# Concurrent separation logic

Ori Lahav Viktor Vafeiadis

1 September 2017

**Sequential separation logic**

- Separating conjunction
- Fault-avoiding interpretation of SL triples

**Concurrent separation logic (CSL)**

- Resource invariants
- Ownership transfer

**Soundness of CSL over SC**

- Interpretation of CSL triples

**Rule of constancy**

$$\frac{\{P\}\ c\ \{Q\} \qquad fv(R) \cap wr(C) = \emptyset}{\{P \wedge R\}\ c\ \{Q \wedge R\}}$$

**Disjoint parallel composition**

$$\frac{\begin{array}{ll}\{P_1\}\ c_1\ \{Q_1\} & fv(P_1, c_1, Q_1) \cap wr(c_2) = \emptyset \\ \{P_2\}\ c_2\ \{Q_2\} & fv(P_2, c_2, Q_2) \cap wr(c_1) = \emptyset\end{array}}{\{P_1 \wedge P_2\}\ c_1 \| c_2\ \{Q_1 \wedge Q_2\}}$$

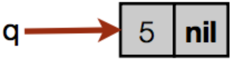
What about programs with pointers?

# Points-to assertions

SL provides a convenient syntax for describing the dynamically allocated memory (aka "heap").

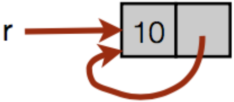

$p \mapsto 5$

$heap(p) = 5$




$q \mapsto 5, \textbf{nil}$

$heap(q) = 5 \wedge$
$heap(q+1) = \textbf{nil}$

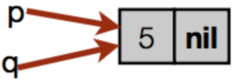



$r \mapsto 10, r$

$heap(r) = 10 \wedge$
$heap(r+1) = r$

# Two types of conjunction

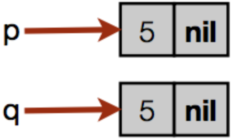
Classical conjunction:



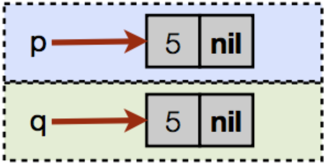

$p \mapsto 5, \textbf{nil} \wedge q \mapsto 5, \textbf{nil}$

$p \mapsto 5, \textbf{nil} \wedge p = q$

Separating conjunction:

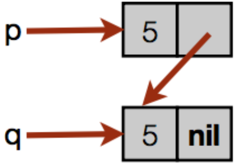


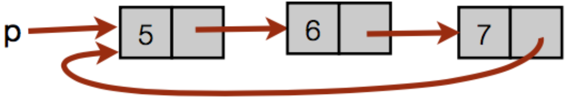

$p \mapsto 5, \textbf{nil} * q \mapsto 5, \textbf{nil}$



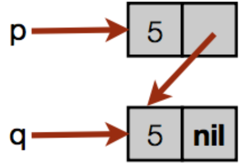

Split the heap in two parts: one satisfying $p \mapsto 5, \textbf{nil}$ and the other satisfying $q \mapsto 5, \textbf{nil}$.

Use a SL assertion to describe the following pictures:

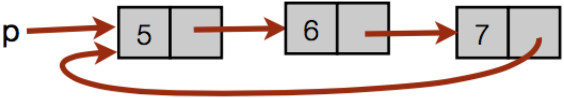


Use a SL assertion to describe the following pictures:

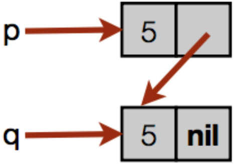



$p \mapsto 5, q \ast q \mapsto 5, \textbf{nil}$

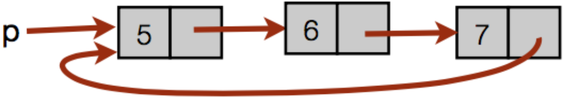


# Quick quiz

Use a SL assertion to describe the following pictures:




$p \mapsto 5, q \ast q \mapsto 5, \textbf{nil}$




$\exists q, r.\ p \mapsto 5, q \ast q \mapsto 6, r \ast r \mapsto 7, p$

Key concept of *ownership* :

