**Separation Logic**

**Hoare logic:** cannot do modular reasoning about C-programs: e.g.

$$\text{list}(x) \land \text{list}(y)$$

**Separation logic:** provides *local reasoning* about C-programs by viewing partial heaps as *resource*: e.g. $x \neq y$

$$\text{list}(x) \ast \text{list}(y)$$

O’Hearn, Reynolds, Yang: CSL 2001; POPL tutorial, O’Hearn

**Origins:** The assertion language came directly from category theory.

**Applications:** Used to verify e.g. device drivers and Linux code.
Local Reasoning about Heaps

Heap model: $h : \text{Loc} \rightarrow_{\text{fin}} \text{Val}$, with $\text{Loc} \subseteq \text{Val}$

Cell assertions:

$x \mapsto y$, the cell at location $x$ has value $y$, and the thread has the right to modify it.

Other assertions:

emp, empty heap

$P \ast Q$, separating conjunction
Local Reasoning about Heaps

Small Hoare axiom:

\[ \{ x \mapsto y \} \text{dispose}(x) \{ \text{emp} \} \]

Frame rule:

\[ \frac{ \{ x \mapsto y \} \text{dispose}(x) \{ \text{emp} \} } { \{ P \ast x \mapsto y \} \text{dispose}(x) \{ P \ast \text{emp} \} } \quad x \notin P \]
Local Reasoning about Sets

Set model: $s : \text{Values} \rightarrow_{\text{fin}} \{0, 1\}$

Value assertions:

$\text{in}(v)$, value $v$ is in the set and the thread has the right to modify it.

$\text{out}(v)$, value $v$ is not in the set and the thread has the right to modify it.

Assertion axiom: e.g.

$\text{in}(v) \star \text{in}(v) \Rightarrow false$
Local Reasoning about Sets

Small Hoare axiom:

$$\begin{align*}
\{ \text{in}(v) \} & \text{remove}(v) \{ \text{out}(v) \} \\
\end{align*}$$

Frame rule:

$$\begin{align*}
\{ \text{in}(v) \} & \text{remove}(v) \{ \text{out}(v) \} \\
\{ P \ast \text{in}(v) \} & \text{remove}(v) \{ P \ast \text{out}(v) \} \\
\end{align*}$$

$$v \notin P$$
Fiction of Separation

Abstract set specification:
\[
\{ \text{in}(v) \} \ \text{remove}(v) \ \{ \text{out}(v) \}
\]

Concrete linked-list implementation:
\[
\{ v \in \text{list}(h) \} \ \text{code_for_remove}(v) \ \{ v \notin \text{list}(h) \}
\]

Fiction of separation: elements not separated in list implementation
Disjoint Concurrency

Value assertions enough for disjoint concurrency: e.g., \( v_1 \neq v_2 \)

\[
\begin{align*}
\{ \text{in}(v_1) \} \times \{ \text{in}(v_2) \} \\
\{ \text{in}(v_1) \} \parallel \{ \text{in}(v_2) \} \\
\text{remove}(v_1) \parallel \text{remove}(v_2) \\
\{ \text{out}(v_1) \} \parallel \{ \text{out}(v_2) \} \\
\{ \text{out}(v_1) \} \times \{ \text{out}(v_2) \}
\end{align*}
\]
Value assertions need adapting for shared concurrency:

**Value assertion:**

\[
\text{in}_{\text{def}}(v)_i, \text{ permission } i \in (0, 1]
\]

- **def**: the value \( v \) is definitely in the set
- \( 0 < i \leq 1 \): no other thread has the right to modify \( v \)
- \( i = 1 \): this thread has the right to modify \( v \)

**out}_{\text{def}}(v)_i \text{ is analogous.}
Shared Concurrency

Value assertions need adapting for shared concurrency:

**Value assertion:**

\[ \text{in}_{\text{rem}}(v)_i, \text{ for } i \in (0, 1] \]

- \textbf{rem}: the value \( v \) is in the set, but might be removed
- \( 0 < i \leq 1 \): all threads have the right to modify \( v \)

