Revisiting Concurrent Separation Logic

Pedro Soares[*], António Ravara[†] and Simão Melo de Sousa[‡]

Abstract

We present a new soundness proof of Concurrent Separation Logic (CSL) based on a structural operational semantics (SOS). We build on two previous proofs and develop new auxiliary notions to achieve the goal. One uses a denotational semantics (based on traces). The other is based on SOS, but was obtained only for a fragment of the logic — the Disjoint CSL — which disallows modifying shared variables between concurrent threads. In this work, we lift such a restriction, proving the soundness of full CSL with respect to a SOS. Thus contributing to the development of tools able of ensuring the correctness of realistic concurrent programs. Moreover, given that we used SOS, such tools can be well-integrated in programming environments and even incorporated in compilers.

Keywords: Concurrent Separation Logic; Structural Operational Semantics; Soundness Proof.

1 Introduction

The aim of this work is to present a new soundness proof for Concurrent Separation Logic [7], with respect to a structural operational semantics [11]. This work adapts and extends the results presented by Brookes [4] and by Vafeiadis [16].

The axiomatic verification of programs goes back to Hoare Logic [6]. This seminal work introduces two key ideas, i) the specification of programs by means of what is known by a Hoare triple: $\{P\}C\{Q\}$, where P and Q are first

[*]Universidade do Porto, PT
[†]CITI & DI-FCT, Universidade Nova de Lisboa, PT
[‡]LISP & LIACC& DI-FE, Universidade da Beira Interior, PT
This work was partially funded by Fundação para a Ciência e Tecnologia through AVIACC project, grant PTDC/EIA-CCO/117590, and CITI/FCT/UNL, grant Pest-OE/EEI/UI0527/2014.

order formulae, called the precondition and postcondition respectively, and C is an imperative program; ii) a deductive proof system to ensure the partial correctness of programs. A program is partially correct, if every execution of C from a state respecting the precondition does not abort and when it terminates the postcondition holds for its final state. The state for this logic is formed only by the store, i.e. a partial function that records the value of each variable. Hoare's work gave rise to numerous deductive systems, for instance the Owicki-Gries method ([9, 10]) and Separation Logic ([8, 13]).

The Owicki-Gries method is one of the first attempts to give a resource sensitive proof system for concurrent programs. To do this, Owicki and Gries augmented the programming language with i) parallel composition, $C \parallel C$; ii) local resources, $\mathsf{resource}\ r\ \mathsf{in}\ C$; and iii) a critical region, $\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ C$, where r denotes a resource. Each resource has a mutual exclusion lock, an assertion, called invariant, and a set of variables, called protected variables.

The execution of parallel composition non-deterministically chooses one of the commands to execute first. As usual, the parallel execution is assumed to be weakly fair, i.e. if a command is continually available to be executed, then this command will be eventually selected. The resource command declares a local variable r to be used in C. The critical region command waits for the availability of the resource r, and when B holds, it acquires r and starts the execution of C; the resource r is released upon the execution of C terminates.

The programs derivable by the Owicki-Gries method have to preserve the resource invariants when the resource is free, and respect the protection of variables by resources, i.e. a program needs to acquire all resources protecting a variable, before the program can change that variable. The parallel rule proposed by Owicki [9] requires that every variable occurring in the derivation proof of one command cannot be changed by another command, except for variables protected by a resource such that the variables only appear inside the critical region's proof. Thus, the Owicki-Gries method is not compositional.

Separation Logic (SL) supports reasoning about imperative programs with shared mutable data and consequently about dynamical data structures, such as lists and trees. In order to do this, the assertion and program languages used by Hoare had to be augmented. The assertions are extended with the constructs `emp`, the empty memory; $e \mapsto e'$, a single memory cell e with the value e'; and $P * Q$, two disjoint memory's parts such that one satisfies P and the other satisfies Q. In this settings, the memory is usually represented by the heap — a partial function from the set of locations to the set of values. The store and the heap together define the state of a program.

The programing language is augmented with commands for memory manipulation. Naturally, the proof system is also extended with a rule for each

new command and with a frame rule, used to enlarge the portion of memory considered in the condition of a specification. This rule is crucial to achieve local reasoning: program specifications only need to consider the relevant memory for their execution. Therefore, this local reasoning mechanism can be used to establish the partial correctness of disjoint concurrent programs, i.e. concurrent program which does not change shared variables.

In order to prove the soundness of the frame rule, and thus of local reasoning, it is sufficient to ensure the validity of two key properties: safety monotonicity and the frame property. Safety monotonicity states that if an execution does not abort for a given memory portion, then the execution does not abort for any memory portion that contains the initial one. The frame property says that if a command does not abort for a given memory portion, then every execution on a larger memory corresponds to an execution on the initial memory.

Recently provers based on separation logic were adopted in real industrial projects, Facebook's infer being the most prominent of such tools [5].

Since the introduction of SL, different authors adapted it to the verification of concurrent programs. Vafeiadis and Parkinson introduced RGSep, combining SL with Rely/Guarantee reasoning [17]. Reddy and Reynolds introduced a syntactic control of interference in SL [12], borrowing ideas from works on fractional permissions [2]. O'Hearn proposed Concurrent Separation Logic (CSL), combining SL with the Owicki-Gries method [7]. Brookes formalized CSL, extending the traditional Hoare triples with a resource context Γ and a rely-set A, what leads to specifications of the form $\Gamma \models_A \{P\}C\{Q\}$. A resource context records the invariant and the protected variables of each resource. A rely-set consists of all variables relevant for its derivation tree. This set ensures that CSL is a compositional proof method, proved sound with respect to a denotational semantics based on traces, where a program state is represented by a store, a heap and sets of resources, expressing resource ownership [4]. Actually, the rely-set was introduced after Wehrman and Berdine discovered a counter-example to the initial version of CSL [3], and it is analogous to the set of variables used by Owicki and Gries to check non-interference in their parallel rule.

Alternatively, Vafeiadis proposed a structural operational semantics (SOS) for concurrent programs synchronizing via resources, and proved the soundness of a part of CSL, the Disjoint CSL (DCSL) [16]. DCSL and CSL have different side conditions for the parallel rule. Concurrent threads, in DCSL, must not modify all variables that appear in other Hoare triples, however concurrent threads, in CSL, can not modify variables that belongs to other rely-sets.

Our aim is to remove the disjointness condition and obtain a soundness

proof using a SOS for the full CSL (Section 6). The goal is relevant because CSL has been adopted as the basis for most modern program logics, and it is a step in the development of more expressive provers well integrated in software development environments and compilers. Not only does it allows proving correct concurrent programs manipulating shared resources, but also provides techniques to equip compilers with mechanisms of detecting data-races. Concretely, the contributions of this work are the following:

1. A novel notion of environment transition that simulates actions made by other threads. We define it taking into account the rely-set, available resources and their invariants (Section 5.1). This relation is crucial to study the soundness of the parallel rule;[1]

2. The resource configuration that expresses ownership. It is defined by three sets: owned resources, locked resources, and available resources (Section 4.1). A program state is formed by a store, a heap and a resource configuration. Brookes also used sets of resources to represent resources ownership [4];

3. Illustrative examples that we prove correct in CSL, showing the proof system's expressiveness (Section 3).

This paper is an extended version of [15]. We present herein more examples and sketches of the proofs of the results reported in the short paper (which does not present proofs). Further examples and proofs in full detail can be found in a technical report [14].

The paper is organized as follows: first, we review the syntax of concurrent resource-oriented programs with shared mutable data (Section 2.2) and Concurrent Separation Logic proof system (Section 2.3), following the work of Brookes [4]. Next, we present a structural operational semantics for the previous programs (Section 4.1), along the lines of the work of Vafeiadis [16]. We state important results over this operational semantics for the soundness proof, including safety monotonicity and frame property (Section 4.2). Afterwards, we introduce the environment transition (Section 5.1). Finally, we prove the soundness of Concurrent Separation Logic with respect to the operational semantics we defined (Section 6).

2 Concurrent Separation Logic

We revisit Concurrent Separation Logic (CSL), as presented by Brookes [4]. First, we define the assertion language, then the syntax of commands for

[1] Vafeadis used a completely different notion of environment transition (in RGSep [17]).

concurrent programs, and finally the inference rules for CSL.

2.1 Assertion Language

Consider a set **Var** of *variables*, ranged over by $\mathtt{x}, \mathtt{y}, \ldots$, and a set **Val** of *values*, that includes the integers and the value *null*. These meta-variables may be indexed or primed.

$$\begin{array}{rcl} e & \equiv & x \mid n \mid e_1 + e_2 \mid e_1 - e_2 \mid e_1 \times e_2 \\ B & \equiv & \mathtt{true} \mid \mathtt{false} \mid e_1 = e_2 \mid e_1 < e_2 \mid B_1 \wedge B_2 \mid \neg B \\ P & \equiv & B \mid \neg P \mid P_1 \wedge P_2 \mid \forall x P \mid \mathtt{emp} \mid e \mapsto e'_1, e'_2, \ldots, e'_n \mid P_1 * P_2 \end{array}$$

Figure 1: Syntax of the Assertion Language

The grammar in Figure 1 defines the syntax of the assertion language, where $e \mapsto e'_1, e'_2, \ldots, e'_n$ denotes $e \mapsto e'_1 * (e+1) \mapsto e'_2 * \cdots * (e+n-1) \mapsto e'_n$. We assume the usual definitions of *free variables* of an assertion (FV).

We use the definition of SL for the *validity* of an assertion with respect to the pair (s, h), where s and h are denoted by *storage* and *heap*, respectively, and given by the functions:

$$s : \mathbf{Var} \to \mathbf{Val}, \quad h : \mathbf{Loc} \rightharpoonup \mathbf{Val},$$

where $\mathbf{Loc} \subset \mathbb{N}$ is the set of current locations, for details see e.g. [8, Section 2]. The set of those pairs is denoted by $\mathcal{S}$.

For an assertion P, we write $s, h \models P$ if the assertion is valid for $(s, h) \in \mathcal{S}$, and we write $\models P$ if $s, h \models P$ for every $(s, h) \in \mathcal{S}$. We state a popular result about the validity of assertion, see e.g. [16, Proposition 4.2].

Proposition 1. *Let P be an assertion, $(s, h), (s', h) \in \mathcal{S}$. If $s(x) = s'(x)$, for every $x \in FV(P)$, then*

$$s, h \models P \quad \textit{iff} \quad s', h \models P.$$

For a given heap and storage, the precise assertions uniquely determine the subheap that verifies it. The heap h' is a *subheap* of h, if the domain of h' is contained in the domain of h and the evaluation of h' coincides with the evaluation of h .

Definition 1. *We say that an assertion P is* precise *if for every $(s, h) \in \mathcal{S}$ there is at most one subheap h' of h such that $s, h' \models P$.*

Let **Res** be the set of resources names, which is disjoint from **Var**. The *resource context* Γ is used to represent a shared state. The resource context Γ has the form

$$r_1(X_1) : R_1, r_2(X_2) : R_2, \ldots, r_n(X_n) : R_n, \tag{1}$$

where $r_i \in \mathbf{Res}$ are distinct, R_i are precise assertions and $X_i \subseteq \mathbf{Var}$ such that $FV(R_i) \subseteq X_i$, for each $i = 1, 2, \ldots, n$. The assertion R_i represents a resource invariant. Since CSL has the conjunction rule, the assertions R_i must be precise, as exemplified by Reynolds [7, Section 11].

Let $Res(\Gamma)$ denote the set of *resources names* appearing in Γ, ranged over by r_i. Furthermore, let $PV(\Gamma)$ denote the set of all *protected variables* by resources in Γ, and $PV(r_i) := X_i$ denote the set of protected variables by r_i.

2.2 Programming Language

The language includes the *basic commands* to manipulate storage and heap:

$$c \equiv \mathsf{x{:=}e} \mid \mathsf{x{:=}[e]} \mid \mathsf{[e]{:=}e'} \mid \mathsf{x{:=}cons}(\overline{\mathsf{e}}) \mid \mathsf{disp(e)}.$$

The basic commands use the notation of SL. The bracket parenthesis denotes an access to a heap location. For a vector $\overline{\mathsf{e}} = (\mathsf{e}_1, \ldots, \mathsf{e}_n)$, $\mathsf{x{:=}cons}(\overline{\mathsf{e}})$ allocates n sequential locations with values $\mathsf{e}_1, \ldots, \mathsf{e}_n$. And $\mathsf{disp(e)}$ frees a location.

The following grammar defines the syntax of the *programming language*, $\mathcal{C}$.

$$\begin{aligned} C \;\equiv\; & \mathsf{skip} \mid c \mid C_1 \,;\, C_2 \mid \mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2 \mid \mathsf{while}\ B\ \mathsf{do}\ C \mid \\ & \mathsf{resource}\ r\ \mathsf{in}\ C \mid \mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ C \mid C_1 \parallel C_2 \end{aligned}$$

The set of *modified variables* by a program C, $mod(C)$, consists of all variables x such that the program C has one of the following commands: $\mathsf{x{:=}e}$, $\mathsf{x{:=}[e]}$ or $\mathsf{x{:=}cons(e)}$.

The set of *resources occurring* in a command C is denoted by $Res(C)$ and it consists of all resources names r such that C has one of the following commands: $\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ C$, $\mathsf{resource}\ r\ \mathsf{in}\ C$.

The substitution on C of a resource name r for a resource name r' not occurring in C is denoted by $C[r'/r]$.

The set of auxiliary variables have been useful to deduce more specific post conditions for a program's specification, see e.g. [10]. Those variables do not impact the flow of the program. Next, we give the definition of auxiliary variables for a command.

Definition 2. *Let $C \in \mathcal{C}$. We say that X is a set of* auxiliary variables *for C if every occurrence of $x \in X$ in C is inside an assignment to a variable in X.*

After we have used the auxiliary variables to deduce a specification, we want to remove them from the program. We replace every assignment to an auxiliary variable by the command skip. This replacement is denoted by $C \setminus X$.

2.3 Inference rules

In this section, we present the most relevant inference rules for CSL as stated by Brookes [4]. First, we define what is a well-formed specification in CSL.

Definition 3. *Let Γ be a resource context, $A \subset \boldsymbol{Var}$, P, Q assertions and $C \in \mathcal{C}$. The* specification of a program *has the form*

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

Moreover we say that the specification of the program is well-formed*, if $FV(P,Q) \subseteq A$ and $FV(C) \subseteq A \cup PV(\Gamma)$.*

In Figure 2, we present some inference rules of CSL. The inference rules are only applied for well-formed specifications. The specifications derivable by SL are denoted by $\vdash^{SL}$.

The rule for basic commands are inherited from SL by adding the rely-set (containing all relevant variables for the derivation) and imposing that protected variables are not modified. The sequential and frame rules are very similar to the respective rules of SL, but the rely-set needs to take into account the rely-sets of both programs or the variables of the framed assertion.