- Resourceful reading of Hoare triples, $\{P\}\ c\ \{Q\}$
- To access a non-atomic location, you must own it:

$$\begin{array}{ccc} \{\mathrm{emp}\} & a := \mathbf{alloc} & \{a \mapsto \_\} \\ \{x \mapsto v\} & a := [x] & \{x \mapsto v \wedge a = v\} \\ \{x \mapsto v\} & [x] := v' & \{x \mapsto v'\} \end{array}$$

- Frame rule:

$$\frac{\{P\}\ c\ \{Q\} \qquad fv(R) \cap wr(C) = \emptyset}{\{P * R\}\ c\ \{Q * R\}}$$

- Disjoint parallelism:

$$\frac{\begin{array}{ll} \{P_1\}\ c_1\ \{Q_1\} & fv(P_1, c_1, Q_1) \cap wr(c_2) = \emptyset \\ \{P_2\}\ c_2\ \{Q_2\} & fv(P_2, c_2, Q_2) \cap wr(c_1) = \emptyset \end{array}}{\{P_1 * P_2\}\ c_1 \| c_2\ \{Q_1 * Q_2\}}$$

# Separation logic: Disjoint parallelism

$$\left\{x \mapsto 0 * y \mapsto 0\right\}$$

$$\begin{array}{l||l} a := [x]; & b := [y]; \\ [x] := a + 1; & [y] := b + 1; \end{array}$$

$$\left\{x \mapsto 1 * y \mapsto 1\right\}$$

Threads mind their own business!

# Separation logic: Disjoint parallelism

$$\left\{x \mapsto 0 * y \mapsto 0\right\}$$

$$\begin{array}{l||l}
\left\{x \mapsto 0\right\} & \left\{y \mapsto 0\right\} \\
a := [x]; & b := [y]; \\
\left\{x \mapsto 0 \land a = 0\right\} & \left\{y \mapsto 0 \land b = 0\right\} \\
[x] := a + 1; & [y] := b + 1; \\
\left\{x \mapsto 1\right\} & \left\{y \mapsto 1\right\}
\end{array}$$

$$\left\{x \mapsto 1 * y \mapsto 1\right\}$$

Simple programs are easy to verify!

$$
\begin{aligned}
E &::= x \mid n \mid E + E \mid E - E \mid \ldots \\
B &::= B \wedge B \mid \neg B \mid E = E \mid E \leq E \mid \ldots \\
C &::= \textbf{skip} \mid x := E \mid x := [E] \mid [E] := E \mid x := \textbf{alloc}(E) \mid \textbf{free}(E) \\
&\quad \mid C_1; C_2 \mid \textbf{if } B \textbf{ then } C_1 \textbf{ else } C_2 \mid \textbf{while } B \textbf{ do } C
\end{aligned}
$$

**Small-step operational semantics:**

$$(C, s, h) \to (C', s', h') \qquad s : \text{Stack} \triangleq \text{VarName} \to \text{Val}$$

$$(C, s, h) \to \textbf{abort} \qquad h : \text{Heap} \triangleq \text{Loc} \rightharpoonup \text{Val}$$

**Rules for sequential composition:**

$$(\textbf{skip}; C, s, h) \to (C, s, h) \qquad \frac{(C_1, s, h) \to (C_1', s', h')}{(C_1; C_2, s, h) \to (C_1'; C_2, s', h')}$$

$$C ::= \ldots \mid C_1 \| C_2$$

- Interleaving semantics:

$$\frac{(C_1, s, h) \to (C_1', s', h')}{(C_1 \| C_2, s, h) \to (C_1' \| C_2, s', h')} \qquad \frac{(C_2, s, h) \to (C_2', s', h')}{(C_1 \| C_2, s, h) \to (C_1 \| C_2', s', h')}$$

- Abort semantics:

$$\frac{(C_1, s, h) \to \textbf{abort}}{(C_1 \| C_2, s, h) \to \textbf{abort}} \qquad \frac{(C_2, s, h) \to \textbf{abort}}{(C_1 \| C_2, s, h) \to \textbf{abort}}$$

