Syntactic Control of Interference for Concurrent Separation Logic

Uday S. Reddy[1]

[1]School of Computer Science
University of Birmingham

Concurrency Workshop, Dublin, Apr 2011

Outline

Concurrent Programming

- Concurrent programming requires a tight control over resources
  - Be clear about what resources are being used by each process
  - Ensure that these resources are disjoint as far as possible
  - Devise suitable protocols for sharing resources when necessary
- Concurrent Separation Logic does all of these very well for heap locations.
- But it brushes under the carpet the very same issues for variables.
- This work is an attempt to change that.

How are variables different?

- Variables are syntactic symbols.
- It should be possible to control their usage in the formulation of "syntax".
- Variables participate in expressions, which represent read-only uses of the resources.
- Need to make this convenient.

Concurrent Separation Logic

Parallel composition

$$\frac{\{P_1\}\ C_1\ \{Q_1\} \quad \{P_2\}\ C_2\ \{Q_2\}}{\{P_1 \star P_2\}\ C_1 \parallel C_2\ \{Q_1 \star Q_2\}}$$

Owicki-Gries:

- if no variable free in P_i or Q_i is changed in C_j ($j \neq i$).
- if a variable x is changed in a process C_i, it cannot appear in C_j ($j \neq i$) unless it belongs to a resource.
- if a variable x belongs to a resource, it cannot appear in a parallel process except in a critical section for r.

Brookes:

- $\mathbf{free}(P_i, Q_i) \cap \mathbf{writes}(C_2) = \mathbf{free}(P_2, Q_2) \cap \mathbf{writes}(C_1) = \emptyset$
- $(\mathbf{free}(C_1) \cap \mathbf{writes}(C_2)) \cup (\mathbf{free}(C_2) \cap \mathbf{writes}(C_1)) \subseteq \mathbf{owned}(\Gamma)$ where Γ lists all the resources in the context

Concurrent Separation Logic

Parallel composition

$$\frac{\{P_1\}\ C_1\ \{Q_1\} \quad \{P_2\}\ C_2\ \{Q_2\}}{\{P_1 \star P_2\}\ C_1 \parallel C_2\ \{Q_1 \star Q_2\}}$$

Owicki-Gries:

- if no variable free in P_i or Q_i is changed in C_j ($j \neq i$).
- if a variable x is changed in a process C_i, it cannot appear in C_j ($j \neq i$) unless it belongs to a resource.
- if a variable x belongs to a resource, it cannot appear in a parallel process except in a critical section for r.

Brookes:

- $\mathbf{free}(P_i, Q_i) \cap \mathbf{writes}(C_2) = \mathbf{free}(P_2, Q_2) \cap \mathbf{writes}(C_1) = \emptyset$
- $(\mathbf{free}(C_1) \cap \mathbf{writes}(C_2)) \cup (\mathbf{free}(C_2) \cap \mathbf{writes}(C_1)) \subseteq \mathbf{owned}(\Gamma)$ where Γ lists all the resources in the context

Concurrent Separation Logic

Parallel composition

$$\frac{\{P_1\}\ C_1\ \{Q_1\} \quad \{P_2\}\ C_2\ \{Q_2\}}{\{P_1 \star P_2\}\ C_1 \parallel C_2\ \{Q_1 \star Q_2\}}$$

Owicki-Gries:

- if no variable free in P_i or Q_i is changed in C_j ($j \neq i$).
- if a variable x is changed in a process C_i, it cannot appear in C_j ($j \neq i$) unless it belongs to a resource.
- if a variable x belongs to a resource, it cannot appear in a parallel process except in a critical section for r.

Brookes:

- $\mathbf{free}(P_i, Q_i) \cap \mathbf{writes}(C_2) = \mathbf{free}(P_2, Q_2) \cap \mathbf{writes}(C_1) = \emptyset$
- $(\mathbf{free}(C_1) \cap \mathbf{writes}(C_2)) \cup (\mathbf{free}(C_2) \cap \mathbf{writes}(C_1)) \subseteq \mathbf{owned}(\Gamma)$ where Γ lists all the resources in the context

$$\frac{\{P_1\}\ C_1\ \{Q_1\} \quad \{P_2\}\ C_2\ \{Q_2\}}{\{P_1 \star P_2\}\ C_1 \parallel C_2\ \{Q_1 \star Q_2\}}$$

Owicki-Gries:

- if no variable free in P_i or Q_i is changed in C_j ($j \neq i$).
- if a variable x is changed in a process C_i, it cannot appear in C_j ($j \neq i$) unless it belongs to a resource.
- if a variable x belongs to a resource, it cannot appear in a parallel process except in a critical section for r.

