Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 286 (2012) 87–102

www.elsevier.com/locate/entcs

# Syntactic Control of Interference and Concurrent Separation Logic

Stephen Brookes

*Department of Computer Science*
*Carnegie Mellon University*
*Pittsburgh, USA*

**Abstract**

At last year's MFPS conference we introduced a revised version of Concurrent Separation Logic in which assertions are tagged with a "rely set" of variables assumed to be unmodified by other processes. We showed that this logic is compositional and sound with respect to an action trace semantics. The revision was motivated by a subtle issue concerning soundness of the original version of the logic, discovered by Ian Wehrman and Josh Berdine. The revised logic fixes this problem and also relaxes the Owicki-Gries constraints on variables, allowing shared variables to be protected by multiple resources rather than a single one, but requiring that a process writing to a shared variable must acquire all resources that protect it, while a process reading a shared variable need only acquire one such resource. This generalization brings concurrent separation logic closer in spirit to permission-based logics, although our formulation makes no explicit mention of permissions. At the same conference, Uday Reddy introduced a concurrent separation logic with static permissions for variables, generalizing John Reynolds's ideas on syntactic control of interference to a concurrent setting. Here we show that there is an extremely close relationship between these two logics. Essentially, every provable assertion in Reddy's logic corresponds to a provable assertion in CSL with the same semantic content; and every provable assertion in CSL corresponds to a multitude of assertions in Reddy's logic, differing only in the choice of specific permission values. We show that every derivation in Reddy's logic can be transformed into a derivation in CSL, by abstracting away from permission details while retaining the relevant information about protection of variables by resources. And we show how to construct, for a given CSL derivation, a family of corresponding derivations in Reddy's logic that differ only in inessential permission choices. These results also imply that one can establish soundness of Reddy's logic by appealing to soundness of CSL, leading to a simpler soundness proof than the one given in Reddy's original paper, which used an augmented form of action trace semantics.

*Keywords:* concurrency, shared memory, resources, separation logic, permissions

## 1 Introduction

Concurrent Separation Logic is a logic for fault-free partial correctness of shared-memory concurrent programs, combining separation logic [10] with Owicki-Gries inference rules for pointer-free shared-memory programs [8], as proposed by Peter O'Hearn [7]. The original Owicki-Gries and O'Hearn logics apply only to programs with rigid parallel structure, because of a constraint that "no other process modifies" certain variables, imposed as a side condition in the rule for conditional critical regions.


doi:10.1016/j.entcs.2012.08.007

In prior work we formulated a more general concurrent separation logic [3], using resource contexts in an attempt to avoid these limitations. We gave a semantic formalization of O'Hearn's notion of "ownership transfer" based on resource invariants, and of O'Hearn's principle that provable processes "mind their own business" [7]. Subsequently, Ian Wehrman and Josh Berdine [12] discovered that the soundness analysis in [3] contains a hidden assumption tantamount to "no other process modifies", so this logic was also only sound for rigid programs.

In response, at last year's MFPS [5] we introduced a revised version of this logic in which assertions are augmented with a "rely set" representing a set of variables deemed to be left unmodified by "other" processes. By making this set an integral part of assertions, we avoid the need for a non-compositional side condition. The revised logic, which we call CSL, deals with these no-modifies assumptions in a syntax-directed and modular manner, without forcing the prover to know the rest of the program in advance. In addition the revised logic relaxes the Owicki-Gries constraints on use of critical variables, allowing such variables to be protected by several resources instead of a single one, thus embodying a more general protection discipline. A process writing a shared variable must acquire *all* resources that protect it, whereas reading a shared variable merely requires acquisition of *some* resource that protects it. We proved soundness of the revised CSL, this time without the hidden assumption and without requiring rigid program structure [5].

At the same MFPS Uday Reddy presented a permission-based version of concurrent separation logic, in which variables (but not heap cells) are given statically controlled permissions. This material appeared later as a joint paper with John Reynolds [9], including an algorithm (due to Reynolds) for inferring permission assignments. This work was motivated by a desire to generalize earlier ideas of Reynolds on syntactic control of interference to the concurrent setting. Reddy's assertions deal elegantly with statically scoped permission contexts, and his formulation elides some of the syntactic side conditions that govern CSL rules dealing with variables, instead replacing them with implicit well-formedness constraints on the syntax of judgements. In this logic, writing to a shared variable requires acquisition of "total" permission for that variable, while reading is allowed with any permission. Static permissions, used this way, enforce syntactic control of interference: every provable program is free of race conditions involving variables (because of permission constraints), and free of races involving heap cells (by judicious use of separate conjunction).

Reddy commented [9] that there seems to be a close relationship between CSL and static permissions, and we made a similar observation at the time [5]. Indeed we deliberately used similar terminology to describe the disciplines for shared variable usage in the two frameworks: total permission surely seems "equivalent" to possession of all protecting resources, since both preclude any other process from interfering. CSL keeps track of the set of resources owned by processes, whereas Reddy's logic maintains book-keeping information on the amount of permission for variables gathered by processes as they acquire and release resources.

In this paper we make a precise and rigorous connection between the two logics,

confirming these intuitions. Informally, each provable assertion in Reddy's logic corresponds to a provable assertion in CSL; and a provable CSL assertion corresponds to a (non-empty) set of provable assertions in Reddy's logic, differing only in essentially irrelevant details concerning specific choices of permission values. To establish these claims formally we first summarize some of the key concepts and definitions from the two logics. We make a careful analysis of the way proof derivations work in the two logics. We will also make clear what we mean by irrelevant permission decisions.

We assume familiarity with separation logic, as defined by Reynolds [10]. The reader can consult [5] for semantic foundations. To make this paper self-contained we recapitulate some material from [5] and [9].

## 2 Syntax

The syntax of commands (or processes) is given by the following abstract grammar:

$$\begin{aligned} C ::= {} & \mathbf{skip} \mid i{:=}e \mid i{:=}[e] \mid [e]{:=}e' \mid i{:=}\mathbf{cons}\, E \mid \mathbf{dispose}\, e \\ \mid {} & C_1; C_2 \mid \mathbf{if}\ b\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2 \mid \mathbf{while}\ b\ \mathbf{do}\ C \\ \mid {} & \mathbf{with}\ r\ \mathbf{when}\ b\ \mathbf{do}\ C \mid \mathbf{resource}\ r\ \mathbf{in}\ C \mid C_1 \| C_2 \end{aligned}$$

where $e$ ranges over integer expressions, $b$ over boolean expressions, and $E$ over list expressions of form $[e_1, \ldots, e_n]$. Expressions are *pure*, i.e. independent of the heap. Further, $i$ ranges over identifiers (or program variables) and $r$ over *resource names*; resources behave like binary semaphores. We use the standard abbreviation **with** $r$ **do** $C$ for **with** $r$ **when true do** $C$.