In the critical region rule, if the command inside the critical region preserves the invariant, when B is initially respected, then the resource context can be expanded by r. Note that the rely-set does not need to include all protected variables, however the well-formedness of the specification must be preserved. In the local resource rule, we are able to take out a resource from the assumption's resource context to the conclusion's local condition. The parallel rule (PAR) has the side condition below that restricts the interference between programs.

$$mod(C_1) \cap A_2 = mod(C_2) \cap A_1 = \emptyset. \qquad (*)$$

$$\frac{}{\Gamma \vdash_A \{P\}\mathsf{skip}\{P\}}\ (\text{SKP}) \quad \frac{\Gamma \vdash_{A_1} \{P_1\}C_1\{P_2\} \quad \Gamma \vdash_{A_2} \{P_2\}C_2\{P_3\}}{\Gamma \vdash_{A_1 \cup A_2} \{P_1\}C_1\,;C_2\{P_3\}}\ (\text{SEQ})$$

$$\frac{mod(c) \notin PV(\Gamma) \quad \vdash^{SL} \{P\}c\{Q\}}{\Gamma \vdash_A \{P\}c\{Q\}}\ (\text{BC}) \quad \frac{\Gamma \vdash_A \{P\}C\{Q\} \quad mod(C) \cap FV(R) = \emptyset}{\Gamma \vdash_{A \cup FV(R)} \{P * R\}C\{Q * R\}}\ (\text{FRA})$$

$$\frac{\Gamma \vdash_A \{P \wedge B\}C\{P\}}{\Gamma \vdash_A \{P\}\mathsf{while}\ B\ \mathsf{do}\ C\{P \wedge \neg B\}}\ (\text{LP}) \quad \frac{\Gamma \vdash_{A_1} \{P_1\}C\{Q_1\} \quad \Gamma \vdash_{A_2} \{P_2\}C\{Q_2\}}{\Gamma \vdash_{A_1 \cup A_2} \{P_1 \wedge P_2\}C\{Q_1 \wedge Q_2\}}\ (\text{CONJ})$$

$$\frac{\Gamma \vdash_{A_1} \{P \wedge B\}C_1\{Q\} \quad \Gamma \vdash_{A_2} \{P \wedge \neg B\}C_2\{Q\}}{\Gamma \vdash_{A_1 \cup A_2} \{P\}\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2\{Q\}}\ (\text{IF})$$

$$\frac{\Gamma \vdash_A \{P\}C\{Q\} \quad \models P' \Rightarrow P \quad \models Q \Rightarrow Q' \quad A \subseteq A'}{\Gamma \vdash_{A'} \{P'\}C\{Q'\}}\ (\text{CONS})$$

$$\frac{\Gamma \vdash_{A \cup X} \{P\}C\{Q\} \quad X \cap FV(P,Q) = \emptyset \quad X \cap PV(\Gamma) = \emptyset \quad X \text{ is auxiliary for } C}{\Gamma \vdash_A \{P\}C \setminus X\{Q\}}\ (\text{AUX})$$

$$\frac{\Gamma[r'/r] \vdash_A \{P\}C[r'/r]\{Q\} \quad r' \notin Res(C) \quad r' \notin Res(\Gamma)}{\Gamma \vdash_A \{P\}C\{Q\}}\ (\text{REN})$$

$$\frac{\Gamma \vdash_{A_1} \{P_1\}C_1\{Q_1\} \quad \Gamma \vdash_{A_2} \{P_2\}C_2\{Q_2\} \quad (*)}{\Gamma \vdash_{A_1 \cup A_2} \{P_1 * P_2\}C_1 \parallel C_2\{Q_1 * Q_2\}}\ (\text{PAR})$$

$$\frac{\Gamma \vdash_{A \cup X} \{(P \wedge B) * R\}C\{Q * R\}}{\Gamma, r(X) : R \vdash_A \{P\}\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ C\{Q\}}\ (\text{CR}) \quad \frac{\Gamma, r(X) : R \vdash_A \{P\}C\{Q\}}{\Gamma \vdash_{A \cup X} \{P * R\}\mathsf{resource}\ r\ \mathsf{in}\ C\{Q * R\}}\ (\text{RES})$$

Figure 2: Rules of the Inference System

In order to obtain the inference rules of DCSL we erase the rely-set from the CSL inference rules and change the side condition in the parallel rule (PAR) to the following condition:

$$mod(C_1) \cap FV(P_2, C_2, Q_2) = mod(C_2) \cap FV(P_1, C_1, Q_1) = \emptyset.$$

Note that every valid specification in DCSL is also valid in CSL, considering that $A_i = FV(P_i, C_i, Q_i)$, for $i = 1, 2$.

3 Motivating examples

3.1 Semaphore

In this example, we present a simple binary semaphore for two threads. Similar examples were studied in [7, Section 4]. We use the resource invariant to infer the properties of mutual exclusion, absence of deadlocks and starvation.

This example is a solution to the critical region problem in [1, Section 3]. In contrast to the usual solutions, we obtain a simpler solution for the critical region problem, due to the command with r when B do C.

We have the following specifications for the thread p:

$$se(\mathsf{p},\mathsf{q}) : S \vdash_{\emptyset} \{\mathtt{emp}\}P(p)\{\mathtt{emp}\},$$
$$se(\mathsf{p},\mathsf{q}) : S \vdash_{\emptyset} \{\mathtt{emp}\}V(p)\{\mathtt{emp}\},$$

where

- $S \equiv [(\mathsf{p}=0 \wedge \mathsf{q}=0) \vee (\mathsf{p}=1 \wedge \mathsf{q}=0) \vee (\mathsf{p}=0 \wedge \mathsf{q}=1)] \wedge \mathtt{emp}$,
- $P(p) \equiv$ with se when $\mathsf{q}=0$ do p:=1,
- $V(p) \equiv$ with se do p:=0.

Consider the next well-formed specification of programs.

$$\begin{array}{ll} \vdash_{\{\mathsf{p},\mathsf{q}\}} & \{(\mathtt{emp} \wedge \mathsf{q}=0) * S\} \\ & \{\mathsf{q}=0 \wedge \mathtt{emp}\} \\ & \mathsf{p}{:=}1 \\ & \{\mathsf{p}=1 \wedge \mathsf{q}=0 \wedge \mathtt{emp}\} \\ & \{\mathtt{emp} * S\} \end{array}$$

and

$$\begin{array}{ll} \vdash_{\{\mathsf{p},\mathsf{q}\}} & \{\mathtt{emp} * S\} \\ & \{[(0=0 \wedge \mathsf{q}=0) \vee (0=0 \wedge \mathsf{q}=1)] \wedge \mathtt{emp}\} \\ & \mathsf{p}{:=}0 \\ & \{[(\mathsf{p}=0 \wedge \mathsf{q}=0) \vee (\mathsf{p}=0 \wedge \mathsf{q}=1)] \wedge \mathtt{emp}\} \\ & \{\mathtt{emp} * S\} \end{array}$$

Applying the (CR) rule we obtain the desired specifications. Considering the analogous programs and derivations for the thread q we obtain:

$$se(\mathsf{p},\mathsf{q}) : S \vdash_{\emptyset} \{\mathtt{emp}\}P(q)\{\mathtt{emp}\},$$

$$se(\mathsf{p},\mathsf{q}) : S \vdash_{\emptyset} \{\mathtt{emp}\}V(q)\{\mathtt{emp}\},$$

where

- $P(q) \equiv$ with se when $\mathsf{p} = 0$ do $\mathsf{q}{:=}1$ and
- $V(q) \equiv$ with se do $\mathsf{q}{:=}0$.

Using the (PAR) rule, we obtain the next specification.

$$se(\mathsf{p},\mathsf{q}) : S \vdash_{\emptyset} \{\mathtt{emp}\}(P(p)\,;V(p)) \parallel (P(q)\,;V(q))\{\mathtt{emp}\}$$

This program is a solution to the critical region problem. Next, we add a Critical Region between the operation P and Q in the previous specification. And, we discuss the properties of mutual exclusion, absence of deadlocks and starvation.

Consider a Critical Region (C.R.) and the following program

p:=0 ; q:=0;
resource se in

while true do	‖	while true do
P(p);	‖	P(q);
C.R.;	‖	C.R.;
V(p)	‖	V(q)

The execution of the program is inside the critical region for the thread p (q), if $p = 1$ ($q = 1$, respectively). The mutual exclusion follows from the resource invariant S.

The execution of this program is free from deadlock, because the resource invariant implies that one of the control variables $\mathsf{p} = 0 \vee \mathsf{q} = 0$.

After the execution of the critical region, the execution of $V(p)$ or $V(q)$ allows the execution of $P(p)$ and $P(q)$. Assuming the fairness of the parallel execution, the program is free from starvation.

3.2 Concurrent stack

First, to show that DCSL is not as expressive as CSL, we present an example of parallel operations over a stack that cannot be proved correct in the former but can be in the latter.

Let us specify a stack with operations *pop* and *push*. The stack is represented by the resource *st* in the following way

$$st(\{\mathsf{z},\mathsf{y}\}) : stack(\mathsf{z}),$$

where $\{\mathsf{z},\mathsf{y}\}$ is the set of variables protected by *st*, and $stack(\mathsf{z})$ is defined by

$$(\mathsf{z} = null \wedge \mathsf{emp}) \vee (\exists_{a,b} \mathsf{z} \mapsto a, b * stack(b)).$$

The operations *pop* and *push* over a stack are defined below.

$$pop(\mathsf{x1}) \equiv \mathsf{with}\ st\ \mathsf{when}\ \neg(\mathsf{z} = null)\ \mathsf{do}\ (\mathsf{y{:=}z}\ ;\ \mathsf{x1{:=}y}\ ;\ \mathsf{z{:=}[y{+}1]}\ ;\ \mathsf{disp(y{+}1)}),$$

$$push(\mathsf{x2}) \equiv \mathsf{with}\ st\ \mathsf{do}\ (\mathsf{y{:=}cons(x2,z)}; \mathsf{z{:=}y}).$$

The operation *pop* picks the first node of a non-empty stack and passes it to the variable x1. In the following specification, the program performs a pop over a shared stack and it disposes the memory space retrieved by the stack.

$$st(\mathsf{z},\mathsf{y}) : stack(z) \vdash \{\mathsf{emp}\} pop(\mathsf{x1})\ ;\ \mathsf{disp(x1)}\{\mathsf{emp}\}. \tag{2}$$

To prove this result in DCSL, we use the rules of SL and the critical region rule, by omitting the rely-set. Consider the following derivation, that proves the validity of the program inside the critical region.

$$\begin{array}{ll} \vdash & \{\mathsf{emp} * \exists_{a,b}\ \mathsf{z} \mapsto a, b * stack(b)\} \\ & \mathsf{y{:=}z}; \\ & \{\mathsf{emp} * \exists_{a,b}\ \mathsf{y} \mapsto a, b * stack(b)\} \\ & \mathsf{x1{:=}y}; \\ & \{\mathsf{emp} * \exists_{a,b}\ \mathsf{x1} \mapsto a * \mathsf{y{+}1} \mapsto b * stack(b))\} \\ & \mathsf{z{:=}[y{+}1]}; \\ & \{\mathsf{emp} * \exists_{a}\ \mathsf{x1} \mapsto a * \mathsf{y{+}1} \mapsto \mathsf{z} * stack(\mathsf{z}))\} \\ & \mathsf{disp(y{+}1)} \\ & \{(\exists_{a}\ \mathsf{x1} \mapsto a) * (stack(\mathsf{z}))\} \end{array}$$

Applying the critical region rule, the resource *st* appears and we obtain,

$$st(\mathsf{z},\mathsf{y}) : stack(\mathsf{z}) \vdash \{\mathsf{emp}\} pop(\mathsf{x1})\{\ \exists_{a}\ \mathsf{x1} \mapsto a\}.$$

Using the sequential and deallocation rules of SL we get the specification (2).

Now, we turn our attention to the *push* operator over a stack, showing that the following specification is valid in the context of DCSL. Let *push* insert an element $x2$ in the top of a stack.

$$st(\mathsf{z},\mathsf{y}) : stack(\mathsf{z}) \vdash \{\mathsf{emp}\} push(\mathsf{x2})\{\mathsf{emp}\}.$$

As before, from SL inference rules, we obtain the specification below. Then we can apply the critical region rule to obtain the specification above.

$$\begin{array}{ll} \vdash & \{\mathsf{emp} * \mathit{stack}(\mathsf{z})\} \\ & \mathsf{y{:=}cons(x2,z)}; \\ & \{\mathsf{y} \mapsto \mathsf{x2}, \mathsf{z} * \mathit{stack}(\mathsf{z})\} \\ & \{\mathsf{emp} * \exists_{a,b}\ \mathsf{y} \mapsto a, b * \mathit{stack}(b)\} \\ & \mathsf{z{:=}y} \\ & \{\mathsf{emp} * \exists_{a,b}\ \mathsf{z} \mapsto a, b * \mathit{stack}(b)\} \end{array}$$

Until now we have shown that each specification is derivable in DCSL; now we want to study their parallel composition. To apply the parallel rule we need that the variables modified by one program cannot occur free in the other.

$$\mathit{mod}(\mathit{pop}(\mathsf{x1})\,;\mathsf{disp(x1)}) = \{\mathsf{x1},\mathsf{y},\mathsf{z}\}, \quad \mathit{mod}(\mathit{push}(\mathsf{x2})) = \{\mathsf{y},\mathsf{z}\}.$$

The variables z and y are used in both specifications. Hence it is not possible to apply the DCSL parallel rule and obtain a specification for the parallel execution of *pop* and *push*.

In order to express the specification above in the context of CSL it is necessary to define the rely-set for the operation of *pop* and *push*, that are $\{\mathsf{x1}\}$ and $\{\mathsf{x2}\}$, respectively.

It is straightforward, using the derivations above, to infer, in CSL, the following specifications:

$$\mathit{st}(\mathsf{z},\mathsf{y}) : \mathit{stack}(\mathsf{z}) \vdash_{\{\mathsf{x1}\}} \{\mathsf{emp}\}\mathit{pop}(\mathsf{x1})\,;\mathsf{disp(x1)}\{\mathsf{emp}\},$$

$$\mathit{st}(\mathsf{z},\mathsf{y}) : \mathit{stack}(\mathsf{z}) \vdash_{\{\mathsf{x2}\}} \{\mathsf{emp}\}\mathit{push}(\mathsf{x2})\{\mathsf{emp}\}.$$

To apply the CSL parallel rule, we need to check that there is no interference between rely-sets and modified variables. Since $\mathit{mod}(\mathit{push}(\mathsf{x2})) \cap \{\mathsf{x1}\} = \emptyset$ and $\mathit{mod}(\mathit{pop}(\mathsf{x1})\,;\mathsf{disp(x1)}) \cap \{\mathsf{x2}\} = \emptyset$, by parallel rule we infer:

$$\mathit{st}(\mathsf{z},\mathsf{y}) : \mathit{stack}(\mathsf{z}) \vdash_{\{\mathsf{x1},\mathsf{x2}\}} \{\mathsf{emp}\}(\mathit{pop}(\mathsf{x1})\,;\mathsf{disp(x1)}) \parallel \mathit{push}(\mathsf{x2})\{\mathsf{emp}\}.$$

As this example shows we can obtain, at least, simpler specifications using CSL than DCSL, and prove correctness of more programs.

4 Operational Semantics

In this section, we describe a structural operational semantics (SOS) that we use to prove the soundness of CSL. We mostly follow the approach of Vafeiadis [16]. Let us introduce the concept of resource configuration, which records the state of each declared resource.

4.1 Program transition

We start by extending the programming language with a command for executions inside a critical region. We denote this command by $\mathsf{within}\ r\ \mathsf{do}\ C$, where r is an acquired resource and C is a command. In the extended programming language, we can associate to each command a set of *locked resources*, $Locked(C)$ which is inductively defined by:

$$
\begin{aligned}
&Locked(C_1\,;C_2) = Locked(C_1),\\
&Locked(C_1 \parallel C_2) = Locked(C_1) \cup Locked(C_2),\\
&Locked(\mathsf{resource}\ r\ \mathsf{in}\ C) = Locked(C) \setminus \{r\},\\
&Locked(\mathsf{within}\ r\ \mathsf{do}\ C) = Locked(C) \cup \{r\},\\
&Locked(C) = \emptyset,\ \textit{otherwise.}
\end{aligned}
$$

Moreover the set of resources occurring in a command is extended with all resources names r such that the command also includes $\mathsf{within}\ r\ \mathsf{do}\ C$.

Let O, L, D be disjoint pairwise subsets of resources names. We say that $\rho = (O, L, D)$ is a *resource configuration*, where O are resources owned by the running program, L are resources locked by others programs and D are available resources. The set of resources configurations is denoted by $\mathcal{O}$.

We write $r \in \rho$ ($r \notin \rho$) if $r \in (O \cup L \cup D)$ ($r \notin (O \cup L \cup D)$, respectively), and $(O, L, D) \setminus \{r\} := (O \setminus \{r\}, L \setminus \{r\}, D \setminus \{r\})$.

Usually the state of a machine in SL consists of a storage, s, and a heap, h. However, we define a program's state by a triple (s, h, ρ). The program transitions, that define the SOS, are represented by the relation $\rightarrow_p$ defined from the tuple $(C, (s, h, \rho))$ to $(C', (s', h', \rho'))$ or the abort state (abort), where $C, C' \in \mathcal{C}$, $(s, h), (s', h') \in \mathcal{S}$ and $\rho, \rho' \in \mathcal{O}$.

For a basic command c we denote by $[c](s, h)$ the result of executing c for the pair (s, h), in the context of SL. The result of the execution of c on a pair (s, h) can be a pair (s', h') or abort. In Figure 3, we display the program transitions.

