An Operational Semantics for Concurrent Separation Logic

Pedro Soares[1] António Ravara[2] Simão Melo de Sousa[3]
[1]LIACC
Universidade do Porto
psoares@fc.up.pt
[2]CITI & DI-FCT
Universidade Nova de Lisboa
aravara@fct.unl.pt
[3]LIACC & DI-FE
Universidade da Beira Interior
desousa@di.ubi.pt

Technical Report Series: DCC-2014-11

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
Rua do Campo Alegre, 1021/1055,
4169-007 PORTO,
PORTUGAL
Tel: 220 402 900 Fax: 220 402 950
http://www.dcc.fc.up.pt/Pubs/

An Operational Semantics for Concurrent Separation Logic

Pedro Soares[1] António Ravara[2] Simão Melo de Sousa[3]
[1]LIACC [2]CITI & DI-FCT
Universidade do Porto Universidade Nova de Lisboa
psoares@fc.up.pt aravara@fct.unl.pt
[3]LIACC & DI-FE
Universidade da Beira Interior
desousa@di.ubi.pt

Abstract

The deductive verification of concurrent programs gained new tools with the advent of Concurrent Separation Logic (CSL). This program logic is a compositional method that combines the Owicki-Gries method with Separation Logic, allowing to reason and prove correct concurrent programs manipulating shared mutable data structure. The soundness of Concurrent Separation Logic had been established using a denotational semantics (based on traces). An alternative proof based on structural operational semantics 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 Concurrent Separation Logic with respect to an operational semantics.

1 Introduction

The aim of this work is to present a new soundness proof for the Concurrent Separation Logic, [7], with respect to a structural operational semantics, [11]. This work adapts and extends the results presented by Vafeiadis, [15].

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 know by a Hoare triple: $\{P\}C\{Q\}$, where P and Q are first 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. We say that a program is partially correct, if every execution of C from a state respecting the precondition does not abort and the postcondition holds for its final state, when the execution terminates. 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 Owicki-Gries method, see e.g. [9, 10], and Separation Logic, see e.g. [8, 13, 14].

The Owicki-Gries method is one of the first attempts to give a resource-sensitive proof system for concurrent programs. Each resource has a mutual exclusion lock, an assertion, called invariant, and a set of variables, called protected variables. To reason about concurrent programs synchronizing via resources, 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.

The execution of parallel composition non-deterministically chooses one of the commands to execute first. As usual, the execution is assumed weakly fair, i.e. if a commands 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 termination of the execution.

The programs derivable by the Owicki-Gries method have to respect resources invariants — properties respected when acquiring and releasing resources — and respect the protection of variables — to change a protected variable, it has to acquire all resources protecting that variable. The parallel rule of the proof system proposed by Owicki, [9], requires that every free variable occurring in one command can not be changed by another, except by protected variables occurring within the derivation proof of its critical region. Thus, the Owicki-Gries method is not compositional.

Separation Logic 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 in the Hoare Logic had to be augmented. The assertions are extended with the constructs `emp`, representing the empty memory; $e \mapsto e'$, representing a single memory location e with the value e'; and $P * Q$, representing a portion of memory that can be divided in two disjoint parts, one satisfying P and the other satisfying Q. The programing language is augmented with commands for reading a memory location, `x:=[e]`; for writing on a memory location, `[e]:=e'`; for allocating a memory cell, `x:=cons(e)`; and for deallocating a memory cell, `dispose(e)`. Naturally, the proof system is also extended with a rule for each new commands and with a frame rule, used to enlarge the portion of memory considered in the specification of a program. This rule is crucial to achieve local reasoning: program specifications only need to consider the memory relevant for their execution; the frame rule allows to extend those memory portions. Therefore, this local reasoning mechanism can be used to establish the partial correctness of disjoint concurrent programs, i.e. concurrent program where each thread do not modified variables used by others threads.

In these 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. 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 portion corresponds to an execution on the initial one.

