A Deep Embedding of a Decidable Fragment of Separation Logic in HOL

Thomas Tuerk
University of Cambridge, Computer Laboratory

UNIVERSITY OF CAMBRIDGE

Separation Logic

Separation logic is an extension of Hoare logic that allows local reasoning about mutable data structures. It's been introduced by O'Hearn, Reynolds and Yang in 2001 [3, 4].

$$\frac{\{P\}\ \mathit{prog}\ \{Q\}}{\{P * R\}\ \mathit{prog}\ \{Q * R\}} \qquad \frac{\{P_1\}\ \mathit{prog}_1\{Q_1\} \qquad \{P_2\}\ \mathit{prog}_2\{Q_2\}}{\{P_1 * P_2\}\ \mathit{prog}_1||\mathit{prog}_2\ \{Q_1 * Q_2\}}$$

Smallfoot

Smallfoot [1] is a tool that allows to automatically verify specifications of programs written in a simple, imperative language. It uses light-weight specifications about dynamically allocated pointer structures on a heap that are expressed in a simple fragment of separation logic:

$$\begin{array}{ccc} e_1 \doteq e_2 & e_1 \neq e_2 & \mathit{emp} \\ e \mapsto [t_1 : e_1, \ldots, t_k : e_k] & \mathit{list\_seg}(tl, e_1, e_2) & \mathit{sf}_1 * \mathit{sf}_2 \\ \ldots & & \end{array}$$

Compared to fragments of separation logic used by other tools or current papers, the fragment used by Smallfoot is quite weak. However, this simplicity leads to decidable verification conditions and therefore allows Smallfoot to be completely automatic.

Example

```
list_remove(l;x) [list(l)] {
  local t;
  if(l!=NULL) {
    if(l==x) {
      l = l->tl;
      dispose(x);
    } else {
      t = l->tl;
      list_remove(t;x);
      l->tl = t;
    }
  }
} [list(l)]
```

- call-by-reference parameter l
- call-by-value parameter x
- intention: remove node x from list l
- precondition: l is start of null-terminated single linked list
- postcondition: l is still start of linked list

Smallfoot is able to verify this annotated function definition completely automatic. During this verification the following verification conditions are checked besides others:

```
emp |- listseg(tl; 0, 0)
```

```
0!=x * l!=x * listseg(tl; l, 0) |-
listseg(tl; l, 0)
```

```
0!=l * l!=t * l!=x *
l |-> tl:t * listseg(tl; t, 0) |- listseg(tl; l, 0)
```

Other examples lead to more interesting verification conditions:

```
0!=t * 0!=x * t!=y * x!=y * t |-> tl:y *
listseg(tl; y, 0) * listseg(tl; x, t) |-
listseg(tl; x, 0)
```

HOL embedding

I deeply embedded the logic, in which verification conditions are expressed, and I implemented a decision procedure for it in HOL. Some documentation and the code can be found in the HOL-repository (`http://hol.sourceforge.net`, subdirectory `examples/decidable_separationLogic`).

The decision procedure is implemented as a set of conversions that simplify verification conditions. It is designed in such a way, that the simplifications will finally terminate and that iff a verification condition is valid, it is simplified to `true`.

Conclusions and Future Work

- all inferences used by Smallfoot have been verified using HOL
- this is likely to increase the trust in Smallfoot
- it can be used as a separation logic calculator
- however: just verification conditions not programs are handled
- I plan to build framework based on abstract separation logic [2]
  - this should include the semantics of programs
  - abstract enough to handle different programming languages / flavours of separation logic

Acknowledgements

I thank Mike Gordon and Magnus Myreen for many discussions.

References

[1] Josh Berdine, Cristiano Calcagno, and Peter W. O'Hearn. Smallfoot: Modular automatic assertion checking with separation logic. In *FMCO*, pages 115–137, 2005.

[2] Cristiano Calcagno, Peter W. O'Hearn, and Hongseok Yang. Local action and abstract separation logic. In *LICS '07: Proceedings of the 22nd Annual IEEE Symposium on Logic in Computer Science*, pages 366–378, Washington, DC, USA, 2007. IEEE Computer Society.

[3] Peter O'Hearn, John Reynolds, and Hongseok Yang. Local reasoning about programs that alter data structures. *Lecture Notes in Computer Science*, 2142:1–??, 2001.

[4] J. Reynolds. Separation logic: a logic for shared mutable data structures, 2002.