Reasoning over Permissions Regions in Concurrent Separation Logic

James Brotherston, Diana Costa, Aquinas Hobor and John Wickerson

PPLV seminar, UCL Dept of Computer Science

Friday 28th October, 2022

Concurrent separation logic (CSL)

- Concurrent separation logic (CSL) is based upon the following concurrency rule:

$$\frac{\{A_1\}\, C_1\, \{B_1\} \quad \{A_2\}\, C_2\, \{B_2\}}{\{A_1 \circledast A_2\}\, C_1 \,||\, C_2\, \{B_1 \circledast B_2\}}$$

Concurrent separation logic (CSL)

- Concurrent separation logic (CSL) is based upon the following concurrency rule:

$$\frac{\{A_1\}\, C_1\, \{B_1\} \quad \{A_2\}\, C_2\, \{B_2\}}{\{A_1 \circledast A_2\}\, C_1 \,||\, C_2\, \{B_1 \circledast B_2\}}$$

- This rule says that concurrent threads behave compositionally with respect to separation ($\circledast$) between their respective memory resources.

Concurrent separation logic (CSL)

- Concurrent separation logic (CSL) is based upon the following concurrency rule:

$$\frac{\{A_1\}\, C_1\, \{B_1\} \quad \{A_2\}\, C_2\, \{B_2\}}{\{A_1 \circledast A_2\}\, C_1 \,||\, C_2\, \{B_1 \circledast B_2\}}$$

- This rule says that concurrent threads behave compositionally with respect to separation ($\circledast$) between their respective memory resources.

- However, separation $\circledast$ typically allows some sharing of read-only resources between threads, which can be controlled using fractional permissions.

Fractional permissions

- Fractional permissions are intended to allow the division of memory into two or more “read-only copies”.

Fractional permissions

- Fractional permissions are intended to allow the division of memory into two or more "read-only copies".
- Permissions can be represented e.g. as rationals in the open interval $(0, 1]$. 1 is the write permission and values in $(0, 1)$ are read-only permissions.

Fractional permissions

- Fractional permissions are intended to allow the division of memory into two or more “read-only copies”.
- Permissions can be represented e.g. as rationals in the open interval $(0, 1]$. 1 is the write permission and values in $(0, 1)$ are read-only permissions.
- Heaps store a data value and permission at each location. Heaps can be composed provided they agree where they overlap; we add the permissions at overlapping locations.

Fractional permissions

- Fractional permissions are intended to allow the division of memory into two or more "read-only copies".
- Permissions can be represented e.g. as rationals in the open interval $(0, 1]$. 1 is the write permission and values in $(0, 1)$ are read-only permissions.
- Heaps store a data value and permission at each location. Heaps can be composed provided they agree where they overlap; we add the permissions at overlapping locations.
- Separation $\circledast$ denotes the division of a heap using this composition. E.g., we have

$$x \overset{0.5}{\mapsto} d \circledast x \overset{0.5}{\mapsto} d \equiv x \mapsto d \ .$$

Typical CSL *proof structure*

$$\{x \overset{0.5}{\mapsto} d\} \qquad \Bigg\| \qquad \{x \overset{0.5}{\mapsto} d\}$$

$$\texttt{foo}(); \qquad \Bigg\| \qquad \texttt{bar}();$$

$$\{x \overset{0.5}{\mapsto} d * A\} \qquad \Bigg\| \qquad \{x \overset{0.5}{\mapsto} d * B\}$$

Typical CSL *proof structure*

$$\{x \mapsto d\}$$

$\{x \overset{0.5}{\mapsto} d\}$	$\parallel$	$\{x \overset{0.5}{\mapsto} d\}$
`foo();`		`bar();`
$\{x \overset{0.5}{\mapsto} d * A\}$		$\{x \overset{0.5}{\mapsto} d * B\}$