Since most of the program transitions are standard, we only emphasize how we manage the resource configuration. First note that it is not changed by any transition of basic commands (BCT). The acquisition of a resource by the transition ($W0$) requires that the resource is available and transfers it to the set of owned resources; the release of a resource made by ($W2$) returns the resource to the set of available resources. The local resource command does not add the resource to the resource configuration, since that would break locality, i.e., the local resource should only be visible to who created it. For the local resource we use the set of locked resources, $Locked(C)$, to determine if a resource should be in the set of owned or available resources. In

$$\frac{}{\mathsf{skip}; C_2, (s,h,\rho) \to_p C_2, (s,h,\rho)}\ (S1) \qquad \frac{C_1, (s,h,\rho) \to_p C_1', (s',h',\rho')}{C_1\,;C_2, (s,h,\rho) \to_p C_1'\,;C_2, (s',h',\rho')}\ (S2)$$

$$\frac{}{\mathsf{while}\ B\ \mathsf{do}\ C, (s,h,\rho) \to_p \mathsf{if}\ B\ \mathsf{then}\ C\ ;\ \mathsf{while}\ B\ \mathsf{do}\ C\ \mathsf{else}\ \mathsf{skip}, (s,h,\rho)}\ (LP)$$

$$\frac{s(B) = \mathtt{true}}{\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, (s,h,\rho) \to_p C_1, (s,h,\rho)}\ (IF1) \qquad \frac{C_1, (s,h,\rho) \to_p C_1', (s',h',\rho')}{C_1 \parallel C_2, (s,h,\rho) \to_p C_1' \parallel C_2, (s',h',\rho')}\ (P1)$$

$$\frac{s(B) = \mathtt{false}}{\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, (s,h,\rho) \to_p C_2, (s,h,\rho)}\ (IF2) \qquad \frac{C_2, (s,h,\rho) \to_p C_2', (s',h',\rho')}{C_1 \parallel C_2, (s,h,\rho) \to_p C_1 \parallel C_2', (s',h',\rho')}\ (P2)$$

$$\frac{r \notin \rho}{\mathsf{resource}\ r\ \mathsf{in}\ \mathsf{skip}, (s,h,\rho) \to_p \mathsf{skip}, (s,h,\rho)}\ (R0) \qquad \frac{}{\mathsf{skip} \| \mathsf{skip}, (s,h,\rho) \to_p \mathsf{skip}, (s,h,\rho)}\ (P3)$$

$$\frac{C, (s,h,(O \cup \{r\}, L, D)) \to_p C', (s',h',\rho') \quad r \notin \rho = (O,L,D) \quad r \in \mathit{Locked}(C)}{\mathsf{resource}\ r\ \mathsf{in}\ C, (s,h,\rho) \to_p \mathsf{resource}\ r\ \mathsf{in}\ C', (s',h',\rho' \setminus \{r\})}\ (R1)$$

$$\frac{C, (s,h,(O, L, D \cup \{r\})) \to_p C', (s',h',\rho') \quad r \notin \rho = (O,L,D) \quad r \notin \mathit{Locked}(C)}{\mathsf{resource}\ r\ \mathsf{in}\ C, (s,h,\rho) \to_p \mathsf{resource}\ r\ \mathsf{in}\ C', (s',h',\rho' \setminus \{r\})}\ (R2)$$

$$\frac{\rho = (O, L, D \cup \{r\}) \quad \rho' = (O \cup \{r\}, L, D) \quad s(B) = \mathtt{true}}{\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ C, (s,h,\rho) \to_p \mathsf{within}\ r\ \mathsf{do}\ C, (s,h,\rho')}\ (W0)$$

$$\frac{r \in O \quad C, (s,h,(O \setminus \{r\}, L, D)) \to_p C', (s',h',(O',L',D'))}{\mathsf{within}\ r\ \mathsf{do}\ C, (s,h,(O,L,D)) \to_p \mathsf{within}\ r\ \mathsf{do}\ C', (s',h',(O' \cup \{r\}, L', D'))}\ (W1)$$

$$\frac{\rho = (O \cup \{r\}, L, D) \quad \rho' = (O, L, D \cup \{r\})}{\mathsf{within}\ r\ \mathsf{do}\ \mathsf{skip}, (s,h,\rho) \to_p \mathsf{skip}, (s,h,\rho')}\ (W2) \qquad \frac{[c](s,h) = (s',h')}{c, (s,h,\rho) \to_p \mathsf{skip}, (s',h',\rho)}\ (BCT)$$

Figure 3: Program Transitions

$$\frac{r \in \rho}{\mathsf{resource}\ r\ \mathsf{in}\ C, (s,h,\rho) \to_p \mathsf{abort}}\ (RA) \qquad \frac{r \notin \rho}{\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ C, (s,h,\rho) \to_p \mathsf{abort}}\ (WA)$$

$$\frac{r \in \mathit{Locked}(C) \quad C, (s,h,(O \cup \{r\}, L, D)) \to_p \mathsf{abort}}{\mathsf{resource}\ r\ \mathsf{in}\ C, (s,h,(O,L,D)) \to_p \mathsf{abort}}\ (RA1) \qquad \frac{[c](s,h) = \mathsf{abort}}{c, (s,h,\rho) \to_p \mathsf{abort}}\ (BCA)$$

$$\frac{r \notin \mathit{Locked}(C) \quad C, (s,h,(O,L,D \cup \{r\})) \to_p \mathsf{abort}}{\mathsf{resource}\ r\ \mathsf{in}\ C, (s,h,(O,L,D)) \to_p \mathsf{abort}}\ (RA2) \qquad \frac{C_1, (s,h,\rho) \to_p \mathsf{abort}}{C_1 \,;\, C_2, (s,h,\rho) \to_p \mathsf{abort}}\ (SA)$$

$$\frac{C, (s,h,\rho \setminus \{r\}) \to_p \mathsf{abort}}{\mathsf{within}\ r\ \mathsf{do}\ C, (s,h,\rho) \to_p \mathsf{abort}}\ (WA1) \qquad \frac{C_1, (s,h,\rho) \to_p \mathsf{abort}}{C_1 \parallel C_2, (s,h,\rho) \to_p \mathsf{abort}}\ (PA1)$$

$$\frac{r \notin O}{\mathsf{within}\ r\ \mathsf{do}\ C, (s,h,(O,L,D)) \to_p \mathsf{abort}}\ (WA2) \qquad \frac{C_2, (s,h,\rho) \to_p \mathsf{abort}}{C_1 \parallel C_2, (s,h,\rho) \to_p \mathsf{abort}}\ (PA2)$$

Figure 4: Abort Transitions

Proposition 3, we prove that $\mathit{Locked}(C)$ is equal to the set of owned resources along an execution starting in a non-extended command.

In Figure 4, we include transitions that abort. As in SL, a memory fault causes the program to abort. The parallel command aborts if one of its commands aborts. The local resource command aborts, if the command tries to create a pre-existing resource. The critical region command aborts if it tries to acquire an undeclared resource, if the execution inside the critical region aborts, or if an acquired resource is not in the set of owned resources.

Next, we check that program transitions are well-defined.

Proposition 2. *Let $C, C' \in \mathcal{C}$, $(s,h), (s',h') \in \mathcal{S}$ and $O, L, D, O', L', D' \subset$ **Res**. If $(O,L,D) \in \mathcal{O}$ and*

$$C, (s,h,(O,L,D)) \to_p C', (s',h',(O',L',D')),$$

then $(O',L',D') \in \mathcal{O}$, $L = L'$ and $O \cup D = O' \cup D'$.

Proof. We prove the result by induction on the rules of $\to_p$.

Let $C, C' \in \mathcal{C}$, $(s,h), (s',h') \in \mathcal{S}$, $O, O', L, L', D, D' \subseteq$ **Res** such that $(O,L,D) \in \mathcal{O}$ and

$$C, (s,h,(O,L,D)) \to_p C', (s',h',(O',L',D')).$$

The proof is immediate for the rules $(S1)$, $(S2)$, (LP), $(IF1)$, $(IF2)$, $(P1)$, $(P2)$, $(P3)$, $(R0)$ and (BCT).

If the transition is $(R1)$. We have that $r \notin (O, L, D)$, $C = \mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}$, $C' = \mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}'$ and $r \in \mathit{Locked}(\tilde{C})$ such that

$$\tilde{C}, (s, h, (O \cup \{r\}, L, D)) \to_p \tilde{C}', (s', h', (O'', L'', D'')),$$

where $(O'', L'', D'') \setminus \{r\} = (O', L', D')$.

By induction $L = L''$, $(O'', L'', D'') \in \mathcal{O}$ and $O \cup \{r\} \cup D = O'' \cup D''$.

From $r \notin L$, it follows that $r \notin L''$. Hence $L = L'$. Because $r \notin (O, L, D)$,

$$O' \cup D' = (O'' \cup D'') \setminus \{r\} = (O \cup \{r\} \cup D) \setminus \{r\} = O \cup D.$$

From $O' = O'' \setminus \{r\}$ and $D' = D'' \setminus \{r\}$, we obtain that

$$O' \cap D' = (O'' \setminus \{r\}) \cap (D'' \setminus \{r\}) \subseteq O'' \cap D'' = \emptyset,$$

$$O' \cap L' = (O'' \setminus \{r\}) \cap L'' \subseteq O'' \cap L'' = \emptyset,$$
$$D' \cap L' = (D'' \setminus \{r\}) \cap L'' \subseteq D'' \cap L'' = \emptyset.$$

Hence $(O', L', D') \in \mathcal{O}$.

The case $(R2)$ is analogous to the previous one.

If the transition is given by $(W0)$. We have $D' = D \setminus \{r\}$, $O' = O \cup \{r\}$, $L = L'$ and $r \in D$. And

$$O' \cap D' = (O \cup \{r\}) \cap (D \setminus \{r\}) = (O \cap (D \setminus \{r\})) \subseteq O \cap D = \emptyset,$$

$$O' \cap L' = (O \cup \{r\}) \cap L = (O \cap L) \cup (\{r\} \cap L) = \{r\} \cap L = \emptyset,$$
$$L' \cap D' = L \cap (D \setminus \{r\}) \subseteq L \cap D = \emptyset.$$

Therefore $(O', L', D') \in \mathcal{O}$. Moreover, $O' \cup D' = (O \cup \{r\}) \cup (D \setminus \{r\}) = O \cup D$ and $L = L'$.

If the transition is given by $(W1)$. We have $r \in O$, $C = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}$ and $C' = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}'$ such that

$$\tilde{C}, (s, h, (O \setminus \{r\}, L, D)) \to_p \tilde{C}', (s', h', (O'', L'', D'')),$$

where $O' = O'' \cup \{r\}$, $L' = L''$, $D' = D''$.

By induction hypothesis, we know that $(O'', L'', D'') \in \mathcal{O}$, $L = L''$ and $(O \setminus \{r\}) \cup D = O'' \cup D''$.

Therefore $L = L'$ and $O \cup D = \{r\} \cup O'' \cup D'' = O' \cup D'$.

It remains to check that $(O', L', D') \in \mathcal{O}$, this follows from

$$O' \cap L' = (O'' \cup \{r\}) \cap L'' = (O'' \cap L'') \cup (\{r\} \cap L) = \emptyset,$$

$$O' \cap D' = (O'' \cup \{r\}) \cap D'' = (O'' \cap D'') \cup (\{r\} \cap D'') = \emptyset,$$

$$L' \cap D' = L'' \cap D'' = \emptyset.$$

If the transition is given by $(W2)$. We have $r \in O$, $L = L'$, $D' = D \cup \{r\}$ and $O' = O \setminus \{r\}$. Note that

$$O' \cap L' \subseteq O \cap L = \emptyset,$$

$$O' \cap D' = (O \setminus \{r\}) \cap (D \cup \{r\}) \subseteq (O \cap D) \cup ((O \setminus \{r\}) \cap \{r\}) = \emptyset,$$

$$L' \cap D' = L \cap (D \cup \{r\}) = (L \cap D) \cup (L \cap \{r\}) = \emptyset.$$

Then $(O', L', D') \in \mathcal{O}$, $L = L'$ and $O' \cup D' = (O \setminus \{r\}) \cup (D \cup \{r\}) = O \cup D$. □

We say that a command C' is *reachable* from a CSL's command C if there are $(s, h, \rho), (s', h', \rho')$ and k such that

$$C, (s, h, \rho) \rightarrow_p^k C', (s', h', \rho')$$

and $C, (s, h, \rho) \not\rightarrow_p^j \mathsf{abort}$ for every $j \leq k$, where $\rightarrow_p^i$ denotes the composition of i transitions. In the next proposition, we see that owned resources are equal to locked resources, along an execution starting from a non-extended command.

Proposition 3. *Let $C, C' \in \mathcal{C}$, $(s, h), (s', h') \in \mathcal{S}$, $\rho' \in \mathcal{O}$, Γ be a resource context, and $k \geq 0$ such that C' is reachable from C.*

If $C, (s, h, (\emptyset, \emptyset, Res(\Gamma)) \rightarrow_p^k C', (s', h', \rho')$, then

$$\rho' = (Locked(C'), \emptyset, Res(\Gamma) \setminus Locked(C')).$$

Proof. Let $C, C' \in \mathcal{C}$, $(s, h), (s', h') \in \mathcal{S}$, $\rho' \in \mathcal{O}$, Γ be a resource context, and $k \geq 0$ such that C' is reachable from C.

The proof is done by induction on k.

Let $k = 0$. Then $C' = C$ is a non-extended command and $Locked(C) = \emptyset$.

Let $k = n + 1$. Then there exist C'', s'', h'', ρ'' such that

$$C, (s, h, (\emptyset, \emptyset, Res(\Gamma)) \rightarrow_p^n C'', (s'', h'', \rho'') \rightarrow_p C', (s', h', \rho').$$

Note that C'' is reachable from C. By the induction hypothesis on k, we have that

$$\rho'' = (Locked(C''), \emptyset, Res(\Gamma) \setminus Locked(C'')).$$

Now, we prove the result from k to $k + 1$ by induction on the program transitions. The proof is immediate or an immediate consequence of the induction on the program transition for all transitions except $(W0)$ and $(W2)$, which change the locked resources.

However in both cases the resulting resource configuration preserves that

$$\rho' = (Locked(C'), \emptyset, Res(\Gamma) \setminus Locked(C')).$$ □

The proposition above reinforces the idea that the transitions $(R1)$ and $(R2)$ are well defined. Furthermore, it completely describes the resource configuration along an execution.

4.2 Properties of program transitions

We state now the main properties of the program transitions. We start with the safety monotonicity and the frame property that are essential to show the soundness of the frame rule, as well as of the parallel rule. The property of safety monotonicity and frame property original appears in the context of Separation Logic and are still valid in the Concurrent Separation Logic.

Let h, g be heaps. If they have disjoint domains we write $h \perp g$, and we denote by $h \uplus g$ the union of disjoint heaps. If g is a subheap of h, $h \setminus g$ denotes the heap h restricted to $dom(h) \setminus dom(g)$.

Proposition 4. *Let $C \in \mathcal{C}$, $(s,h) \in \mathcal{S}$, and $\rho \in \mathcal{O}$. Suppose h_F is a heap such that $h \perp h_F$. If $C, (s,h,\rho) \not\to_p \mathsf{abort}$, then $C, (s, h \uplus h_F, \rho) \not\to_p \mathsf{abort}$.*

Proposition 5. *Let $C, C' \in \mathcal{C}$, $(s,h), (s',h') \in \mathcal{S}$, and $\rho, \rho' \in \mathcal{O}$.*

Suppose h_F is a heap such that $h \perp h_F$. If $C, (s, h \uplus h_F, \rho) \to_p C', (s', h', \rho')$ and $C, (s,h,\rho) \not\to_p \mathsf{abort}$, then h_F is a subheap of h' and

$$C, (s,h,\rho) \to_p C', (s', h' \setminus h_F, \rho').$$

The proofs follow a standard pattern (See 7 for the proofs).

By safety monotonicity and frame property we know that the execution of parallel commands only affects his own heap; however it is necessary to have dual properties for the resource configuration. Next, we state these dual properties.

Proposition 6. *Let $C \in \mathcal{C}$, $(s,h) \in \mathcal{S}$, and $(O_1 \cup O_2, L, D), (O_1, L \cup O_2, D) \in \mathcal{O}$. If $C, (s,h,(O_1, L \cup O_2, D)) \not\to_p \mathsf{abort}$, then*

$$C, (s,h,(O_1 \cup O_2, L, D)) \not\to_p \mathsf{abort}.$$

Proof. We will prove the contra-position by induction on the rules of $\to_p$.

Let $C \in \mathcal{C}$, $(s,h) \in \mathcal{S}$, and $(O_1 \cup O_2, L, D), (O_1, L \cup O_2, D) \in \mathcal{O}$ such that

$$C, (s,h,(O_1 \cup O_2, L, D)) \to_p \mathsf{abort}.$$

If the transition is given by (BCA). The transition is independent from the resource configuration. Then

$$C, (s,h,(O_1, L \cup O_2, D)) \to_p \mathsf{abort}.$$

If the transition is given by (RA) or (WA). The conclusion follows from

$$r \in (O_1, L \cup O_2, D) \text{ iff } r \in (O_1 \cup O_2, L, D).$$

If the transition is given by $(WA2)$. We have $r \notin O_1 \cup O_2$. Then $r \notin O_1$ and

$$C, (s, h, (O_1, L \cup O_2, D)) \to_p \mathsf{abort}.$$

If the transition is given by $(PA1)$. We have

$$C_1, (s, h, (O_1 \cup O_2, L, D)) \to_p \mathsf{abort}.$$

Using the induction hypotheses, $C_1, (s, h, (O_1, L \cup O_2, D)) \to_p \mathsf{abort}$. Then

$$C, (s, h, (O_1, L \cup O_2, D)) \to_p \mathsf{abort}.$$

The cases $(PA2)$, (SA), $(RA1)$, $(RA2)$ and $(WA1)$ are similar to the previous case. □

The dual of the frame property for resource configurations is the following.