Brookes:

- $\mathbf{free}(P_i, Q_i) \cap \mathbf{writes}(C_2) = \mathbf{free}(P_2, Q_2) \cap \mathbf{writes}(C_1) = \emptyset$
- $(\mathbf{free}(C_1) \cap \mathbf{writes}(C_2)) \cup (\mathbf{free}(C_2) \cap \mathbf{writes}(C_1)) \subseteq \mathbf{owned}(\Gamma)$ where Γ lists all the resources in the context

$$\frac{\{P_1\}\ C_1\ \{Q_1\} \quad \{P_2\}\ C_2\ \{Q_2\}}{\{P_1 \star P_2\}\ C_1 \parallel C_2\ \{Q_1 \star Q_2\}}$$

Owicki-Gries:

- if no variable free in P_i or Q_i is changed in C_j ($j \neq i$).
- if a variable x is changed in a process C_i, it cannot appear in C_j ($j \neq i$) unless it belongs to a resource.
- if a variable x belongs to a resource, it cannot appear in a parallel process except in a critical section for r.

Brookes:

- $\mathbf{free}(P_i, Q_i) \cap \mathbf{writes}(C_2) = \mathbf{free}(P_2, Q_2) \cap \mathbf{writes}(C_1) = \emptyset$
- $(\mathbf{free}(C_1) \cap \mathbf{writes}(C_2)) \cup (\mathbf{free}(C_2) \cap \mathbf{writes}(C_1)) \subseteq \mathbf{owned}(\Gamma)$ where Γ lists all the resources in the context

Concurrent Separation Logic (contd)

$$\frac{\Gamma \vdash \{P \star R_r \wedge B\}\ C\ \{Q \star R_r\}}{\Gamma, r(X) : R_r \vdash \{P\}\ \textbf{with}\ r\ \textbf{when}\ B\ \textbf{do}\ C\ \textbf{od}\ \{Q\}}\ \textit{CRIT}$$

Owicki-Gries:

- No variable free in P or Q is changed in any "other process".

The reference to "other processes" makes this rule non-compositional.

Brookes:

- $r \notin \mathrm{dom}(\Gamma)$
- $X \cap \textbf{owned}(\Gamma) = \emptyset$
- $\textbf{free}(R_i) \cap \textbf{owned}(\Gamma) = \emptyset$
- $\textbf{free}(P, Q) \cap X = \emptyset$

These conditions are compositional. But they are not enough!

Concurrent Separation Logic (contd)

$$\frac{\Gamma \vdash \{P \star R_r \wedge B\}\ C\ \{Q \star R_r\}}{\Gamma, r(X) : R_r \vdash \{P\}\ \textbf{with}\ r\ \textbf{when}\ B\ \textbf{do}\ C\ \textbf{od}\ \{Q\}}\ \textit{CRIT}$$

Owicki-Gries:

- No variable free in P or Q is changed in any "other process".

The reference to "other processes" makes this rule non-compositional.

Brookes:

- $r \notin \mathrm{dom}(\Gamma)$
- $X \cap \textbf{owned}(\Gamma) = \emptyset$
- $\textbf{free}(R_i) \cap \textbf{owned}(\Gamma) = \emptyset$
- $\textbf{free}(P, Q) \cap X = \emptyset$

These conditions are compositional. But they are not enough!

Concurrent Separation Logic (contd)

$$\frac{\Gamma \vdash \{P \star R_r \wedge B\}\ C\ \{Q \star R_r\}}{\Gamma, r(X) : R_r \vdash \{P\}\ \textbf{with}\ r\ \textbf{when}\ B\ \textbf{do}\ C\ \textbf{od}\ \{Q\}}\ \textit{CRIT}$$

Owicki-Gries:

- No variable free in P or Q is changed in any "other process".

The reference to "other processes" makes this rule non-compositional.

Brookes:

- $r \notin \mathrm{dom}(\Gamma)$
- $X \cap \textbf{owned}(\Gamma) = \emptyset$
- $\textbf{free}(R_i) \cap \textbf{owned}(\Gamma) = \emptyset$
- $\textbf{free}(P, Q) \cap X = \emptyset$

These conditions are compositional. But they are not enough!