Since the introducing of Separation Logic, different authors adapted Separation Logic to the verification of concurrent programs. Vafeiadis introduced RGSep, combining Separation Logic with Rely/Guarantee reasoning [16]. Reddy and Reynolds introduced a syntactic control of interference in Separation Logic [12], borrowing ideas from works on fractional permissions [2]. O'Hearn proposed Concurrent Separation Logic (CSL), combining Separation Logic with the Owicki-Gries method [7]. Brookes formalized this version of the logic, 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 free variable of a specification, with the exception of the variables occurring inside a critical region. 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 the store, the heap and the sets of resources, expressing variable ownership [4]. Actually, the rely-set was introduced after Wehrman and Berdine discovered a counter-example to the initial proof of soundness 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, playing the same role in the parallel rule used by Brookes.

Alternatively, Vafeiadis proposed a structural operational semantics for concurrent programs synchronizing via resources, and proved the soundness of a fragment of CSL, the Disjoint Concurrent Separation Logic, where concurrent threads are not allowed to modify shared variables [15].

Our motivation for this work was to remove the disjointness condition and obtain a soundness proof using a structural operational semantics for the full Concurrent Separation Logic. The goal is relevant because it is a step in the development of more expressive provers well integrated in software development environments.

Concretely, the contributions of this work are the following.

- 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 4.1). This relation is important to study the interferences made by other threads. Furthermore it

allows to show the soundness of the parallel rule. Note that Vafeadis uses a different notion of environment transition [16].

- The resource configuration that expresses ownership. It is defined with three sets: owned resources, locked resources, and available resources (Definition 24). A program state is formed by a store, a heap and a resource configuration.

- A collection of examples of programs which we prove correct in Concurrent Separation Logic. The aim is to illustrate the expressiveness of the proof system; hence we selected "classical" problems. In particular, we show correct programs to push and pop over a stack in parallel; to lock and release of a binary semaphore in parallel; and to multiply two matrices in parallel (Section 2.6). These examples together with the programs proved by O'Hearn and Brookes, [7, 3, 4], support the usefulness of Concurrent Separation Logic.

In the rest of this paper, we review the syntax of concurrent resource-oriented program with shared mutable data (Section 2.4) and Concurrent Separation Logic proof system (Section 2.5), following the work of Brookes [4]. Next, we present a structural operational semantics for the previous programs (Section 3.4), along the lines of the work of Vafeiadis [15]. We prove that this operational semantics respects safety monotonicity and the frame property (Section 3.6) and we give a specific formula for the resource configuration along an execution (Section 3.7). Afterwards, we introduce the environment transition relation (Section 4.1). Finally, we prove the soundness of Concurrent Separation Logic with respect to the operational semantic we defined (Section 4.4).

2 Concurrent Separation Logic

In the following sections, Concurrent Separation Logic is revisited, as presented by Brookes [4]. First, the syntax of expressions, Boolean expressions and assertions are defined. This part follows the work done for Separation Logic, see e.g. [13]. Next, the syntax of commands for concurrent programs is specified. Then, the inference rules for Concurrent Separation Logic are given. To finish some examples of derivable programs in Concurrent Separation Logic are presented.

2.1 Expressions and Boolean expressions

We write **Var** for the set of *variables*, which are usually expressed with lowercase letter, e.g. $x, y, \ldots$.

Let **Val** denote the set of *values*. The set of values include at least the natural numbers and the representation of *null*, i.e. $\mathbb{N}_0 \cup \{null\} \subseteq \mathbf{Val}$.

The set of *expressions* is denoted by **Exp** and it is given by the following grammar

$$e := x \mid n \mid e_1 + e_2 \mid e_1 - e_2 \mid e_1 \times e_2, \quad x \in \mathbf{Var}, \quad n \in \mathbf{Val}.$$

The set of *Boolean expressions* is denoted by **Bool** and it is defined by

$$B := \texttt{true} \mid \texttt{false} \mid e_1 = e_2 \mid e_1 < e_2 \mid B_1 \wedge B_2 \mid B_1 \vee B_2 \mid \neg B, \quad e_1, e_2 \in \mathbf{Exp}.$$

Next, we define the set of variables, that occur free inside an expression.