\( \text{out}_{\text{rem}}(v)_i \) is analogous.
Shared Concurrency

Assertion Axioms

\[ \text{in}_{\text{def}}(v)_1 \iff \text{in}_{\text{rem}}(v)_1 \]

\[ \text{in}_{\text{rem}}(v)_i \ast \text{in}_{\text{rem}}(v)_j \iff \text{in}_{\text{rem}}(v)_{i+j}, \text{ if } i + j \leq 1 \]

\[ \text{in}_{\text{rem}}(v)_i \ast \text{in}_{\text{rem}}(v)_j \Rightarrow \text{false}, \text{ if } i + j > 1 \]

\[ \vdots \]
Shared Concurrency

Small Hoare axioms:

\[
\begin{align*}
\{ \text{in}_{\text{def}}(v)_1 \} & \quad \text{remove}(v) & \{ \text{out}_{\text{def}}(v)_1 \} \\
\{ \text{in}_{\text{rem}}(v)_i \} & \quad \text{remove}(v) & \{ \text{out}_{\text{rem}}(v)_i \} \\
\vdots & & \vdots
\end{align*}
\]
Shared Concurrency

\[
\begin{align*}
\{ \text{in}_{\text{def}}(v) \} & \\
\{ \text{in}_{\text{rem}}(v) \} & \\
\{ \text{in}_{\text{rem}}(v) \frac{1}{2} \} & \times \{ \text{in}_{\text{rem}}(v) \frac{1}{2} \} \\
\{ \text{in}_{\text{rem}}(v) \frac{1}{2} \} & \parallel \{ \text{in}_{\text{rem}}(v) \frac{1}{2} \} \\
\text{remove}(v) & \parallel \text{remove}(v) \\
\{ \text{out}_{\text{rem}}(v) \frac{1}{2} \} & \parallel \{ \text{out}_{\text{rem}}(v) \frac{1}{2} \} \\
\{ \text{out}_{\text{rem}}(v) \frac{1}{2} \} & \times \{ \text{out}_{\text{rem}}(v) \frac{1}{2} \} \\
\{ \text{out}_{\text{def}}(v) \} & 
\end{align*}
\]
Parallel Sieve of Eratosthenes

worker$_2$

worker$_3$

worker$_4$

worker$_5$

\[ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ \cdots \]
Parallel Sieve of Eratosthenes

Sieve Specification

\[
\begin{align*}
\{ \bigodot_{2 \leq n \leq \text{max}} \text{in}_{\text{def}}(n)_1 & \land \text{max} > 1 \}\ \\
\{ \bigotimes_{2 \leq n \leq \text{max}} \text{in}_{\text{rem}}(n)_1 & \land \text{max} > 1 \}\ \\
\text{worker}(2, \text{max}) & \parallel \text{worker}(3, \text{max}) \parallel \ldots \parallel \text{worker}(m, \text{max})
\end{align*}
\]

where \( m = \lfloor \sqrt{\text{max}} \rfloor \)
Parallel Sieve of Eratosthenes

Worker thread \( \bigotimes \) is iterated separating conjunction.

\[
\begin{align*}
\{ & 2 \leq v \land \bigotimes_{2 \leq n \leq \text{max}} \text{in}_{\text{rem}}(n)_i \\
\text{worker}(v, \text{max}) \{ \\
& \quad c := v + v; \\
& \quad \text{while}(c \leq \text{max}) \{ \\
& \quad \quad \text{remove}(c); \\
& \quad \quad c := c + v; \\
& \quad \} \\
\} \\
\{ & \bigotimes_{2 \leq n \leq \text{max}} \text{fac}(n, v) \implies \text{out}_{\text{rem}}(n)_i \land \\
& \neg \text{fac}(n, v) \implies \text{in}_{\text{rem}}(n)_i
\end{align*}
\]
Another assertion axiom:

\[ \text{in}_{\text{rem}}(v)_i \ast \text{out}_{\text{rem}}(v)_j \Rightarrow \text{out}_{\text{rem}}(v)_{i+j}, \text{ if } i + j \leq 1 \]
Parallel Sieve of Eratosthenes

Sieve Specification

\[ \{ \bigotimes_{2 \leq n \leq \text{max}} \text{in}_{\text{def}}(n) \land \text{max} > 1 \} \]

\text{worker}(2, \text{max}) \parallel \text{worker}(3, \text{max}) \parallel \ldots \parallel \text{worker}(m, \text{max})

\[ \left\{ \bigotimes_{2 \leq n \leq \text{max}} \text{isPrime}(n) \implies \text{in}_{\text{def}}(n) \land \right. \]

\[ \left. \neg \text{isPrime}(n) \implies \text{out}_{\text{def}}(n) \right\} \]

where \( m = \lfloor \sqrt{\text{max}} \rfloor \)

Application: concurrent indexes, verified concurrent B-tree implementation
Local Resource Reasoning at POPL

Life-time achievement award, Hoare

Parallization of sequential programs, Dodds

Syntactic control of interference for separation logic, Reddy

Towards a program logic for JavaScript Smith

Separation logic: O’Hearn, POPL tutorial

Termination: Cook, POPL mentoring tutorial and POPL tutorial
Long-term Questions

What do you know?

How much are you learning?

What is your research voice?

Who is your intended audience?