Holfoot

Thomas Tuerk

26th January 2010

Separation Logic

- Separation logic is an extension of Hoare Logic
- successfully used to reason about programs using pointers
- allows local reasoning, scales nicely
- there are some implementations
  - Smallfoot (Calcagno, Berdine, O'Hearn)
  - Slayer (MSR, B. Cook, J. Berdine et al.)
  - Space Invader
  - ...
- there are formalisation in theorem provers
  - *Concurrent C-Minor Project*, Coq (Appel et al.)
  - *Practical Tactics for Separation Logic* (McCreight)
  - *Types, Bytes, and Separation Logic*, Isabelle/HOL (Tuch, Klein, Norrish)

Motivation

- I developed a general separation logic framework inside HOL 4
- special interest in generality and easy reuse
- 3 levels of abstraction
  - abstract separation logic (Calcagno, O'Hearn and Yang)
  - variables as resource (Parkinson, Bornat and Calcagno)
  - **Holfoot**, a tool similar to Smallfoot
- this talk: high level presentation of Holfoot

Examples - Webinterface I

- `list_length.sf`
- `copy.sf`

Features - programming language

- simple, low-level imperative toy language
- shared by Smallfoot and Holfoot
- features
  - variable assignments / pure expressions
  - heap lookups / assignments
  - conditional execution
  - while loops
  - explicit allocation / deallocation of heap cells
  - local and global variables
  - mutually recursive procedures with call-by-reference and call-by-value parameters
  - parallel procedure calls
  - conditional critical regions
  - ...

Features - specification language

- interested in partial correctness, i. e. termination is not proved
- Hoare style procedure specifications
- there are predicates for
  - conditions on variables
  - single heap cells (points to)
  - single linked lists
  - trees
  - existentially quantified expressions
  - ghost variables
  - ...

Features - comparison

- Holfoot supports
  - explicit read- / write-permissions
  - data content of lists and trees
  - user defined side-conditions
- Holfoot lacks in comparison to Smallfoot
  - XOR- and double-linked-lists
  - reasoning about resource invariants that share variables with other parts of the program
- Holfoot is implemented inside HOL 4
  - formal semantics of programming language and specifications
  - everything is done by proof

Examples - extended demo

Loop invariants

- using loop invariants doesn't exploit local reasoning
- each iterations has to work on the part of the state
- this leads to complicated specifications using partial data-structures like list-segments and existential quantification
- trees are very hard to handle
- **proposed solution**: unroll loops / introduce new functions

```
while c {
   prog1;
}
prog2
```

==>

```
newfun () {
  if c then {
     prog1; newfun();
  } else {
    prog2;
  }}
...
newfun();
```

Future work

- add support for arrays
- add some more examples
- finish my PhD