Typical CSL *proof structure*

$$\{x \mapsto d\}$$

$$\{x \overset{0.5}{\mapsto} d \circledast x \overset{0.5}{\mapsto} d\}$$

$$
\begin{array}{c|c}
\{x \overset{0.5}{\mapsto} d\} & \{x \overset{0.5}{\mapsto} d\} \\
\texttt{foo}(); & \texttt{bar}(); \\
\{x \overset{0.5}{\mapsto} d * A\} & \{x \overset{0.5}{\mapsto} d * B\}
\end{array}
$$

Typical CSL *proof structure*

$$\{x \mapsto d\}$$

$$\{x \overset{0.5}{\mapsto} d \circledast x \overset{0.5}{\mapsto} d\}$$

$$\begin{array}{c|c} \{x \overset{0.5}{\mapsto} d\} & \{x \overset{0.5}{\mapsto} d\} \\ \texttt{foo}(); & \texttt{bar}(); \\ \{x \overset{0.5}{\mapsto} d * A\} & \{x \overset{0.5}{\mapsto} d * B\} \end{array}$$

$$\{x \overset{0.5}{\mapsto} d \circledast x \overset{0.5}{\mapsto} d \circledast A \circledast B\}$$

Typical CSL *proof structure*

$$\{x \mapsto d\}$$

$$\{x \overset{0.5}{\mapsto} d \circledast x \overset{0.5}{\mapsto} d\}$$

$$\begin{array}{c|c} \{x \overset{0.5}{\mapsto} d\} & \{x \overset{0.5}{\mapsto} d\} \\ \texttt{foo}(); & \texttt{bar}(); \\ \{x \overset{0.5}{\mapsto} d * A\} & \{x \overset{0.5}{\mapsto} d * B\} \end{array}$$

$$\{x \overset{0.5}{\mapsto} d \circledast x \overset{0.5}{\mapsto} d \circledast A \circledast B\}$$

$$\{x \mapsto d \circledast A \circledast B\}$$

Typical CSL *proof structure*

$$\{x \mapsto d\}$$

$$\{x \overset{0.5}{\mapsto} d \circledast x \overset{0.5}{\mapsto} d\}$$

$$\begin{array}{c|c} \{x \overset{0.5}{\mapsto} d\} & \{x \overset{0.5}{\mapsto} d\} \\ \texttt{foo}(); & \texttt{bar}(); \\ \{x \overset{0.5}{\mapsto} d * A\} & \{x \overset{0.5}{\mapsto} d * B\} \end{array}$$

$$\{x \overset{0.5}{\mapsto} d \circledast x \overset{0.5}{\mapsto} d \circledast A \circledast B\}$$

$$\{x \mapsto d \circledast A \circledast B\}$$

BUT... we hit problems when we use permissions to describe regions of memory and not just pointers.

The first difficulty

Suppose we define linked list segments using $\circledast$:

$$\mathsf{ls}\, x\, y \;=_{\mathrm{def}}\; (x = y \wedge \mathsf{emp}) \vee (\exists z.\ x \mapsto z \circledast \mathsf{ls}\, z\, y)\ .$$

The first difficulty

Suppose we define linked list segments using $\circledast$:

$$\mathsf{ls}\, x\, y \ =_{\mathrm{def}} \ (x = y \wedge \mathsf{emp}) \vee (\exists z.\ x \mapsto z \circledast \mathsf{ls}\, z\, y)\ .$$

Now consider traversal procedure `foo(x,y)`:

```
foo(x,y) { if x=y then return; else foo([x],y); }
```

The first difficulty

Suppose we define linked list segments using $\circledast$:

$$\mathsf{ls}\, x\, y \;=_{\mathrm{def}}\; (x = y \wedge \mathsf{emp}) \vee (\exists z.\ x \mapsto z \circledast \mathsf{ls}\, z\, y)\ .$$

Now consider traversal procedure `foo(x,y)`:

```
foo(x,y) { if x=y then return; else foo([x],y); }
```

This satisfies the following Hoare triple:

$$\left\{(\mathsf{ls}\, x\, y)^{0.5}\right\} \texttt{foo(x,y);} \left\{(\mathsf{ls}\, x\, y)^{0.5}\right\}\ .$$

However, we will have difficulties proving so!

