ETH zürich

Alexander J. Summers

Program Verification

Exercise Sheet 12: Permissions and Concurrent Programs

Assignment 1 (Encoding Non-Determinism)

We've already seen one way to encode a `havoc x` statement in Viper: by calling an abstract method which returns the appropriate type (slide 274). It might be tempting to simulate a `havoc x` statement by adding additional *parameters* to the enclosing method (without any constraints in the precondition, these parameters will have unknown values, which could then be assigned to e.g. the `x` at the point of the intended `havoc`). This has the obvious disadvantage that a caller of the current method will have to provide values for these parameters. Ignoring this problem, the approach is also insufficient for reflecting the correct behaviour of `havoc x` statements, in general.

1. How many extra parameters would be needed, to eliminate the `havoc` statements from a given method body?
2. Why does this mean that *some* method bodies could not be handled by this approach?
3. Give a different approach for encoding a `havoc x` statement, using `inhale` and `exhale` operations.
4. Does your approach suffer from the same problems?
5. Show how to encode a non-deterministic choice statement $s_1 \, [] \, s_2$, using your ideas.

Assignment 2 (Graph Marking)

On the course webpage you can find a file `GraphMarking.vpr` that contains an encoding of a recursive graph marking algorithm. The encoding is incomplete: all quantifiers in the encoding are missing triggers. Please complete the example by specifying the missing triggers and answer the following questions:

1. Why do we need to explicitly ensure that the nodes are not modified? Can you think of a different way of ensuring this property, other than the one used in the example?

2. Under which conditions is the method `trav_rec` guaranteed to terminate? How would you informally justify that? Does the method precondition already require the necessary conditions, or would you need to explicitly write them?