- Termination:

$$(\textbf{skip} \| \textbf{skip}, s, h) \to (\textbf{skip}, s, h)$$

# Atomic blocks

$$C ::= \ldots \mid \textbf{atomic } C$$

Atomic blocks execute in one step

$$\frac{(C, s, h) \rightarrow^* (\textbf{skip}, s', h')}{(\textbf{atomic } C, s, h) \rightarrow (\textbf{skip}, s', h')}$$

$$\frac{(C, s, h) \rightarrow^* \textbf{abort}}{(\textbf{atomic } C, s, h) \rightarrow \textbf{abort}}$$

**Note**

Normally, we also need a rule for non-terminating atomic blocks:

$$\frac{(C, s, h) \rightarrow^\omega}{(\textbf{atomic } C, s, h) \rightarrow (\textbf{atomic } C, s, h)}$$

# Multiple resources

- Lock declarations & conditional critial regions (CCRs)

$$C ::= \ldots \mid \textbf{resource } r \textbf{ in } C \mid \textbf{with } r \textbf{ when } B \textbf{ do } C$$
$$\mid \textbf{within } r \textbf{ do } C$$

- Enter a CCR

$$\frac{[\![B]\!](s)}{(\textbf{with } r \textbf{ when } B \textbf{ do } C, s, h) \rightarrow (\textbf{within } r \textbf{ do } C, s, h)}$$

- Execute body of a CCR

$$\frac{(C, s, h) \rightarrow (C', s', h') \qquad r \notin Locked(C')}{(\textbf{within } r \textbf{ do } C, s, h) \rightarrow (\textbf{within } r \textbf{ do } C', s', h')}$$

- Exit the CCR

$$(\textbf{within } r \textbf{ do skip}, s, h) \rightarrow (\textbf{skip}, s, h)$$

# Operational semantics for parallel composition

$$\frac{(C_1, s, h) \to (C_1', s', h') \qquad Locked(C_1') \cap Locked(C_2) = \emptyset}{(C_1 \| C_2, s, h) \to (C_1' \| C_2, s', h')}$$

$$\frac{(C_2, s, h) \to (C_2', s', h') \qquad Locked(C_1) \cap Locked(C_2') = \emptyset}{(C_1 \| C_2, s, h) \to (C_1 \| C_2', s', h')}$$

where

$$Locked(C) \triangleq \{r \mid \exists C'. \ (\textbf{within } r \textbf{ do } C') \text{ is a subterm of } C\}$$

Ensures that commands are well-formed:

$$\begin{aligned}
&wf(\textbf{skip}) \triangleq true \\
&wf(C_1; C_2) \triangleq wf(C_1) \wedge wf(C_2) \wedge (Locked(C_2) = \emptyset) \\
&wf(C_1 \| C_2) \triangleq wf(C_1) \wedge wf(C_2) \wedge (Locked(C_1) \cap Locked(C_2) = \emptyset) \\
&wf(\textbf{with } r \textbf{ when } B \textbf{ do } C) \triangleq wf(C) \wedge (Locked(C) = \emptyset) \\
&wf(\textbf{within } r \textbf{ do } C) \triangleq wf(C) \wedge r \notin Locked(C)
\end{aligned}$$

# Some proof rules

$$\frac{\Gamma \vdash \{P_1\}\ C_1\ \{Q_1\} \qquad fv(\Gamma, P_1, C_1, Q_1) \cap wr(C_2) = \emptyset \quad \Gamma \vdash \{P_2\}\ C_2\ \{Q_2\} \qquad fv(\Gamma, P_2, C_2, Q_2) \cap wr(C_1) = \emptyset}{\Gamma \vdash \{P_1 * P_2\}\ C_1 \| C_2\ \{Q_1 * Q_2\}}\ (\textsc{Par})$$

$$\frac{\Gamma \vdash \{(P * J) \wedge B\}\ C\ \{Q * J\}}{\Gamma, r : J \vdash \{P\}\ \textbf{with}\ r\ \textbf{when}\ B\ \textbf{do}\ C\ \{Q\}}\ (\textsc{With})$$

$$\frac{\Gamma, r : J \vdash \{P\}\ C\ \{Q\} \qquad fv(J) \cap wr(C) = \emptyset}{\Gamma \vdash \{P * J\}\ \textbf{resource}\ r\ \textbf{in}\ C\ \{Q * J\}}\ (\textsc{Res})$$

$$\frac{\Gamma \vdash \{P\}\ C\ \{Q\} \qquad fv(R) \cap wr(C) = \emptyset}{\Gamma \vdash \{P * R\}\ C\ \{Q * R\}}\ (\textsc{Frame})$$

**Note**

The rules have draconian variable side-conditions.