Definition 1. *Let $e \in$ **Exp**. The set of free variables in e is denoted by $FV_E(e)$ and it is given by:*

- $FV_E(x) = \{x\}$,
- $FV_E(n) = \emptyset$,
- $FV_E(e_1 + e_2) = FV_E(e_1 - e_2) = FV_E(e_1 \times e_2) = FV_E(e_1) \cup FV_E(e_2)$.

For the set of expressions, $\{e_1, e_2, \ldots, e_n\}$, we write $FV_E(e_1, e_2, \ldots, e_n)$ to denote the set $FV_E(e_1) \cup FV_E(e_2) \cup \ldots \cup FV_E(e_n)$. The set of free variables for a Boolean expression is defined below.

Definition 2. *Let $B \in \textbf{\textit{Bool}}$. The set of free variables in B is denoted by $FV_{BE}(B)$ and it is given by:*

- $FV_{BE}(\mathtt{true}) = FV_{BE}(\mathtt{false}) = \emptyset$,
- $FV_{BE}(e_1 = e_2) = FV_{BE}(e_1 < e_2) = FV_E(e_1) \cup FV_E(e_2)$,
- $FV_{BE}(B_1 \wedge B_2) = FV_{BE}(B_1 \vee B_2) = FV_{BE}(B_1) \cup FV_{BE}(B_2)$,
- $FV_{BE}(\neg B) = FV_{BE}(B)$.

For the set of Boolean expressions, $\{B_1, B_2, \ldots, B_n\}$, we write $FV_{BE}(B_1, B_2, \ldots, B_n)$ to denote the set $FV_{BE}(B_1) \cup FV_{BE}(B_2) \cup \ldots \cup FV_{BE}(B_n)$.

In both definition, when no confusing arise we omit the subscript.

Next, we see how the partial substitution interact with expressions and Boolean expressions.

Definition 3. *Let $e \in \textbf{\textit{Exp}}$, $x \in \textbf{\textit{Var}}$ and $v \in \textbf{\textit{Exp}}$. The partial substitution in the expression e of x by v is denoted by $e[v/x]_E$, and it is defined by:*

- *if $y = x$, then $y[v/x]_E = v$, else $y[v/x] = y$*
- $n[v/x]_E = n$,
- $(e_1 + e_2)[v/x]_E = e_1[v/x]_E + e_2[v/x]_E$,
- $(e_1 - e_2)[v/x]_E = e_1[v/x]_E - e_2[v/x]_E$,
- $(e_1 \times e_2)[v/x]_E = e_1[v/x]_E \times e_2[v/x]_E$.

We denote the partial substitutions $e[v_1/x_1]_E[v_2/x_2]_E \ldots [v_n/x_n]_E$ by $e[v_1/x_1, v_2/x_2, \ldots, v_n/x_n]_E$.

Definition 4. *Let $B \in \textbf{\textit{Bool}}$, $x \in \textbf{\textit{Var}}$ and $v \in \textbf{\textit{Exp}}$. The partial substitution in the Boolean expression B of x by v is denoted by $B[v/x]_{BE}$, and it is defined by:*

- $\mathtt{true}[v/x]_{BE} = \mathtt{true}$,
- $\mathtt{false}[v/x]_{BE} = \mathtt{false}$,
- $(e_1 = e_2)[v/x]_{BE} = (e_1[v/x]_E = e_2[v/x]_E)$,
- $(e_1 < e_2)[v/x]_{BE} = (e_1[v/x]_E < e_2[v/x]_E)$,
- $(B_1 \wedge B_2)[v/x]_{BE} = (B_1[v/x]_{BE} \wedge B_2[v/x]_{BE})$,
- $(B_1 \vee B_2)[v/x]_{BE} = (B_1[v/x]_{BE} \vee B_2[v/x]_{BE})$,
- $(\neg B)[v/x]_{BE} = \neg B[v/x]_{BE}$.

The notation $B[v_1/x_1, v_2/x_2, \ldots, v_n/x_n]_B$ stands for $B[v_1/x_1]_B[v_2/x_2]_B \ldots [v_n/x_n]_B$.