Let $\mathit{free}(C) \subseteq \mathbf{Ide}$ be the set of identifiers with a free occurrence in $C$. Let $\mathit{mod}(C)$ be the set of identifiers with a free write occurrence in $C$, defined as usual, by structural induction:

$$\begin{aligned} & \mathit{mod}(\mathbf{skip}) = \{\} \\ & \mathit{mod}(i{:=}e) = \mathit{mod}(i{:=}\mathbf{cons}\, E) = \mathit{mod}(i{:=}[e]) = \{i\} \\ & \mathit{mod}([e]{:=}e') = \mathit{mod}(\mathbf{dispose}\ e) = \{\} \\ & \mathit{mod}(C_1; C_2) = \mathit{mod}(C_1 \| C_2) = \mathit{mod}(C_1) \cup \mathit{mod}(C_2) \\ & \mathit{mod}(\mathbf{if}\ b\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2) = \mathit{mod}(C_1) \cup \mathit{mod}(C_2) \\ & \mathit{mod}(\mathbf{while}\ b\ \mathbf{do}\ C) = \mathit{mod}(C) \\ & \mathit{mod}(\mathbf{with}\ r\ \mathbf{when}\ b\ \mathbf{do}\ C) = \mathit{mod}(C) \\ & \mathit{mod}(\mathbf{resource}\ r\ \mathbf{in}\ C) = \mathit{mod}(C) \end{aligned}$$

# 3 Concurrent Separation Logic

As in Owicki-Gries [8], we associate to each resource name $r$ a set $X \subseteq \mathbf{Ide}$ of "protected variables" and a "resource invariant" $R$ [6], which is required to be a *precise* separation logic formula as in [7]. A separation logic formula $R$ is precise iff, for all stores $s$ and heaps $h$, there is at most one sub-heap $h' \subseteq h$ such that $(s, h') \models R$. Instead of assuming a statically fixed association of resource invariants and protection sets to resource names, we extend the syntax of partial correctness assertions to include a *resource context*, as in [3]. We also relax the variable usage constraints from the earlier logics by not insisting that protection sets be pairwise disjoint.

**Definition 3.1** A *well-formed* resource context has the form

$$r_1(X_1) : R_1, \ldots, r_n(X_n) : R_n$$

where $r_1, \ldots, r_n$ are distinct resource names, $X_1, \ldots, X_n$ are sets of identifiers, each $R_i$ is precise, and $\mathit{free}(R_i) \subseteq X_i$ for each $i$.

We let $\Gamma$ range over the set of well-formed resource contexts. We say that $r$ *protects* $x$ in $\Gamma$ when $r(X) : R$ is in $\Gamma$ and $x \in X$. Let $\mathit{owned}(\Gamma) = \bigcup_{i=1}^{n} X_i$, and $\mathit{inv}(\Gamma) = R_1 * \cdots * R_n$. Let $\mathit{dom}(\Gamma)$ be $\{r_1, \ldots, r_n\}$. We let $\Gamma, r(X) : R$ be the context that combines $\Gamma$ with $r(X) : R$, when well-formed.

**Definition 3.2** A *well-formed* CSL *assertion* has form

$$\Gamma \vdash_A \{P\}C\{Q\},$$

where $A$ is a (finite) set of identifiers, $\Gamma$ is a well-formed resource context, $\mathit{free}(P, Q) \subseteq A$, and $\mathit{free}(C) \subseteq \mathit{owned}(\Gamma) \cup A$.

In an assertion $\Gamma \vdash_A \{P\}C\{Q\}$ we refer to $A$ as the *rely set*, $P$ as the pre-condition, and $Q$ as the post-condition. The constraints allow $P$ and $Q$ to mention identifiers owned by resources in $\Gamma$, but only if they belong to the rely set; the command $C$ may use variables owned by resources or belonging to the rely set. The rules use rely sets to keep track of variables used (outside of critical regions, so without protection): for soundness and the avoidance of race conditions such variables must not be modified by any other process, and this requirement is enforced as a side condition in the CSL parallel rule. The CSL inference rules will constrain where $C$ is allowed to read and write protected variables: every write in $C$ to a protected variable must be inside (nested) critical regions naming *all* resources that protect it; and every read occurrence in $C$ of a protected variable must be inside a critical region naming *some* resource that protects it. In the special case where the protection sets are pairwise disjoint, this coincides with the usual Owicki-Gries discipline.

**Definition 3.3** The assertion $\Gamma \vdash_A \{P\}C\{Q\}$ is valid iff every finite interactive computation of $C$, from a state (with values for all variables in $\Gamma, A$) satisfying

$P * inv(\Gamma)$, in an environment that respects $\Gamma$ and does not write to variables in $A$, is fault-free, respects $\Gamma$, and ends in a state satisfying $Q * inv(\Gamma)$.

Respect for $\Gamma$ means obedience to the protection regime and preservation of each resource invariant (separately). Fault-freedom means no runtime errors such as dangling pointers, and no race conditions involving concurrent writes to shared variables or heap cells.

Validity of $\Gamma \vdash_A \{P\}C\{Q\}$ also implies fault-free partial correctness: every terminating execution of $C$ *in isolation*, from a state satisfying $P * inv(\Gamma)$, is fault-free and ends in a final state satisfying $Q * inv(\Gamma)$. This is because the empty environment vacuously respects $\Gamma$ and does not write to any variable.

The CSL rules for assignment, parallel composition, critical regions, and local resource blocks are summarized below, together with the Frame rule and the Rule of Consequence, which allows weakening of post-conditions and strengthening of pre-conditions as usual and allows expansion of a rely set.

Only well-formed instances of these rules are allowed.

- Assignment

$$\frac{}{\Gamma \vdash_A \{[e/i]P\}i{:=}e\{P\}} \quad \text{if } i \notin owned(\Gamma),\ free(P, e) \subseteq A$$

- Parallel

$$\frac{\Gamma \vdash_{A_1} \{P_1\}C_1\{Q_1\} \quad \Gamma \vdash_{A_2} \{P_2\}C_2\{Q_2\}}{\Gamma \vdash_{A_1 \cup A_2} \{P_1 * P_2\}C_1 \| C_2\{Q_1 * Q_2\}}$$

  if $mod(C_1) \cap free(C_2) \subseteq owned(\Gamma)$, $mod(C_2) \cap free(C_1) \subseteq owned(\Gamma)$, $mod(C_1) \cap A_2 = \{\}$ and $mod(C_2) \cap A_1 = \{\}$.

