ETH zürich

Alexander J. Summers

Program Verification

Exercise Sheet 11: Unbounded Heap Data Structures

Assignment 1 (List Segments)

Consider the `lseg` predicate from Slide 242.

1. Write an assertion using this predicate to describe a cyclic list.
2. The `lseg` predicate does not allow for "empty" segments; permission to the `start` reference's fields is always contained inside. Write a definition for an alternative `mylseg` predicate, which allows for possibly-empty list segments.
3. Can you use `mylseg` to describe cyclic lists?
4. Consider the `addAtEnd` and `prependLseg` methods, which were needed in order to verify the iterative version of list append, in the lectures. These methods were not implemented in the lecture; you can find the appropriate file `list-examples.vpr` on the course webpage, in which the method declarations are included, but no bodies. Implement these methods, such that your resulting code verifies.

Assignment 2 (Heap-based Matrices)

On slide 256, a Viper encoding of arrays is shown, using a custom domain and quantified permission assertions. Suppose that we want to implement a similar encoding for heap-based *square matrices*: a special case of two-dimensional arrays.

1. Write a corresponding `Matrix` domain definition (you might want to borrow ideas from the `Array` domain).
2. What assertion would you use to describe full permission to all elements of a particular matrix? (Hint: it should involve two quantifiers).
3. The current version of Carbon (one of the Viper verification back-ends) does not support quantified permissions under multiple (nested) quantifiers; only single quantifiers are supported. Describe an alternative representation of matrices which requires only a single

quantifier, using the original `Array` domain. Can you think of any practical disadvantages of this encoding using single quantifiers, compared to the more-direct two-dimensional quantification? (Hint: there may be more than one problem).