Concurrent Separation Logic (contd)

$$\frac{\Gamma \vdash \{P \star R_r \wedge B\}\ C\ \{Q \star R_r\}}{\Gamma, r(X) : R_r \vdash \{P\}\ \textbf{with}\ r\ \textbf{when}\ B\ \textbf{do}\ C\ \textbf{od}\ \{Q\}}\ \textit{CRIT}$$

Owicki-Gries:

- No variable free in P or Q is changed in any "other process".

The reference to "other processes" makes this rule non-compositional.

Brookes:

- $r \notin \mathrm{dom}(\Gamma)$
- $X \cap \textbf{owned}(\Gamma) = \emptyset$
- $\textbf{free}(R_i) \cap \textbf{owned}(\Gamma) = \emptyset$
- $\textbf{free}(P, Q) \cap X = \emptyset$

These conditions are compositional. But they are not enough!

Concurrent Separation Logic (contd)

$$\frac{\Gamma \vdash \{P \star R_r \wedge B\}\ C\ \{Q \star R_r\}}{\Gamma, r(X) : R_r \vdash \{P\}\ \textbf{with}\ r\ \textbf{when}\ B\ \textbf{do}\ C\ \textbf{od}\ \{Q\}}\ \textit{CRIT}$$

Owicki-Gries:

- No variable free in P or Q is changed in any "other process".

The reference to "other processes" makes this rule non-compositional.

Brookes:

- $r \notin \mathrm{dom}(\Gamma)$
- $X \cap \textbf{owned}(\Gamma) = \emptyset$
- $\textbf{free}(R_i) \cap \textbf{owned}(\Gamma) = \emptyset$
- $\textbf{free}(P, Q) \cap X = \emptyset$

These conditions are compositional. But they are not enough!

Syntactic Control of Interference

[Reynolds 1978]
Two terms

$$T_1 \qquad T_2$$

are deemed to *interfere*:

- if any free variable actively used in one term is used in the other term (as a free variable again).
- The two terms can share passively used free variables.
- **Procedure call:** $F \quad (\quad A \quad)$
- **local declarations:** $\textbf{let} \quad x = A \quad \textbf{in} \quad B$
- **Parallel composition:** $C_1 \quad \| \quad C_2$

Further work on SCI

[O'Hearn 1991] *Linear logic and interference control*, CTCS
[O'Hearn 1993] *A model for syntactic control of interference*, MSCS
[Reddy 1996] *Global state considered unnecessary: An introduction to object-based semantics*, J. LSP
[O'Hearn, Power, Takeyama, Tennent 1995] *Syntactic control of interference Revisited*, MFPS
[McCusker, 2007] *Categorical models of syntactic control of interference revisited, revisited*, LMSJCM
[McCusker, 2010] *A graph model for imperative computation*, LMCS
[Ghica, Murawski, Ong] *Syntactic control of concurrency*, TCS
[Ghica 2007] *Geometry of synthesis: A structured approach to VLSI design*, POPL.

Further work on SCI (contd)

- Bunched typing arose from an effort combine SCI with regular function application:
  [O'Hearn 2003] *On bunched typing*, JFP
- BI arose from viewing these type systems as logics:
  [Pym, O'Hearn 1999] *The logic of bunched implications*, BSL
- Separation Logic uses BI as its assertion logic.
- **However:** Remarkably, SCI has never been used to structure programming logics, which was its original motivation!

O'Hearn's formulation of SCI

(inspired by Linear Logic)

The contexts are combined multiplicatively.

$$\frac{\Sigma_1 \vdash F : \tau_1 \to \tau_2 \quad \Sigma_2 \vdash A : \tau_1}{\Sigma_1, \Sigma_2 \vdash F(A) : \tau_2}$$

$$\frac{\Sigma_1 \vdash A : \tau_1 \quad \Sigma_2, x : \tau_1 \vdash B : \tau_2}{\Sigma_1, \Sigma_2 \vdash \textbf{let } x = A \textbf{ in } B : \tau_2}$$

$$\frac{\Sigma_1 \vdash C_1 \textbf{ Comm} \quad \Sigma_2 \vdash C_2 \textbf{ Comm}}{\Sigma_1, \Sigma_2 \vdash C_1 \parallel C_2}$$