Proposition 7. *Let $C, C' \in \mathcal{S}$, $(s, h), (s', h') \in \mathcal{S}$, and $\rho_1, \rho_2, \rho' \in \mathcal{O}$ such that $\rho' = (O', L, D')$, $\rho_1 = (O_1 \cup O_2, L, D)$ and $\rho_2 = (O_1, L \cup O_2, D)$. If $C, (s, h, \rho_2) \not\to_p \mathsf{abort}$ and $C, (s, h, \rho_1) \to_p C', (s', h', \rho')$, then $O_2 \subseteq O'$ and*

$$C, (s, h, \rho_2) \to_p C', (s', h', (O' \setminus O_2, L \cup O_2, D')).$$

Proof. Let $C, C', s, s', h, h', O_1, L, O_2, D, O', D'$ as stated in the proposition.

Suppose that $C, (s, h, (O_1, L \cup O_2, D)) \not\to_p \mathsf{abort}$ and

$$C, (s, h, (O_1 \cup O_2, L, D)) \to_p C', (s', h', (O', L, D')).$$

The prove is done by induction on the program transitions.

If the transition is given by $(S1)$, (LP), $(IF1)$, $(IF2)$, (BCT) or $(P3)$, the transition does not depend on the resource configuration. Then, the conclusion is immediate.

If the transition is given by $(W0)$.

We have $C = \mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}$, $C' = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}$, $r \in D$, $s(B) = \texttt{true}$, $s' = s$, $h' = h$, $O' = O_1 \cup O_2 \cup \{r\}$ and $D' = D \setminus \{r\}$.

Then $O_2 \subseteq O'$ and $O' \setminus O_2 = O_1 \cup \{r\}$. Therefore

$$C, (s, h, (O_1, L \cup O_2, D)) \to_p C', (s', h', (O' \setminus O_2, L \cup O_2, D')).$$

If the transition is given by $(W1)$.

We have $C = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}$, $C' = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}'$, $r \in (O_1 \cup O_2) \cap O'$ and

$$\tilde{C}, (s, h, (O_1 \cup O_2 \setminus \{r\}, L, D)) \rightarrow_p \tilde{C}', (s', h', (O' \setminus \{r\}, L, D')).$$

From $C, (s, h, (O_1, L \cup O_2, D)) \not\rightarrow_p \mathsf{abort}$, we know that $r \in O_1$ and

$$\tilde{C}, (s, h, (O_1 \setminus \{r\}, L \cup O_2, D)) \not\rightarrow_p \mathsf{abort}.$$

Now, we can apply the induction hypothesis to conclude that $O_2 \subseteq O' \setminus \{r\}$ and

$$\tilde{C}, (s, h, (O_1 \setminus \{r\}, L \cup O_2, D)) \rightarrow_p \tilde{C}', (s', h', ((O' \setminus \{r\}) \setminus O_2, L \cup O_2, D')).$$

Note that $O_2 \subseteq O' \setminus \{r\} \subseteq O'$ and $(O' \setminus \{r\}) \setminus O_2 = (O' \setminus O_2) \setminus \{r\}$.
From $r \in O_1 \cap O'$ and $r \notin O_2$, we know that $r \in O_1 \cap (O' \setminus O_2)$. Therefore

$$C, (s, h, (O_1, L \cup O_2, D)) \rightarrow_p C', (s', h', (O' \setminus O_2, L \cup O_2, D')).$$

If the transition is given by $(W2)$.
We have $C = \mathsf{within}\ r\ \mathsf{do}\ \mathsf{skip}$, $C' = \mathsf{skip}$, $s' = s$, $h' = h$, $r \in O_1 \cup O_2$, $O' = (O_1 \cup O_2) \setminus \{r\}$ and $D' = D \cup \{r\}$.
As before, we know that $r \in O_1$. Hence we can rewrite the set of owned resources in the following expression

$$O' = (O_1 \setminus \{r\}) \cup O_2.$$

Then $O_2 \subseteq O'$ and

$$C, (s, h, (O_1, L \cup O_2, D)) \rightarrow_p C', (s', h', (O' \setminus O_2, L \cup O_2, D')).$$

If the transition is given by $(R0)$.
We have $C = \mathsf{resource}\ r\ \mathsf{in}\ \mathsf{skip}$, $C' = \mathsf{skip}$, $s' = s$, $h' = h$, $r \notin (O_1 \cup O_2, L, D)$, $O' = O_1 \cup O_2$ and $D' = D$. Then $O_2 \subseteq O'$.
From $r \notin (O_1 \cup O_2, L, D)$, we know that $r \notin (O_1, L \cup O_2, D)$. Therefore

$$C, (s, h, (O_1, L \cup O_2, D)) \rightarrow_p C', (s', h', (O' \setminus O_2, L \cup O_2, D')).$$

If the transition is given by $(R1)$.
We have $C = \mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}$, $C' = \mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}'$, $r \notin (O_1 \cup O_2, L, D)$, $r \in \mathit{Locked}(\tilde{C})$ and

$$\tilde{C}, (s, h, (O_1 \cup O_2 \cup \{r\}, L, D)) \rightarrow_p \tilde{C}', (s', h', (O'', L, D'')),$$

such that $O'' \cup D'' = O' \cup D' \cup \{r\}$.

From $C, (s, h, (O_1, L \cup O_2, D)) \not\rightarrow_p \mathsf{abort}$, we know that $r \notin (O_1, L \cup O_2, D)$ and

$$\tilde{C}, (s, h, (O_1 \cup \{r\}, L \cup O_2, D)) \not\rightarrow_p \mathsf{abort}.$$

By induction hypothesis, we have that $O_2 \subseteq O''$ and

$$\tilde{C}, (s, h, (O_1 \cup \{r\}, L \cup O_2, D)) \rightarrow_p \tilde{C}, (s, h, (O'' \setminus O_2, L \cup O_2, D'')).$$

From $O'' \subseteq O' \cup \{r\}$ and $r \notin O_2$, we have that $O_2 \subseteq O'$.
Moreover, we get that $(O'' \setminus O_2) \cup D'' = (O' \setminus O_2) \cup D' \cup \{r\}$. Therefore

$$C, (s, h, (O_1, L \cup O_2, D)) \rightarrow_p C', (s', h', (O' \setminus O_2, L \cup O_2, D')).$$

The case $(R2)$ is similar to the previous case.
If the transition is given by $(P1)$.
We have $C = C_1 \parallel C_2$, $C' = C_1' \parallel C_2$ and

$$C_1, (s, h, (O_1 \cup O_2, L, D)) \rightarrow_p C_1', (s', h', (O', L, D')).$$

And $C_1, (s, h, (O_1, L \cup O_2, D)) \not\rightarrow_p \mathsf{abort}$, since $C, (s, h, (O_1, L \cup O_2, D)) \not\rightarrow_p \mathsf{abort}$.

By the induction hypothesis, we conclude that $O_2 \subseteq O'$ and

$$C_1, (s, h, (O_1, L \cup O_2, D)) \rightarrow_p C_1', (s', h', (O' \setminus O_2, L \cup O_2, D')).$$

Therefore

$$C, (s, h, (O_1, L \cup O_2, D)) \rightarrow_p C', (s', h', (O' \setminus O_2, L \cup O_2, D')).$$

The cases $(P2)$ and $(S2)$ are similar to the previous case. □

The previous propositions allow us to make a correspondence between the transitions in a parallel execution to transitions of its commands executed independently.

For the soundness of the renaming rule, we prove that the execution of a program and its renaming version are equivalents.

Proposition 8. *Let $C, C' \in \mathcal{C}$, $(s, h), (s', h') \in \mathcal{S}$, $\rho, \rho' \in \mathcal{O}$, and $r, r' \in \boldsymbol{Res}$ such that $r' \notin Res(C)$ and $r' \notin \rho$.*

1. *$C, (s, h, \rho) \not\rightarrow_p \mathsf{abort}$ if and only if $C[r'/r], (s, h, \rho[r'/r]) \not\rightarrow_p \mathsf{abort}$.*
2. *$C, (s, h, \rho) \rightarrow_p C', (s', h', \rho')$ if and only if*

$$C[r'/r], (s, h, \rho[r'/r]) \rightarrow_p C'[r'/r], (s', h', \rho'[r'/r]).$$

Proof. Let $C, C' \in \mathcal{C}$, $(s,h),(s',h') \in \mathcal{S}$, $\rho, \rho' \in \mathcal{O}$, and $r, r' \in \mathbf{Res}$ such that $r' \notin Res(C)$ and $r' \notin \rho$.

Note that $C[r'/r][r/r'] = C$ and $\rho[r'/r][r/r'] = \rho$. Hence, we only need to prove one direction of the equivalence. We prove both by induction on $\rightarrow_p$.

For the first one, we suppose that $C,(s,h,\rho) \rightarrow_p \mathsf{abort}$ and prove that $C[r'/r],(s,h,\rho[r'/r]) \rightarrow_p \mathsf{abort}$.

Suppose that the transition to the abort state is given by (BCA).

Note that the transitions is independent from the resource configuration and $C[r'/r] = C$. Then

$$C[r'/r],(s,h,\rho[r'/r]) \rightarrow_p \mathsf{abort}.$$

Suppose that the transition is given by $(PA1)$. Then $C = C_1 \parallel C_2$ and

$$C_1,(s,h,\rho) \rightarrow_p \mathsf{abort}.$$

We know that $r' \notin Res(C_1)$, because $r' \notin Res(C)$. Using the induction hypothesis, we have that

$$C_1[r'/r],(s,h,\rho[r'/r]) \rightarrow_p \mathsf{abort}.$$

Hence

$$C[r'/r],(s,h,\rho[r'/r]) \rightarrow_p \mathsf{abort}.$$

The cases $(P2)$ and (SA) are identical to the previous case.

Suppose that the transition is given by (RA). We have $C = \mathsf{resource}\ \hat{r}\ \mathsf{in}\ \tilde{C}$ and $\hat{r} \in \rho$.

Note that $C[r'/r] = \mathsf{resource}\ \hat{r}[r'/r]\ \mathsf{in}\ \tilde{C}[r'/r]$ and

$$\hat{r} \in \rho \ \text{ iff } \ \hat{r}[r'/r] \in \rho[r'/r].$$

Therefore

$$C[r'/r],(s,h,\rho[r'/r]) \rightarrow_p \mathsf{abort}.$$

The cases (WA) and $(WA2)$ are analogous to the case before.

Suppose that the transition is given by $(RA1)$. We have $C = \mathsf{resource}\ \hat{r}\ \mathsf{in}\ \tilde{C}$, $\hat{r} \in Locked(\tilde{C})$ $\hat{r} \notin \rho = (O,L,D)$ and

$$\tilde{C},(s,h,(O \cup \{\hat{r}\},L,D)) \rightarrow_p \mathsf{abort}.$$

We have that $r' \notin Res(\tilde{C}) \subset Res(C)$. From the induction hypothesis, we conclude that

$$\tilde{C}[r'/r],(s,h,(O[r'/r] \cup \{\hat{r}[r'/r]\},L[r'/r],D[r'/r])) \rightarrow_p \mathsf{abort}.$$

Note that $C[r'/r] = \mathsf{resource}\ \hat{r}[r'/r]\ \mathsf{in}\ \tilde{C}[r'/r]$ and

$$\hat{r} \in \mathit{Locked}(\tilde{C}) \ \text{ iff } \ \hat{r}[r'/r] \in \mathit{Locked}(\tilde{C}[r'/r]).$$

Therefore

$$C[r'/r], (s, h, \rho[r'/r]) \rightarrow_p \mathsf{abort}.$$

The case $(RA2)$ is analogous to the previous case.

Suppose that the transition is $(WA1)$. We have $C = \mathsf{within}\ \hat{r}\ \mathsf{do}\ \tilde{C}$ and

$$\tilde{C}, (s, h, \rho \setminus \{\hat{r}\}) \rightarrow_p \mathsf{abort}.$$

We know that $r' \notin \mathit{Res}(\tilde{C})$ and $r' \notin \rho \setminus \{\hat{r}\}$, because $r' \notin \mathit{Res}(C)$ and $r' \notin \rho$ respectively. By induction hypothesis, we have the following transition

$$\tilde{C}[r'/r], (s, h, (\rho \setminus \{\hat{r}\})[r'/r]) \rightarrow_p \mathsf{abort}.$$

Note that $(\rho \setminus \{\hat{r}\})[r'/r] = \rho[r'/r] \setminus \{\hat{r}[r'/r]\}$. Therefore

$$C[r'/r], (s, h, \rho[r'/r]) \rightarrow_p \mathsf{abort}.$$

This concludes the proof of the first equivalence. For the second equivalence, we suppose that $C, (s, h, \rho) \rightarrow_p C', (s', h', \rho')$. And show that

$$C[r'/r], (s, h, \rho[r'/r]) \rightarrow_p C'[r'/r], (s', h', \rho'[r'/r]).$$

Suppose that the transition is given by (BCT), (LP), $(IF1)$, $(IF2)$, $(S1)$ or $(P3)$. The transition does not depend in the resource context or the resource names. So, $C[r'/r], (s, h, \rho[r'/r]) \rightarrow_p C'[r'/r], (s', h', \rho'[r'/r])$.

Suppose that the transition is given by $(P1)$, $(P2)$ or $(S2)$. Using the induction hypothesis, it is straightforward that

$$C[r'/r], (s, h, \rho[r'/r]) \rightarrow_p C'[r'/r], (s', h', \rho'[r'/r]).$$

Suppose that the transition is given by $(R0)$. Then $C = \mathsf{resource}\ \hat{r}\ \mathsf{in}\ \mathsf{skip}$, $C' = \mathsf{skip}$ and $\hat{r} \notin \rho$. Note that $\hat{r}[r'/r] \notin \rho[r'/r]$. Hence

$$C[r'/r], (s, h, \rho[r'/r]) \rightarrow_p C'[r'/r], (s', h', \rho'[r'/r]).$$

Suppose that the transition is given by $(R1)$. Then $C = \mathsf{resource}\ \hat{r}\ \mathsf{in}\ \tilde{C}$, $C' = \mathsf{resource}\ \hat{r}\ \mathsf{in}\ \tilde{C}'$, $\hat{r} \notin \rho$, $\hat{r} \in \mathit{Locked}(C)$ and

$$\tilde{C}, (s, h, (O \cup \{\hat{r}\}, L, D)) \rightarrow_p \tilde{C}', (s', h', \rho''),$$

where $\rho'' \setminus \{\hat{r}\} = \rho'$.

Note that $r' \notin Res(\tilde{C})$ and $r' \notin (O \cup \{\hat{r}\}, L, D)$. By the induction hypothesis, we have the following transition

$$\tilde{C}[r'/r], (s, h, (O \cup \{\hat{r}\}, L, D)[r'/r]) \rightarrow_p \tilde{C}'[r'/r], (s', h', \rho''[r'/r]).$$

Moreover, we know that $(\rho'' \setminus \{\hat{r}\})[r'/r] = \rho''[r'/r] \setminus \{\hat{r}[r'/r]\} = \rho'[r'/r]$ and $\hat{r}[r'/r] \in Locked(C[r'/r])$. Therefore,

$$C[r'/r], (s, h, \rho[r'/r]) \rightarrow_p C'[r'/r], (s', h', \rho'[r'/r]).$$

The cases $(R2)$ and $(W1)$ are analogous to the previous case.

Suppose that it is $(W0)$. Then $s(B) = \texttt{true}$, $C = \mathsf{with}\ \hat{r}\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}$, $C' = \mathsf{within}\ \hat{r}\ \mathsf{do}\ \tilde{C}$, $\rho = (O, L, D \cup \{\hat{r}\})$ and $\rho' = (O \cup \{\hat{r}\}, L, D)$.

When $\hat{r} \neq r$, the conclusion is immediate. Suppose that $\hat{r} = r$.

We have $C[r'/r] = \mathsf{with}\ r'\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}[r'/r]$, $C'[r'/r] = \mathsf{within}\ r'\ \mathsf{do}\ \tilde{C}[r'/r]$, $\rho[r'/r] = (O, L, D \cup \{r'\})$ and $\rho'[r'/r] = (O \cup \{r'\}, L, D)$.

It follows that

$$C[r'/r], (s, h, \rho[r'/r]) \rightarrow_p C'[r'/r], (s', h', \rho'[r'/r]).$$

The case $(W2)$ is analogous to the previous case. □

5 Validity

In this section, we start by defining the validity of specifications in the SOS presented before. This captures the idea that a specification is valid if and only if every execution starting from a state that respects the precondition and the shared state is not faulty and if it terminates, then the postcondition and the shared state are respected.

