Verification: Local Resource Reasoning

Philippa Gardner

Imperial College London

Separation Logic

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

$$\texttt{list}(x) \wedge \texttt{list}(y)$$

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

$$\texttt{list}(x) * \texttt{list}(y)$$

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: $h : \mathtt{Loc} \rightharpoonup_{\mathtt{fin}} \mathtt{Val}$, with $\mathtt{Loc} \subseteq \mathtt{Val}$

Cell assertions:

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

Other assertions:

$\mathtt{emp}$, empty heap

$P * Q$, separating conjunction

Local Reasoning about Heaps

Small Hoare axiom:

$$\left\{x \mapsto y\right\}\ \mathtt{dispose}(x)\ \left\{\mathsf{emp}\right\}$$

Frame rule:

$$\frac{\left\{x \mapsto y\right\}\ \mathtt{dispose}(x)\ \left\{\mathsf{emp}\right\}}{\left\{P * x \mapsto y\right\}\ \mathtt{dispose}(x)\ \left\{P * \mathsf{emp}\right\}}\ x \notin P$$

Local Reasoning about Sets

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

Value assertions:

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

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

Assertion axiom: e.g.

$\mathtt{in}(v) * \mathtt{in}(v) \Rightarrow \mathit{false}$

Local Reasoning about Sets

Small Hoare axiom:

$$\left\{\mathtt{in}(v)\right\}\ \mathtt{remove}(v)\ \left\{\mathtt{out}(v)\right\}$$

Frame rule:

$$\frac{\left\{\mathtt{in}(v)\right\}\ \mathtt{remove}(v)\ \left\{\mathtt{out}(v)\right\}}{\left\{P * \mathtt{in}(v)\right\}\ \mathtt{remove}(v)\ \left\{P * \mathtt{out}(v)\right\}}\ v \notin P$$

Fiction of Separation

Abstract set specification:

$$\left\{\texttt{in}(v)\right\}\ \texttt{remove}(v)\ \left\{\texttt{out}(v)\right\}$$

Concrete linked-list implementation:

$$\left\{v \in \texttt{list}(h)\right\}\ \texttt{code\_for\_remove}(v)\ \left\{v \notin \texttt{list}(h)\right\}$$

Fiction of separation: elements not separated in list implementation

Disjoint Concurrency

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

$$\left\{\mathtt{in}(v_1) * \mathtt{in}(v_2)\right\}$$

$$\begin{array}{c|c} \left\{\mathtt{in}(v_1)\right\} & \left\{\mathtt{in}(v_2)\right\} \\ \mathtt{remove}(\mathtt{v}_1) & \mathtt{remove}(\mathtt{v}_2) \\ \left\{\mathtt{out}(v_1)\right\} & \left\{\mathtt{out}(v_2)\right\} \end{array}$$

$$\left\{\mathtt{out}(v_1) * \mathtt{out}(v_2)\right\}$$

Shared Concurrency

Value assertions need adapting for shared concurrency:

Value assertion:

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

- $\mathbf{def}$: 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

$\mathtt{out}_{\mathtt{def}}(v)_i$ is analogous.

Shared Concurrency

Value assertions need adapting for shared concurrency:

Value assertion:

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

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

$\texttt{out}_{\texttt{rem}}(v)_i$ is analogous.

Shared Concurrency

Assertion Axioms

$$\mathsf{in}_{\mathsf{def}}(v)_1 \Leftrightarrow \mathsf{in}_{\mathsf{rem}}(v)_1$$

$$\mathsf{in}_{\mathsf{rem}}(v)_i * \mathsf{in}_{\mathsf{rem}}(v)_j \Leftrightarrow \mathsf{in}_{\mathsf{rem}}(v)_{i+j}\,, \ \text{if } i + j \leq 1$$

$$\mathsf{in}_{\mathsf{rem}}(v)_i * \mathsf{in}_{\mathsf{rem}}(v)_j \Rightarrow \mathsf{false}, \ \text{if } i + j > 1$$

$$\vdots$$

Shared Concurrency

Small Hoare axioms:

$$\left\{\mathsf{in}_{\mathsf{def}}(v)_1\right\} \ \ \texttt{remove(v)} \ \ \left\{\mathsf{out}_{\mathsf{def}}(v)_1\right\}$$
$$\left\{\mathsf{in}_{\mathsf{rem}}(v)_i\right\} \ \ \texttt{remove(v)} \ \ \left\{\mathsf{out}_{\mathsf{rem}}(v)_i\right\}$$
$$\vdots$$

