Incremental Symbolic Execution
- Project Description -

Roger Koradi
Supervisors: Malte Schwerhoff, Valentin Wüstholz, Peter Müller
ETH Zürich, Switzerland

Wednesday 1st April, 2015

1 Motivation

Most often, programs under verification are changed incrementally, i.e. they gradually evolve to implement the desired functionality. Therefore, most parts of the program are likely to be unaffected by some code changes. It would be wasteful, for example, to re-verify the entire program just because the programmer appended one more statement.
The Silicon verifier—as part of the Viper infrastructure[1]—is supposed to be constantly running in the background during IDE sessions, providing periodic feedback to the programmers as they are editing the code. As such, response time is crucial to IDE integration. This is why we will optimise Silicon's responsiveness and performance by reducing the amount of code that must be re-verified after changes by caching and re-using earlier verification results where appropriate.

We will be adopting a caching technique that was developed for the Dafny IDE and its Boogie backend. The key challenge in this is generalising and extending the solution to the new setting of the permission-based symbolic-execution tool Silicon. This might also allow for several optimisations in the caching.

[1] http://www.pm.inf.ethz.ch/research/viper

2 Core Goals

Identify and implement ways of re-verifying as little as possible after code has been changed. As of now, we will use two such approaches:

Coarse-grained caching refers to deciding which methods are affected by current changes in code and which methods can be assumed to not need re-verification. This will presumably require the implementation of checksums of some kind, two in total, one that will change if the method itself is modified and one that will change if a callee's specification is. A change in either checksum may indicate the need for re-verification of the method. Note that the computation of checksums may also need to deal with changes to other global entities in Silver, such as domains, axioms and fields. To attach these checksums to methods and functions, we will add support for attributes on methods and functions.

Fine-grained caching means taking into account that we may not need to re-verify the entirety of a method's implementation if we can prove parts of it were not affected by the change that triggered the re-verification. For example, contract changes to a callee may be harmless for the caller if they strengthen a postcondition or weaken a precondition.

The existing technique for fine-grained caching makes it possible to mark assertions as fully-verified or as partially-verified under certain conditions that usually depend on whether callees' postconditions from the cached version of the program still hold in the current version. We will thus add support for attributes on assertions to so mark them, allowing Silicon to later simplify their checking if these conditions hold.

Expressing the conditions will require checking if an "old" postcondition still holds after a call to the new, changed callee. Unlike in Boogie, this is more involved in a setting with postconditions that mention permissions and will require support for an "exhalable" construct that will allow us to check if a given condition—possibly containing permissions—holds at a given program point. Note that this is different from the existing "exhale"-statement which checks that a given condition holds and gives up any permissions the condition requires.

We believe the technique is beneficial not only to the simplification of the checking of assertions but to statements in general. For, unlike in Boogie, some statements in Silver require implicit checks during verification. For instance, the receiver of a method invocation must not be null or the index during an array access must be within a valid range.

Therefore, the project will:

- Add language support for statement, method and function attributes, for the "exhalable" construct and for partially-verified statements.
- Implement and evaluate effectiveness of coarse-grained caching.
- Implement and evaluate effectiveness of fine-grained caching.

3 Extension Goals

- Integrate caching infrastructure into the IDE.
- If the simplified checking for assertions and exhale statements yields a benefit, extend it to work for all partially-verified statements.
- Evaluate effectiveness over real IDE sessions.
- Add support for new language features in Carbon.

4 Timeline

Start: Monday, 23rd February 2015
Deadline: Sunday, 23rd August 2015

19th	March	Project Presentation
11th	May	Implemented language support for statement/method/function attributes, for the "exhalable" construct and for partially-verified statements, as well as method/function entity & dependency checksums and evaluated the merit of simplified checking for assertions and exhalable statements
25th	May	Implemented coarse-grained caching
1st	June	Evaluated coarse-grained caching
15th	June	Implemented fine-grained caching
22nd	June	Evaluated fine-grained caching
29th	June	Extension goals
6th	July	Start writing report
23rd	August	deadline