- Region

$$\frac{\Gamma \vdash_{A \cup X} \{(P * R) \wedge b\}C\{Q * R\}}{\Gamma, r(X) : R \vdash_A \{P\}\textbf{with } r \textbf{ when } b \textbf{ do } C\{Q\}}$$

- Resource

$$\frac{\Gamma, r(X) : R \vdash_A \{P\}C\{Q\}}{\Gamma \vdash_{A \cup X} \{P * R\}\textbf{resource } r \textbf{ in } C\{Q * R\}}$$

- Frame

$$\frac{\Gamma \vdash_A \{P\}C\{Q\}}{\Gamma \vdash_{A \cup free(R)} \{P * R\}C\{Q * R\}} \quad \text{if } mod(C) \cap free(R) = \{\}$$

- Consequence

$$\frac{\Gamma \vdash_A \{P\}C\{Q\}}{\Gamma \vdash_{A'} \{P'\}C\{Q'\}} \quad \text{if } A \subseteq A',\ P' \Rightarrow P,\ Q \Rightarrow Q'$$

In Assignment note that well-formedness implies that $i \in A$.

In Parallel, the side condition ensures that neither process modifies any identifiers in the other process's rely set.

In Region the premiss relies on $A \cup X$ because mutual exclusion for $r$ implies that the identifiers in $X$ cannot be modified by other processes while the current process is inside the critical region.

In Resource the conclusion relies on $A \cup X$, which ensures well-formedness of the conclusion whenever the premiss is well-formed, because $\mathit{free}(R) \subseteq X$.

In Frame, as usual, $C$ must not write to any identifier occurring free in $R$. There is no need to insist, as was done in [3], that $\mathit{free}(R) \cap \mathit{owned}(\Gamma) = \{\}$; instead we add the variables occurring free in $R$ to the rely set, reflecting the assumption that no concurrent processes modify these variables.

## 4 Example

Here is an example that facilitates the coming comparison with Reddy's logic.

The following assertion is provable in CSL. Note the rely set $\{x\}$.

$$
\begin{array}{l}
\vdash_{\{x\}} \{42 \mapsto \_\} \\
\quad \textbf{resource } r_1 \textbf{ in} \\
\quad\quad \textbf{resource } r_2 \textbf{ in} \\
\quad\quad\quad (\textbf{with } r_1 \textbf{ do } (\textbf{with } r_2 \textbf{ do } x{:=}1); [42]{:=}1) \\
\quad\quad \parallel (\textbf{with } r_2 \textbf{ do } (\textbf{with } r_1 \textbf{ do } x{:=}2); [42]{:=}2) \\
\quad \{42 \mapsto \_\}
\end{array}
$$

Validity of this assertion implies that this program, when executed from a state in which $x$ has a value and 42 is a heap cell, is error-free, provided no other process modifies $x$; in particular, there is no race condition involving $x$, and no race condition involving the heap cell. Of course, the rely assumption is needed: if the code is run concurrently with a process that modifies $x$ there could be a race condition. Let $C_1$ and $C_2$ be:

$$C_1 :: \textbf{with } r_1 \textbf{ do } ((\textbf{with } r_2 \textbf{ do } x{:=}1); [42]{:=}1)$$

$$C_2 :: \textbf{with } r_2 \textbf{ do } ((\textbf{with } r_1 \textbf{ do } x{:=}2); [42]{:=}2).$$

Let $R_1$ and $R_2$ be the assertions

$$R_1 :: (x = 1 \wedge 42 \mapsto 1) \vee (x = 2 \wedge \textbf{emp})$$

$$R_2 :: (x = 1 \wedge \textbf{emp}) \vee (x = 2 \wedge 42 \mapsto 2).$$

As shown in [5] the following assertions are derivable in CSL:

$$r_1(x) : R_1, r_2(x) : R_2 \vdash_{\{\}} \{\textbf{emp}\} C_1 \{\textbf{emp}\}$$

$$r_1(x) : R_1, r_2(x) : R_2 \vdash_{\{\}} \{\textbf{emp}\} C_2 \{\textbf{emp}\}.$$

By PARALLEL we can then get

$$r_1(x) : R_1, r_2(x) : R_2 \vdash_{\{\}} \{\mathbf{emp}\} C_1 \| C_2 \{\mathbf{emp}\}$$

and finally, by RESOURCE,

$$\vdash_{\{x\}} \{R_1 * R_2\} \mathbf{resource}\ r_1\ \mathbf{in\ resource}\ r_2\ \mathbf{in}\ (C_1 \| C_2) \{R_1 * R_2\}.$$

This derivation employs a resource context $r_1(x) : R_1.r_2(x) : R_2$ in which the protection lists are not disjoint.

## 5 Reddy's logic

Reddy's version of Concurrent Separation Logic uses static *permissions* to ensure proper usage of shared variables.

Following [1], a permission algebra $(\mathcal{P}, \oplus, \top)$ is a partial commutative, cancellative semi-group such that

$$\forall p_1, p_2 \in \mathcal{P}.\ p_1 \oplus p_2 \neq p_2 \qquad \text{(non-zero)}$$

$$\forall p \in \mathcal{P}.\ p \oplus \top \text{ is undefined} \qquad \text{(top)}$$

$$\forall p \in \mathcal{P}. \exists p_1, p_2 \in \mathcal{P}.\ p = p_1 \oplus p_2 \qquad \text{(divisibility)}$$

A permission context $\Sigma$ has the form $x_1^{p_1}, \ldots, x_n^{p_n}$ and is well-formed if and only if whenever $x$ occurs multiple times in $\Sigma$, with permissions $p_{i_1}, \ldots, p_{i_k}$, then $p_{i_1} \oplus \cdots \oplus p_{i_k}$ is defined. We write $\Sigma_1, \ldots, \Sigma_n$ for the obvious composite context combining the permission entries of the $\Sigma_i$, when this is well-formed.

Let $dom(\Sigma) = \{x \in \mathbf{Ide} \mid \exists p.\ x^p \in \Sigma\}$ and, for $x \in dom(\Sigma)$, let $\Sigma(x)$ be the sum of all permissions occurring for $x$ in the entries of $\Sigma$. Let $|\Sigma| = \{(x, \Sigma(x)) \mid x \in dom(\Sigma)\}$. Permission contexts $\Sigma_1$ and $\Sigma_2$ are *equivalent* iff they give the same permissions to all identifiers, i.e. when $|\Sigma_1| = |\Sigma_2|$.