Failed proof attempt

$\{(\mathsf{ls}\, x\, y)^{0.5}\}$

```
foo(x,y) {
if x=y then return;
else

foo([x],y);



}
```

$\{(\mathsf{ls}\, x\, y)^{0.5}\}$

Failed proof attempt

$\{(\mathsf{ls}\, x\, y)^{0.5}\}$

```
foo(x,y) {
if x=y then return;
```

$\{(\mathsf{ls}\, x\, y)^{0.5}\}$

```
else

foo([x],y);

}
```

$\{(\mathsf{ls}\, x\, y)^{0.5}\}$

Failed proof attempt

$\left\{(\mathsf{ls}\, x\, y)^{0.5}\right\}$

```
foo(x,y) {
if x=y then return;
else
```

$\left\{(\mathsf{ls}\, x\, y)^{0.5}\right\}$

$\left\{x \neq y \wedge (x \mapsto z \circledast \mathsf{ls}\, z\, y)^{0.5}\right\}$

```
foo([x],y);
```

```
}
```

$\left\{(\mathsf{ls}\, x\, y)^{0.5}\right\}$

Failed proof attempt

$\left\{(\mathsf{ls}\,x\,y)^{0.5}\right\}$

```
foo(x,y) {
if x=y then return;
else
```

$\left\{(\mathsf{ls}\,x\,y)^{0.5}\right\}$

$\left\{x \neq y \wedge (x \mapsto z \circledast \mathsf{ls}\,z\,y)^{0.5}\right\}$

$\left\{x \neq y \wedge (x \overset{0.5}{\mapsto} z \circledast (\mathsf{ls}\,z\,y)^{0.5})\right\}$

```
foo([x],y);

}
```

$\left\{(\mathsf{ls}\,x\,y)^{0.5}\right\}$

Failed proof attempt

$\left\{(\mathsf{ls}\,x\,y)^{0.5}\right\}$

```
foo(x,y) {
if x=y then return;
else



foo([x],y);




}
```

$\left\{(\mathsf{ls}\,x\,y)^{0.5}\right\}$ (after `if x=y then return;`)

$\left\{x \neq y \wedge (x \mapsto z \circledast \mathsf{ls}\,z\,y)^{0.5}\right\}$

$\left\{x \neq y \wedge (x \overset{0.5}{\mapsto} z \circledast (\mathsf{ls}\,z\,y)^{0.5})\right\}$

$\left\{x \overset{0.5}{\mapsto} z \circledast (\mathsf{ls}\,z\,y)^{0.5}\right\}$ (before `foo([x],y);`)

`}` $\left\{(\mathsf{ls}\,x\,y)^{0.5}\right\}$

Failed proof attempt

$\{(\mathsf{ls}\,x\,y)^{0.5}\}$

```
foo(x,y) {
if x=y then return;
else


foo([x],y);


}
```

$\{(\mathsf{ls}\,x\,y)^{0.5}\}$ (after `if x=y then return;`)

$\{x \neq y \wedge (x \mapsto z \circledast \mathsf{ls}\,z\,y)^{0.5}\}$

$\left\{x \neq y \wedge (x \overset{0.5}{\mapsto} z \circledast (\mathsf{ls}\,z\,y)^{0.5})\right\}$

$\left\{x \overset{0.5}{\mapsto} z \circledast (\mathsf{ls}\,z\,y)^{0.5}\right\}$

$\{(x \mapsto z \circledast \mathsf{ls}\,z\,y)^{0.5}\}$

$\}\quad \{(\mathsf{ls}\,x\,y)^{0.5}\}$

Failed proof attempt

$\left\{(\mathsf{ls}\,x\,y)^{0.5}\right\}$

```
foo(x,y) {
if x=y then return;
else


foo([x],y);



}
```

