Specifying and Verifying Concurrent Programs with Ghost State

William Mansky, Princeton University

at DeepSpec 2018
Concurrency in Verifiable C

\{x \mapsto v\} \text{ (read-only)}

```c
void f(int *x){
    ...
    y = *x + 1;
    ...
}
\{\exists p. p \mapsto v + 1 * x \mapsto v\}
```
Concurrency in Verifiable C

\{\text{lock}(l, x \mapsto v)\} \quad \text{(writable)}

void f(int *x) {
    acquire(l);
    y = *x + 1;
    release(l);
}

\{\exists p. p \mapsto v + 1 \ast \text{lock}(l, x \mapsto v)\}
Concurrency in Verifiable C

• Distribute memory between threads with shares
• Lock-based concurrency library (makelock, freelock, acquire, release, spawn)
• Newest concurrent feature: ghost state
What Is Ghost State?

\[
\begin{align*}
\{x = N\} \\
y &= 1; \\
\{x = N \land y = 1\} \\
\text{while } (x > 0) \{ \\
\quad y &= 3 \times y; \\
\quad x &= x / 2; \\
\} \\
\{x = 0 \land y = 3^{\lg(N)}\}
\end{align*}
\]
What Is Ghost State?

\[
\begin{align*}
\{x = N\} \\
y = 1; \\
\{x = N \land y = 1 \land z = 0\} \\
\text{while } (x > 0) \{ \\
\{x = N / (2^z) \land y = 3^z\} \\
y = 3 \times y; \\
x = x / 2; \\
\{x = N / (2^{(z+1)}) \land y = 3^{(z+1)}\} \\
\} \\
\{x = 0 \land y = 3^{\log(N)}\}
\end{align*}
\]
Ghost State in Concurrent Separation Logic

\[ *x = 0; \]

release(l);
{lock(l, R)}
acquire(l); \|\| acquire(l);
||
x = x + 1; \|\| x = x + 1;
||
release(l); \|\| release(l);
{lock(l, R)}
acquire(l);
{lock(l, R) * R}

\[ R := \exists v. x \mapsto v \]
Ghost State in Concurrent Separation Logic

\*x = 0;
{lock(l, R) \* x \mapsto 0 \* g_1 \mapsto 0 \* g_2 \mapsto 0}
release(l);
{lock(l, R) \* g_1 \mapsto 0 \* g_2 \mapsto 0}
acquire(l); || acquire(l);
{... \* g_1 \mapsto 0 \* x \mapsto 0 + v_2 \* ...} ||
x = x + 1; || x = x + 1;
{... \* g_1 \mapsto 1 \* x \mapsto 1 + v_2 \* ...} ||
release(l); || release(l);
{lock(l, R) \* g_1 \mapsto 1 \* g_2 \mapsto 1}
acquire(l);
{lock(l, R) \* x \mapsto 2 \* g_1 \mapsto 1 \* g_2 \mapsto 1}

\text{\( R := \exists v_1, v_2. x \mapsto v_1 + v_2 \* g_1 \mapsto v_1 \* g_2 \mapsto v_2 \)
What Can Ghost State Be?

• Ghost state can be anything with the right algebraic structure!
• Any \((M, \oplus)\) such that

\[
a \oplus b = b \oplus a \quad \text{and} \quad (a \oplus b) \oplus c = a \oplus (b \oplus c)
\]

(partial commutative monoid, resource algebra, separation algebra)

• Essentially, anything that can be inserted into separation logic:

\[
\text{ghost}(a) \ast \text{ghost}(b) \iff \text{ghost}(a \oplus b)
\]

• Examples: variables, shares, graphs, state machines, histories, ...

\[
(g, \pi_1, v) \oplus (g, \pi_2, v) = (g, \pi_1 \oplus \pi_2, v) \quad g \xrightarrow{\pi} v ::= \text{ghost}((g, \pi, v))
\]
Updating Ghost State

• We can change the value of the ghost variable when we have all of it.
• More generally, we can make frame-preserving updates:
  \[ a \sim b ::= \forall c, a \oplus c \text{ defined } \Rightarrow b \oplus c \text{ defined} \]
• I.e., we can’t invalidate anyone else’s ghost state
• Shared variables: \( \pi_1 \oplus \pi_2 ::= \pi_1 + \pi_2 \) when \( \pi_1, \pi_2 > 0, \pi_1 + \pi_2 \leq 1 \)

• In CSL:
  \[
  \frac{a \sim b}{\text{ghost}(a) \Rightarrow \text{ghost}(b)}
  \]
What Can Ghost State Do?