# Proof rules for atomic blocks

$$\frac{J \vdash \{P_1\}\ C_1\ \{Q_1\} \qquad fv(J, P_1, C_1, Q_1) \cap wr(C_2) = \emptyset \qquad J \vdash \{P_2\}\ C_2\ \{Q_2\} \qquad fv(J, P_2, C_2, Q_2) \cap wr(C_1) = \emptyset}{J \vdash \{P_1 * P_2\}\ C_1 \| C_2\ \{Q_1 * Q_2\}} \ (\textsc{Par})$$

$$\frac{\mathbf{emp} \vdash \{P * J\}\ C\ \{Q * J\}}{J \vdash \{P\}\ \mathbf{atomic}\ C\ \{Q\}} \ (\textsc{Atom})$$

$$\frac{J * R \vdash \{P\}\ C\ \{Q\} \qquad fv(R) \cap wr(C) = \emptyset}{J \vdash \{P * R\}\ C\ \{Q * R\}} \ (\textsc{Share})$$

$$\frac{J \vdash \{P\}\ C\ \{Q\} \qquad fv(R) \cap wr(C) = \emptyset}{J \vdash \{P * R\}\ C\ \{Q * R\}} \ (\textsc{Frame})$$

# Hoare triples (partial correctness)

$$\models \{P\}\ C\ \{Q\}$$

if and only if

$$\forall s\, h\, s'\, h'.\ s, h \models P\ \wedge\ (C, s, h) \rightarrow^* (\mathbf{skip}, s', h') \Rightarrow s', h' \models Q$$

# Hoare triples (partial correctness)

$$\models \{P\}\ C\ \{Q\}$$

if and only if

$$\forall s\, h\, s'\, h'.\ s, h \models P\ \wedge\ (C, s, h) \to^* (\mathbf{skip}, s', h') \Rightarrow s', h' \models Q$$

if and only if

$$\forall s\, h.\ s, h \models P \Rightarrow \\ (\forall s'\, h'.\ (C, s, h) \to^* (\mathbf{skip}, s', h') \Rightarrow s', h' \models Q)$$

# Hoare triples (partial correctness)

$$\models \{P\}\ C\ \{Q\}$$

if and only if

$$\forall s\, h\, s'\, h'.\ s, h \models P \ \wedge\ (C, s, h) \rightarrow^{*} (\mathbf{skip}, s', h') \Rightarrow s', h' \models Q$$

if and only if

$$\forall s\, h.\ s, h \models P \Rightarrow$$
$$(\forall s'\, h'.\ (C, s, h) \rightarrow^{*} (\mathbf{skip}, s', h') \Rightarrow s', h' \models Q)$$

if and only if

$$\forall s\, h.\ s, h \models P \Rightarrow$$
$$(\forall m.\ \forall s'\, h'.\ (C, s, h) \rightarrow^{m} (\mathbf{skip}, s', h') \Rightarrow s', h' \models Q)$$