As before, we omit the subscript in partial substitutions when no confusing arise.

2.2 Assertions

We begin this section by defining the syntax of assertions.

Definition 5. *The set of* assertions *is denoted by* ***Astn*** *and it is given by:*

$$\begin{array}{rcl} P & := & B \mid \neg P \mid P_1 \wedge P_2 \mid P_1 \vee P_2 \mid P_1 \Rightarrow P_2 \mid \forall x P \mid \exists x P \\ & \mid & \mathtt{emp} \mid e \mapsto e' \mid P_1 * P_2 \mid P_1 -\!\!* P_2, \end{array}$$

where $B \in \textbf{\textit{Bool}}$, $x \in \textbf{\textit{Var}}$ and $e, e' \in \textbf{\textit{Exp}}$.

Besides the assertion usually used in Hoare Logic, we also consider the assertion introduced by Separation Logic. They intent to describe the following ideas:

`emp`	the empty memory.
$e \mapsto e'$	a memory with the location e, which has the value e'.
$P_1 * P_2$	a memory that can be divided in two disjoint parts such that one part verifies P_1 and the other part verifies P_2.
$P_1 -\!\!* P_2$	a memory such that, for every disjoint memory that verifies P_1, the union of memories verifies P_2 .

The set of variables that occur free inside an assertion are defined below.

Definition 6. *Let $P \in \boldsymbol{Astn}$. The set of free variables in P is denoted by $FV_A(P)$ and it is given by:*

- $FV_A(B) = FV_{BE}(B)$,
- $FV_A(\mathit{emp}) = \emptyset$,
- $FV_A(\neg P) = FV_A(P)$,
- $FV_A(\forall x P) = FV_A(\exists x P) = FV_A(P) \setminus \{x\}$,
- $FV_A(P_1 \wedge P_2) = FV_A(P_1 \vee P_2) = FV_A(P_1 \Rightarrow P_2) = FV_A(P_1 * P_2) = FV_A(P_1 -\!\!* P_2) = FV_A(P_1) \cup FV_A(P_2)$,
- $FV_A(e \mapsto e') = FV_E(e, e')$.

As before, for a set of assertions $\{P_1, P_2, \ldots, P_n\}$, we write $FV_A(P_1, P_2, \ldots, P_n)$ to represent the set $FV_A(P_1) \cup FV_A(P_2) \cup \ldots \cup FV_A(P_n)$. And, when no confusing arise we omit the subscript.

Next, we define the partial substitution on assertions.

Definition 7. *Let $P \in \boldsymbol{Astn}$. We denote the partial substitution in the assertion P of $x \in \boldsymbol{Var}$ for $v \in \boldsymbol{Exp}$ by $P[v/x]_A$, and it is inductively defined in the following way:*

- $B[v/x]_A = B[v/x]_{BE}$,
- $(\neg P)[v/x]_A = \neg P[v/x]_A$,
- $(P_1 \wedge P_2)[v/x]_A = P_1[v/x]_A \wedge P_2[v/x]_A$,
- $(P_1 \vee P_2)[v/x]_A = P_1[v/x]_A \vee P_2[v/x]_A$,
- $(P_1 \Rightarrow P_2)[v/x]_A = P_1[v/x]_A \Rightarrow P_2[v/x]_A$,
- *if $x' = x$, then $(\forall x' P)[v/x]_A = \forall x' P$, else $(\forall x' P)[v/x]_A = \forall x'(P)[v/x]_A$,*
- *if $x' = x$, then $(\exists x' P)[v/x]_A = \exists x' P$, else $(\exists x' P)[v/x]_A = \exists x'(P)[v/x]_A$,*
- $\mathit{emp}[v/x]_A = \mathit{emp}$,
- $(e \mapsto e')[v/x]_A = (e[v/x]_E \mapsto e'[v/x]_E)$,
- $(P_1 * P_2)[v/x]_A = P_1[v/x]_A * P_2[v/x]_A$,
- $(P_1 -\!\!* P_2)[v/x]_A = P_1[v/x]_A -\!\!* P_2[v/x]_A$.