SCI Revisited

deals with passive uses

Add a separate zone of passively used free variables:

$$\frac{\Sigma_1 \mid \Pi \vdash F : \tau_1 \to \tau_2 \quad \Sigma_2 \mid \Pi \vdash A : \tau_1}{\Sigma_1, \Sigma_2 \mid \Pi \vdash F(A) : \tau_2}$$

$$\frac{\Sigma_1 \mid \Pi \vdash A : \tau_1 \quad \Sigma_2, x : \tau_1 \mid \Pi \vdash B : \tau_2}{\Sigma_1, \Sigma_2 \mid \Pi \vdash \textbf{let } x = A \textbf{ in } B : \tau_2}$$

$$\frac{\Sigma_1 \mid \Pi \vdash C_1 \textbf{ Comm} \quad \Sigma_2 \mid \Pi \vdash C_2 \textbf{ Comm}}{\Sigma_1, \Sigma_2 \mid \Pi \vdash C_1 \parallel C_2}$$

SCI Revisited (contd)

Normal free variables can be regarded as passively used free variables in limited contexts:

$$\frac{\Sigma, x : \tau \mid \Pi \vdash E\ \mathbf{Exp}}{\Sigma \mid x : \tau, \Pi \vdash E\ \mathbf{Exp}}$$

However, once a free variable is marked as passive, it cannot be used actively any more.
Fractional permissions to the rescue!

- we can annotate passively used variables with fractional permissions, and
- combine permissions to recover the whole (active) variable again.

Example

$\{x = 0\}$

```
resource r(x) {true} in begin
   with r do                          ||      with r do
      x := x+1;                       ||         x := x+1;
   od                                 ||      od
end
```

$\{x = 2\}$

Example (with auxiliary variables)

a := 0; b := 0;
resource r(x, a, b) $\{x = a + b\}$ **in begin**

$\{a = 0\}$		$\{b = 0\}$
with r **do**		**with** r **do**
x := x+1;	$\|\|$	x := x+1;
a := 1		b := 1
od		**od**
$\{a = 1\}$		$\{b = 1\}$

end
$\{x = a + b \star a = 1 \star b = 1\}$
$\{x = 2\}$

Question: How can we use a and b outside critical regions?

Example (with permissions)

a := 0; b := 0;

resource $r(x^1, a^{\frac{1}{2}}, b^{\frac{1}{2}})$ $\{x = a + b\}$ **in begin**

// owns $a^{\frac{1}{2}}$		// owns $b^{\frac{1}{2}}$
$\{a = 0\}$		$\{b = 0\}$
with r **do**		**with** r **do**
x := x+1;	$\|\|$	x := x+1;
a := 1		b := 1
od		**od**
$\{a = 1\}$		$\{b = 1\}$

end

Since the left process keeps $a^{\frac{1}{2}}$ permission, there is no way that any "other process" can modify a. (Similarly for b.)

Permission algebra

[Boyland], [Bornat et al.]
A partial commutative semigroup $(\mathcal{P}, \oplus, \top)$.

- cancellative: $x \oplus y = x \oplus y' \implies y = y'$.
- totality: $\top \oplus x$ is undefined.
- no unit: $x \oplus y \neq x$.
- divisibility: $\forall x.\, \exists y_1, y_2.\, x = y_1 \oplus y_2$.

Example: real interval $(0, 1]$ with addition as $\oplus$ and 1 as $\top$.

Well-formedness judgements

$$x_1^{p_1}, \ldots, x_n^{p_n} \vdash E \textbf{ Exp}$$
$$x_1^{p_1}, \ldots, x_n^{p_n} \vdash P \textbf{ Assert}$$

Variable contexts (Σ)

- The same variable can have multiple occurrences in Σ:

$$x^{p_{i_1}}, \ldots, x^{p_{i_k}}$$

- But the permissions should be combinable:

$$p_{i_1} \oplus \cdots \oplus p_{i_k} \text{ defined}$$

E.g., $x^1, y^{\frac{1}{2}}, x^{\frac{1}{2}} \vdash \cdots$ is illegal.