Reddy introduces syntactic judgements, defined with structural inference rules, with the following forms:

- $\Sigma \vdash i$ Var
- $\Sigma \vdash e$ Exp
- $\Sigma \vdash P$ Assert

In each case $\Sigma$ is required to be a well-formed permission context. Using validity here to mean derivability from Reddy's inference rules, the following key facts are easy to establish:

- If $\Sigma \vdash i$ Var is valid, then $i \in dom(\Sigma)$ and $\Sigma(i) = \top$.
- If $\Sigma \vdash e$ Exp is valid, then $\mathtt{free}(e) \subseteq dom(\Sigma)$.
- If $\Sigma \vdash P$ Assert is valid, then $\mathtt{free}(P) \subseteq dom(\Sigma)$.

Although Reddy's paper does not provide a full set of inference rules for these judgements, it seems reasonable to assume that the converse implications also hold, since the purpose of these syntactic judgements is to formalize the indicated static constraints on variables and permissions.

Reddy also introduces *permissive resource contexts*, of the form

$$r_1(\Sigma_1) : R_1, \ldots, r_n(\Sigma_n) : R_n$$

where $r_1, \ldots, r_n$ are distinct resource names, $\Sigma_1, \ldots, \Sigma_n$ is a well-formed permission context, each $R_i$ is precise, and for each $i$, $\Sigma_i \vdash R_i$ Assert is valid. We will use $\Delta$ to range over (well-formed) permissive resource contexts [1].

Reddy's proof system uses assertions of the form

$$\Sigma \mid r_1(\Sigma_1) : R_1, \ldots, r_n(\Sigma_n) : R_n \vdash \{P\}C\{Q\},$$

subject to the constraint that $\Sigma, \Sigma_1 \ldots, \Sigma_n$ is a well-formed permission context, $r_1, \ldots, r_n$ are distinct resource names, each $R_i$ is precise, and $\Sigma_i \vdash R_i$ Assert is valid, for each $i$. Note that this implies that for each $i$, $\mathit{free}(R_i) \subseteq \mathit{dom}(\Sigma_i)$.

The Reddy rules for assignment, parallel composition, critical regions, and local resource blocks are listed below, together with the Frame Rule. There are also rules for the other program constructs and a Rule of Consequence. We use the same labels as for the corresponding CSL rules. Only well-formed instances are allowed.

- Assignment
$$\frac{\Sigma \vdash i \text{ Var} \quad \Sigma \vdash e \text{ Exp} \quad \Sigma \vdash P \text{ Assert}}{\Sigma \mid \Delta \vdash \{[e/i]P\}i{:=}e\{P\}}$$
- Parallel
$$\frac{\Sigma_1 \mid \Delta \vdash \{P_1\}C_1\{Q_1\} \quad \Sigma_2 \mid \Delta \vdash \{P_2\}C_2\{Q_2\}}{\Sigma_1, \Sigma_2 \mid \Delta \vdash \{P_1 * P_2\}C_1 \| C_2\{Q_1 * Q_2\}}$$
- Region
$$\frac{\Sigma \vdash P, Q \text{ Assert} \quad \Sigma, \Sigma_0 \mid \Delta \vdash \{(P * R) \wedge b\}C\{Q * R\}}{\Sigma \mid \Delta, r(\Sigma_0) : R \vdash \{P\}\textbf{with } r \textbf{ when } b \textbf{ do } C\{Q\}}$$
- Resource
$$\frac{\Sigma_0 \vdash R \text{ Assert} \quad \Sigma \mid \Delta, r(\Sigma_0) : R \vdash \{P\}C\{Q\}}{\Sigma, \Sigma_0 \mid \Delta \vdash \{P * R\}\textbf{resource } r(\Sigma_0) \textbf{ in } C\{Q * R\}}$$
- Frame
$$\frac{\Sigma \mid \Delta \vdash \{P\}C\{Q\} \quad \Sigma' \vdash R \text{ Assert}}{\Sigma, \Sigma' \mid \Delta \vdash \{P * R\}C\{Q * R\}}$$

The implicit well-formedness constraints on these rules codify many of the static side conditions that occur in the corresponding CSL rules. For example, in Frame the premiss $\Sigma' \vdash R$ Assert implies that $\mathit{free}(R) \subseteq \mathit{dom}(\Sigma')$, and when $\Sigma \mid \Delta \vdash \{P\}C\{Q\}$ is provable and $\Sigma, \Sigma'$ is well-formed this implies $\mathit{mod}(C) \cap \mathit{free}(R) = \{\}$, the side condition imposed in the CSL Frame rule.

[1] Reddy uses $\Gamma$ for permissive contexts, but we prefer $\Delta$ to avoid confusion with CSL.

We should point out a subtle issue that differentiates Reddy's set-up from ours. Although this may seem to be only a minor difference, it has significant consequences. In Reddy's framework we must attach permission contexts to local resource names. This is evident in the RESOURCE rule: unless $|\Sigma_0| = |\Sigma_1|$, the decorated programs **resource** $r(\Sigma_0)$ **in** $C$ and **resource** $r(\Sigma_1)$ **in** $C$ are logically distinguishable, since they are usable in different proof contexts. To prove $\Sigma' \mid \Delta \vdash \{P'\}$**resource** $r(\Sigma_0)$ **in** $C\{Q'\}$ it must be possible to "slice out" $\Sigma_0$ from $\Sigma'$, whereas it may not be possible to slice out $\Sigma_1$ instead. Each of these judgements concerns a decorated version of the same original program, and in establishing connections between the two logics we should distinguish between *commands* (in the original programming language) and *decorated commands* that arise in this manner. We will use $C', C''$ to range over decorated commands, $C$ over commands; and we say that $C'$ *is a decoration of* $C$ if $C$ is obtained from $C'$ by erasing permission contexts. The rules of Reddy's logic presented above should really employ $C'$ rather than $C$ as meta-variable, since the judgements involve decorated commands rather than commands, although we refrain from re-stating them in amended form.

Obviously we can characterize the decoration relationship by structural induction. In particular, if $C$ contains no local resource blocks then the only decoration of $C$ is $C$ itself; $C''$ is a decoration of **resource** $r$ **in** $C$ if and only if $C''$ has form **resource** $r(\Sigma)$ **in** $C'$, where $C'$ is a decoration of $C$. Similarly $C_1'; C_2'$ is a decoration of $C_1; C_2$ if and only if $C_1'$ is a decoration of $C_1$ and $C_2'$ is a decoration of $C_2$; and $C_1' \| C_2'$ is a decoration of $C_1 \| C_2$ if and only if $C_1'$ is a decoration of $C_1$ and $C_2'$ is a decoration of $C_2$. It is obvious that when $C'$ is a decoration of $C$, $mod(C') = mod(C)$ and $free(C') = free(C)$.