Let Γ be a resource context and $D = \{r_{i_1}, \ldots, r_{i_k}\} \subseteq Res(\Gamma)$, we define

$$\underset{r \in D}{\circledast} \Gamma(r) := R_{i_1} * \ldots * R_{i_k}, \quad inv(\Gamma) := \underset{r \in Res(\Gamma)}{\circledast} \Gamma(r).$$

Definition 4. *We write $\Gamma \models \{P\}C\{Q\}$, if for every $(s, h) \in \mathcal{S}$ such that $s, h \models P * inv(\Gamma)$, we have that*

- *$C, (s, h, (\emptyset, \emptyset, res(\Gamma)) \not\rightarrow_p^k \mathsf{abort}$, for every $k \geq 0$. And*
- *If there exist $(s', h') \in \mathcal{S}$ and $k \geq 0$ such that*

$$C, (s, h, (\emptyset, \emptyset, res(\Gamma))) \rightarrow_p^k \mathsf{skip}, (s', h', (\emptyset, \emptyset, res(\Gamma))),$$

then $s', h' \models Q * inv(\Gamma)$.

However we were not able to inductively prove the soundness of CSL using this notion, because we can not emulate the change of parallel execution in all its parts. The rest of this section is devoted to see how we overcome this difficulty. Thus we introduce the environment transition, that will be essential to spread changes made in the state by one program to other parallel programs. And we give a refined notion of validity for the SOS extended with the environment transition, this new notion is called safety. We finish this section by seeing that safety implies validity.

5.1 Environment transition

In order to define the environment transition, we define the environment transformation respecting a set of variables. This transformation modifies the storage and the resource configuration, afterwards the environment transition combines this transformations with modification in the shared state.

Definition 5. *Let* $(s, h, (O, L, D)), (s', h', (O', L', D'))$ *be states and* $A \subseteq \boldsymbol{Var}$*. We say that the* environment transforms $(s, h, (O, L, D))$ *into* $(s', h, (O, L', D'))$ *respecting* A *and we write* $(s, h, (O, L, D)) \overset{A}{\rightsquigarrow} (s', h, (O, L', D'))$ *if* $s(x) = s'(x)$*, for every* $x \in A$*, and* $L' \cup D' = L \cup D$*.*

Note that the environment transformation preserves the local heap and the owned resources, since other programs cannot change them. Furthermore, the environment transformation, $\overset{A}{\rightsquigarrow}$, naturally defines a relation between states. It is easy to see that this relation is an equivalence relation and it is order reversing with respect to A. In the next proposition, we state this properties.

Proposition 9. *Let* $A', A \subseteq \boldsymbol{Var}$*. The relation* $\overset{A}{\rightsquigarrow}$ *is an equivalence relation. If* $A' \subseteq A$ *and* $(s, h, \rho) \overset{A}{\rightsquigarrow} (s', h', \rho')$*, then* $(s, h, \rho) \overset{A'}{\rightsquigarrow} (s', h', \rho')$*.*

We denote the *environment transition* by $\xrightarrow{A,\Gamma}_e$, it is a relation between $(C, (s, h \uplus h_G, \rho))$ and $(C, (s', h \uplus h'_G, \rho'))$, where C is a command and $(s, h \uplus h_G, \rho)$, $(s', h \uplus h'_G, \rho')$ are states, and it is defined by the rule below. Consider the set $A' = A \cup \bigcup_{r \in Locked(C)} PV(r)$. If $(s, h, \rho) \overset{A'}{\rightsquigarrow} (s', h, \rho')$, $s, h_G \models \circledast_{r \in D} \Gamma(r)$ and $s', h'_G \models \circledast_{r \in D'} \Gamma(r)$, then

$$\frac{}{C, (s, h \uplus h_G, \rho) \xrightarrow{A,\Gamma}_e C, (s', h \uplus h'_G, \rho')} \ (E) \ .$$

As noted before the environment transition is used to simulate modification done by parallel program. The environment transition can be used

to: change the storage, except for variables in the rely-set A or variables protected by a locked resources; interchange locked resources and available resource; and modify the available shared heap.

We extend the transitions on the SOS with the environment transition, and we define the relation $\xrightarrow{A,\Gamma}$ from $(C,(s,h,\rho))$ to $(C',(s',h',\rho'))$ or to abort, where C, C' are commands and $(s,h,\rho),(s',h',\rho')$ are states. This relation is given by

$$\xrightarrow{A,\Gamma} = \rightarrow_p \cup \xrightarrow{A,\Gamma}_e .$$

5.2 Safety

For a command C, we associate the set of variables passive to be *changed* by C in the next transition, and we denote it by $chng(C)$. This set consists of all variables x such that C can perform a transition using $\mathsf{x{:=}e}$; $\mathsf{x{:=}[e]}$ or $\mathsf{x{:=}cons(e)}$.

In the next definition of a program's safety with respect to a state for the following n transitions, we include some additional properties that will be useful to prove the soundness of CSL.

Definition 6. *Let $C \in \mathcal{C}$, $(s,h) \in \mathcal{S}$, $\rho \in \mathcal{O}$, Γ be a resource context, Q be an assertion and $A \subseteq \boldsymbol{Var}$. We say that $Safe_0(C,s,h,\rho,\Gamma,Q,A)$ is always valid, and $Safe_{n+1}(C,s,h,\rho,\Gamma,Q,A)$ is valid if:*

(i) *If $C = \mathsf{skip}$, then $s,h \models Q$;*

(ii) *$C,(s,h,\rho) \not\rightarrow_p \mathsf{abort}$;*

(iii) *$chng(C) \cap \bigcup_{r\in L\cup D} PV(r) = \emptyset$;*

(iv) *For every h_G such that $h \perp h_G$, $s,h_G \models \circledast_{r\in D} \Gamma(r)$ and*

$$C,(s,h\uplus h_G,\rho) \xrightarrow{A,\Gamma} C',(s',\hat{h},\rho'),$$

then there exist h' and h'_G such that $\hat{h} = h' \uplus h'_G$,

$$s',h'_G \models \circledast_{r\in D'} \Gamma(r), \quad Safe_n(C',s',h',\rho',\Gamma,Q,A) \text{ is valid.}$$

The property (i) states that if the execution terminates, then Q is respected. In the property (ii), we ensure that the next transition of C does not abort for the state (s,h,ρ). The property (iii) guarantees that the next transition of C does not change variables protected by resources not owned.

In the final condition (iv), we require that the available shared state is preserved after every transition and that the posterior transitions respects this conditions.

In the next theorem, we see that if a program is safe for every number of transitions and for every state that respects the pre-condition, then the corresponding specification is valid with respect to the SOS. The theorem is proved by induction on the number of program's transitions.

Theorem 1. *Let $C \in \mathcal{C}$, let P, Q be assertions, let Γ be a resource context and $A \subseteq \boldsymbol{Var}$. If for every $(s,h) \in \mathcal{S}$ and $n \geq 0$ such that $s, h \models P$, we have that $Safe_n(C, s, h, (\emptyset, \emptyset, Res(\Gamma)), \Gamma, Q, A)$ is valid, then*

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

In order to prove the soundness of CSL, by the theorem above, is sufficient to show that every derivable specification on CSL implies safety, a result we prove in the next section.

6 Soundness

We sketch here the soundness of CSL with respect to the SOS. First, we state the main result of this work, the soundness of CSL. Next we present an intermediate theorem that, together with the Theorem 1, proves the main result. The intermediate theorem says that every derivable specification in CSL is safe in the extended operational semantics.

Theorem 2. *If $\Gamma \vdash_A \{P\}C\{Q\}$, then $\Gamma \models \{P\}C\{Q\}$.*

Theorem 2 is an immediate consequence of the next theorem and Theorem 1.

Theorem 3. *Let C be a command, let P, Q be assertions, let Γ be a resource context and $A \subseteq \boldsymbol{Var}$. If $\Gamma \vdash_A \{P\}C\{Q\}$, then for every $(s,h) \in \mathcal{S}$ and $n \geq 0$ such that $s, h \models P$, we have that $Safe_n(C, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is valid, where $L \cup D = Res(\Gamma)$.*

In the next lines, we present a proof of this theorem by studying the inference rules of CSL. The proof is carried by induction on the inference rules and uses auxiliary results about the inference rules.

Proposition 10. *Let $(s,h) \in \mathcal{S}$, let $\rho \in \mathcal{O}$, let Γ be a resource context, let Q be an assertion and $A \subseteq \boldsymbol{Var}$ such that $FV(Q) \subseteq A$. If $s, h \models Q$, then $Safe_n(\mathsf{skip}, s, h, \rho, \Gamma, Q, A)$ is valid for every $n \geq 0$.*

Proof. Let $(s,h) \in \mathcal{S}$, let $\rho \in \mathcal{O}$, let Γ be a resource context, let Q be an assertion and $A \subseteq \mathbf{Var}$ such that $FV(Q) \subseteq A$ and $s,h \models Q$.

We prove the result by induction on n. For $n = 0$, the result is trivial.

Let $n = k+1$. The first properties of safety are immediate, because $s,h \models Q$, the command skip does not abort and it does not modify protected variables.

For the last property, let h_G, C', s', h' and ρ' be such that $s, h_G \models \circledast_{r \in D} \Gamma(r)$ and

$$\mathsf{skip}, (s, h \uplus h_G, \rho) \xrightarrow{A,\Gamma} C', (s', h', \rho').$$

We note that the only possible transition of skip is an environment transition. Then $C' = \mathsf{skip}$, $s(x) = s'(x)$, for every $x \in A$, and there is h'_G such that $h' = h \uplus h'_G$ and

$$s', h'_G \models \circledast_{r \in D'} \Gamma(r).$$

It is enough to check that $Safe_k(\mathsf{skip}, s', h, \rho', \Gamma, Q, A)$ is valid. But the environment transition does not modify the variables in the rely-set neither the local heap. Therefore $s', h \models Q$, by Proposition 1. And $Safe_k(\mathsf{skip}, s', h, \rho', \Gamma, Q, A)$ is valid, by induction. □

In order to check the safety of basic commands rules (BC), we argue mostly as in the context of SL. Like in SL, we know that if a state respects the precondition, then the execution does not abort and the state reached after the program transition (BCT) respects the post condition.

Proposition 11. *Let c be a basic command, let P, Q be assertions, let Γ be a resource context, let $(s,h) \in \mathcal{S}$, let $\rho \in \mathcal{O}$ and $A \subseteq$ **Var** such that $\vdash^{SL} \{P\}c\{Q\}$ and $FV(P,c,Q) \subset A$. If $s,h \models P$ and $mod(c) \notin PV(\Gamma)$, then $Safe_n(c,s,h,\rho,\Gamma,Q,A)$ is valid, for all $n \geq 0$.*

Proof. Let c be a basic command, let P, Q be assertions, let Γ be a resource context, let $(s,h) \in \mathcal{S}$, let $\rho \in \mathcal{O}$ and $A \subseteq \mathbf{Var}$ such that $\vdash^{SL} \{P\}c\{Q\}$, $FV(P,c,Q) \subset A$, $s,h \models P$ and $mod(c) \notin PV(\Gamma)$.

From $\vdash^{SL} \{P\}c\{Q\}$, we know that $[c](s,h) \neq \mathsf{abort}$ and $[c](s,h) \models Q$.

We prove by induction on n that $Safe_n(c,s,h,\rho,\Gamma,Q,A)$ is valid. For $n = 0$, the result is trivial.

Let $n = k+1$. The first properties of safety are immediate, because $c \neq \mathsf{skip}$, $[c](s,h) \neq \mathsf{abort}$ and $mod(c) \notin PV(\Gamma)$.

For the last property, let h_G, C', s', h' and ρ' be such that $s, h_G \models \circledast_{r \in D} \Gamma(r)$ and

$$c, (s, h \uplus h_G, \rho) \xrightarrow{A,\Gamma} C', (s', h', \rho').$$

We have two possibilities: a transition by the environment or by (BCT).

Suppose that it is an environment transition, then $C' = c$, $s(x) = s'(x)$, for every $x \in A \supseteq FV(P)$, and there is h'_G such that $h' = h \uplus h'_G$ and

$$s', h'_G \models \circledast_{r \in D'} \Gamma(r).$$

Moreover, the precondition is preserved, $s', h \models P$. So we can apply the induction hypothesis to see that $Safe_n(c, s', h, \rho', \Gamma, Q, A)$ is valid.

Suppose that it is a transition by (BCT), then $C' = \mathsf{skip}$, $(s', h') = [c](s, h \uplus h_G)$ and $\rho' = \rho$. Using the frame property, Proposition 4, we know that $h' = h'' \uplus h_G$, where $[c](s, h) = (s', h'') \models Q$.

Therefore, by Proposition 10, we have that $Safe_n(\mathsf{skip}, s', h'', \rho, \Gamma, Q, A)$ is valid. This concludes the proof. □

The soundness of the rules (SEQ), (LP), $(CONJ)$, (IF) and $(CONS)$ are a direct consequence of the inductive process.

The soundness of the frame rule (FRA) is supported by the following proposition. It follows from the safety monotonicity and frame property (Propositions 4 and 5). We note that R is valid after every transition, because $FV(R)$ is not modified by the command and the rely-set includes it.

Proposition 12. *Let C be a reachable command, let Γ be a resource context, let $(s, h \uplus h_R) \in \mathcal{S}$, let $\rho \in \mathcal{O}$, let Q, R be assertions and $A \subseteq \boldsymbol{Var}$ such that $s, h_R \models R$. If $Safe_n(C, s, h, \rho, \Gamma, Q, A)$ is valid and $mod(C) \cap FV(R) = \emptyset$, then $Safe_n(C, s, h \uplus h_R, \rho, \Gamma, Q * R, A \cup FV(R))$ is valid.*

Proof. Let C a reachable command, Γ a resource context, $(s, h \uplus h_R) \in \mathcal{S}$, $\rho \in \mathcal{O}$, Q, R assertions and $A \subseteq \mathbf{Var}$ such that $s, h_R \models R$, $Safe_n(C, s, h, \rho, \Gamma, Q, A)$ is valid and $mod(C) \cap FV(R) = \emptyset$.

We just show the inductive step of the proof. Let $n = k + 1$.

If $C = \mathsf{skip}$, then $s, h \models Q$ and $s, h \uplus h_R \models Q * R$.

By $Safe_n(C, s, h, \rho, \Gamma, Q, A)$ and Proposition 4, we have that

$$C, (s, h \uplus h_R, \rho) \not\to \mathsf{abort},$$

and

$$chng(C) \cap \bigcup_{r \in L \cup D} PV(r) = \emptyset.$$

Therefore, the first properties of $Safe_n(C, s, h \uplus h_R, \rho, \Gamma, Q * R, A \cup FV(R))$ are established.

In order to check the last property. Let h_G, C', s', h' and ρ' such that $h_G \perp (h \uplus h_R)$, $s, h_G \models \circledast_{r \in D} \Gamma(r)$ and

$$C, (s, h \uplus h_R \uplus h_G, \rho) \xrightarrow{A \cup FV(R), \Gamma} C', (s', h', \rho').$$

This transition can be a program transition or a environment transition.
Suppose that it is a program transition.
By frame property, Proposition 4, there is h'' such that $h' = h'' \uplus h_R$ and

$$C, (s, h \uplus h_G, \rho) \rightarrow_p C', (s', h'', \rho').$$

From the validity of $Safe_n(C, s, h, \rho, \Gamma, Q, A)$ and the transition above, we know that there exists $h'_G \subseteq h''$, such that $Safe_k(C', s', h'' \setminus h'_G, \rho', \Gamma, Q, A)$ is valid and

$$s', h'_G \models \circledast_{r \in D'} \Gamma(r).$$

From $mod(C) \cap FV(R) = \emptyset$, we have that $mod(C') \cap FV(R) = \emptyset$ and

$$s', h_R \models R.$$

By induction, $Safe_k(C', s', h' \setminus h'_G, \rho', \Gamma, Q * R, A \cup FV(R))$ is valid.
Suppose that it occurs an environment transition.
There exists $h'_G \subseteq h'$ such that $h' = h \uplus h_R \uplus h'_G$, $s', h'_G \models \circledast_{r \in D'} \Gamma(r)$, and

$$(s, h \uplus h_R, \rho) \overset{A'}{\rightsquigarrow} (s', h \uplus h_R, \rho'),$$

where $A' = A \cup FV(R) \cup \bigcup_{r \in Locked(C)} PV(r)$. And $s', h_R \models R$, because $FV(R) \subseteq A'$.
Let $A'' = A \cup \bigcup_{r \in Locked(C)} PV(r) \subset A'$. Then $(s, h, \rho) \overset{A''}{\rightsquigarrow} (s', h, \rho')$, and

$$C, (s, h \uplus h_G, \rho) \xrightarrow{A, \Gamma}_e C, (s', h \uplus h_G, \rho').$$

From the validity of $Safe_n(C, s, h, \rho, \Gamma, Q, A)$ and the transition above, we have that $Safe_k(C, s', h, \rho', \Gamma, Q, A)$ is valid.
By induction, we conclude that $Safe_k(C, s', h \uplus h_R, \rho', \Gamma, Q * R, A \cup FV(R))$ is valid. □

Next we study the parallel rule (PAR).