We write $P[v_1/x_1, v_2/x_2, \ldots, v_n/x_n]_A$ for the partial substitution $P[v_1/x_1]_A[v_2/x_2]_A \ldots [v_n/x_n]_A$. When no confusing arise, we omit the subscript.

Despite we did not give the semantic of assertions, we introduce the notions of validity and precise assertions. The interested readers can see the sections 3.1 and 3.2 for the formal definitions.

We say that an assertion P is valid and write $\models P$, if P is satisfied by every storage and heap (definition 21).

If for a given storage and heap, the subheap that satisfies P is uniquely determined, then we say that P is precise (definition 22).

For instance, we give examples of valid and precise assertions.

Example 1. *The following assertions are valid:*

- $\models P \Rightarrow P$,
- $\models P * \mathsf{emp} \Rightarrow P$,
- $\models P \Rightarrow P * \mathsf{emp}$.

Example 2.

- emp *is a precise assertion.*
- $e \mapsto e'$ *is a precise assertion.*
- *if* P *is precise and* $B \in \boldsymbol{B}$*, then* $P \wedge B$ *is precise.*
- *if* P_1 *and* P_2 *are precise assertions, then* $P_1 * P_2$ *is a precise assertion.*

2.3 Resources context

The set of *resources names* is denoted by **Res**. The resources names are usually denoted by lowercase letters different from those used for variables, i.e. $\mathbf{Res} \cap \mathbf{Var} = \emptyset$.

Next, we define the resource context as it is defined in [4]. The global properties are represented by resource contexts.

Definition 8. *A* resource context Γ *has the following form*

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

where $r_1, r_2, \ldots, r_n$ *are distinct resources names,* $R_1, R_2, \ldots, R_n$ *are assertions and* $X_1, X_2, \ldots, X_n$ *are sets of variables such that* $FV(R_i) \subseteq X_i$*, for each* $i = 1, 2, \ldots, n$.

We say that the resource context Γ *is* well-formed *if* R_i *is a precise assertion, for every* i.

As noted by others authors, because we'll consider the conjunction rule in the inferences rules, it is necessarily to use precise assertion to define the resource context. This is illustrated with an example in [7, Section 11]. Moreover, the results presented in this text do not require that the resource context is precise, except for the conjunction rule.

The following examples are instances of well-formed resources context.

Example 3.

- *A resource context that establish the equality of two variables has the form*

  $$r(\mathsf{x}, \mathsf{y}) : \mathsf{x} = \mathsf{y} \wedge \mathsf{emp}.$$

- *The resource st represents a stack*

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

  where $stack(\mathsf{z})$ *is inductively defined by:*

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

  and $z \mapsto a, b$ *is abbreviation to* $\mathsf{z} \mapsto a * \mathsf{z} + 1 \mapsto b$.

- *Let* $a_{i,j} \in \boldsymbol{Val}$*,* $i = 1, \ldots, n$ *and* $j = 1, \ldots, m$*. A matrix,* $n \times m$*, with coefficients* $a_{i,j}$ *is represented by the following resource context*

  $$mat(X, n, m) : X \mapsto a_{1,1}, a_{1,2} \ldots, a_{n,m}.$$

- *Let $i = 1, \ldots, m$. The next resource context represents the i-column of the matrix above*

$$col_i(X, n, m) : X + (i-1) \mapsto a_{1,i} * X + (i-1) + m \mapsto a_{2,i} * \ldots * X + (i-1) + (n-1)m \mapsto a_{n,i}.$$

- *Let $j = 1, \ldots, n$. The j-line of the matrix above is represent by*

$$lin_j(X, m) : X + (j-1)m \mapsto a_{j,1}, a_{j,2}, \ldots a_{j,m}.$$

- *Let p and q denote different threads. A binary semaphore for the threads p and q can be represented in the following way*

$$se(\mathsf{wantp}, \mathsf{wantq}) : \mathsf{wantp} + \mathsf{wantq} \leq 1 \wedge (\mathsf{wantp} = 0 \vee \mathsf{wantq} = 0) \wedge \mathsf{emp}.$$

In the next definition, we fix some notation for resource contexts.