Shared Concurrency

$$\left\{\mathsf{in}_{\mathsf{def}}(v)_1\right\}$$

$$\left\{\mathsf{in}_{\mathsf{rem}}(v)_1\right\}$$

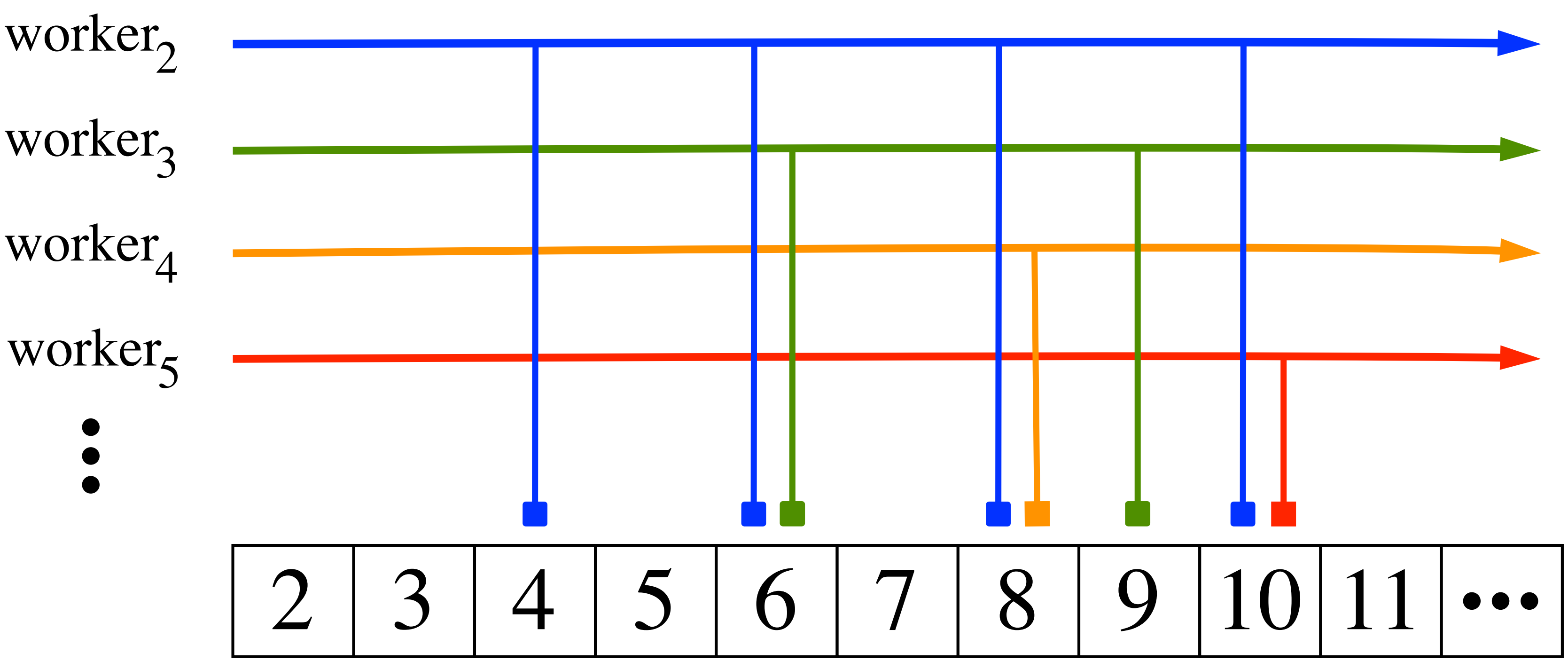
$$\left\{\mathsf{in}_{\mathsf{rem}}(v)_{\frac{1}{2}} * \mathsf{in}_{\mathsf{rem}}(v)_{\frac{1}{2}}\right\}$$

