Interfacing TVLA and SAMPLE

Problem Description

Raphael Fuchs
fuchsra@student.ethz.ch

February 21, 2011

Background

TVLA [1] is a framework to define, implement and execute shape analyses. Shape analysis allows to statically infer properties about heap-allocated data structures, such as reachability of heap cells through pointers. Over the past 10 years, several shape analyses have been defined and implemented in TVLA. The results of this scientific work have often been published at the best conferences.

Sample (Static Analysis of Multiple LanguagEs) is a generic static analyzer based on the abstract interpretation theory, developed at the Chair of Programming Methodology at ETH in the last 2 years. It supports and already contains a wide range of different analyses and can be used for multiple programming languages.

Core Task

The goal of this bachelor thesis is to extend Sample with more powerful heap analysis capabilities. This will be achieved by interfacing it with TVLA.

The implementation will consist of an infrastructure to communicate with the TVLA tool. In particular, this will mean to implement a parser for tvs files (these are the files used by TVLA to represent the shape of the heap) and to represent these heap data structures in Sample. Furthermore, a set of *update formulae* which describe the effect of executing program statements on heap structures needs to be provided in order to fully exploit the power of the TVLA engine. These formulae will represent the semantics of the operators required by Sample (e.g., field access and field update).

[1] Three-Valued-Logic Analyzer, http://www.cs.tau.ac.il/~tvla/

Possible Extensions

Depending on the time left, possible extensions of this work may include the following:

- TVLA's abstraction of program semantics is guided by *instrumentation predicates*. The right choice of these predicates is crucial in order to obtain precise results. Therefore, various predicates should be evaluated, possibly balancing efficiency with precision.
- Apply the analysis to a wide set of benchmarks
- Develop a set of common predicates to specify more precise analyses of the most common data structures (such as lists, trees and DAGs).