Loop invariant inference from postconditions in EVE

BACHELOR THESIS PROJECT PLAN

Project period: 29.10.2012 – 02.04.2013

Student name: Michael Ameri

Email address: mameri@student.ethz.ch

Supervisor name: Carlo A. Furia

PROJECT DESCRIPTION

Overview

Loop invariants are an indispensable ingredient of the correctness proofs of programs. At the same time, annotating a loop with suitable invariants is often a non-trivial task that resists complete automation. The availability of postconditions can help in this regard, as loop invariants are often expressible as mutations of the loop's postcondition. The goal of this project is implementing a loop invariant inference technique based on *postcondition mutation*.

Scope of the work

The idea of relating a loop's invariant to its postcondition originates in the classic work on refinement and constructive program development [1,2,3]. Recently, this idea has been applied [4] to the reverse problem of inferring a suitable loop invariant given an implementation annotated with postcondition. The basic idea of such "postcondition mutation" approach is to take a routine's postcondition, mutate it according to several heuristics, and generate a set of candidate loop invariant formulae. Using a static program prover such as Boogie [5], one can check if some of the postcondition mutations are correct invariant of the loop and retain those that pass the check.

This thesis work will implement this postcondition mutation approach for Eiffel programs, by integrating it within AutoProof [6], Eiffel's static program prover available as part of EVE [7] (the research branch of the EiffelStudio IDE). AutoProof already provides an automatic translation from Eiffel to Boogie. The implementation of postcondition mutation will rely on AutoProof's translation and will be integrated within EVE's GUI to provide user suggestions for loop invariants.

Intended results

Completion of the project will provide an integrated solutions consisting of the following components:

1. An implementation of the postcondition mutation algorithm integrated with AutoProof. AutoProof works by translating Eiffel programs to Boogie. It does so by maintaining an intermediate representation, which simplifies the Eiffel source code and is directly translatable to Boogie code. The implementation of the postcondition mutation algorithm will mainly work at the level of this intermediate representation to take advantage of the rest of AutoProof's features.

2. A simple GUI interface integrated within EVE.
   Through the GUI, users will be able to invoke the postcondition mutation algorithm on a specific routine or loop. They will receive back a list of suggestions for loop invariants. They will be able to select some (or all) of the suggestions, which will then be injected into the Eiffel source code as **invariant** clauses. The GUI will also allow users to select specific heuristics to be used to mutate postconditions.
3. A succinct API offering the same services as the GUI in item 2.
   For further developments and integration with other components of the environment, the same basic services accessible through the GUI (invocation of postcondition mutation on a specific routine, list of invariant suggestions, injection of invariants into the Eiffel code, etc.) will also be made available through a simple API.
4. An experimental demonstration of the final implementation based on a series of examples.
   A prototype implementation of the postcondition mutation approach [8] works directly on Boogie files and has been tested with a number of examples (mostly consisting of algorithms on arrays). The evaluation of the new implementation in EVE will be tested on as many examples from [8] as possible. It may also introduce new demonstrator examples that exercise specific features of the new implementation.
5. The final project report.

BACKGROUND MATERIAL

Reading list

Carlo A. Furia and Bertrand Meyer. **Inferring Loop Invariants Using Postconditions**. In *Fields of Logic and Computation: Essays Dedicated to Yuri Gurevich on the Occasion of His 70th Birthday*. *Lecture Notes in Computer Science*, 6300:277-300, Springer, August 2010.

K. Rustan M. Leino. **This is Boogie 2**. Manuscript KRML 178.
http://research.microsoft.com/en-us/um/people/leino/papers/krml178.pdf

Julian Tschannen. **Automatic verification of Eiffel programs**. Master's thesis. ETH, April 2009.
http://se.inf.ethz.ch/old/projects/julian_tschannen/report.pdf

Julian Tschannen, Carlo A. Furia, Martin Nordio, and Bertrand Meyer. **Usable Verification of Object-Oriented Programs by Combining Static and Dynamic Techniques**. In *9th International Conference on Software Engineering and Formal Methods (SEFM'11)*. *Lecture Notes in Computer Science*, 7041:382--398, Springer, November 2011.

PROJECT MANAGEMENT

Objectives and priorities

The primary objectives are:

- The implementation of the postcondition mutation approach within EVE.
- The demonstration of its functionalities on examples (item 4 in *Intended results* section).

Criteria for success

The overall success will be assessed as follows:

1. It should be possible to carry out a demonstration of the final implementation, targeting the examples mentioned in item 4 of the *Intended results* section.

2. The final implementation should be usable through EVE's GUI by users familiar with Eiffel and the notion of loop invariants, but without detailed knowledge of the postcondition mutation approach (except possibly for selecting which mutation heuristics should be applied). That is, the postcondition mutation implementation should work transparently as a black-box mechanism for providing loop invariant suggestions.

Method of work

The postcondition mutation algorithm will be implemented in Eiffel and integrate with the rest of EVE. The implementation should reuse the existing components of EVE (and especially AutoProof) as much as possible, focusing on the implementation of the new functionality. The prototype stand-alone implementation of the postcondition mutation technique [8] can be used as reference but need not be directly reused within EVE.

Quality management: documentation

The project documentation consists of the final report which will include:

1. A description of the salient features of the implementation, in particular pointing out the features where it diverges from the high-level description in [4].
2. Documentation of the implemented API (based on the comments and contracts whenever appropriate, but possibly also including other informal descriptions).
3. A brief tutorial that explains how the postcondition mutation is used in EVE through the GUI.
4. A description of experiments with the examples (item 4 in *Intended results* section), discussing in detail which mutation heuristics have been applied in each case and also possible failed attempts.

PLAN WITH MILESTONES

Project steps

1. Become familiar with AutoProof's principles and general architecture in EVE, the Boogie language and prover (and its command-line interface), and the postcondition mutation approach.
2. Implement the postcondition mutation approach using AutoProof's features, including a mechanism to propagate back the inferred loop invariants into the source Eiffel code.
3. Develop a GUI interface within EVE for the postcondition mutation implementation.
4. Experiment with examples to demonstrate that the overall implementation works and to assess its strengths and limits.
5. Write the project report.

Deadline

The deadline for the project is 02.04.2013.

Tentative schedule

November	December	January	February	March

Step/ week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1.																				
2.																				
3.																				
4.																				
5.																				

References

[1] E.W. Dijkstra. **A Discipline of Programming**. *Prentice Hall*, 1976.

[2] D. Gries. **The science of programming**. *Springer-Verlag*, 1981.

[3] B. Meyer. **A basis for the constructive approach to programming**. In *Proceedings of IFIP Congress 1980*. pp. 293-298, 1980.

[4] C. A. Furia and B. Meyer. **Inferring Loop Invariants Using Postconditions**. In *Fields of Logic and Computation: Essays Dedicated to Yuri Gurevich on the Occasion of His 70th Birthday*. *Lecture Notes in Computer Science*, 6300:277-300, Springer, 2010.

[5] Microsoft Research Boogie. http://boogie.codeplex.com/

[6] J. Tschannen. **Automatic verification of Eiffel programs**. Master's thesis. ETH, 2009. http://se.inf.ethz.ch/old/projects/julian_tschannen/report.pdf

[7] **EVE**: the Eiffel Verification Environment. http://se.inf.ethz.ch/research/eve/

[8] **Gin-Pink**: loop invariants from postconditions.
http://se.inf.ethz.ch/people/furia/software/gin-pink.html