Fine-Grained Concurrent Separation Logic (FCSL)

Irene Yoon
PLClub 06/05/2020
It’s a **Family Tree**
Rely-Guarantee

- “Rely” and “Guarantee” transitions (relations on states)
  - **Rely** — transitions allowed to **others**
  - **Guarantee** — transitions allowed to **myself**

Idea: **forking shuffle** — *Parallel threads become each other’s “other”*
Subjectivity: self and other
Concurrent Resources

- State transition systems with subjective auxiliary state

- Self - state controlled by “me”

- Other - state controlled by “all other threads”

- Joint - modified by everyone, “as allowed by transitions”

[A. Nanevski et al. 2014]
Ownership (*Subjectivity*)

- Thread-relative view of resources: self, other, resource

\[ \ell \overset{s}{\leftrightarrow} v \quad \ell \overset{o}{\leftrightarrow} v \quad \ell \overset{j}{\leftrightarrow} v \]
Ownership (Subjectivity)

- Thread-relative view of resources: self, other, resource

\[
\ell^s \rightarrow v \quad \ell^o \rightarrow v \quad \ell^j \rightarrow v
\]

- Shared lock
Concurrent Resources

- State transition systems with subjective auxiliary state (*Concurroids*)

\[ \{p\} c \{q\} @ U \]

- \( U \) defines resources touched by \( c \), their transitions, and invariants
Logical State Split

• Values of resources form a Partial Commutative Monoid \((\oplus, \text{unit})\)

Illustration by Ilya Sergey
Concurrent Resources

- State transition systems with subjective auxiliary state (*Concurroids*)

\[
\{p\} \ c \ \{q\} @ U
\]

- U defines resources touched by c, their transitions, and invariants
Modeling fine-grained changes to state

- Relation between states of possibly *multiple resources*
Modeling fine-grained changes to state

- Relation between states of possibly multiple resources

- Internal (preserves “heap footprint”, i.e. domain of heap remains constant)

\[
\begin{align*}
\text{priv} & \mapsto h_S \quad * (\text{lock} \mapsto (\text{Own}, a_S) \land \text{lock} \mapsto ((lk \rightarrow \text{false}) \cup h)) \leadsto \\
\text{priv} & \mapsto (h_S \cup h) \quad * (\text{lock} \mapsto (\text{Own}, a_S) \land \text{lock} \mapsto (lk \rightarrow \text{true}))
\end{align*}
\]
Modeling fine-grained changes to state

• Relation between states of possibly *multiple resources*

• Internal (preserves “heap footprint”, i.e. domain of heap remains constant)

\[
\begin{align*}
\text{priv} & \mapsto h_S \\
\text{priv} & \mapsto (h_S \cup h) \\
\text{lock} & \mapsto (\text{Own}, a_S) \land \text{lock} \mapsto ((lk \rightarrow \text{false}) \cup h)) \\
\text{lock} & \mapsto (\text{Own}, a_S) \land \text{lock} \mapsto (lk \rightarrow \text{true})
\end{align*}
\]

*Transition for threads specified by concurroid*

“What might I do?”
Modeling fine-grained changes to state

- Relation between states of possibly multiple resources

- External (can release or acquire heaps, can build internal from external)

\[
\begin{align*}
\text{priv} & \xrightarrow{s} h_S \xrightarrow{+h} \text{priv} \xrightarrow{s} (h_S \cup h) \\
\text{lock} & \xrightarrow{s} (\text{Own}, a_S) \land \text{lock} \xrightarrow{j} ((lk \rightarrow \text{false}) \cup h) \xrightarrow{-h} \text{lock} \xrightarrow{s} (\text{Own}, a_S) \land \text{lock} \xrightarrow{j} lk \rightarrow \text{true}
\end{align*}
\]
Modeling fine-grained changes to state

• Relation between states of possibly *multiple resources*

• External (can release or acquire heaps, can build internal from external)

\[
\begin{align*}
\text{priv} & \xleftrightarrow{s} h_S \quad \xrightarrow{+h} \quad \text{priv} \xleftrightarrow{s} (h_S \cup h) \\
\text{lock} & \xleftrightarrow{s} (\text{Own}, a_S) \land \text{lock} \xrightarrow{j} ((\text{lk} \rightarrow \text{false}) \cup h) \quad \xrightarrow{-h} \quad \text{lock} \xleftrightarrow{s} (\text{Own}, a_S) \land \text{lock} \xrightarrow{j} \text{lk} \rightarrow \text{true}
\end{align*}
\]

“dangling wires” for composing polarized transitions

“How might I communicate with others?”
Modeling fine-grained changes to state

- Relation between states of possibly *multiple resources*

- External (can release or acquire heaps, can build internal from external)

\[
\begin{align*}
\text{priv} & \xleftarrow{s} h_S \xrightarrow{+h} \text{priv} \xleftarrow{s} (h_S \cup h) \\
\text{lock} & \xleftarrow{s} (\text{Own, a}_S) \land \text{lock} \xrightarrow{j} ((l_k \rightarrow \text{false}) \cup h) \xrightarrow{-h} \text{lock} \xleftarrow{s} (\text{Own, a}_S) \land \text{lock} \xrightarrow{j} l_k \rightarrow \text{true}
\end{align*}
\]