$\left\{(\mathsf{ls}\,x\,y)^{0.5}\right\}$

$\left\{x \neq y \wedge (x \mapsto z \circledast \mathsf{ls}\,z\,y)^{0.5}\right\}$

$\left\{x \neq y \wedge (x \overset{0.5}{\mapsto} z \circledast (\mathsf{ls}\,z\,y)^{0.5})\right\}$

$\left\{x \overset{0.5}{\mapsto} z \circledast (\mathsf{ls}\,z\,y)^{0.5}\right\}$

$\left\{(x \mapsto z \circledast \mathsf{ls}\,z\,y)^{0.5}\right\}$

$\left\{(\mathsf{ls}\,x\,y)^{0.5}\right\}$

$\left\{(\mathsf{ls}\,x\,y)^{0.5}\right\}$

Failed proof attempt

$\{(\mathsf{ls}\,x\,y)^{0.5}\}$

`foo(x,y) {`

`if x=y then return;` $\{(\mathsf{ls}\,x\,y)^{0.5}\}$

`else` $\{x \neq y \wedge (x \mapsto z \circledast \mathsf{ls}\,z\,y)^{0.5}\}$

$\left\{x \neq y \wedge (x \overset{0.5}{\mapsto} z \circledast (\mathsf{ls}\,z\,y)^{0.5})\right\}$

`foo([x],y);` $\left\{x \overset{0.5}{\mapsto} z \circledast (\mathsf{ls}\,z\,y)^{0.5}\right\}$

$\{(x \mapsto z \circledast \mathsf{ls}\,z\,y)^{0.5}\}$

$\{(\mathsf{ls}\,x\,y)^{0.5}\}$

`}` $\{(\mathsf{ls}\,x\,y)^{0.5}\}$

Reason for failure

- The highlighted inference step is not sound:

$$x \overset{0.5}{\mapsto} z \circledast (\mathsf{ls}\, z\, y)^{0.5} \not\models (x \mapsto z \circledast \mathsf{ls}\, z\, y)^{0.5} \ .$$

Reason for failure

- The highlighted inference step is not sound:

$$x \overset{0.5}{\mapsto} z \circledast (\mathsf{ls}\, z\, y)^{0.5} \not\models (x \mapsto z \circledast \mathsf{ls}\, z\, y)^{0.5} \ .$$

- This is because the pointer and list segment can overlap on the LHS, but not on the RHS. In general,

$$A^{\pi} \circledast B^{\pi} \not\models (A \circledast B)^{\pi} \ .$$

Reason for failure

- The highlighted inference step is not sound:

$$x \overset{0.5}{\mapsto} z \circledast (\mathsf{ls}\, z\, y)^{0.5} \not\models (x \mapsto z \circledast \mathsf{ls}\, z\, y)^{0.5} \ .$$

- This is because the pointer and list segment can overlap on the LHS, but not on the RHS. In general,

$$A^{\pi} \circledast B^{\pi} \not\models (A \circledast B)^{\pi} \ .$$

- But if we use strong separation $*$, which enforces disjointness of heaps, to define our list segments, the proof above goes through (since $(A * B)^{\pi} \equiv A^{\pi} * B^{\pi}$).

The second difficulty

The triple $\{\mathsf{ls}\,x\,y\}$ `foo(x,y);` $\|$ `foo(x,y);` $\{\mathsf{ls}\,x\,y\}$ is correct, but again the proof fails:

$$\begin{array}{c|c} \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} & \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} \\ \texttt{foo(x,y);} & \texttt{foo(x,y);} \\ \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} & \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} \end{array}$$

The second difficulty

The triple $\{\mathsf{ls}\,x\,y\}$ `foo(x,y);` $\|$ `foo(x,y);` $\{\mathsf{ls}\,x\,y\}$ is correct, but again the proof fails:

$$\{\mathsf{ls}\,x\,y\}$$

$$\begin{array}{c|c} \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} & \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} \\ \texttt{foo(x,y);} & \texttt{foo(x,y);} \\ \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} & \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} \end{array}$$