## 6 Example

Let $\mathcal{P} = (0, 1]$ be *fractional permissions*, with $p_1 \oplus p_2 = p_1 + p_2$ iff $p_1 + p_2 \leq 1$, and $\top = 1$, as in [2]. One can derive the following assertion in Reddy's logic, as shown in [9]:

$$
\begin{array}{l}
\vdash \{42 \mapsto \_\} \\
\quad \textbf{resource } r_1(x^{1/2}) \textbf{ in} \\
\quad\quad \textbf{resource } r_2(x^{1/2}) \textbf{ in} \\
\quad\quad\quad (\textbf{with } r_1 \textbf{ do } (\textbf{with } r_2 \textbf{ do } x{:=}1); [42]{:=}1) \\
\quad\quad \| \ (\textbf{with } r_2 \textbf{ do } (\textbf{with } r_1 \textbf{ do } x{:=}2); [42]{:=}2) \\
\quad \{42 \mapsto \_\}
\end{array}
$$

But equally well for each pair $(p, q)$ such that $0 < p, q < 1$ and $p + q = 1$ one can derive

$$\begin{array}{l} \vdash \{42 \mapsto \_\} \\ \quad \textbf{resource } r_1(x^p) \textbf{ in} \\ \quad\quad \textbf{resource } r_2(x^q) \textbf{ in} \\ \quad\quad\quad (\textbf{with } r_1 \textbf{ do } (\textbf{with } r_2 \textbf{ do } x{:=}1); [42]{:=}1) \\ \quad\quad \| \ (\textbf{with } r_2 \textbf{ do } (\textbf{with } r_1 \textbf{ do } x{:=}2); [42]{:=}2) \\ \{42 \mapsto \_\} \end{array}$$

and in each case the derivations are essentially the same, up to minor details concerning fractions.

The results of the next section will establish that all of these derivations correspond in a precise manner to the single CSL derivation shown earlier as an example.

# 7 Connecting the logics

We first show that every provable judgement in Reddy's logic corresponds to an assertion provable in CSL. To start, we note without proof following general properties, which are echoed in Reddy's development.

**Theorem 7.1**

*(i) If* $\Sigma \mid r_1(\Sigma_1) : R_1, \ldots, r_n(\Sigma_n) : R_n \vdash \{P\}C'\{Q\}$ *is provable, and* $x \in \mathit{free}(C')$*, then* $x \in \mathit{dom}(\Sigma, \Sigma_1, \ldots, \Sigma_n)$*.*

*(ii) If* $\Sigma \mid r_1(\Sigma_1) : R_1, \ldots, r_n(\Sigma_n) : R_n \vdash \{P\}C'\{Q\}$ *is provable, and* $x \in \mathit{mod}(C')$*, then* $(\Sigma, \Sigma_1, \ldots, \Sigma_n)(x) = \top$*.*

When $(\Sigma, \Sigma_1, \ldots, \Sigma_n)$ is a tuple of permission contexts we can construct a rely set and a tuple of protection lists by applying *dom* to each component and rearranging, to get $(\mathit{dom}(\Sigma), (\mathit{dom}(\Sigma_1), \ldots, \mathit{dom}(\Sigma_n)))$. Similarly, given a (well-formed) Reddy context $\Delta$ we can construct a (well-formed) CSL resource context by applying *dom* inside each term of $\Delta$, leaving resource names and invariants unchanged. When $\Delta$ has the form $r_1(\Sigma_1) : R_1, \ldots, r_n(\Sigma_n) : R_n$ this produces the resource context $\Gamma$ of form $r_1(\mathit{dom}\,\Sigma_1) : R_1, \ldots, r_n(\mathit{dom}\,\Sigma_n) : R_n$. Let us write $\Gamma = \mathit{map}\,\mathit{dom}\,\Delta$ when this relationship holds.

**Theorem 7.2**
*If* $\Sigma \mid \Delta \vdash \{P\}C'\{Q\}$ *is provable in Reddy's logic,* $C'$ *is a decoration of* $C$*,* $\Gamma = \mathit{map}\,\mathit{dom}\,\Delta$ *and* $A = \mathit{dom}(\Sigma)$*, then* $\Gamma \vdash_A \{P\}C\{Q\}$ *is provable in CSL.*

Proof: by induction on the proof height of the derivation. We show that for each (well-formed instance of an) inference rule in Reddy logic, if the relevant side conditions hold and the premisses are provable in Reddy's logic, then the CSL assertion corresponding to the rule's conclusion is provable from the CSL assertions representing the rule's premisses. We sketch the details for the rules listed above.

We make use of Theorem 7.1 in various places.

- Every well-formed instance of the Reddy ASSIGNMENT rule has the form

$$\frac{\Sigma \vdash i \text{ Var} \quad \Sigma \vdash e \text{ Exp} \quad \Sigma \vdash P \text{ Assert}}{\Sigma \mid \Delta \vdash \{[e/i]P\}i{:=}e\{P\}}$$

  Since we assume that the premisses are provable, we have $|\Sigma|(i) = \top$, $\mathit{free}(e) \subseteq \mathit{dom}(\Sigma)$, and $\mathit{free}(P) \subseteq \mathit{dom}(\Sigma)$. Let $A = \mathit{dom}(\Sigma)$ and let $\Gamma = \mathit{map}\,\mathit{dom}\,\Delta$ be the CSL resource context determined by $\Delta$. By well-formedness of $\Sigma, \Delta$ it follows that $i \notin \mathit{owned}(\Gamma)$. So the side conditions for the appropriate instance of the CSL ASSIGNMENT rule hold, and the CSL assertion $\Gamma \vdash_A \{[e/i]P\}i{:=}e\{P\}$ is provable.