Proposition 13. *Let $C_1 \parallel C_2$ be a reachable command, let Q_1, Q_2 be assertions, let $(s, h), (s, h_1), (s, h_2) \in \mathcal{S}$, let $\rho, \rho_1, \rho_2 \in \mathcal{O}$ and $A_1, A_2 \subseteq \boldsymbol{Var}$. Suppose that $h = h_1 \uplus h_2$, $\rho = (O_1 \cup O_2, L, D)$, $\rho_1 = (O_1, L \cup O_2, D)$, $\rho_2 = (O_2, L \cup O_1, D)$, $FV(Q_i) \subseteq A_i$, for $i = 1, 2$, and $A_1 \cap mod(C_2) = A_2 \cap mod(C_1) = \emptyset$.*

If $Safe_n(C_1, s, h_1, \rho_1, \Gamma, Q_1, A_1)$ and $Safe_n(C_2, s, h_2, \rho_2, \Gamma, Q_2, A_2)$ are valid , then $Safe_n(C_1 \parallel C_2, s, h, \rho, \Gamma, Q_1 * Q_2, A_1 \cup A_2)$ is valid.

As before we prove this result by induction on n. The firsts three properties of safety are immediate from the safety of C_1 and C_2, and Propositions 4 and 6.

In order to apply the induction step we use the environment transition. If the parallel execution transits by a program transition, then there are three cases. First case, a transition is done by C_1. We perform the same transition on C_1 (by Propositions 5 and 7) and an environment transition on C_2, that replicates the changes performed by the program transition. This environment transition exists because the variables modified by the program C_1 are different from the rely-set A_2. In the second case, a transition is done by C_2, and we do analogous transitions. The third case is the joint of parallel commands. In this case, we do a reflexive environment transition on C_1 and C_2. If the program transits by an environment transition, then we perform the same environment transition on C_1 and C_2. This environment transition can be used because the rely-set of $C_1 \parallel C_2$ includes the rely-set of each command. Therefore we can apply the inductive hypothesis and obtain the proposition.

Proof. Let $C_1 \parallel C_2$ be a reachable command, let $(s,h)(s,h_1),(s,h_2) \in \mathcal{S}$, let $\rho, \rho_1, \rho_2 \in \mathcal{O}$, let Q_1, Q_2 be assertions and $A_1, A_2 \subseteq \mathbf{Var}$ such that $h = h_1 \uplus h_2$, $\rho = (O_1 \cup O_2, L, D)$, $\rho_1 = (O_1, L \cup O_2, D)$, $\rho_2 = (O_2, L \cup O_1, D)$, $FV(Q_1) \subseteq A_1$, $FV(Q_2) \subseteq A_2$ and $A_1 \cap mod(C_2) = A_2 \cap mod(C_1) = \emptyset$.

We just prove the induction step for $n = k + 1$.

Suppose that the hypothesis is valid, i.e. $Safe_{k+1}(C_1, s, h_1, \rho_1, \Gamma, Q_1, A_1)$ and $Safe_{k+1}(C_2, s, h_2, \rho_2, \Gamma, Q_2, A_2)$ are valid.

The first property of $Safe_n(C_1 \parallel C_2, s, h, \rho, \Gamma, Q_1 * Q_2, A_1 \cup A_2)$ is trivial, because $C_1 \parallel C_2 \neq \mathsf{skip}$.

Applying the safety monotonicity (Proposition 4) to the hypothesis, we see that

$$C_1, (s, h_1 \uplus h_2, \rho_1) \not\rightarrow_p \mathsf{abort} \quad C_2, (s, h_1 \uplus h_2, \rho_2) \not\rightarrow_p \mathsf{abort}.$$

Using Proposition 6, we obtain that

$$C_1, (s, h, \rho) \not\rightarrow_p \mathsf{abort} \quad C_2, (s, h, \rho) \not\rightarrow \mathsf{abort}.$$

Hence, we respect the second property $C_1 \parallel C_2, (s, h, \rho) \not\rightarrow_p \mathsf{abort}$.

Using the hypothesis, we derive that

$$\begin{aligned}
chng(C) \cap \bigcup_{\substack{r\in \\ L\cup D}} PV(r) =& (chng(C_1) \cup chng(C_2)) \cap \bigcup_{\substack{r\in \\ L\cup D}} PV(r) \\
=& (chng(C_1) \cap \bigcup_{\substack{r\in \\ L\cup D}} PV(r)) \cup (chng(C_2) \cap \bigcup_{\substack{r\in \\ L\cup D}} PV(r)) \\
\subseteq& (chng(C_1) \cap \bigcup_{\substack{r\in L\cup \\ O_2\cup D}} PV(r)) \cup (chng(C_2) \cap \bigcup_{\substack{r\in L\cup \\ O_1\cup D}} PV(r)) \\
\subseteq& \emptyset.
\end{aligned}$$

And the third condition of $Safe_n(C_1 \parallel C_2, s, h, \rho, \Gamma, Q_1 * Q_2, A_1 \cup A_2)$ follows.

Now, we check the fourth condition. Let h_G, C', s', h' and ρ' such that $s, h_G \models \circledast_{r\in D} \Gamma(r)$ and

$$C_1 \parallel C_2, (s, h_1 \uplus h_2 \uplus h_G, \rho) \xrightarrow{A_1\cup A_2,\Gamma} C', (s', h', \rho').$$

There are four possible transitions.

Suppose that the transition is given by $(P1)$. We have that $C' = C_1' \parallel C_2$ and

$$C_1, (s, h_1 \uplus h_2 \uplus h_G, \rho) \rightarrow_p C_1', (s', h', \rho').$$

The validity of $Safe_{k+1}(C_1, s, h_1, \rho_1, \Gamma, Q_1, A_1)$ implies that the command C_1 does not abort. Using the frame property (Proposition 5) and Proposition 7 we have that $O_2 \subseteq O'$, $h_2 \subseteq h'$ and

$$C_1, (s, h_1 \uplus h_G, \rho_1) \rightarrow_p C_1', (s', h' \setminus h_2, (O' \setminus O_2, L \cup O_2, D')).$$

Define $\rho_1' := (O' \setminus O_2, L \cup O_2, D')$. From the hypothesis, we know that there are h_1' and h_G' such that $h' \setminus h_2 = h_1' \uplus h_G'$, $Safe_k(C_1', s', h_1', \rho_1', \Gamma, Q_1, A_1)$ is valid and

$$s', h_G' \models \circledast_{r\in D'} \Gamma(r).$$

In order to apply the induction hypothesis and conclude the validity of $Safe_k(C_1' \parallel C_2, s', h_1' \uplus h_2, \rho', \Gamma, Q_1 * Q_2, A_1 \cup A_2)$, it remains to check that $Safe_k(C_2, s', h_2, \rho_2', \Gamma, Q_2, A_2)$ is valid, where $\rho_2' = (O_2, L \cup (O' \setminus O_2), D')$.

We know that $s(x) = s'(x)$, for every $x \notin chng(C_1)$.

Note that $chng(C_1) \subset mod(C_1)$ and $Locked(C_2) \subset O_2$, otherwise C_2 will abort by $(WA2)$. Let $A_2' = A_2 \cup \bigcup_{r\in Locked(C_2)} PV(r)$, using the hypothesis we have that

$$chng(C_1) \cap A'_2 \subseteq (mod(C_1) \cap A_2) \cup (chng(C_1) \cap \bigcup_{r \in O_2} PV(r)) \subseteq \emptyset.$$

Hence $s(x) = s'(x)$, for every $x \in A'_2$.

From Proposition 2, we know that $O_1 \cup D = (O' \setminus O_2) \cup D'$ and

$$L \cup O_1 \cup D = L \cup (O' \setminus O_2) \cup D'.$$

We have the environment transformation and environment transition

$$(s, h_2, \rho_2) \overset{A'_2}{\rightsquigarrow} (s', h_2, \rho'_2),$$

$$C_2, (s, h_2 \uplus h_G, \rho_2) \xrightarrow{A_2, \Gamma}_e C_2, (s', h_2 \uplus h'_G, \rho'_2),$$

By hypothesis we conclude that $Safe_k(C_2, s', h_2, \rho'_2, \Gamma, Q_2, A_2)$ is valid.

Therefore $Safe_k(C'_1 \parallel C_2, s', h'_1 \uplus h_2, \rho', \Gamma, Q_1 * Q_2, A_1 \cup A_2)$ is valid.

The transition $(P2)$ is analogous to the transition $(P1)$.

Suppose that the transition is given by $(P3)$. We have $C' = \mathsf{skip}$, $C_1 = \mathsf{skip}$, $C_2 = \mathsf{skip}$, $s' = s$, $h' = h \uplus h_G$ and $\rho' = \rho$.

Taking $h'_G = h_G$. We know that

$$s, h_G \models \circledast_{r \in D} \Gamma(r).$$

Because $Safe_n(\mathsf{skip}, s, h_1, \rho_1, \Gamma, Q_1, A_1)$ and $Safe_n(\mathsf{skip}, s, h_2, \rho_2 \Gamma, Q_2, A_2)$ are valid, we have that

$$s, h_1 \uplus h_2 \models Q_1 * Q_2.$$

From Proposition 10, $Safe_k(\mathsf{skip}, s, h_1 \uplus h_2, \rho, \Gamma, Q_1 * Q_2, A_1 \cup A_2)$ is valid.

Suppose that the transition is given by (E). Let

$$A' = A_1 \cup A_2 \cup \bigcup_{r \in Locked(C_1 \parallel C_2)} PV(r).$$

We have that $C' = C_1 \parallel C_2$, $\rho' = (O_1 \cup O_2, L', D')$, $(s, h, \rho) \overset{A'}{\rightsquigarrow} (s', h, \rho')$, and there exists h'_G such that $h' = h_1 \uplus h_2 \uplus h'_G$ and

$$s', h'_G \models \circledast_{r \in D'} \Gamma(r).$$

We know that $s(x) = s'(x)$, for every $x \in A'$ and $L \cup D = L' \cup O_2 \cup D'$. Then we have the following environment transformation

$$(s, h_1, \rho_1) \overset{A'_1}{\rightsquigarrow} (s', h_1, \rho'_1),$$

where $\rho_1' = (O_1, L' \cup O_2, D')$ and $A_1' = A_1 \cup \bigcup_{r \in Locked(C_1)} PV(r) \subseteq A'$.

By the hypothesis, we conclude that $Safe_k(C_1, s', h_1, \rho_1', \Gamma, Q_1, A_1)$ is valid.

Analogous, we obtain that $Safe_k(C_2, s', h_2, \rho_2', \Gamma, Q_2, A_2)$ is valid, where $\rho_2' = (O_2, L' \cup O_1, D')$.

Therefore $Safe_k(C_1 \parallel C_2, s', h_1 \uplus h_2, \rho', \Gamma, Q_1 * Q_2, A_1 \cup A_2)$ is valid, by the induction hypothesis. □

The safety of the critical region rule follows from the safety inside the critical region. Because any environment transition performed before the critical region does not break the precondition's validity and when a program enters a critical region its invariant is valid. Therefore the next result establishes safety for the critical region rule.

Proposition 14. *Let C be a reachable command, let $(s, h) \in \mathcal{S}$, let Γ be a resource context, let $\rho = (O, L, D) \in \mathcal{O}$, let Q, R be assertions and $A \subseteq$ **Var**. Suppose that $r \in O$, $\Gamma' = \Gamma, r(X) : R$ is a resource context and $FV(Q) \subseteq A$.*

If $Safe_n(C, s, h, \rho \setminus \{r\}, \Gamma, Q * R, A \cup X)$ is valid, then

$$Safe_n(\textsf{within}\ r\ \textsf{do}\ C, s, h, \rho, \Gamma', Q, A) \text{ is valid.}$$

Proof. Let C be a reachable command, let $(s, h) \in \mathcal{S}$, let Γ be a resource context, let $\rho = (O, L, D) \in \mathcal{O}$, let Q, R be assertions and $A \subseteq$ **Var** such that $r \in O$, $\Gamma' = \Gamma, r(X) : R$ is a resource context, $Safe_n(C, s, h, \rho \setminus \{r\}, \Gamma, Q * R, A \cup X)$ is valid and $FV(Q) \subseteq A$.

We prove by induction on n that $Safe_n(\textsf{within}\ r\ \textsf{do}\ C, s, h, \rho, \Gamma', Q, A)$ is valid. For $n = 0$, it is trivial.

Let $n = k + 1$. The first property is immediate, because $\textsf{within}\ r\ \textsf{do}\ C \neq \textsf{skip}$.

From $Safe_n(C, s, h, \rho \setminus \{r\}, \Gamma, Q * R, A \cup X)$ be valid, we know that

$$C, (s, h, \rho \setminus \{r\}) \not\to_p \textsf{abort}.$$

Together with $r \in O$, we have the second property

$$\textsf{within}\ r\ \textsf{do}\ C, (s, h, \rho) \not\to_p \textsf{abort}.$$

From $Safe_n(C, s, h, \rho \setminus \{r\}, \Gamma, Q * R, A \cup X)$ be valid and $r \in O$, we know that $\textsf{within}\ r\ \textsf{do}\ C$ does not change variable protected by resource in $L \cup D$. Then, the third condition is respected.

Let h_G, C', s', h' and ρ' such that $s, h_G \models \circledast_{\hat{r} \in D} \Gamma'(\hat{r})$ and

$$\textsf{within}\ r\ \textsf{do}\ C, (s, h \uplus h_G, \rho) \xrightarrow{A, \Gamma'} C', (s', h', \rho').$$

There are three possible transition: $(W1)$, $(W2)$ or (E).

Suppose that it is $(W1)$. We have $C' = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}$, $r \in O \cap O'$ and

$$C, (s, h \uplus h_G, \rho \setminus \{r\}) \rightarrow_p \tilde{C}, (s', h', \rho' \setminus \{r\}).$$

From $Safe_n(C, s, h, \rho \setminus \{r\}, \Gamma, Q * R, A \cup X)$, we know that there are h'_G, h'_L such that $h' = h'_L \uplus h'_G$, $Safe_k(\tilde{C}, s', h'_L, \rho' \setminus \{r\}, \Gamma, Q * R, A \cup X)$ is valid and

$$s', h'_G \models \circledast_{\hat{r} \in D'} \Gamma(\hat{r}).$$

From $r \in O'$, we conclude that $s', h'_G \models \circledast_{\hat{r} \in D'} \Gamma'(\hat{r})$.

From $Safe_k(\tilde{C}, s', h'_L, \rho' \setminus \{r\}, \Gamma, Q * R, A \cup X)$ and the induction hypothesis, we obtain that $Safe_k(\mathsf{within}\ r\ \mathsf{do}\ \tilde{C}, s', h'_L, \rho', \Gamma', Q, A)$ is valid.

Suppose that the transition is given by $(W2)$.

We have that $C' = \mathsf{skip}$, $C = \mathsf{skip}$, $s = s'$, $h' = h \uplus h_G$, $O' = O \setminus \{r\}$, $L' = L$ and $D' = D \cup \{r\}$. From $Safe_n(\mathsf{skip}, s, h, \rho, \Gamma, Q * R, A \cup X)$, we have that

$$s, h \models Q * R.$$

Then there exists $h_R \subseteq h$ such that $s, h_R \models R$ and $s, h \setminus h_R \models Q$. Then

$$s, h_G \uplus h_R \models \circledast_{\hat{r} \in D'} \Gamma'(\hat{r}).$$

By Proposition 10, $FV(Q) \subseteq A$ and $s, h \setminus h_R \models Q$, we conclude that $Safe_k(\mathsf{skip}, s, h \setminus h_R, \rho', \Gamma', Q, A)$ is valid.

Suppose that the transition is given by (E). Let

$$A' = A \cup PV(r) \cup \bigcup_{\hat{r} \in Locked(C)} PV(\hat{r}).$$

We have $C' = \mathsf{within}\ r\ \mathsf{do}\ C$, $(s, h, \rho) \overset{A'}{\rightsquigarrow} (s', h, \rho')$ and there exists $h'_G \subseteq h'$ such that $h' = h \uplus h'_G$ and

$$s', h'_G \models \circledast_{\hat{r} \in D'} \Gamma'(\hat{r}).$$

From $r \in O \cap O'$, we get that $s', h'_G \models \circledast_{\hat{r} \in D'} \Gamma(\hat{r})$ and $s, h_G \models \circledast_{\hat{r} \in D} \Gamma(\hat{r})$.