Definition 9. *Let Γ be a well-formed resource context with the form*

$$r_1(X_1) : R_1, r_2(X_2) : R_2, \ldots, r_n(X_n) : R_n.$$

- *We denote by $Res(\Gamma)$ the set $\{r_1, r_2, \ldots r_n\}$,*
- *For $r_i \in Res(\Gamma)$, we say that the variables in X_i are protected by the resource r_i and we denote X_i by $PV(r_i)$,*
- *We denote by $PV(\Gamma)$ the set $\bigcup_{r \in Res(\Gamma)} PV(r)$,*
- *For $r_i \in Res(\Gamma)$, the assertion R_i is denoted by $\Gamma(r_i)$,*
- *For a set $D = \{r_{j_1}, r_{j_2}, \ldots, r_{j_k}\} \subseteq Res(\Gamma)$, we define*

$$\circledast_{r \in D} \Gamma(r) = R_{j_1} * R_{j_2} * \ldots * R_{j_k},$$

- *We write $inv(\Gamma)$ to represent $\circledast_{r \in Res(\Gamma)} \Gamma(r)$.*

Below, we see how to change the resources names inside a resource context.

Definition 10. *Let $r, r' \in \mathbf{Res}$ and Γ be a resource context with the form*

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

such that $r' \notin Res(\Gamma)$. We define the substitution $\Gamma[r'/r]$ by:

- *if $r = r_i$, $\Gamma[r'/r] = r_1(X_1) : R_1, r_2(X_2) : R_2, \ldots, r'(X_i) : R_i, \ldots, r_n(X_n) : R_n$.*
- *$\Gamma[r'/r] = \Gamma$, otherwise.*

For every resource context, we associate to each resource a set of protected variables. Then the resource context determines a system of permission, as noted in [12] and [5].

2.4 Programming Language

The *commands* C are given by the following grammar:

$$\begin{array}{rcl} C & := & \mathsf{skip} \mid \mathsf{x{:=}e} \mid \mathsf{x{:=}[e]} \mid \mathsf{[e]{:=}e'} \mid \mathsf{x{:=}cons}(\overline{\mathsf{e}}) \mid \mathsf{dispose(e)} \\ & \mid & C_1 \,;\, C_2 \mid \mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2 \mid \mathsf{while}\ B\ \mathsf{do}\ C \mid C_1 \parallel C_2 \\ & \mid & \mathsf{resource}\ r\ \mathsf{in}\ C \mid \mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ C, \end{array}$$

where $\mathsf{x} \in \mathbf{Var}$, $\mathsf{e}, \mathsf{e'} \in \mathbf{Exp}$, $B \in \mathbf{Bool}$ and $\overline{\mathsf{e}} = (\mathsf{e}_1, \ldots, \mathsf{e}_n)$ is a vector of expressions.

The set of commands that result from the grammar above is denoted by **Com**.

The set of commands is composed by the commands consider by Hoare Logic, Separation Logic, and Concurrent Separation Logic. The commands appended by Separation Logic capture the following ideas:

x:=[e]	it loads the value on the location e to the variable x.
[e]:=e'	it changes the value on the location e to e'.
x:=cons($\overline{e}$)	it allocates n contiguous locations in the memory with the values $e_1, \ldots, e_n$ and assigns the first location to the variable x.
dispose(e)	it deallocates the location e from the memory.

The commands added by Concurrent Separation Logic express the following ideas:

$C_1 \parallel C_2$	it computes in parallel the commands C_1 and C_2, with weakly fairness (i.e. every enable command is eventually considered in the execution).
with r when B do C	if B is satisfied and the resource is not locked, it acquires the resource r and execute C, after the execution it releases the resource r.
resource r in C	it executes the command C with the local resource r.

Next, we give some examples of program in **Com**.

Example 4.

