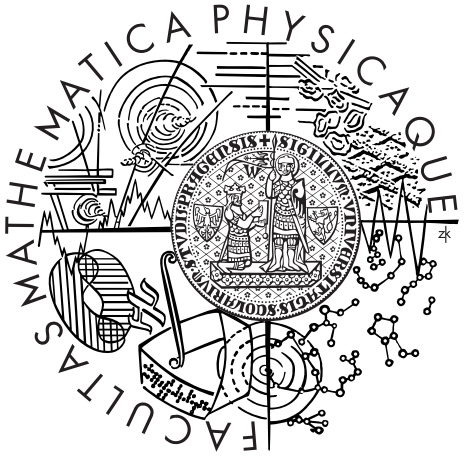
Charles University in Prague
Faculty of Mathematics and Physics

MASTER THESIS



Štěpán Šindelář

Implementing control flow resolution in dynamic language

Department of Software Engineering of the Charles University in Prague

Supervisor of the master thesis: Filip Zavoral, Ph.D.
Study programme: Software Systems
Specialization: specialization

Prague 2014

Dedication.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In ..... date ..... signature of the author

Název práce:

Autor: Bc. Štěpán Šindelář

Katedra: Katedra softwarového inženýrství Univerzity Karlovy

Vedoucí diplomové práce: Filip Zavoral, Ph.D., TODO: pracoviště

Abstrakt:

Klíčová slova: dynamické programovací jazyky, statická analýza, PHP, Phalanger, .NET

Title: Implementing control flow resolution in dynamic language

Author: Bc. Štěpán Šindelář

Department: Department of Software Engineering of the Charles University in Prague

Supervisor: Filip Zavoral, Ph.D., TODO: pracoviště

Abstract:

Keywords: dynamic programming languages, static analysis, PHP, Phalanger, .NET

Contents

1. Introduction

1.1 Problem Description

Static type analysis of a dynamic language, benefits: compiler optimizations, IDE support - bug hunting analysis.

1.2 Implementation Constraints

Use Phalanger front end, provide support for compiler back end and IDE (re-analysis), focus on object oriented PHP 5 style projects.

1.3 Thesis structure

2. Static Analysis

More detailed description of what static analysis is (as opposed to for example explicit model checking, verification, etc.). Terminology: context sensitive, path sensitive, etc.

2.1 Data Flow Analysis

2.1.1 Abstract Interpretation

Detailed description of Data Flow analysis or just reference to the literature in the previous chapter and this chapter would be left out.

2.1.2 Rapid Analysis

2.2 The PHP Programming Language

Note: maybe a new chapter?

What makes static analysis in PHP harder than of statically typed language

what are the caveats of PHP we had to deal with

PHPDoc annotations and their significance for type analysis (especially for the bug hunting)

3. Existing Software

This project can be compared to other projects on more levels:

- RPython, Ecstatic (Ruby, [1]): comparison to approaches used for type analysis in other dynamic languages, e.g. Python, Ruby. Probably more interesting from the academic point of view.
- Weverca [2], Pixy[3]: comparison to other tools that analyse PHP, e.g. Weverca, Pixy, but those tools focus on different aims, namely security vulnerabilities through user input, XSS attacks, etc.
- NetBeans, Zend Studio: comparison to real world tools that analyze PHP for the same type of bugs (unused variable, use of unassigned variable, etc.), but those tools usually do not provide any documentation and any comparison would have to be based either on benchmarking and evaluation or on detailed study of their source code, which would be time consuming.
- HipHop: its type analysis is meant only for compiler back end, so it tries to find only one type for a variable, if that fails, it does not continue with analysis of that variable. (As far as I understood from the source codes, because there are no articles or documentation).
- Phantm [4]: very similar project, but uses Scala. What PHP versions it supports? Intra-procedural? Integration into a compiler/interpreter.

4. Implementation

specific constraints:

must use Phalanger front end and its AST data structures,

should be ready to be connected in between the Phalanger front end and back end as a middle end,

should be ready to be used continuously: interactively re-analyze updated code when used inside IDE,

overall description:

Control Flow graph construction

discussion of the choice of intermediate representation (or in fact, why we do not use any intermediate representation).

generic framework for Data Flow analyses

type analysis

5. Results

Analyses of well known OSS PHP projects – few examples of discovered errors plus tables with data (as an appendix?). Possibly comparison to other tools (NetBeans, Zend Studio).

6. Conslusion

6.1 Future Work

Branched Data Flow analysis, integer interval analysis, integration with the compiler back-end.

Bibliography

[1] M. Madsen, P. Sørensen, and K. Kristensen, *Ecstatic–type inference for Ruby using the cartesian product algorithm*. PhD thesis, Master Thesis, Jun, 2007.

[2] D. Hauzar and J. Kofron, "Hunting bugs inside web applications,"

[3] N. Jovanovic, C. Kruegel, and E. Kirda, "Pixy: A static analysis tool for detecting web application vulnerabilities," in *Security and Privacy, 2006 IEEE Symposium on*, pp. 6–pp, IEEE, 2006.

[4] E. Kneuss, P. Suter, and V. Kuncak, "On using static analysis to detect type errors in php applications," tech. rep., 2010.

List of Tables

Attachments