Software Verification (Fall 2011)
Course Project

Hand-out date: 17 October 2011
Due date: 28 November 2011

1. Project description

The goal of this project is to specify and implement a well-known, non-trivial algorithm and prove its functional correctness using a program verifier. Depending on the chosen algorithm and on your personal preferences, you may either use the Boogie language and tool [1], or C/Java and the VeriFast tool [2].

You are free to choose the algorithm to implement and prove correct. However, the assistant (Stephan) must approve your choice of algorithm before you hand in the project. Not all algorithms will be equally easy to implement and prove correct, and the difficulty level (*Standard* or *Advanced*) will influence the project's grading (see Section 3).

To give you an idea of what may be the difficulty level of an algorithm, consider the following examples:

- *Standard*: most array-based sorting algorithms such as *insertion sort* [3], recursive algorithms on trees;
- *Advanced*: *topological sort* [4], algorithms on doubly-linked lists, iterative algorithms on trees.

The project can be done individually or in groups of up to three students.

Please submit the source code of your verified program (a zip file) and your project report (a pdf file) by email to `Stephan.vanStaden@inf.ethz.ch`.

2. Tools

If you haven't done so yet in the exercise sessions, you may download the Boogie tool from the URL given in [1], and the VeriFast tool from [2].

3. Grading

The final mark will depend on the following criteria:

Implementation (up to 20 points)

- Quality of the implementation (5 points)
- Accuracy and completeness of the proved specification (15 points)

Difficulty level: The points obtained for the implementation are scaled by a factor determined by the difficulty level of the chosen algorithm:

- *Standard*: factor of 0.8 (i.e., max. 16 points)
- *Advanced*: factor of 1 (i.e., max. 20 points)

Project report (10 points)

- Description of the algorithm and your implementation (2 points)
  - What other implementations of the same algorithm could you have used?
  - What are the motivations for your particular implementation choices?
- Discussion of your proved specification (4 points)
  - Which other specifications could you have used?
  - Why did you choose your particular specification?
  - Is your specification complete or there are aspects of correctness that it does not cover?
- Discussion of your experiences of proving programs using the chosen tool (2 points)
  - Please report:
    - the settings/command-line options you used with your program
    - the configuration of the PC you tried it on (processor, amount of RAM, etc.)
    - the verification time on your PC
  - Where do you see major difficulties using the tool?
  - Did you have to tweak your original design to make it amenable to the tool?
    - If yes, which aspects and why?
  - Compare the chosen approach to other approaches and proof tools.
- Structure, grammar and spelling of the report (2 points)

4. Support

You can ask questions about the project at the exercise sessions on Mondays. Additionally, you can arrange a meeting with the assistant (please send requests to `Stephan.vanStaden@inf.ethz.ch`).

References

[1] *Boogie: An Intermediate Verification Language*. `http://research.microsoft.com/en-us/projects/boogie/`
[2] *VeriFast*. `http://people.cs.kuleuven.be/~bart.jacobs/verifast/`
[3] Wikipedia. *Insertion sort*. `http://en.wikipedia.org/wiki/Insertion_sort`
[4] Wikipedia. *Topological sorting*. `http://en.wikipedia.org/wiki/Topological_sorting`