# Hoare triples (partial correctness)

$$\models \{P\}\ C\ \{Q\}$$

if and only if

$$\forall s\, h\, s'\, h'.\ s, h \models P\ \land\ (C, s, h) \rightarrow^{*} (\mathbf{skip}, s', h') \Rightarrow s', h' \models Q$$

if and only if

$$\forall s\, h.\ s, h \models P \Rightarrow (\forall s'\, h'.\ (C, s, h) \rightarrow^{*} (\mathbf{skip}, s', h') \Rightarrow s', h' \models Q)$$

if and only if

$$\forall s\, h.\ s, h \models P \Rightarrow (\forall m.\ \forall s'\, h'.\ (C, s, h) \rightarrow^{m} (\mathbf{skip}, s', h') \Rightarrow s', h' \models Q)$$

if and only if

$$\forall s\, h\, n.\ s, h \models P \Rightarrow (\forall m < n.\ \forall s'\, h'.\ (C, s, h) \rightarrow^{m} (\mathbf{skip}, s', h') \Rightarrow s', h' \models Q)$$

$$\models \{P\}\ C\ \{Q\} \text{ iff } \forall s\, h\, n.\ s, h \models P \Rightarrow \mathrm{safe}_n(C, s, h, Q)$$

where

$$\mathrm{safe}_n(C, s, h, Q) \triangleq$$
$$(\forall m < n.\ \forall s'\, h'.\ (C, s, h) \to^m (\mathbf{skip}, s', h') \Rightarrow s', h' \models Q)$$

As an inductive definition:

$$\mathrm{safe}_0(C, s, h, Q) = \mathit{true}$$
$$\mathrm{safe}_{n+1}(C, s, h, Q) =$$
$$(C = \mathbf{skip} \Rightarrow s, h \models Q)$$
$$\wedge\ (\forall C'\, s'\, h'.\ (C, s, h) \to (C', s', h')$$
$$\Rightarrow \mathrm{safe}_n(C', s', h', Q))$$

# Configuration safety

$$\models \{P\}\ C\ \{Q\} \text{ iff } \forall s\, h\, n.\ s, h \models P \Rightarrow \mathrm{safe}_n(C, s, h, Q)$$

$$\begin{aligned}
&\mathrm{safe}_0(C, s, h, Q) \triangleq \mathit{true} \\
&\mathrm{safe}_{n+1}(C, s, h, Q) \triangleq \\
&\qquad (C = \textbf{skip} \Rightarrow s, h \models Q) \\
&\quad \wedge (\forall C'\, s'\, h'.\ (C, s, h) \rightarrow (C', s', h') \\
&\qquad\quad \Rightarrow \mathrm{safe}_n(C', s', h', Q))
\end{aligned}$$

$$\models \{P\}\ C\ \{Q\} \text{ iff } \forall s\, h\, n.\ s, h \models P \Rightarrow \mathrm{safe}_n(C, s, h, Q)$$

$$\begin{aligned}
&\mathrm{safe}_0(C, s, h, Q) \triangleq \mathit{true} \\
&\mathrm{safe}_{n+1}(C, s, h, Q) \triangleq \\
&\qquad (C = \textbf{skip} \Rightarrow s, h \models Q) \\
&\quad \wedge (\neg(C, s, h) \rightarrow \textbf{abort}) \\
&\quad \wedge (\forall C'\, s'\, h'.\ (C, s, h) \rightarrow (C', s', h') \\
&\qquad\quad \Rightarrow \mathrm{safe}_n(C', s', h', Q))
\end{aligned}$$

"Well-specified programs don't go wrong."

## "Bake in" the frame rule

$$\models \{P\}\ C\ \{Q\} \text{ iff } \forall s\, h\, n.\ s, h \models P \Rightarrow \mathrm{safe}_n(C, s, h, Q)$$

$$\mathrm{safe}_0(C, s, h, Q) \triangleq \mathit{true}$$
$$\begin{aligned}
\mathrm{safe}_{n+1}(C, s, h, Q) \triangleq&\\
&(C = \mathbf{skip} \Rightarrow s, h \models Q)\\
&\wedge (\forall h_{\mathrm{F}}.\ \neg(C, s, h \uplus h_{\mathrm{F}}) \to \mathbf{abort})\\
&\wedge (\forall h_{\mathrm{F}}\ C'\ s'\ h'.\ (C, s, h \uplus h_{\mathrm{F}}) \to (C', s', h')\\
&\qquad \Rightarrow \exists h''.\ h' = h'' \uplus h_{\mathrm{F}} \wedge \mathrm{safe}_n(C', s', h'', Q))
\end{aligned}$$