Example rules of well-formednes

$$\frac{}{\Sigma, x^{p} \vdash x\ \mathbf{Exp}} \qquad \frac{\Sigma \vdash E_1\ \mathbf{Exp} \quad \Sigma \vdash E_2\ \mathbf{Exp}}{\Sigma \vdash E_1 + E_2\ \mathbf{Exp}}$$

$$\frac{\Sigma \vdash E_1\ \mathbf{Exp} \quad \Sigma \vdash E_2\ \mathbf{Exp}}{\Sigma \vdash E_1 \overset{p}{\longmapsto} E_2\ \mathbf{Assert}} \qquad \frac{\Sigma \vdash P_1\ \mathbf{Assert} \quad \Sigma \vdash P_2\ \mathbf{Assert}}{\Sigma \vdash P_1 \star P_2\ \mathbf{Assert}}$$

$$\frac{\Sigma, x^{\top} \vdash P\ \mathbf{Assert}}{\Sigma \vdash \exists x.\, P\ \mathbf{Assert}}$$

Structural rules

Contraction rule:

$$\frac{\Sigma, x^p, x^q \vdash \mathcal{S}}{\Sigma, x^{p \oplus q} \vdash \mathcal{S}}$$

Weakening (an admissible rule):

$$\frac{\Sigma \vdash \mathcal{S}}{\Sigma, \Sigma' \vdash \mathcal{S}}$$

Substitution (an admissible rule)

$$\frac{\Sigma \vdash E \ \mathbf{Exp} \quad \Sigma, x^{\top} \vdash \mathcal{S}}{\Sigma \vdash \mathcal{S}[E/x]}$$

Well-formedness of commands

$$\Sigma \mid \Gamma \vdash C \textbf{ Comm}$$

Σ is a variable context: $x_1^{p_1}, \ldots, x_n^{p_n}$
Γ is a resource context: $r_1(\Sigma_1), \ldots, r_m(\Sigma_m)$
For a legal context:

- All the r_i's are distinct.
- $\Sigma, \Sigma_1, \ldots, \Sigma_m$ is legal.

Example: $a^{\frac{1}{2}}, b^{\frac{1}{2}} \mid r(x^1, a^{\frac{1}{2}}, b^{\frac{1}{2}}) \vdash C_1 \parallel C_2$ **Comm**

Example rules for commands

$$\frac{\Sigma \mid \Gamma \vdash E\ \mathbf{Exp}}{\Sigma \mid \Gamma \vdash (x := E)\ \mathbf{Comm}} \quad \text{if } x^{\top} \in \Sigma$$

$$\frac{\Sigma \mid \Gamma \vdash E_1\ \mathbf{Exp} \quad \Sigma \mid \Gamma \vdash E_2\ \mathbf{Exp}}{\Sigma \mid \Gamma \vdash ([E_1] := E_2)\ \mathbf{Comm}}$$

$$\frac{\Sigma \mid \Gamma \vdash C_1\ \mathbf{Comm} \quad \Sigma \mid \Gamma \vdash C_2\ \mathbf{Comm}}{\Sigma \mid \Gamma \vdash (C_1; C_2)\ \mathbf{Comm}}$$

$$\frac{\Sigma_1 \mid \Gamma \vdash C_1\ \mathbf{Comm} \quad \Sigma_2 \mid \Gamma \vdash C_2\ \mathbf{Comm}}{\Sigma_1, \Sigma_2 \mid \Gamma \vdash (C_1 \parallel C_2)\ \mathbf{Comm}}$$

There are no side conditions for the parallel rule!
A bit deceptive because Σ_1, Σ_2 should be legal.

An aside on natural deduction

A natural deduction starts with assumptions and applies rules to derive conclusions:

$$\begin{array}{ccc} A_1 & \ldots & A_n \\ \vdots & & \vdots \\ & \mathcal{S} & \end{array}$$

For the sake of clarity on how the assumptions are handled, we write it in sequent form:

$$A_1, \ldots, A_n \vdash \mathcal{S}$$

So the parallel rule is really saying:

$$\frac{\begin{array}{cc} \Sigma_1 & \Sigma_2 \\ \vdots & \vdots \\ C_1\ \textbf{Comm} & C_2\ \textbf{Comm} \end{array}}{(C_1 \parallel C_2)\ \textbf{Comm}}$$

Resources and critical regions

$$\frac{\Sigma \mid \Gamma, r(\Sigma_0) \vdash C\ \textbf{Comm}}{\Sigma, \Sigma_0 \mid \Gamma \vdash (\textbf{resource}\ r(\Sigma_0)\ \textbf{in}\ C)\ \textbf{Comm}}$$

$$\frac{\Sigma, \Sigma_0 \vdash B\ \textbf{Exp} \quad \Sigma, \Sigma_0 \mid \Gamma \vdash C\ \textbf{Comm}}{\Sigma \mid \Gamma, r(\Sigma_0) \vdash (\textbf{with}\ r\ \textbf{when}\ B\ \textbf{do}\ C\ \textbf{od})\ \textbf{Comm}}$$

- The resource declaration slices off a part of the current variable context (Σ_0) and locks it up in the resource.
- A critical region unlocks the resource's context and provides it to the body of **with** .

