Strong Support for Pure Functions in an Automated Program Verifier

Project Plan

Benjamin Fischer

17 March 2013

Bachelor thesis: 11 March 2013 to 1 September 2013
Supervisor: Dr. Ioannis Kassios

ETH Computer Science
Prof. Dr. Peter Müller

1 Project description

1.1 Overview

The Semper Intermediate Language (abbreviated to SIL) is a verification language for concurrent programs. SIL offers functions and predicates in the strict sense, that is, they are free from side effects and thus also called pure functions. Central to formal verification are assertions, which in SIL can be abstracted with the help of functions. A verifier for SIL is Silicon. The goal of this project is to improve the reasoning of Silicon about functions.

1.2 Scope of the work

1.2.1 Core

- Two common approaches for the construction of automatic program verifiers are Verification Condition Generation (VCG) and Symbolic Execution (SE). Experiments suggest that SE is generally more efficient than VCG, but less complete [2]. SE simulates a program execution with symbolic values, accumulating them in a logical formula known as the path condition, which is given to the prover whenever an assertion needs to be verified. Silicon is based on SE. We plan to change Silicon by additionally giving the prover relevant functions as axioms, an idea taken from VCG. Since this provides the prover with stronger assumptions, it should improve the completeness of the verifier.
- To strengthen the ability of Silicon to prove partial correctness of functions, it should be extended with a technique based on the induction principle [3]. There might be a need to make SIL more expressive, for instance to give the verifier a hint about whether to apply an inductive proof.

1.2.2 Extensions

- Verfiers translate functions into axioms, thereby facilitating the verification. If recursive functions are represented as recursive axioms, then they might be instantiated indefinitely. Being unable to terminate, the prover enters a so-called matching loop. Recent research proposes a solution for VCG [1]. We could integrate it into Silicon.
- The termination of certain recursive functions should be proven, namely if the function provides a well-founded termination measure on its arguments.

1.3 Intended results

- A test suite should be developed with program examples covering the problems.
- The improved verifier should do well on the test suite.
- A report in English should be completed. It is to introduce the concepts around SIL, document the techniques incorporated into Silicon and evaluate the changes.

2 Project management

2.1 Objectives and priorities

The main objective is to improve the completeness of the verifier. It should be sound, so any verified assertion must be correct indeed. The program examples form a basis for the experimental evaluation.

2.2 Criteria for success

A chosen subset of the program examples are required to be verified without a specific time constraint. Together with the completion of the report and the two presentations, this guarantees the lowest passing grade 4 or more.

2.3 Method of work

At the beginning of the project, there are weekly meetings with the supervisor to identify problems early. If desirable, this frequency can be loosened later to once a fortnight.

Two presentations should take place. The initial presentation is to be given about one month after the start of the project. The intention is to inform the research group, receive fruitful feedback and exchange thoughts related to the topic. The final presentation at the end of the project should summarise the whole project.

2.4 Quality management

2.4.1 Documentation

The report should document the techniques realised in the verifier on an abstract level. The source code should be documented by employing a good programming style.

2.4.2 Validation steps

The software should be validated with code reviews and the test suite.

3 Plan with milestones

	Time / weeks		
Project step	Budget	Accumulated	Deadline
Ideas and studies completed	1	1	2013-03-17
Background material collected	1	2	2013-03-24
Program examples created	1	3	2013-03-31
Milestone 1: requirements completed	0	3	2013-03-31
Design completed	2	5	2013-04-14
Initial presentation given	1	6	2013-04-21
Implementation completed	7	13	2013-06-09
Time buffer	2	15	2013-06-23
Validation completed	1	16	2013-06-30
Experimental evaluation completed	1	17	2013-07-07
Milestone 2: software completed	0	17	2013-07-07
Report completed	5	22	2013-08-11
Final presentation given	1	23	2013-08-18
Time buffer	2	25	2013-09-01
Milestone 3: project completed	0	25	2013-09-01

References

[1] Stefan Heule et al. *Verification Condition Generation for Permission Logics with Abstract Predicates and Abstraction Functions*. Technical report. ETH Zurich, 2012. URL: http://pm.inf.ethz.ch/publications/getpdf.php?bibname=Own&id=HeuleKassiosMuellerSummers12.pdf.

[2] Ioannis T. Kassios, Peter Müller, and Malte Schwerhoff. "Comparing Verification Condition Generation with Symbolic Execution: an Experience Report". In: *Proceedings of the 4th international conference on Verified Software: theories, tools, experiments*. Ed. by Rajeev Joshi, Peter Müller, and Andreas Podelski. 2012. URL: http://pm.inf.ethz.ch/publications/getpdf.php?bibname=Own&id=KassiosMuellerSchwerhoff12.pdf.

[3] K. Rustan M. Leino. "Automating Induction with an SMT Solver". In: *Proceedings of the 13th international conference on Verification, Model Checking, and Abstract Interpretation*. Ed. by Viktor Kuncak and Andrey Rybalchenko. 2012. URL: http://research.microsoft.com/en-us/um/people/leino/papers/krml218.pdf.