**Note**

- No need for safety monotonicity & frame property.
- The same definition works for permissions.
  (where $\uplus$ becomes the addition of permission-heaps)

$$J \models \{P\}\ C\ \{Q\} \text{ iff } \forall s\ h\ n.\ s, h \models P \Rightarrow \mathrm{safe}_n(C, s, h, J, Q)$$

$$\begin{aligned}
&\mathrm{safe}_0(C, s, h, J, Q) \triangleq \mathit{true} \\
&\mathrm{safe}_{n+1}(C, s, h, J, Q) \triangleq \\
&\quad (C = \mathbf{skip} \Rightarrow s, h \models Q) \\
&\quad \wedge (\forall h_{\mathrm{J}}\ h_{\mathrm{F}}.\ s, h_{\mathrm{J}} \models J \Rightarrow \neg(C, s, h \uplus h_{\mathrm{J}} \uplus h_{\mathrm{F}}) \to \mathbf{abort}) \\
&\quad \wedge (\forall h_{\mathrm{J}}\ h_{\mathrm{F}}\ C'\ s'\ h'.\ (C, s, h \uplus h_{\mathrm{J}} \uplus h_{\mathrm{F}}) \to (C', s', h') \\
&\qquad\qquad \wedge s, h_{\mathrm{J}} \models J \\
&\qquad \Rightarrow \exists h''\ h'_{\mathrm{J}}.\ h' = h'' \uplus h'_{\mathrm{J}} \uplus h_{\mathrm{F}} \\
&\qquad\qquad \wedge s, h'_{\mathrm{J}} \models J \\
&\qquad\qquad \wedge \mathrm{safe}_n(C', s', h'', J, Q))
\end{aligned}$$

- Add heap $h_{\mathrm{J}}$ satisfying the resource invariant, $J$.
- Resource invariant must be re-established in $h'_{\mathrm{F}}$.

# Multiple resources

$$\Gamma \models \{P\}\ C\ \{Q\} \text{ iff } \forall s\, h\, n.\ s, h \models P \Rightarrow \mathrm{safe}_n(C, s, h, \Gamma, Q)$$

$$\begin{aligned}
&\mathrm{safe}_0(C, s, h, \Gamma, Q) \triangleq \mathit{true}\\
&\mathrm{safe}_{n+1}(C, s, h, \Gamma, Q) \triangleq\\
&\qquad (C = \textbf{skip} \Rightarrow s, h \models Q)\\
&\quad \wedge (\forall h_{\mathrm{F}}.\ \neg(C, s, h \uplus h_{\mathrm{F}}) \to \textbf{abort})\\
&\quad \wedge (\forall h_{\mathrm{J}}\, h_{\mathrm{F}}\, C'\, s'\, h'.\ (C, s, h \uplus h_{\mathrm{J}} \uplus h_{\mathrm{F}}) \to (C', s', h')\\
&\qquad\qquad\qquad\qquad \wedge s, h_{\mathrm{J}} \models \circledast_{r \in Locked(C') \setminus Locked(C)} \Gamma(r)\\
&\qquad\quad \Rightarrow \exists h''\, h'_{\mathrm{J}}.\ h' = h'' \uplus h'_{\mathrm{J}} \uplus h_{\mathrm{F}}\\
&\qquad\qquad\qquad\qquad \wedge s, h'_{\mathrm{J}} \models \circledast_{r \in Locked(C) \setminus Locked(C')} \Gamma(r)\\
&\qquad\qquad\qquad\qquad \wedge \mathrm{safe}_n(C', s', h'', \Gamma, Q))
\end{aligned}$$

- Assume res. invariant satisfied only for acquired locks ($h_{\mathrm{J}}$).
- Ensure res. invariant satisfied for released locks ($h'_{\mathrm{J}}$).

# Further reading


- Separation logic: A logic for shared mutable data structures.
  John C. Reynolds: LICS 2002: 55-74
- Resources, concurrency, and local reasoning.
  Peter W. O'Hearn, TCS 375(1-3): 271-307 (2007)
- Concurrent separation logic and operational semantics.
  Viktor Vafeiadis, ENTCS 276: 335-351 (2011)