- *With our granularity of commands, we can increment a variable in parallel using the following program*

$$x{:=}x{+}1 \parallel x{:=}x{+}1$$

- *Let st denotes a stack. To pop an element from st, we use*

$$\textit{with } st \textit{ when } \neg(z = null) \textit{ do } y{:=}z\,;\ x1{:=}y\,;\ z{:=}[y{+}1]\,;\ \textit{dispose}(y{+}1),$$

and to push the element x to the stack

$$\textit{with } st \textit{ do } y{:=}\textit{cons}(x,z)\,;\ z{:=}y.$$

- *Let X be a matrix $n \times k$, Y a matrix $k \times m$, $i = 1, \ldots, n$ and $j = 1, \ldots, m$. The i-line of X, lin_i, multiplied by the j-column of Y, col_j, is obtained by*

$$\textit{with } lin_i \textit{ do with } col_j \textit{ do } x{:=}0; l{:=}0; \textit{while } l < k \textit{ do } a{:=}[X{+}(i{-}1)k + l]; b{:=}[Y{+}(j{-}1) + k\ l]; x{:=}x{+}\ a\ b; l{:=}l{+}1.$$

- *The resource se represents a binary semaphore for the threads p and q. In the thread p, the semaphore is acquired using the command above*

$$\textit{with } se \textit{ when } wantq = 0 \textit{ do } wantp{:=}1,$$

and the semaphore is released by

$$\textit{with } se \textit{ do } wantp{:=}0.$$

We use analogous commands, for the thread q.

Next, we define the set of variables that occur free in a command.

Definition 11. *Let $C \in$ **Com**. The set of free variables in C is denoted by $FV(C)$ and it is given by:*

- $FV(skip) = \emptyset$,
- $FV(x{:=}e) = FV(x{:=}[e]) = \{x\} \cup FV_E(e)$,
- $FV(x{:=}cons((e_1, \ldots, e_n))) = \{x\} \cup FV_E(e_1, \ldots, e_n)$,

- $FV(\mathsf{[e]:=e'}) = FV_E(\mathsf{e}, \mathsf{e'})$,
- $FV(\mathsf{dispose(e)}) = FV_E(\mathsf{e})$,
- $FV(C_1 \,;C_2) = FV(C_1 \parallel C_2) = FV(C_1) \cup FV(C_2)$,
- $FV(\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2) = FV_{BE}(B) \cup FV(C_1) \cup FV(C_2)$,
- $FV(\mathsf{while}\ B\ \mathsf{do}\ C) = FV(\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ C) = FV_{BE}(B) \cup FV(C)$,
- $FV(\mathsf{resource}\ r\ \mathsf{in}\ C) = FV(C)$,

Let $\{C_1, C_2, \ldots, C_n\} \subset \mathbf{Com}$, we denote $FV(C_1) \cup FV(C_2) \cup \ldots \cup FV(C_n)$ by $FV(C_1, C_2, \ldots, C_n)$. In the following definition, we define the set of variables that can be modified by a command.

Definition 12. *Let $C \in \boldsymbol{Com}$. The set of variables* modified *by C is named $mod(C)$ and it is given by:*

- $mod(\mathsf{x:=e}) = mod(\mathsf{x:=[e]}) = mod(\mathsf{x:=cons(e)}) = \{\mathsf{x}\}$,
- $mod(\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2) = mod(C_1 \,;C_2) = mod(C_1 \parallel C_2) = mod(C_1) \cup mod(C_2)$,
- $mod(\mathsf{while}\ B\ \mathsf{do}\ C) = mod(\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ C) = mod(\mathsf{resource}\ r\ \mathsf{in}\ C) = mod(C)$,
- $mod(\alpha) = \emptyset$, *otherwise.*

The set of auxiliary variables have been useful to deduce more specific post conditions for a program's specification, see e.g. [10]. Next, we give the definition of auxiliary variables for a command.

Definition 13. *Let $C \in \boldsymbol{Com}$. We say that X is a set of* auxiliary variables *for C if every occurrence of $x \in X$ in C is inside a 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 the assignments to auxiliary variables by the command skip. This is formalized in the next definition.