- Consider a well-formed instance of PARALLEL, where $C_1' \| C_2'$ is a decoration of $C_1 \| C_2$:

$$\frac{\Sigma_1 \mid \Delta \vdash \{P_1\}C_1'\{Q_1\} \quad \Sigma_2 \mid \Delta \vdash \{P_2\}C_2'\{Q_2\}}{\Sigma_1, \Sigma_2 \mid \Delta \vdash \{P_1 * P_2\}C_1' \| C_2'\{Q_1 * Q_2\}}$$

  Let $A_1 = \mathit{dom}(\Sigma_1)$, $A_2 = \mathit{dom}(\Sigma_2)$, $\Gamma = \mathit{map}\,\mathit{dom}\,\Delta$. The CSL assertions corresponding to the premisses of this rule are then $\Gamma \vdash_{A_1} \{P_1\}C_1\{Q_1\}$ and $\Gamma \vdash_{A_2} \{P_1\}C_2\{Q_2\}$. By Theorem 7.1 and well-formedness it follows that $\mathit{mod}(C_2) \cap A_1 = \{\}$, $\mathit{mod}(C_1) \cap A_2 = \{\}$, $\mathit{mod}(C_1) \cap \mathit{free}(C_2) \subseteq \mathit{owned}(\Gamma)$, and $\mathit{mod}(C_1) \cap \mathit{free}(C_1) \subseteq \mathit{owned}(\Gamma)$. So by the CSL PARALLEL rule we get $\Gamma \vdash_{A_1 \cup A_2} \{P_1 * P_2\}C_1 \| C_2\{Q_1 * Q_2\}$. This is the CSL assertion corresponding to $\Sigma_1, \Sigma_2 \mid \Delta \vdash \{P_1 * P_2\}C_1' \| C_2'\{Q_1 * Q_2\}$.

- Consider a well-formed instance of REGION in which $C'$ is a decoration of $C$:

$$\frac{\Sigma \vdash P, Q \text{ Assert} \quad \Sigma, \Sigma_0 \mid \Delta \vdash \{(P * R) \wedge b\}C'\{Q * R\}}{\Sigma \vdash \Delta, r(\Sigma_0) : R \vdash \{P\}\mathbf{with}\ r\ \mathbf{when}\ b\ \mathbf{do}\ C'\{Q\}}$$

  Let $A = \mathit{dom}(\Sigma)$, $X = \mathit{dom}(\Sigma_0)$, and $\Gamma = \mathit{map}\,\mathit{dom}\,\Delta$. The CSL assertion corresponding to the (second) premiss is $\Gamma \vdash_{A \cup X} \{(P * R) \wedge b\}C\{Q * R\}$. Using the CSL REGION rule we can deduce

$$\Gamma, r(X) : R \vdash_A \{P\}\mathbf{with}\ r\ \mathbf{when}\ b\ \mathbf{do}\ C\{Q\}.$$

  This is the CSL assertion corresponding to the conclusion of the Reddy rule, and is well-formed; the first premiss implies $\mathit{free}(P, Q) \subseteq A$.

- Consider a well-formed instance RESOURCE in which $C'$ decorates $C$:

$$\frac{\Sigma_0 \vdash R \text{ Assert} \quad \Sigma \mid \Delta, r(\Sigma_0) : R \vdash \{P\}C'\{Q\}}{\Sigma, \Sigma_0 \mid \Delta \vdash \{P * R\}\mathbf{resource}\ r(\Sigma_0)\ \mathbf{in}\ C'\{Q * R\}}$$

  Assume that the premisses are provable. Let $A = \mathit{dom}(\Sigma)$, $X = \mathit{dom}(\Sigma_0)$, $\Gamma = \mathit{map}\,\mathit{dom}\,\Delta$. Then $\mathit{free}(R) \subseteq X$, because $\Sigma_0 \vdash R$ Assert is provable. The CSL assertion determined by the second premiss is

$$\Gamma, r(X) : R \vdash_A \{P\}C\{Q\}.$$

Using the CSL Resource rule we can deduce from this the assertion

$$\Gamma \vdash_{A \cup X} \{P * R\}\mathbf{resource}\ r\ \mathbf{in}\ C\{Q * R\},$$

which corresponds to the conclusion in the Reddy rule.

- For the Frame rule suppose that $\Sigma \mid \Delta \vdash \{P\}C'\{Q\}$ and $\Sigma' \vdash R$ Assert are provable in Reddy's logic, so that $\Sigma, \Sigma' \mid \Delta \vdash \{P * R\}C'\{Q * R\}$ is provable from the Frame rule. Let $C'$ decorate $C$. Then $free(R) \subseteq dom(\Sigma')$ and $mod(C') \cap dom(\Sigma') = \{\}$, using Theorem 7.1 again. Let $A = dom(\Sigma)$ and $\Gamma = map\ dom\ \Delta$. The CSL version of the premiss is $\Gamma \vdash_A \{P\}C\{Q\}$. From the above, we have $mod(C) \cap free(R) = \{\}$. So we can use the CSL Frame rule to derive $\Gamma \vdash_{A \cup free(R)} \{P * R\}C\{Q * R\}$. Using Consequence we can then deduce $\Gamma \vdash_{A \cup dom(\Sigma')} \{P * R\}C\{Q * R\}$, which corresponds as required to $\Sigma, \Sigma' \mid \Delta \vdash \{P * R\}C'\{Q * R\}$.

(*End of Proof*)

To establish a converse connection between the logics we argue as follows.

For a tuple of identifier sets $(A, X_1, \ldots, X_n)$ and a subset $Y \subseteq A \cup \bigcup_{i=1}^{n} X_i$, the set of permission contexts $\Sigma, \Sigma_1, \ldots, \Sigma_n$ such that $dom(\Sigma) = A$, $dom(\Sigma_i) = X_i$ for each $i$, and $(\Sigma, \Sigma_1, \ldots, \Sigma_n)(x) = \top$ for all $x \in Y$, is non-empty. We use this fact to guide the choices of permission contexts when constructing a Reddy judgement $\Sigma \mid \Delta \vdash \{P\}C'\{Q\}$ to match a CSL assertion $\Gamma \vdash_A \{P\}C\{Q\}$.

When $\Delta$ is $r_1(\Sigma_1) : R_1, \ldots, r_n(\Sigma_n) : R_n$ and $\Sigma$ is a permission context we may use the abbreviation $\Sigma, \Delta$ for the permission context $\Sigma, \Sigma_1, \ldots, \Sigma_n$. We will also say that the combination $\Sigma, \Delta$ is well-formed when this permission context is well-formed and $\Delta$ is a well-formed permissive resource context.

**Theorem 7.3**
*Let $\Gamma \vdash_A \{P\}C\{Q\}$ be a provable assertion in CSL. Let $\Sigma, \Delta$ be well-formed and suppose that $dom(\Sigma) = A$, $map\ dom\ \Delta = \Gamma$, and for all $x \in mod(C)$, $(\Sigma, \Delta)(x) = \top$. Then there is a decoration $C'$ of $C$ such that the judgement $\Sigma \mid \Delta \vdash \{P\}C'\{Q\}$ is provable in Reddy's logic.*

Proof: For simplicity we will assume that $\mathcal{P}$ is the fractional permissions algebra, although it is easy to adjust the proof details to encompass a general permissions algebra; in the general proof divisibility plays a crucial rôle, and with fractional permissions we can get by with division by 2.