The second difficulty

The triple $\{\mathsf{ls}\,x\,y\}$ `foo(x,y);` $\|$ `foo(x,y);` $\{\mathsf{ls}\,x\,y\}$ is correct, but again the proof fails:

$$\{\mathsf{ls}\,x\,y\}$$

$$\left\{(\mathsf{ls}\,x\,y)^{0.5} \circledast (\mathsf{ls}\,x\,y)^{0.5}\right\}$$

$$\begin{array}{c|c} \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} & \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} \\ \texttt{foo(x,y);} & \texttt{foo(x,y);} \\ \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} & \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} \end{array}$$

The second difficulty

The triple $\{\mathsf{ls}\, x\, y\}$ `foo(x,y);` $\|$ `foo(x,y);` $\{\mathsf{ls}\, x\, y\}$ is correct, but again the proof fails:

$$\{\mathsf{ls}\, x\, y\}$$

$$\left\{(\mathsf{ls}\, x\, y)^{0.5} \circledast (\mathsf{ls}\, x\, y)^{0.5}\right\}$$

$$\begin{array}{c|c} \left\{(\mathsf{ls}\, x\, y)^{0.5}\right\} & \left\{(\mathsf{ls}\, x\, y)^{0.5}\right\} \\ \texttt{foo(x,y);} & \texttt{foo(x,y);} \\ \left\{(\mathsf{ls}\, x\, y)^{0.5}\right\} & \left\{(\mathsf{ls}\, x\, y)^{0.5}\right\} \end{array}$$

$$\left\{(\mathsf{ls}\, x\, y)^{0.5} \circledast (\mathsf{ls}\, x\, y)^{0.5}\right\}$$

The second difficulty

The triple $\{\mathsf{ls}\,x\,y\}$ `foo(x,y);` $\|$ `foo(x,y);` $\{\mathsf{ls}\,x\,y\}$ is correct, but again the proof fails:

$$\{\mathsf{ls}\,x\,y\}$$

$$\left\{(\mathsf{ls}\,x\,y)^{0.5} \circledast (\mathsf{ls}\,x\,y)^{0.5}\right\}$$

$$\begin{array}{c|c} \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} & \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} \\ \texttt{foo(x,y);} & \texttt{foo(x,y);} \\ \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} & \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} \end{array}$$

$$\left\{(\mathsf{ls}\,x\,y)^{0.5} \circledast (\mathsf{ls}\,x\,y)^{0.5}\right\}$$

$$\{\mathsf{ls}\,x\,y\}$$

The second difficulty

The triple $\{\mathsf{ls}\,x\,y\}$ `foo(x,y);` $\|$ `foo(x,y);` $\{\mathsf{ls}\,x\,y\}$ is correct, but again the proof fails:

$$\{\mathsf{ls}\,x\,y\}$$

$$\left\{(\mathsf{ls}\,x\,y)^{0.5} \circledast (\mathsf{ls}\,x\,y)^{0.5}\right\}$$

$$\begin{array}{c|c} \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} & \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} \\ \texttt{foo(x,y);} & \texttt{foo(x,y);} \\ \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} & \left\{(\mathsf{ls}\,x\,y)^{0.5}\right\} \end{array}$$

$$\left\{(\mathsf{ls}\,x\,y)^{0.5} \circledast (\mathsf{ls}\,x\,y)^{0.5}\right\}$$

$$\not\hookrightarrow \{\mathsf{ls}\,x\,y\}$$

Reason for second failure

- The highlighted inference step is not sound:

$$(\mathsf{Is}\,x\,y)^{0.5} \circledast (\mathsf{Is}\,x\,y)^{0.5} \not\models \mathsf{Is}\,x\,y\ .$$

Reason for second failure

- The highlighted inference step is not sound:

$$(\mathsf{ls}\,x\,y)^{0.5} \circledast (\mathsf{ls}\,x\,y)^{0.5} \not\models \mathsf{ls}\,x\,y \ .$$

