Separation logic adapted for proofs by rewriting

Magnus O. Myreen
University of Cambridge

short paper at ITP'10
Motivation

In this talk: separation logic adapted to ACL2-like proofs (rewriting)

ITP

TPHOIs
more expressive logics

ACL2
high degree of automation, scales better
An extension to Hoare logic due to Reynolds et al. (~2002)

Its separating conjunction $\mathbin{\&}$ prevents pointer aliasing:

$$(a \leftrightarrow b) \mathbin{\&} (a+1 \leftrightarrow x) \mathbin{\&} (b \leftrightarrow 0) \mathbin{\&} (b+1 \leftrightarrow y)$$

Separation logic in one slide

Its frame rule makes reasoning local:

$$\{p\} \ C \ {q} \implies \forall r. \ {p \mathbin{\&} r} \ C \ {q \mathbin{\&} r}$$
Problematic quantifiers

• Definition of separating conjunction:

\[(p \ast q) s = \exists s_1 s_2. (s = s_1 \uplus s_2) \land p s_1 \land q s_2\]

• Quantifiers also in frame rule, linked-list predicate, etc.
Avoiding quantifiers

- Wrote an interpreter for -separated predicates

\[ ((a \mapsto x) \cdot (b \mapsto y) \cdot (c \mapsto z)) \text{ state} \]

\[ \text{separate } [(a, x), (b, y), (c, z)] \parallel \text{ state} \]

\[
\begin{align*}
\text{separate } & t \text{ state } = \text{ all_distinct } t \\
\text{separate } & (a, x):l) \text{ t state } = (\text{state}(a) = x) \land \text{separate } l \ (a::t) \text{ state }
\end{align*}
\]

- The linked-list example and frame:

\[ \text{separate } ([[(a, b), (a+1, x), (b, 0), (b+1, y)] \tikz[baseline=-1.5pt] \fill (0,0) circle (2pt) ++ (0pt,-3pt) node[anchor=north]{frame}) \parallel \text{ state} \]
Powerful proof automation

• Example: destructive list reversal

  rewriting alone can automatically prove body of loop:

  separate (llist 1 (x::xs) ++ llist 2 ys ++ frame ++ ...) [3] state
  list p1 (x :: xs) * list p2 (ys)

  p3 = p1->tail;
  p1->tail = p2;
  p2 = p1;
  p1 = p3;

  mem[mem[1]] := mem[2];
  mem[2] := mem[1];

  list p1 (xs) * list p2 (x :: ys)

  separate (llist 1 xs ++ llist 2 (x::ys) ++ frame ++ ...) [3] state

• toy language where pc, code, regs are kept in memory
  (potential for pointer aliasing)
Verified example

\[
\text{separate (} \llist 1 \; xs \leftrightarrow frame \leftrightarrow \ldots \; \text{)} \; [2,3] \; \text{state}
\]

0: \text{mem}[2] := 0;
3: \text{jump to 18;}
6: \text{mem}[3] := \text{mem}[\text{mem}[1]];
9: \text{mem}[\text{mem}[1]] := \text{mem}[2];
12: \text{mem}[2] := \text{mem}[1];
15: \text{mem}[1] := \text{mem}[3];
18: \text{jump to 6, if not (} \text{mem}[1] = 0 \text{)}

\[
\text{separate (} \llist 2 \; (\text{reverse } xs) \leftrightarrow frame \leftrightarrow \ldots \; \text{)} \; [1,3] \; \text{state}
\]
Summary

- **Rewriting = powerful automation for separation logic if quantifiers are avoided**

- **Lesson learnt:** HOL4's simplifier expands outermost match, ACL2's simplifier expands innermost match.

  \[
  \text{next(next(next(state))))}
  \]

Ack. Matt Kaufmann ported my HOL4 implementation into ACL2.