Programming logic with SCI

$$\Sigma \mid \Gamma \vdash \{P\}\ C\ \{Q\}$$

requires well-formedness:

- $\Sigma \vdash P$ **Assert** and $\Sigma \vdash Q$ **Assert**.
- $\Sigma \mid \Gamma \vdash C$ **Comm**

For example:

$$a^{\frac{1}{2}}, b^{\frac{1}{2}} \mid r(x^1, a^{\frac{1}{2}}, b^{\frac{1}{2}}) \vdash \{a = 0 \star b = 0\}\ C_1 \parallel C_2\ \{a = 1 \star b = 1\}$$

Examples of Logic rules

$$\textit{ASSIGN}\quad \frac{\Sigma \mid \Gamma \vdash E\ \mathbf{Exp} \quad \Sigma \mid \Gamma \vdash P\ \mathbf{Assert}}{\Sigma \mid \Gamma \vdash \{P[E/x]\}\ x := E\ \{P\}} \quad \text{if } x^{\top} \in \Sigma$$

$$\textit{FRAME}\quad \frac{\Sigma \mid \Gamma \vdash \{P\}\ C\ \{Q\} \quad \Sigma' \mid \Gamma \vdash R\ \mathbf{Assert}}{\Sigma, \Sigma' \mid \Gamma \vdash \{P \star R\}\ C\ \{Q \star R\}}$$

$$\textit{PAR}\quad \frac{\Sigma_1 \mid \Gamma \vdash \{P_1\}\ C_1\ \{Q_1\} \quad \Sigma_2 \mid \Gamma \vdash \{P_2\}\ C_2\ \{Q_2\}}{\Sigma_1, \Sigma_2 \mid \Gamma \vdash \{P_1 \star P_2\}\ C_1 \parallel C_2\ \{Q_1 \star Q_2\}}$$

- In *FRAME*, the well-formedness of Σ, Σ' is equivalent to the O'Hearn et al. side condition "C does not modify **free**(R)."
- In *PAR*, the well-formedness of Σ_1, Σ_2 is equivalent to the Owicki-Gries side condition "C_i does not modify **free**(P_j, Q_j) (for $j \neq i$)."

Examples of Logic rules (contd)

CRIT rule for critical regions:

$$\frac{\begin{array}{c}\Sigma \vdash P\ \textbf{Assert} \quad \Sigma \vdash Q\ \textbf{Assert} \\ \Sigma, \Sigma_0 \vdash B\ \textbf{Exp} \quad \Sigma, \Sigma_0 \mid \Gamma \vdash \{P \star R \wedge B\}\ C\ \{Q \star R\}\end{array}}{\Sigma \mid \Gamma,\ r(\Sigma_0) : R \vdash \{P\}\ \textbf{with}\ r\ \textbf{when}\ B\ \textbf{do}\ C\ \textbf{od}\ \{Q\}}$$

Since P and Q are well-formed in the context Σ, it is obvious that no "other process" can modify the variables in Σ.
However, "this process" can modify them, because Σ is part of the context for C.
Thus, we have a *compositional formulation* of the Owicki-Gries side conditions.

Comparison with Owicki-Gries-O'Hearn system

All Owicki-Gries-O'Hearn proof outlines can be transformed to our system.
Every resource declaraion

$$\textbf{resource } r(x_1, \ldots, x_n) \textbf{ in } C$$

must be annotated with permissions for the owned variables.

1. If x occurs only inside critical regions:
   - if it is modified there, annotate it as x^1.
   - if it is not modified there, annotate it with a possibly partial permisison p.
2. If x occurs outside critical regions, it can only do so in a *single* process and it must be *passively* used.
   - Annotate the resource with x^p (for some partial permision p).
   - Give the process $x^{p'}$ (where $p \oplus p' = \top$).