"dangling wires" for composing polarized transitions

"How might I communicate with others?"

\[
CSL_{\{\text{lock, l}_k, I\}} = P \times L_{\{\text{lock, l}_k, I\}}
\]
Injection : Combining Concurroids

- **Idea**: Generalization of resource context weakening
Injection : Combining Concurroids

- **Idea**: Generalization of resource context weakening

- Injection (or *coercion*) of concurroids into a larger concurroid

  \[
  \{p\} c \{q\}@U \quad r \text{ stable under } V \\
  \{p * r\} \text{ inject } c \{q * r\}@U \bowtie V \quad \text{INJECT}
  \]

- “**stable**” : remains valid under arbitrary transitions the *other* thread takes over labels in V

- V can change U, but only on external transitions
Private Heap

read $x$

write $x \nu$
Lock on Private Heap

acquire
Combining Concurroids

\[
\begin{align*}
\{ \text{priv } \mapsto^S \text{ empty } \land \text{ lock } \mapsto^S (\text{Own}, 0) \} \\
\text{acquire} \\
\{ \exists a_0. \text{priv } \mapsto^S x \rightarrow a_0 \land (\text{lock } \mapsto^S (\text{Own}, 0) \land \text{lock } \mapsto^o (\neg, a_0)) \} \\
\text{read } x
\end{align*}
\]
Hiding

- **Idea**: Private heap of a thread is *promoted* into a (freshly allocated) shared resource
Hiding

- **Idea**: Private heap of a thread is *promoted* into a (freshly allocated) shared resource

- Generalization of scoped resources to fine-grained resources

\[
\frac{\Gamma, r : I \vdash \{p\} c \{q\}}{\Gamma \vdash \{p \ast I\} \text{ resource } r \text{ in } c \{q \ast I\}} \quad \text{RESOURCECSL}
\]
Hiding

- **Idea:** Private heap of a thread is *promoted* into a (freshly allocated) shared resource

- Generalization of scoped resources to fine-grained resources

\[
\Gamma, r : I \vdash \{p\} c \{q\} \\
\Gamma \vdash \{p \times I\} \text{ resource } r \text{ in } c \{q \times I\}
\]

- **Hiding** generalizes this to concurroids, where a concurroid is constructed from a private heap in the scope of c.

\[
\text{hide}_{\phi,g} c @ (P \times U) \times V \quad \text{HIDE} \quad \text{(simplified****)}
\]
Concurrent Resources

- State transition systems with subjective auxiliary state (*Concurroids*)

- Subjectivity
- Composing resources
- Scoped reasoning

\[ \{p\} c \{q\} \oplus U \]

[Illustration by Ilya Sergey]
Logical State Split

- Values of resources form a **Partial Commutative Monoid** \((\oplus, \text{unit})\)

![Diagram](Illustration by Ilya Sergey)
History: Temporal Specification
Keeping track of History

[I. Sergey et al. 2015]

- Viewing atomic operations as “timestamps”
- Time increases at every abstract operation
Keeping track of History

[I. Sergey et al. 2015]

- Viewing atomic operations as “timestamps”
- Time increases at every abstract operation

\[ \text{push}(x) \]
History as PCM’s

• Partial finite maps (nat -> AbsOp)

• Join: disjoint union

• Unit: empty history
History as PCM’s

- Partial finite maps (nat -> AbsOp)
- Join: disjoint union
- Unit: empty history

- Subjective History
Stack Specification

\text{push}(x)
FCSL

- Subjectivity
- PCM’s
- Temporal Specification
- Fine-grain Concurrent Resources

https://ilyasergey.net/projects/fcsl/

Thanks!

Fig. 1. CSL Family Tree (courtesy of Ilya Sergey)
Parallel Composition

- Subjective separating conjunction

\[
\frac{\{p_1\} c_1 \{q_1\} @ U \quad \{p_2\} c_2 \{q_2\} @ U}{\{p_1 \otimes p_2\} c_1 \parallel c_2 \{q_1 \otimes q_2\} @ U} \quad \text{PAR}
\]

- Parent thread forks and splits self component

\[
\ell \xrightarrow{s} a \oplus b \land \ell \xrightarrow{o} c \implies (\ell \xrightarrow{s} a \land \ell \xrightarrow{o} c \oplus b) \otimes (\ell \xrightarrow{s} b \land \ell \xrightarrow{o} c \oplus a)
\]

- Typically, only need to think about this bi-implication:

\[
\ell \xrightarrow{s} a \oplus b \iff \ell \xrightarrow{s} a \otimes \ell \xrightarrow{s} b
\]

- Reminiscent of “forking shuffle” of rely-guarantee