• Ghost state is useful for tracking facts we could have recorded in the original program.

• With a non-trivial join operation, it also places restrictions on the possible states of other parts of the program. I.e., it enforces protocols between threads.

• Ghost state is essentially a generalization of the SL heap to an arbitrary mathematical object.

• We can use it to do global reasoning locally!
Ghost State and Global Invariants

\{\text{inv}(\exists a \ b. x \mapsto a \cdot y \mapsto b \cdot a \mid b)\}
\quad *y = *x + *y;
\{\text{inv}(\exists a \ b. x \mapsto a \cdot y \mapsto b \cdot a \mid b)\}
\quad *x = 1;
\{\text{inv}(\exists a \ b. x \mapsto a \cdot y \mapsto b \cdot a \mid b)\}

\begin{align*}
(\text{I} \cdot \text{ghost(\text{off})}) & \lor \text{ghost(\text{on})}
\end{align*}

• We can define $\text{lock}(l, R) ::= \text{inv}(l \mapsto 0 \cdot R) \lor l \mapsto 1)"

Based on “Iris from the Ground Up”, Jung et al., to appear in JFP ’18
Histories and Linearizability

• A concurrent data structure is linearizable if its behavior cannot be distinguished from that of a sequential implementation

\[ R ::= \exists H. \text{hashtable}(H, p) \land \exists h. \text{consistent}(H, h) \land \text{ghost}(h) \]

\[ \{\text{inv}(R) \land \text{ghost}(h)\} x = \text{get}(p, k); \{\text{inv}(R) \land \text{ghost}(h; \text{Get } k \ x)\} \]

\[ \{\text{inv}(R) \land \text{ghost}(h)\} \text{set}(p, k, x); \{\text{inv}(R) \land \text{ghost}(h; \text{Set } k \ x)\} \]
Ghost State in Verifiable C

ghost_alloc (ghost_var Tsh 0).

... 

viewshift_SEP 2 (ghost_var Tsh 1 g).
{ prove \( g_1 \mapsto 0 \sim g_1 \mapsto 1 \) }

- Double-increment example shipped with VST, brief manual available online

\[
\begin{align*}
*x &= 0; \\
\{ x \mapsto 0 * lock(l, R) \} \\
\{ x \mapsto 0 * lock(l, R) * g_1 \mapsto 0 \}
\end{align*}
\]

\[
\begin{align*}
\{ lock(l, R) * g_1 \mapsto 0 * \cdots \} \\
\{ lock(l, R) * g_1 \mapsto 1 * \cdots \}
\end{align*}
\]
More Verified Concurrent Programs

• “Mailbox” communication system – atomic exchange, ghost variables

• Hashtable and key-value node – atomic load/store/CAS, histories and linearizability

• Future: high-performance database implementation (Masstree) – relaxed load/store, hand-over-hand locking, etc.
Conclusion

• Concurrent separation logic without ghost state lets us prove race freedom, and verify steady-state programs with lock invariants.

• But ghost state unlocks the true power of CSL, letting us use the same local reasoning to prove global properties like linearizability.

• Ghost state fits neatly into VST, so we can now prove linearizability of concurrent C programs.

• Looking forward to applying it to weak memory, fine-grained data structures, and any concurrent applications you might have!
Note: On Sequential Consistency

• All of this only works under SC!
• Global invariants are unsound under more relaxed models (*)
• The earlier approach falls apart when we try to connect the ghost state to the values of data in memory (what does “data in memory” even mean?)
• RSL and GPS provide more limited, per-location invariant reasoning