Matching Logic

Software Health Management TIM
November 19, 2009

Grigore Rosu
Charles Ellison
Wolfram Schulte

Matching Logic

- An alternative to Hoare logics in which the state structure plays a crucial role
- States represented as algebraic types called configurations; state specifications are represented as configuration terms with variables and constraints called patterns
- Can reason about traditional correctness properties as well as heap properties, so only one verifier needs to be created for each language
- Logic separate from underlying state config., as long as it is expressible algebraically

Comparison With Hoare Logics

- Similarities
  - Specifies program states as logical formulae and gives an axiomatic semantics to a programming language in terms of pre- and post-conditions
  - Generically extended to a formal, syntax-oriented compositional proof system
- Differences
  - Configurations not flattened to arbitrary first order logic (FOL) formulas; instead they are kept as symbolic configurations (restricted $\mathrm{FOL}_{=}$ formulae)

Continued...

Comparison With Hoare Logics

- Differences (continued)
  - Pre- and post-conditions are patterns over configurations, possibly containing both free and bound variables
  - A configuration matches a pattern if it is obtained as an instance of the pattern
  - Matching logic achieves heap separation without having to extend the logic with special connectives; e.g., the very fact that one can match two trees in a heap means, by definition, that the two trees are separate

Reverse Example

```
//@ assume a != null && [list(seq)(a) ** rest] ;
x = a ;
y = *(a + 1) ;
*(x + 1) = null ;
//@ inv [list(?sx)(x) ** list(?sy)(y) ** ?frame]
      && reverse(seq) == reverse(?sy) :: ?sx
while (y != null) {
    t = *(y + 1) ;
    *(y + 1) = x ;
    x = y ;
    y = t
} ;
result = x ;
//@ assert [list(reverse(seq))(result) ** rest]
```

Our Results

- Practical
  - Can derive Matching Logic (ML) verifiers from algebraically defined language semantics
  - Have executable verification tool for a subset of C with which we automatically verified Schorr-Waite graph marking algorithm (and many more!)
- Theoretical
  - Shown a correspondence between Hoare Logic and (a limited version of) ML for various languages
  - Soundness of the verifier w.r.t. language semantics
  - Soundness and completeness of verifier w.r.t. ML

Future Work

- The theory is basically complete. What is left is to provide stronger tools based on the theory
    - Verification tool for a language people use—C
    - Automated/assisted tools for deriving verifiers from formal specifications of languages
- Collection of programs used to compare the efficacy of different verification tools (a program verification benchmark)