- This is because the list segments on the LHS might be (partially) non-overlapping. In general,

$$A^{0.5} \circledast A^{0.5} \not\models A \ .$$

Reason for second failure

- The highlighted inference step is not sound:

$$(\mathsf{ls}\, x\, y)^{0.5} \circledast (\mathsf{ls}\, x\, y)^{0.5} \not\models \mathsf{ls}\, x\, y\ .$$

- This is because the list segments on the LHS might be (partially) non-overlapping. In general,

$$A^{0.5} \circledast A^{0.5} \not\models A\ .$$

- When splitting the list segment $\mathsf{ls}\, x\, y$, we lost the info that the two formulas $(\mathsf{ls}\, x\, y)^{0.5}$ are copies of the same region.

Proposed solution: nominal labels

- We introduce nominal labels (from hybrid logic), where a nominal α is interpreted as denoting a unique heap.

Proposed solution: nominal labels

- We introduce nominal labels (from hybrid logic), where a nominal α is interpreted as denoting a unique heap.
- Any formula of the form $\alpha \wedge A$ then obeys the principle

$$(\alpha \wedge A)^{\sigma} \circledast (\alpha \wedge A)^{\pi} \equiv (\alpha \wedge A)^{\sigma \oplus \pi}$$

where $\oplus$ is addition on permissions.

Proposed solution: nominal labels

- We introduce nominal labels (from hybrid logic), where a nominal α is interpreted as denoting a unique heap.
- Any formula of the form $\alpha \wedge A$ then obeys the principle

$$(\alpha \wedge A)^{\sigma} \circledast (\alpha \wedge A)^{\pi} \equiv (\alpha \wedge A)^{\sigma \oplus \pi}$$

  where $\oplus$ is addition on permissions.
- Thus we can repair the faulty CSL proof above by replacing every instance of $\mathsf{ls}\,x\,y$ by $\alpha \wedge \mathsf{ls}\,x\,y$ (and adding an initial step in which we introduce the fresh label α).

What's in the paper?

- We define an assertion language including both weak $\circledast$ and strong $*$ separating conjunctions, and nominal labels α.

What's in the paper?

- We define an assertion language including both weak $\circledast$ and strong $*$ separating conjunctions, and nominal labels α.
- We also include hybrid logic's jump modality $@_\alpha A$, meaning A is true at α, which is useful in treating more complex sharing examples.

What's in the paper?

- We define an assertion language including both weak $\circledast$ and strong $*$ separating conjunctions, and nominal labels α.
- We also include hybrid logic's jump modality $@_\alpha A$, meaning A is true at α, which is useful in treating more complex sharing examples.
- We formally establish the needed principles, including

$$\begin{array}{rcl} (A * B)^\pi & \equiv & A^\pi * B^\pi \\ (\alpha \wedge A)^\sigma \circledast (\alpha \wedge A)^\pi & \equiv & (\alpha \wedge A)^{\sigma \oplus \pi} \end{array}$$

What's in the paper?

- We define an assertion language including both weak $\circledast$ and strong $*$ separating conjunctions, and nominal labels α.
- We also include hybrid logic's jump modality $@_\alpha A$, meaning A is true at α, which is useful in treating more complex sharing examples.
- We formally establish the needed principles, including

$$\begin{array}{rcl} (A * B)^\pi & \equiv & A^\pi * B^\pi \\ (\alpha \wedge A)^\sigma \circledast (\alpha \wedge A)^\pi & \equiv & (\alpha \wedge A)^{\sigma \oplus \pi} \end{array}$$

- Finally we show how our assertion language can be used in CSL to verify various concurrent programs with sharing.

Directions for future work

- Implementation and automation

Directions for future work

- Implementation and automation
- Specification inference and biabduction

Directions for future work

- Implementation and automation
- Specification inference and biabduction
- Identify tractable fragments

Thanks for listening!

James Brotherston, Diana Costa, Aquinas Hobor and John Wickerson.
Reasoning over Permissions Regions in Concurrent Separation Logic.
In *Proc. CAV-2020.*