Note that $A' = A \cup X \cup \bigcup_{\hat{r} \in Locked(C)} PV(\hat{r})$ and the environment transition

$$C, (s, h \uplus h_G, \rho \setminus \{r\}) \xrightarrow{A \cup X, \Gamma}_e C, (s', h \uplus h'_G, \rho' \setminus \{r\}).$$

From the validity of $Safe_n(C, s, h, \rho \setminus \{r\}, \Gamma, Q * R, A \cup X)$, we have that $Safe_k(C, s', h, \rho' \setminus \{r\}, \Gamma, Q * R, A \cup X)$ is valid.

Therefore, by induction, $Safe_k(\mathsf{within}\ r\ \mathsf{do}\ C, s', h, \rho', \Gamma', Q, A)$ is valid. □

In the proposition below, we give properties for the local resource when the resource is available or locked, similar to [16, Lemma 4.3] in DCSL. The soundness of the local resource rule follows from the second property of the proposition.

Proposition 15. *Let C be a reachable command, let $(s,h) \in \mathcal{S}$, let Γ be a resource context, $\rho = (O, L, D) \in \mathcal{O}$, let Q, R be assertions and $A, X \subseteq \boldsymbol{Var}$. Suppose that $r \notin \rho$, $\Gamma' = \Gamma, r(X) : R$ is a resource context and $FV(Q) \subseteq A$. We have the following statements:*

- *Suppose that $r \in Locked(C)$. If $Safe_n(C, s, h, (O \cup \{r\}, L, D), \Gamma', Q, A)$ is valid, then $Safe_n(\mathsf{resource}\ r\ \mathsf{in}\ C, s, h, \rho, \Gamma, Q * R, A \cup X)$ is valid.*
- *Suppose that $r \notin Locked(C)$ and that there exists h_R such that $h_R \perp h$ and $s, h_R \models R$. If $Safe_n(C, s, h, (O, L, D \cup \{r\}), \Gamma', Q, A)$ is valid, then $Safe_n(\mathsf{resource}\ r\ \mathsf{in}\ C, s, h \uplus h_R, \rho, \Gamma, Q * R, A \cup X)$ is valid.*

This proposition is proved by induction on both properties in the following way: first we prove that both properties are true when $n = 0$; then we assume that both properties are true for $n \geq 0$ and prove that each property is true for $n + 1$.

The program transitions inside the local resource have an equivalent program transition for the command C, except for the transition $(R0)$. In those cases we apply one of the inductive step depending on resource's ownership. For the case $(R0)$, we note that the execution inside the local resource had terminated and the invariant R is respected. If the local resource transits by an environment transition, then there is an equivalent environment transition in C.

Proof. Let C be a reachable command, let $(s,h) \in \mathcal{S}$, let Γ be a resource context, $\rho = (O, L, D) \in \mathcal{O}$, let Q, R be assertions and $A, X \subseteq \mathbf{Var}$ such that $r \notin \rho$, $\Gamma' = \Gamma, r(X) : R$ is a resource context and $FV(Q) \subseteq A$.

Consider the next statements:

$P(n)$ If $r \in Locked(C)$ and $Safe_n(C, s, h, (O \cup \{r\}, L, D), \Gamma', Q, A)$ is valid, then $Safe_n(\mathsf{resource}\ r\ \mathsf{in}\ C, s, h, \rho, \Gamma, Q * R, A \cup X)$ is valid.

$Q(n)$ If $r \notin Locked(C)$, $Safe_n(C, s, h, (O, L, D \cup \{r\}), \Gamma', Q, A)$ is valid and there exist $h_R \perp h$ such that $s, h_R \models R$, then $Safe_n(\mathsf{resource}\ r\ \mathsf{in}\ C, s, h \uplus h_R, \rho, \Gamma, Q * R, A \cup X)$ is valid.

We prove the result in three steps. First, we note that $P(0)$ and $Q(0)$ are true. Next, we see that $P(n) \wedge Q(n) \Rightarrow Q(n+1)$, for every $n \geq 0$. Last, we show that $P(n) \wedge Q(n) \Rightarrow P(n+1)$, for every $n \geq 0$.

Next, we suppose that $P(n) \wedge Q(n)$ is valid and show that $Q(n+1)$ is valid.

Suppose that $Safe_{n+1}(C, s, h, (O, L, D \cup \{r\}), \Gamma', Q, A)$ is valid, $s, h_R \models R$, $h_R \perp h$ and $r \notin Locked(C)$.

The first property of $Safe_{n+1}(\mathsf{resource}\ r\ \mathsf{in}\ C, s, h \uplus h_R, \rho, \Gamma, Q * R, A \cup X)$ is immediate, because $\mathsf{resource}\ r\ \mathsf{in}\ C \neq \mathsf{skip}$.

From $Safe_{n+1}(C, s, h, (O, L, D \cup \{r\}), \Gamma', Q, A)$ and Proposition 4, we have

$$C, (s, h \uplus h_R, (O, L, D \cup \{r\})) \not\to_p \mathsf{abort}.$$

Note that $r \notin \rho \cup Locked(C)$. Hence $\mathsf{resource}\ r\ \mathsf{in}\ C, (s, h \uplus h_R, \rho) \not\to_p \mathsf{abort}$. And the second property is valid.

From $Safe_{n+1}(C, s, h, (O, L, D \cup \{r\}), \Gamma', Q, A)$, we know that

$$chng(\mathsf{resource}\ r\ \mathsf{in}\ C) \cap \bigcup_{\hat{r} \in L \cup D} PV(\hat{r}) \subseteq chng(C) \cap \bigcup_{\hat{r} \in L \cup D \cup \{r\}} PV(\hat{r}) = \emptyset.$$

Hence, the third property is respected.

Let h_G, C', s', h' and ρ' such that $s, h_G \models \circledast_{\hat{r} \in D} \Gamma(\hat{r})$ and

$$\mathsf{resource}\ r\ \mathsf{in}\ C, (s, h \uplus h_R \uplus h_G, \rho) \xrightarrow{A \cup X, \Gamma} C', (s', h', \rho').$$

Next, we study the possible transitions: $(R0)$, $(R2)$ or (E).

Suppose that the transition is given by $(R0)$. We have that $C = C' = \mathsf{skip}$, $s' = s$, $h' = h \uplus h_R \uplus h_G$ and $\rho' = \rho$. Consider $h'_G = h_G$. Then $h'_G \subseteq h'$ and

$$s, h'_G \models \circledast_{\hat{r} \in D} \Gamma(\hat{r}).$$

From $Safe_{n+1}(\mathsf{skip}, s, h, (O, L, D \cup \{r\}), \Gamma', Q, A)$, we have that $s, h \models Q$. And $s, h \uplus h_R \models Q * R$.

Hence $Safe_n(\mathsf{skip}, s, h \uplus h_R, \Gamma, Q * R, A \cup X)$ is valid, by Proposition 10.

Suppose that it is $(R2)$. We have that $C' = \mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}$ and

$$C, (s, h \uplus h_R \uplus h_G, (O, L, D \cup \{r\})) \to_p \tilde{C}, (s', h', \rho''),$$

where $\rho' = \rho'' \setminus \{r\}$. Note that $s, h_R \uplus h_G \models \circledast_{\hat{r} \in D \cup \{r\}} \Gamma'(\hat{r})$.

From $Safe_{n+1}(C, s, h, (O, L, D \cup \{r\}), \Gamma', Q, A)$ and the transition above, we know that there is h'_G such that $h'_G \subseteq h'$, $Safe_n(\tilde{C}, s', h' \setminus h'_G, \rho'', \Gamma', Q, A)$ is valid and

$$s', h'_G \models \circledast_{\hat{r} \in D''} \Gamma'(\hat{r}).$$

In order to prove that $Safe_n(\mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}, s', h' \setminus h'_G, \rho', \Gamma, Q * R, A \cup X)$ is valid, we need to apply the hypothesis $P(n)$ or $Q(n)$, respectively, if $r \in O''$ or $r \in D''$. Note that $r \notin L''$, by Proposition 2.

We observe that $r \in O''$ if and only if the resource r was acquired in the transition above.

If $r \in O''$, then $r \in Locked(\tilde{C})$. From $Safe_n(\tilde{C}, s', h' \setminus h'_G, \rho'', \Gamma', Q, A)$ and $P(n)$, we have $Safe_n(\mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}, s', h' \setminus h'_G, \rho', \Gamma, Q * R, A \cup X)$ is valid.

If $r \in D''$, then $r \notin Locked(\tilde{C})$. We remark that

$$s', h'_G \models \circledast_{\hat{r} \in D''} \Gamma'(\hat{r}) = R * \left(\circledast_{\hat{r} \in D'} \Gamma(\hat{r}) \right).$$

Then there exists $h'_R \subseteq h'_G$ such that $s', h'_R \models R$. Therefore, by $Q(n)$, $Safe_n(\mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}, s', h' \setminus h'_G \uplus h'_R, \rho', \Gamma, Q * R, A \cup X)$ is valid.

Suppose that the transition is given by (E). Let

$$A' = A \cup X \cup \bigcup_{\hat{r} \in Locked(\mathsf{resource}\ r\ \mathsf{in}\ C)} PV(\hat{r}).$$

We have $C' = \mathsf{resource}\ r\ \mathsf{in}\ C$, $(s, h \uplus h_R, \rho) \overset{A'}{\rightsquigarrow} (s', h \uplus h_R, \rho')$ and there exists $h'_G \subseteq h'$ such that $h' = h \uplus h_R \uplus h'_G$ and

$$s', h'_G \models \circledast_{\hat{r} \in D'} \Gamma(\hat{r}).$$

Let $A'' = A \cup \bigcup_{\hat{r} \in Locked(C)} PV(\hat{r})$. From $r \notin Locked(C)$, we have that $Locked(C) = Locked(\mathsf{resource}\ r\ \mathsf{in}\ C)$ and $A'' \subseteq A'$. Therefore

$$(s, h, (O, L, D \cup \{r\})) \overset{A''}{\rightsquigarrow} (s', h, (O', L', D' \cup \{r\})).$$

From $FV(R) \subseteq X \subseteq A'$ and Proposition 1, we have that $s', h_R \models R$. Moreover, we know that

$$s, h_G \uplus h_R \models \circledast_{\hat{r} \in D \cup \{r\}} \Gamma'(\hat{r}), \quad s', h'_G \uplus h_R \models \circledast_{\hat{r} \in D' \cup \{r\}} \Gamma'(\hat{r}).$$

Then, we have the following environment transition

$$C, (s, h \uplus h_R \uplus h_G, (O, L, D \cup \{r\})) \xrightarrow{A, \Gamma'}_e C, (s', h \uplus h_R \uplus h'_G, (O', L', D' \cup \{r\})).$$

From $Safe_{n+1}(C, s, h, (O, L, D \cup \{r\}), \Gamma', Q, A)$ and the environment transition above, it follows that $Safe_n(C, s', h, (O', L', D' \cup \{r\}), \Gamma', Q, A)$ is valid.

By $Q(n)$, we have $Safe_n(\mathsf{resource}\ r\ \mathsf{in}\ C, s', h \uplus h_R, \rho', \Gamma, Q * R, A \cup X)$.

To finish, we prove that $P(n) \wedge Q(n)$ implies $P(n+1)$.

Suppose that $r \in Locked(C)$ and $Safe_{n+1}(C, s, h, (O \cup \{r\}, L, D), \Gamma', Q, A)$ is valid. Analogous to the previous case, we prove the first three properties of $Safe_{n+1}(\mathsf{resource}\ r\ \mathsf{in}\ C, s, h, (O, L, D), \Gamma, Q * R, A \cup X)$.

Let h_G, C', s', h' and ρ' such that $s, h_G \models \circledast_{\hat{r} \in D} \Gamma(\hat{r})$ and

$$\mathsf{resource}\ r\ \mathsf{in}\ C, (s, h \uplus h_G, \rho) \xrightarrow{A \cup X, \Gamma} C', (s', h', \rho').$$

There are two possible transitions: $(R1)$ or (E).

Suppose that the transition is $(R1)$. We have $C' = \mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}$ and

$$C, (s, h \uplus h_G, (O \cup \{r\}, L, D)) \rightarrow_p \tilde{C}, (s', h', \rho''),$$

where $\rho' = \rho'' \setminus \{r\}$. Note that $\circledast_{\hat{r} \in D} \Gamma(\hat{r}) = \circledast_{\hat{r} \in D} \Gamma'(\hat{r})$.

From the validity of $Safe_{n+1}(C, s, h, (O \cup \{r\}, L, D), \Gamma', Q, A)$ and the transition above, there is h'_G such that $h'_G \subseteq h'$, $Safe_n(\tilde{C}, s', h' \backslash h'_G, \rho'', \Gamma', Q, A)$ is valid and

$$s', h'_G \models \circledast_{\hat{r} \in D''} \Gamma'(\hat{r}).$$

As before, we study two cases: $r \in O''$ or $r \in D''$. In this case, we observe that $r \in D''$ if and only if the resource r was released in the transition above.

If $r \in O''$, then $r \in Locked(\tilde{C})$ and $s', h'_G \models \circledast_{\hat{r} \in D'} \Gamma(\hat{r})$.

By $P(n)$, we obtain $Safe_n(\mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}, s', h' \setminus h'_G, \rho', \Gamma, Q * R, A \cup X)$.

If $r \in D''$, then $r \notin Locked(\tilde{C})$ and $\circledast_{\hat{r} \in D''} \Gamma'(\hat{r}) = R * \circledast_{\hat{r} \in D'} \Gamma(\hat{r})$. Moreover there exists $h'_R \subseteq h'_G$ such that $s', h'_R \models R$.

By $Q(n)$, we have $Safe_n(\mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}, s', h' \backslash h'_G \uplus h'_R, \rho', \Gamma, Q * R, A \cup X)$.

Suppose that the transition is given by (E). Let

$$A' = A \cup X \cup \bigcup_{\hat{r} \in Locked(\mathsf{resource}\ r\ \mathsf{in}\ C)} PV(\hat{r}).$$

We have that $C' = \mathsf{resource}\ r\ \mathsf{in}\ C$, $(s, h \uplus h_R, \rho) \overset{A'}{\rightsquigarrow} (s', h \uplus h_R, \rho')$ and there exists $h'_G \subseteq h'$ such that $h' = h \uplus h_R \uplus h'_G$ and

$$s', h'_G \models \circledast_{\hat{r} \in D'} \Gamma(\hat{r}).$$

From $Locked(\mathsf{resource}\ r\ \mathsf{in}\ C) \cup \{r\} = Locked(C)$ and $PV(r) = X$,

$$A' = A \cup \bigcup_{\hat{r} \in Locked(C)} PV(\hat{r}).$$

We have the environment transformation

$$(s, h, (O \cup \{r\}, L, D)) \overset{A'}{\rightsquigarrow} (s', h, (O' \cup \{r\}, L', D')).$$

And the environment transition

$$C, (s, h \uplus h_G, (O \cup \{r\}, L, D)) \xrightarrow{A,\Gamma'}_e C, (s', h \uplus h'_G, (O' \cup \{r\}, L', D')).$$

From $Safe_{n+1}(C, s, h, (O \cup \{r\}, L, D), \Gamma', Q, A)$ and the environment transition above, it follows that $Safe_n(C, s', h, (O' \cup \{r\}, L', D'), \Gamma', Q, A)$ is valid.

By $P(n)$, $Safe_n(\mathsf{resource}\ r\ \mathsf{in}\ C, s', h, \rho', \Gamma, Q * R, A \cup X)$ is valid. □

The soundness of the renaming rule follows from the next proposition which is a consequence of Proposition 8.

Proposition 16. *Let C be a reachable command, $(s, h, (O, L, D))$ be a state, $A \subseteq$ **Var**, Γ a well-formed resource context and r, r' resource names such that $r' \notin Res(C)$, $r' \notin Res(\Gamma)$ and $O \cup L \cup D = Res(\Gamma)$.*

If $Safe_n(C[r'/r], s, h, \rho[r'/r], \Gamma[r'/r], Q, A)$ is valid, then

$$Safe_n(C, s, h, \rho, \Gamma, Q, A) \text{ is valid.}$$

Proof. We prove the proposition by induction on n. Let C be a reachable command, $(s, h, (O, L, D)$ be a state, $A \subseteq$ **Var**, Γ a well-formed resource context and $r, r' \in$ **Res** such that $r' \notin Res(C)$, $r' \notin Res(\Gamma)$, $O \cup L \cup D = Res(\Gamma)$ and $Safe_n(C[r'/r], s, h, \rho[r'/r], \Gamma[r'/r], Q, A)$ is valid.