The proof is by induction on proof height. We show that for each CSL inference rule, if the premisses have this property and the side conditions hold, then the conclusion has this property. We give the details for assignment, for parallel composition (where division is needed), regions, and local resource blocks (where we must choose an appropriate decoration); the other rules are simpler and can be handled in similar manner.

- Assignment: Consider a well-formed instance of the CSL assignment rule:

$$\overline{\Gamma \vdash_A \{[e/i]P\}i{:=}e\{P\}}$$

where $\mathit{free}(i{:=}e) \subseteq \mathit{owned}(\Gamma) \cup A$, $i \notin \mathit{owned}(\Gamma)$, and $\mathit{free}(P, e) \subseteq A$. Then $i \in A$. We can pick any combination of $\Sigma$ and $\Delta$ such that $\mathit{dom}(\Sigma) = A, |\Sigma|(i) = \top$, and $\mathit{map}\ \mathit{dom}\ \Delta = \Gamma$. (The set of such combinations is non-empty.) Then we have $\Sigma \vdash i$ Var, $\Sigma \vdash e$ Exp, and $\Sigma \vdash P$ Assert. So the premisses of the relevant instance of Reddy's assignment rule are valid, hence provable, and

$$\Sigma \mid \Delta \vdash \{[e/i]P\}i{:=}e\{P\}$$

is provable.

- Parallel: Consider a well-formed instance of the CSL parallel rule:

$$\frac{\Gamma \vdash_{A_1} \{P_1\}C_1\{Q_1\} \quad \Gamma \vdash_{A_2} \{P_2\}C_2\{Q_2\}}{\Gamma \vdash_{A_1 \cup A_2} \{P_1 * P_2\}C_1 \| C_2\{Q_2\}}$$

with $\mathit{mod}(C_1) \cap A_2 = \mathit{mod}(C_2) \cap A_1 = \{\}$, $\mathit{mod}(C_1) \cap \mathit{free}(C_2) \subseteq \mathit{owned}(\Gamma)$ and $\mathit{mod}(C_2) \cap \mathit{free}(C_2) \subseteq \mathit{owned}(\Gamma)$. We can choose $\Sigma$ and $\Delta$ such that $\mathit{dom}(\Sigma) = A_1 \cup A_2$, $\mathit{map}\ \mathit{dom}\ \Delta = \Gamma$, and $\forall x \in \mathit{mod}(C_1 \| C_2)$, $(\Sigma, \Delta)(x) = \top$. Define permission contexts $\Sigma_1$ and $\Sigma_2$ as follows:

$$\Sigma_1 = \{x^p \in \Sigma \mid x \in A_1 - A_2\} \cup \{x^{p/2} \mid x^p \in \Sigma, x \in A_1 \cap A_2\}$$
$$\Sigma_2 = \{x^p \in \Sigma \mid x \in A_2 - A_1\} \cup \{x^{p/2} \mid x^p \in \Sigma, x \in A_1 \cap A_2\}.$$

Then $|\Sigma| = |\Sigma_1, \Sigma_2|$, and $\mathit{dom}(\Sigma_1) = A_1$, $\mathit{dom}(\Sigma_2) = A_2$. By assumption $\mathit{mod}(C_1) \cap A_2 = \mathit{mod}(C_2) \cap A_1 = \{\}$, so $\forall x \in \mathit{mod}(C_1).\ (\Sigma_1, \Delta)(x) = \top$ and $\forall x \in \mathit{mod}(C_2).\ (\Sigma_2, \Delta)(x) = \top$. It follows by the induction hypothesis that there are decorations $C_1'$ of $C_1$ and $C_2'$ of $C_2$ such that the judgements $\Sigma_1 \mid \Delta \vdash \{P_1\}C_1'\{Q_1\}$ and $\Sigma_2 \mid \Delta \vdash \{P_2\}C_2'\{Q_2\}$ are provable. So by the Reddy Parallel rule we can deduce

$$\Sigma_1, \Sigma_2 \mid \Delta \vdash \{P_1 * P_2\}C_1' \| C_2'\{Q_1 * Q_2\},$$

and hence $\Sigma \mid \Delta \vdash \{P_1 * P_2\}C_1' \| C_2'\{Q_1 * Q_2\}$. Finally, note that $C_1' \| C_2'$ is a decoration of $C_1 \| C_2$, so the result holds as required.

- For Region, consider a well-formed instance

$$\frac{\Gamma \vdash A \cup X\{(P * R) \wedge b\}C\{Q * R\}}{\Gamma, r(X) : R \vdash_A \{P\}\mathbf{with}\ r\ \mathbf{when}\ b\ \mathbf{do}\ C\{Q\}}$$

Let $\Sigma, \Sigma_0, \Delta$ be well-formed and $\mathit{dom}(\Sigma) = A$, $\mathit{dom}(\Sigma_0) = X$, $\mathit{map}\ \mathit{dom}\ \Delta = \Gamma$, and $\forall x \in \mathit{mod}(C).\ (\Sigma, \Sigma_0, \Delta)(x) = \top$. Then $\mathit{dom}(\Sigma, \Sigma_0) = A \cup X$. So by induction hypothesis there is a decoration $C'$ of $C$ such that

$$\Sigma, \Sigma_0 \mid \Delta \vdash \{(P * R) \wedge b\}C'\{Q * R\}$$

is provable. By well-formedness we have $\mathit{free}(P, Q) \subseteq A$, so $\Sigma \vdash P, Q$ Assert is valid. Hence

$$\Sigma \mid \Delta, r(\Sigma_0) : R \vdash \{P\}\mathbf{with}\ r\ \mathbf{when}\ b\ \mathbf{do}\ C'\{Q\}$$

is provable by the Reddy region rule. This judgement corresponds to the conclusion of the CSL rule.

- For RESOURCE, consider a well-formed instance

$$\frac{\Gamma, r(X) : R \vdash_A \{P\}C\{Q\}}{\Gamma \vdash_{A \cup X} \{P * R\}\mathbf{resource}\ r\ \mathbf{in}\ C\{Q * R\}.}$$

By well-formedness we have $free(R) \subseteq X$, and $mod(C) \subseteq A \cup X \cup owned(\Gamma)$. We can choose $\Sigma'$ and $\Delta$ such that $dom(\Sigma') = A \cup X$, $map\, dom\, \Delta = \Gamma$, and $(\Sigma', \Delta)(x) = \top$ for all $x$ in $mod(C)$. We can then split $\Sigma'$ as $\Sigma, \Sigma_0$ with $dom(\Sigma_0) = X$, $dom(\Sigma) = A$. Then by the induction hypothesis there is a decoration $C'$ of $C$ such that such that $\Sigma \mid \Delta, r(\Sigma_0) : R \vdash \{P\}C'\{Q\}$ is provable and matches the CSL assertion $\Gamma, r(X) : R \vdash_A \{P\}C\{Q\}$. Using the Reddy RESOURCE rule we can deduce

$$\Sigma \mid \Delta \vdash \{P * R\}\mathbf{resource}\ r(\Sigma_0)\ \mathbf{in}\ C'\{Q * R\},$$

and since $\mathbf{resource}\ r(\Sigma_0)\ \mathbf{in}\ C'$ is a decoration of $\mathbf{resource}\ r\ \mathbf{in}\ C$, that completes the proof.