$$\begin{array}{c|c} \left\{\mathsf{in}_{\mathsf{rem}}(v)_{\frac{1}{2}}\right\} & \left\{\mathsf{in}_{\mathsf{rem}}(v)_{\frac{1}{2}}\right\} \\ \texttt{remove(v)} & \texttt{remove(v)} \\ \left\{\mathsf{out}_{\mathsf{rem}}(v)_{\frac{1}{2}}\right\} & \left\{\mathsf{out}_{\mathsf{rem}}(v)_{\frac{1}{2}}\right\} \end{array}$$

$$\left\{\mathsf{out}_{\mathsf{rem}}(v)_{\frac{1}{2}} * \mathsf{out}_{\mathsf{rem}}(v)_{\frac{1}{2}}\right\}$$

$$\left\{\mathsf{out}_{\mathsf{def}}(v)_1\right\}$$

Parallel Sieve of Eratosthenes



Parallel Sieve of Eratosthenes

Sieve Specification

$$\left\{ \circledast_{2 \le n \le \mathtt{max}} \mathsf{in}_{\mathsf{def}}(n)_1 \wedge \mathtt{max} > 1 \right\}$$
$$\left\{ \circledast_{2 \le n \le \mathtt{max}} \mathsf{in}_{\mathsf{rem}}(n)_1 \wedge \mathtt{max} > 1 \right\}$$

`worker(2, max)` $\|$ `worker(3, max)` $\| \ldots \|$ `worker(m, max)`

where $m = \lfloor \sqrt{\mathtt{max}} \rfloor$

Parallel Sieve of Eratosthenes

Worker thread $\circledast$ is iterated separating conjunction.

$$\left\{2 \leq \mathtt{v} \wedge \circledast_{2 \leq n \leq \mathtt{max}} \mathsf{in}_{\mathsf{rem}}(n)_i\right\}$$

$$\begin{array}{l}
\mathtt{worker}(\mathtt{v}, \mathtt{max})\ \{ \\
\quad \mathtt{c} := \mathtt{v} + \mathtt{v}; \\
\quad \mathtt{while}(\mathtt{c} \leq \mathtt{max})\ \{ \\
\quad\quad \mathtt{remove}(\mathtt{c}); \\
\quad\quad \mathtt{c} := \mathtt{c} + \mathtt{v}; \\
\}\}
\end{array}$$

$$\left\{\circledast_{2 \leq n \leq \mathtt{max}} \begin{array}{lcl} \mathsf{fac}(n, \mathtt{v}) & \Longrightarrow & \mathsf{out}_{\mathsf{rem}}(n)_i \wedge \\ \neg\mathsf{fac}(n, \mathtt{v}) & \Longrightarrow & \mathsf{in}_{\mathsf{rem}}(n)_i \end{array}\right\}$$

Another assertion axiom:

$$\mathsf{in}_{\mathsf{rem}}(v)_i * \mathsf{out}_{\mathsf{rem}}(v)_j \Rightarrow \mathsf{out}_{\mathsf{rem}}(v)_{i+j},\ \text{ if } i + j \leq 1$$

Parallel Sieve of Eratosthenes

Sieve Specification

$$\left\{\circledast_{2\leq n\leq \mathtt{max}}\ \mathsf{in}_{\mathsf{def}}(n)_1 \wedge \mathtt{max} > 1\right\}$$

$$\texttt{worker(2, max)} \,\|\, \texttt{worker(3, max)} \,\|\, \ldots \,\|\, \texttt{worker(m, max)}$$

$$\left\{\circledast_{2\leq n\leq \mathtt{max}} \begin{array}{lcl} \mathsf{isPrime}(n) & \implies & \mathsf{in}_{\mathsf{def}}(n)_1 \wedge \\ \neg\mathsf{isPrime}(n) & \implies & \mathsf{out}_{\mathsf{def}}(n)_1 \end{array}\right\}$$

where $m = \lfloor \sqrt{\mathtt{max}} \rfloor$

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?