ETH zürich

Alexander J. Summers

Program Verification

Exercise Sheet 9: Heap Reasoning and Permissions

Assignment 1 (Pure Assertions)

An IDF assertion is called *pure* if doesn't contain any accessibility predicates. For example, $x.f = 4$ is a pure assertion. In practice, pure assertions are typically expressions, but can technically also include the separating conjunction $*$.

1. Prove that, for all states H, P, σ and pure assertions A, if $H, P, \sigma \vDash A$ then $H, \varnothing, \sigma \vDash A$ (where $\varnothing$ represents the empty permissions mask, which maps all field locations to 0).
2. Prove that, for any assertion A' and any pure assertion A the assertions $A * A'$ and $A \wedge A'$ are logically equivalent (i.e. they are true in the same states)[1].

Assignment 2 (Old Expressions and Procedure Calls)

Suppose that we include procedures into our small language. When using IDF for our small language, the global (shared) state consists only of the heap itself (we don't support global variables). Just as in Boogie, it is convenient for procedure specifications to be able to relate the values of this state before and after a procedure call. In IDF, we again have *old expressions*, but here these affect *heap-dependent expressions*. For example, the meaning of an expressions *old*(x.val) is the value that the heap location *had* in the pre-state of the current procedure.

A procedure which increments the val field of its parameter, could, for example, be specified as follows:

```
procedure inc(x)
  requires acc(x.val)
  ensures acc(x.val) * x.val==old(x.val)+1
{
  x.val := x.val + 1
}
```

[1] Note that, as a simple consequence of this latter result, any pure assertion can be equivalently represented as a (boolean-typed) expression.

1. Write an appropriate procedure (including a specification) which takes two parameters and assigns the value of the `val` field of the second parameter to the `val` field of the first.

2. In the absence of global variables, we don't need Boogie-style modifies clauses. Why are these not needed in IDF for reasoning about the heap? Does your procedure specification in the previous part precisely characterise which heap locations might be modified by the procedure?

3. What might an appropriate Hoare Logic procedure call rule be, in this setting?