(*End of Proof*)

The choices of $\Sigma_0$ and so on in the above proof details are arbitrary up to some explicit constraints on their domains (i.e. on which variables are given a permission) and the insistence that certain variables get total permission, collectively.

## 8 Conclusions

CSL, using rely sets, relaxes the rather stringent syntactic constraints on shared variable usage from Owicki-Gries [8,7]. This logic can handle programs in which shared variables are protected by multiple resources; the inference rules require that a process must acquire *all* protecting resources before writing a shared variable, and must acquire *some* protecting resource before reading a shared variable. In Owicki-Gries logic each shared variable was protected by a single resource. The use of rely sets allows us to avoid the need for non-compositional side conditions dealing with "no other process modifies" constraints.

We have demonstrated a strong connection between CSL and Reddy's logic based on static permissions for variables: each provable judgement in Reddy's logic corresponds to a provable assertion in CSL, and vice versa. The relationship is asymmetric: in general many Reddy judgements correspond to the same CSL assertion, the differences arising because of arbitrary choices of permission or different distributions of permission among the permission contexts appearing in a judgement. Arguably this abundance of derivations is unattractive: a single proof, without book-keeping concerning arbitrary choices, is preferable to a plethora. This connection also shows that CSL provides a form of syntactic control of interference in the same sense as Reddy's logic does.

Our analysis, in the proof details that establish the above connection, indicates a systematic strategy for choosing permission contexts to match a given resource

context and rely set while ensuring that all variables written by a program get total permission, starting from a proof for that program in CSL. In contrast the algorithm in [9] infers which resources need to protect which identifiers (i.e. discovers a suitable resource context) while looking for permission assignments that ensure that each variable occurrence in the program gets "enough" permission, starting from a putative judgement for that program. Our strategy starts with more information, a proven program rather than a potentially provable program, so solves a simpler problem. It would be interesting to explore the Reynolds algorithm further in the light of our results.

Although Reddy's logic was originally formulated in [9] with inference rules that deal with decorated commands, our results show that it is possible to dispense entirely with decorations and use Reddy-style rules for undecorated commands, relying instead implicitly on the structure of a *derivation* to keep track of permission choices. Essentially, the only place where this makes a difference is in the RESOURCE rule, which we can replace by

$$\frac{\Sigma_0 \vdash R \text{ Assert} \quad \Sigma \mid \Delta, r(\Sigma_0) : R \vdash \{P\}C\{Q\}}{\Sigma, \Sigma_0 \mid \Delta \vdash \{P * R\}\mathbf{resource}\ r\ \mathbf{in}\ C\{Q * R\}}$$

The choice of $\Sigma_0$ to "decorate" the local resource name $r$ in the original Reddy rule does not need to be made explicit here; instead we can infer $\Sigma_0$ from the premisses. Of course, the cost of doing this would be that the undecorated command is less informative by itself. With this reformulation of Reddy's logic we would again obtain an analogous connection with CSL: there is a many-one relationship between Reddy derivations and CSL derivations *for the same command*. This version of Reddy's logic has the advantage of involving only conventional commands, not decorated commands; consequently we can use conventional semantic models, such as action traces, to establish soundness. In fact it is straightforward, having done this, to combine action trace semantics with permissions [4] and thereby obtain a soundness proof for Reddy's logic, adjusting the notion of *local enabling* to manage permissions appropriately. One can also deduce soundness by appealing to the inter-derivability of the two logics, and our existing soundness proof of CSL [5]. Either approach seems simpler than the method of [9], which deals with decorated commands and seeks to generalize action traces by introducing pre-actions, pre-traces, busy markers, and "extended" contexts. We believe it is more natural to work with a semantic model in which logical concepts such as permissions, invariants, protection lists, and decorated commands play no rôle.

## References

[1] R. Bornat, C. Calcagno, P. W. O'Hearn, and M. Parkinson. *Permission accounting in separation logic.* POPL 2005, SIGPLAN Notices, 40(1), 259-270.

[2] J. Boyland. *Checking interference with fractional permissions.* Proc. $10^{th}$ Symposium on Static Analysis, R. Cousot, editor. Springer LNCS vol. 2694, pp. 55-72, 2003.

[3] S. Brookes. *A semantics for concurrent separation logic.* Invited paper, CONCUR 2004, London. Springer LNCS 3170, August 2004. Journal version in: *Theoretical Computer Science*, 375(1-3), May 2007.

[4] S. Brookes. *Variables as Resource for Shared-Memory Programs: Semantics and Soundness*. MFPS 2006, Genova, Italy. ENTCS, Volume 158, 123-150. May 2006

[5] S. Brookes. *A Revisionist History of Concurrent Separation Logic*. MFPS 2011, Pittsburgh, USA. ENTCS, 2011.

[6] C.A.R. Hoare, *Towards a Theory of Parallel Programming*. In **Operating Systems Techniques**, C. A. R. Hoare and R. H. Perrott, editors, pp. 61-71, Academic Press, 1972.

[7] P. W. O'Hearn. *Resources, Concurrency, and Local Reasoning*. Invited paper, CONCUR 2004, London. Springer LNCS 3170, pp. 49-67, August 2004. Journal version in: *Theoretical Computer Science*, 375(1-3), 271-307, May 2007.

[8] S. Owicki and D. Gries, *Verifying properties of parallel programs: An axiomatic approach*, Comm. ACM. 19(5):279-285, May 1976.

[9] U. Reddy and J. C. Reynolds. *Syntactic Control of Interference for Separation Logic*. POPL 2012. January, 2012.

[10] J.C. Reynolds, *Separation logic: a logic for shared mutable data structures*, Invited paper. Proc. $17^{th}$ IEEE Conference on Logic in Computer Science, LICS 2002, pp. 55-74. IEEE Computer Society, 2002.

[11] J, C. Reynolds. Invited Talk, MFPS 2011. Pittsburgh, USA, May 2011.

[12] I. Wehrman and J. Berdine. Private communication. January 2011.