First, note that $r \notin \rho$, because $r' \notin Res(\Gamma)$ and $O \cup L \cup D = Res(\Gamma)$.

For $n = 0$, it is trivially true. Let $n = k + 1$.

Because $Safe_n(C[r'/r], s, h, \rho[r'/r], \Gamma[r'/r], Q, A)$ is valid. If $C = \mathsf{skip}$, then $C[r'/r] = \mathsf{skip}$ and $s, h \models Q$, .

So the property (i) of $Safe_n(C, s, h, \rho, \Gamma, Q, A)$ is verified.

From $Safe_n(C[r'/r], s, h, \rho[r'/r], \Gamma[r'/r], Q, A)$ and Proposition 8,

$$C[r'/r], (s, h, \rho[r'/r]) \not\to_p \mathsf{abort}.$$

Hence, we have the property (ii) of $Safe_n(C, s, h, \rho, \Gamma, Q, A)$.

From $Safe_n(C[r'/r], s, h, \rho[r'/r], \Gamma[r'/r], Q, A)$ be valid, we have

$$chng(C) \cap \bigcup_{\hat{r} \in L \cup D} PV(\hat{r}) = chng(C[r'/r]) \cap \bigcup_{\hat{r} \in (L \cup D)[r'/r]} PV(\hat{r}) = \emptyset.$$

Then the property (iii) of $Safe_n(C, s, h, \rho, \Gamma, Q, A)$ is respected.

Let h_G, C', s', h' and ρ' such that $h_G \perp h$, $s, h_G \models \circledast_{\hat{r} \in D} \Gamma(\hat{r})$ and

$$C, (s, h \uplus h_G, \rho) \xrightarrow{A,\Gamma} C', (s', h', \rho').$$

Note that C' is a reachable command. By Proposition 8, we have that

$$C[r'/r], (s, h \uplus h_G, \rho[r'/r]) \xrightarrow{A,\Gamma[r'/r]} C'[r'/r], (s', h', \rho'[r'/r]).$$

Moreover, we know that

$$s, h_G \models \circledast_{\hat{r} \in D[r'/r]} \Gamma[r'/r](\hat{r}).$$

By $Safe_n(C[r'/r], s, h, \rho[r'/r], \Gamma[r'/r], Q, A)$ and the transition above, there exists h'_G such that $h'_G \subseteq h'$, $Safe_k(C'[r'/r], s', h' \setminus h'_G, \rho'[r'/r], \Gamma[r'/r], Q, A)$ is valid and

$$s', h'_G \models \circledast_{\hat{r} \in D'[r'/r]} \Gamma[r'/r](\hat{r}).$$

It is easy to see that $s', h'_G \models \circledast_{\hat{r} \in D'} \Gamma(\hat{r})$.

By induction hypothesis, $Safe_k(C', s', h' \setminus h'_G, \rho', \Gamma, Q, A)$ is valid. □

To prove the soundness of the rule for auxiliary variables, we have the following result.

Proposition 17. *Let C be a reachable command, $(s, h), (s', h) \in \mathcal{S}$, $\rho \in \mathcal{O}$, Q an assertion, $A, X \subseteq$ **Var**, Γ a resource context and $l \in \mathbb{N}_0$ such that X is a set of auxiliary variables for C, l is the number of assignments to auxiliary variables in C, $FV(Q) \subseteq A$ and $X \cap (PV(\Gamma) \cup FV(Q)) = \emptyset$.*

If $Safe_{n+l}(C, s, h, \rho, \Gamma, Q, A \cup X)$ is valid and $s(x) = s'(x)$, for every $x \notin X$, then $Safe_n(C \setminus X, s', h, \rho, \Gamma, Q, A)$ is valid.

Proof. Let C be a reachable command, $(s, h), (s', h) \in \mathcal{S}$, $\rho \in \mathcal{O}$, Q an assertion, $A, X \subseteq \mathbf{Var}$, Γ a resource context and $l \in \mathbb{N}_0$ such that X is a set of auxiliary variables for C, l is the number of assignments to auxiliary variables in C, $FV(Q) \subseteq A$, $X \cap (PV(\Gamma) \cup FV(Q)) = \emptyset$, $s'(x) = s(x)$, for every $x \notin X$ and $Safe_{n+l}(C, s, h, \rho, \Gamma, Q, A \cup X)$ is valid.

The proof is done by induction on n. If $n = 0$, the conclusion is valid.

Let $n = k + 1$.

Suppose that $C \setminus X = \mathsf{skip}$. Then $C = \mathsf{skip}$ or $C = \mathsf{x{:=}e}$, where $\mathsf{x} \in X$.

First, we assume $C = \mathsf{skip}$. From $Safe_{n+l}(\mathsf{skip}, s, h, \rho, \Gamma, Q, A \cup X)$, we have $s, h \models Q$. If $s(y) = s'(y)$, for every $y \in FV(Q) \subset \mathbf{Var} \setminus X$, then $s', h \models Q$.

Now, we assume $C = \mathsf{x}{:=}\mathsf{e}$, $\mathsf{x} \in X$. Consider the transition given by (BCT)

$$C, (s, h, \rho) \rightarrow_p \mathsf{skip}, (s'', h, \rho),$$

where $s(y) = s''(y)$, for every $y \notin X$.

Then $Safe_{n+l-1}(\mathsf{skip}, s'', h, \rho, \Gamma, Q, A \cup X)$ is valid and $n+l-1 > 0$. Hence $s'', h \models Q$ and $s', h \models Q$.

The above leads to the first property of $Safe_n(C \setminus X, s', h, \rho, \Gamma, Q, A)$.

From $Safe_{n+l}(C, s, h, \rho, \Gamma, Q, A \cup X)$, we know that the execution of C does not abort, for (s, h, ρ). The command $C \setminus X$ substitutes assignments by skip, so the execution of $C \setminus X$ does not abort, for (s, h, ρ). Changing the value of auxiliary variables X does not introduce aborts in the execution of $C \setminus X$. Hence

$$C \setminus X, (s', h, \rho) \not\rightarrow_p \mathsf{abort}.$$

The second property of $Safe_n(C \setminus X, s', h, \rho, \Gamma, Q, A)$ is verified.

Note that $chng(C \setminus X) = chng(C) \setminus X$. From $Safe_{n+l}(C, s, h, \rho, \Gamma, Q, A \cup X)$,

$$chng(C \setminus X) \cap \bigcup_{r \in L \cup D} PV(r) \subseteq chng(C) \cap \bigcup_{r \in L \cup D} PV(r) = \emptyset.$$

This establishes the third property of $Safe_n(C \setminus X, s', h, \rho, \Gamma, Q, A)$.

Let h_G, $\hat{C}$, $\hat{s}$, $\hat{h}$ and $\hat{\rho}$ such that $h_G \perp h$, $s', h_G \models \circledast_{r \in D} \Gamma(r)$ and

$$C \setminus X, (s', h \uplus h_G, \rho) \xrightarrow{A,\Gamma} \hat{C}, (\hat{s}, \hat{h}, \hat{\rho}).$$

Suppose that it is a program transition.

We observe that there exists $\hat{C}'$, $\hat{s}'$ and j such that $\hat{C}' \setminus X = \hat{C}$, $\hat{s}'(x) = \hat{s}(x)$, for every $x \notin X$, $j = l(C) - l(\hat{C}')$ and

$$C, (s, h \uplus h_G, \rho) \rightarrow_p^{j+1} \hat{C}', (\hat{s}', \hat{h}, \hat{\rho}).$$

From $Safe_{n+l}(C, s, h, \rho, \Gamma, Q, A \cup X)$, we know that there is $h'_G \subseteq \hat{h}$ such that $Safe_{n+l-j-1}(\hat{C}', \hat{s}', \hat{h} \setminus h'_G, \hat{\rho}, \Gamma, Q, A \cup X)$ is valid and $\hat{s}', h'_G \models \circledast_{r \in \hat{D}} \Gamma(r)$. Then $\hat{s}, h'_G \models \circledast_{r \in \hat{D}} \Gamma(r)$, because $\hat{s}'(x) = \hat{s}(x)$, for every $x \in PV(\Gamma) \subseteq \mathbf{Var} \setminus X$.

Note that j is the number of assignments to auxiliary variables in $\hat{C}'$.

By the induction hypothesis, we conclude $Safe_k(\hat{C}' \setminus X, \hat{s}, \hat{h} \setminus h'_G, \hat{\rho}, \Gamma, Q, A)$.

Last, suppose that it is an environment transition. Let

$$A' = A \cup \bigcup_{r \in Locked(C \setminus X)} PV(r).$$

We have $\hat{C} = C \setminus X$, $(s', h, \rho) \overset{A'}{\rightsquigarrow} (\hat{s}, h, \hat{\rho})$, and there exists $h'_G \subseteq \hat{h}$ such that $\hat{h} = h \uplus h'_G$ and

$$\hat{s}, h'_G \models \circledast_{r \in \hat{D}} \Gamma(r).$$

Note that $Locked(C) = Locked(C \setminus X)$. And consider the storage $\hat{s}'$ such that $\hat{s}'(x) = \hat{s}(x)$, if $x \notin X$, and $\hat{s}'(x) = s(x)$, if $x \in X$ and

$$A'' = A \cup X \cup \bigcup_{r \in Locked(C)} PV(r).$$

We have the following environment transformation $(s, h, \rho) \overset{A''}{\rightsquigarrow} (\hat{s}', h, \hat{\rho})$, and the environment transition

$$C, (s, h, \rho) \xrightarrow{A \cup X, \Gamma}_e C, (\hat{s}', h, \hat{\rho}).$$

By the previous transition, we obtain $Safe_{k+l}(C, \hat{s}', h, \hat{\rho}, \Gamma, Q, A \cup X)$.

Note that $\hat{s}'(x) = \hat{s}(x)$, for every $x \notin X$. Therefore by the induction hypothesis, we have that $Safe_k(C \setminus X, \hat{s}, h, \hat{\rho}, \Gamma, Q, A)$ is valid. □

The last proposition exhausts the inferences rules of CSL, completing the proof of Theorem 3.

7 Conclusion

This work presents a proof of correctness of CSL based on SOS, the first we are aware of. We build on two previous proofs, one for the full logic, using a denotational semantics based on traces, and another for a fragment of CSL, the DCSL. An immediate extension to this work is its formalization in a theorem prover.

A proof based on SOS is important, as this form of semantics closer mimics the execution of an imperative program. Therefore, it paves the way to the development of more expressive proving tools that are able to deal with truly concurrent programs manipulating shared resources. Our work may also provide insight on how to develop provably correct compilers able of detecting data-races.

Our aim was lifting the (severe) restriction of forcing concurrent threads to manipulate only disjoint sets of variables, since it does not allow proving correct many interesting and useful programs. To attain this goal, we re-used the notion of "rely-set" , a notion crucial to obtain the soundness result of CSL with respect to the denotational semantics. The adaptation was not trivial and required developing several auxiliary notions, but established a proof technique that may now be used in other contexts.

References

[1] M. Ben-Ari. *Principles of concurrent and distributed programming(Second Edition)*. Addison-Wesley, 2006.

[2] J. Boyland. Checking interference with fractional permissions. In *SAS*, volume 2694 of *Lecture Notes in Computer Science*, pages 55–72. Springer, 2003.

[3] S. Brookes. A semantics for concurrent separation logic. *Theoretical Computer Science*, 375(1-3):227–270, 2007.

[4] S. Brookes. A revisionist history of concurrent separation logic. *ENTCS*, 276:5–28, 2011.

[5] C. Calcagno, D. Distefano, J. Dubreil, D. Gabi, P. Hooimeijer, M. Luca, P. W. O'Hearn, I. Papakonstantinou, J. Purbrick, and D. Rodriguez. Moving fast with software verification. In *NASA Formal Methods - 7th International Symposium, NFM 2015, Pasadena, CA, USA, April 27-29, 2015, Proceedings*, pages 3–11, 2015.

[6] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969.

[7] P. W. O'Hearn. Resources, concurrency, and local reasoning. *Theoretical Computer Science*, 375(1-3):271–307, 2007.

[8] P. W. O'Hearn, J. C. Reynolds, and H. Yang. Local reasoning about programs that alter data structures. In *CSL*, volume 2142 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2001.

[9] S. S. Owicki. A consistent and complete deductive system for the verification of parallel programs. In *STOC*, pages 73–86. ACM, 1976.

[10] S. S. Owicki and D. Gries. Verifying properties of parallel programs: An axiomatic approach. *Communications of the ACM*, 19(5):279–285, 1976.

[11] G. D. Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Programming*, 60–61:17–139, 2004.

[12] U. S. Reddy and J. C. Reynolds. Syntactic control of interference for separation logic. In *POPL*, pages 323–336. ACM, 2012.

[13] J. C. Reynolds. Separation logic: A logic for shared mutable data structures. In *LICS*, pages 55–74. IEEE Computer Society, 2002.

[14] P. Soares, A. Ravara, and S. Melo de Sousa. An operational semantics for concurrent separation logic. Technical Report RR-DCC-2014-11, Department of Computer Science, Faculty of Science, University of Porto, 2014.

[15] P. Soares, A. Ravara, and S. Melo de Sousa. Revisiting concurrent separation logic and operational semantics. In *PDP*, pages 484–491. IEEE, 2015.

[16] V. Vafeiadis. Concurrent separation logic and operational semantics. *ENTCS*, 276:335–351, 2011.

[17] V. Vafeiadis and M. J. Parkinson. A marriage of rely/guarantee and separation logic. In *CONCUR*, volume 4703 of *Lecture Notes in Computer Science*, pages 256–271. Springer, 2007.

Appendix

Proof of Proposition 4. Let $C \in \mathcal{C}$, $(s,h) \in \mathcal{S}$, and $\rho \in \mathcal{O}$. Suppose h_F is a heap such that $h \perp h_F$ and

$$C, (s, h \uplus h_F, \rho) \to_p \mathsf{abort}.$$

We'll prove that $C, (s, h, \rho) \to_p \mathsf{abort}$ by induction on the relation, $\to_p$.

Suppose that the transition to abort is given by (RA), (WA) or $(WA2)$. Then the transitions does not depended on the heap and $C, (s, h, \rho) \to_p \mathsf{abort}$.

Suppose that it is given by (BCA). Then there is a faulty memory access of $h \uplus h_F$ and, consequently, of h. Hence $C, (s, h, \rho) \to_p \mathsf{abort}$.

Suppose that it is given by (SA). Then $C = C_1 \,;\, C_2$ and $C_1, (s, h \uplus h_F, \rho) \to_p \mathsf{abort}$.

By induction, we know that $C_1, (s, h, \rho) \to_p \mathsf{abort}$. Hence $C, (s, h, \rho) \to_p \mathsf{abort}$.

The remaining cases are similar to the previous case. □

Proof of Proposition 5. Let $C, C' \in \mathcal{C}$, $(s,h), (s',h') \in \mathcal{S}$, $\rho, \rho' \in \mathcal{O}$ and h_F such that $h \perp h_F$, $C, (s, h, \rho') \not\to_p \mathsf{abort}$ and

$$C, (s, h \uplus h_F, \rho) \to_p C', (s', h', \rho').$$

We prove by induction on the program rules that h_F is a subheap of h' and $C, (s, h, \rho) \rightarrow_p C', (s', h' \setminus h_F, \rho')$.

Suppose that the transition is given by (BCT). Because the Separation Logic respects the frame property, we know that h_F is a subheap of h' and

$$C, (s, h, \rho) \rightarrow_p C', (s', h' \setminus h_F, \rho').$$

Suppose that the transition is given by $(S1)$, (LP), $(IF1)$, $(IF2)$, $(R0)$, $(P3)$, $(W0)$ or $(W2)$. Then the transition neither depends nor changes the heap function. So we obtain that h' and

$$C, (s, h, \rho) \rightarrow_p C', (s', h' \setminus h_F, \rho').$$

Suppose that the transition is given by $(S2)$. Then $C = C_1 ; C_2$ and $C' = C_1' ; C_2$ such that

$$C_1, (s, h \uplus h_F, \rho) \rightarrow_p C_1', (s', h', \rho').$$

If $C_1, (s, h, \rho) \rightarrow \mathsf{abort}$, then $C, (s, h, \rho) \rightarrow \mathsf{abort}$. Hence

$$C_1, (s, h, \rho) \not\rightarrow \mathsf{abort}.$$

From the induction hypothesis, we conclude that $h_F \subseteq h'$ and

$$C_1, (s, h, \rho) \rightarrow_p C_1', (s', h' \setminus h_F, \rho').$$

Therefore

$$C, (s, h, \rho) \rightarrow_p C', (s', h' \setminus h_F, \rho').$$

The cases $(P1)$, $(P2)$, $(R1)$, $(R2)$ and $(W1)$ are similar to the previous case. □