Comparision with Brookes's system

Brookes allows every resource to deal with two sets of variables:

$$\textbf{resource}\ r(x_1, \ldots, x_n) : R\ \textbf{in}\ C$$

- $\textbf{owned}(r) = \{x_1, \ldots, x_n\}$
- $\textbf{free}(R)$ is the set of free variables in the resource invariant, which can include more variables than $\textbf{owned}(r)$.

Example: resource $r(x) : \{x = a + b\}$ **in** $C_1 \parallel C_2$
Rewrite the resource declaration as:

$$\textbf{resource}\ r(x_1^\top, \ldots, x_n^\top, y_1^{p_1}, \ldots, y_m^{p_m}) : R\ \textbf{in}\ C$$

- Variables in $\textbf{owned}(r)$ are fully owned by the resource.
- The additional variables in $\textbf{free}(R)$ are partially owned by the resource.

However not all Brookes's proof outlines can be translated.

Unsoundness in Brookes's rules

[Ian Wehrman]

x := a;
resource r(x) $\{x = a\}$ **in**
begin

$\{true\}$		$\{true\}$
with r **do**		**with** r **do**
t := x	$\|\|$	x := x+1;
od		a := a+1
$\{t = a\}$		**od**
with r **do**		$\{true\}$
x := t		
od		
$\{true\}$		

end
$\{x = a\}$

Comparison with “Variables as Resource” systems

[Parkinson et al.], [Brookes]
“Variable as resource” systems treat variable usage in *assertions* instead of the syntax.
Our rules can be translated into “Variables as resource” logics.
(Hence, the latter are more general.)

$$\Sigma = (x_1^{p_1}, \ldots, x_n^{p_n}) \quad \rightsquigarrow \quad O_\Sigma \equiv \mathbf{own}_{p_1}(x_1) \star \cdots \star \mathbf{own}_{p_n}(x_n)$$

$$\Sigma \mid \Gamma \vdash \{P\}\, C\, \{Q\} \quad \rightsquigarrow \quad \Gamma \vdash \{O_\Sigma \wedge P\}\, C\, \{O_\Sigma \wedge Q\}$$

But “Variables as resource” logics are strange in practice:

- $E = E$ is not universally true.
- $\neg(E_1 = E_2)$ and $E_1 \neq E_2$ are not the same thing.
- Substitution is not always legal.
- Program variables cannot be treated as logical variables.

Semantics

- SCI is something like a type system.
- So, we would expect it to streamline the denotational semantics, and rule out unwanted behaviours.

Brookes's action traces:

$$\lambda ::= \delta \mid x = v \mid x := v \mid [l] = v \mid [l] := v \mid try(r) \mid acq(r) \mid rel(r) \mid abort$$

Actions are enabled in contexts, and may transform them.

$$\begin{array}{ll}
\Sigma \mid \widetilde{\Gamma} \xrightarrow{x=v} \Sigma \mid \widetilde{\Gamma} & \text{iff} \quad x^p \in \Sigma \text{ for some } p \\
\Sigma \mid \widetilde{\Gamma} \xrightarrow{x:=v} \Sigma \mid \widetilde{\Gamma} & \text{iff} \quad x^\top \in norm(\Sigma) \\
\Sigma \mid \widetilde{\Gamma}, r(\Sigma_0) \xrightarrow{acq(r)} \Sigma, \Sigma_0 \mid \widetilde{\Gamma}, [r(\Sigma_0)] & \\
\Sigma, \Sigma_0 \mid \widetilde{\Gamma}, [r(\Sigma_0)] \xrightarrow{rel(r)} \Sigma \mid \widetilde{\Gamma}, r(\Sigma_0) &
\end{array}$$

Theorem: The trace set T of every command $\Sigma \mid \Gamma \vdash C$ **Comm** satisfies $\Sigma \mid \Gamma \xrightarrow{T} \Sigma \mid \Gamma$.

Summary

- We have produced a clean, simple, compositional system of proof rules for Concurrent Separation Logic.
- The system is expressive, fitting somewhere between Owicki-Gries-O'Hearn rules and "Variables as resource" rules.
- It is sound and has a direct representation in the semantics.

Further work

- Algorithms for parsing/type-checking.
- Integration with type systems.
- Extensions to procedures and objects.