Definition 14. *Let $C \in \boldsymbol{Com}$ and X a set of auxiliary variables for C. We denote by $C \setminus X$ the substitution of every assignments to auxiliary variables in C by skip. It is inductively defined by:*

- *If* $\mathsf{x} \in X$, *then* $(\mathsf{x:=e}) \setminus X = \mathsf{skip}$, *else* $(\mathsf{x:=e}) \setminus X = \mathsf{x:=e}$,
- $(\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2) \setminus X = \mathsf{if}\ B\ \mathsf{then}\ C_1 \setminus X\ \mathsf{else}\ C_2 \setminus X$,
- $(\mathsf{while}\ B\ \mathsf{do}\ C) \setminus X = \mathsf{while}\ B\ \mathsf{do}\ (C \setminus X)$,
- $(C_1 \,;C_2) \setminus X = (C_1 \setminus X) \,;(C_2 \setminus X)$,
- $(C_1 \parallel C_2) \setminus X = (C_1 \setminus X) \parallel (C_2 \setminus X)$,
- $(\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ C) \setminus X = \mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ (C \setminus X)$,
- $(\mathsf{resource}\ r\ \mathsf{in}\ C) \setminus X = \mathsf{resource}\ r\ \mathsf{in}\ (C \setminus X)$,
- $C \setminus X = C$, *otherwise.*

Below, we define the set of resources that appear in a command.

Definition 15. *Let $C \in \mathcal{C}$. The set of resources that occur in a command C is denoted by $Res(C)$ and it is defined by:*

- $Res(\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ C) = Res(\mathsf{resource}\ r\ \mathsf{in}\ C) = Res(C) \cup \{r\}$,
- $Res(C_1 \,;C_2) = Res(\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2) = Res(C_1 \parallel C_1) = Res(C_1) \cup Res(C_2)$,

- $Res(\textsf{while } B \textsf{ do } C) = Res(C)$,
- $Res(C) = \emptyset$, *otherwise.*

To change the name of resources inside a command, we use the next definition.

Definition 16. *Let $C \in \boldsymbol{Com}$ and $r, r' \in \boldsymbol{Res}$ such that $r' \notin Res(C)$. We denote the substitution of r for r' on C by $C[r'/r]$ and it is defined by:*

- $(\textsf{if } B \textsf{ then } C_1 \textsf{ else } C_2)[r'/r] = \textsf{if } B \textsf{ then } C_1[r'/r] \textsf{ else } C_2[r'/r]$,
- $(\textsf{while } B \textsf{ do } C)[r'/r] = \textsf{while } B \textsf{ do } (C[r'/r])$,
- $(C_1 \,;\, C_2)[r'/r] = (C_1[r'/r]) \,;\, (C_2[r'/r])$,
- $(C_1 \parallel C_2)[r'/r] = (C_1[r'/r]) \parallel (C_2[r'/r])$,
- $(\textsf{with } \hat{r} \textsf{ when } B \textsf{ do } C)[r'/r] = \textsf{with } \hat{r}[r'/r] \textsf{ when } B \textsf{ do } (C[r'/r])$,
- $(\textsf{resource } \hat{r} \textsf{ in } C)[r'/r] = \textsf{resource } \hat{r}[r'/r] \textsf{ in } (C[r'/r])$,
- $C[r'/r] = C$, *otherwise,*

where $\hat{r}[r'/r] = r'$, if $\hat{r} = r$, and $\hat{r}[r'/r] = \hat{r}$, otherwise.

2.5 Inference rules

In this section, we write the inference rules for Concurrent Separation Logic, proposed in [4]. We start by defining the specification of programs that will be used in the inference rules.

Definition 17. *Let Γ be a resource context, $A \subseteq \boldsymbol{Var}$, $P, Q \in \boldsymbol{Astn}$ and $C \in \boldsymbol{Com}$. The* specification of a program *have the form*

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

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

The set A used in the specification of program above is called the *rely-set.*
In the rules below, we only use well-formed specifications of programs.

($SKIP$)

$$\frac{}{\Gamma \vdash_A \{P\}\textsf{skip}\{P\}}$$

($ASSIGNMENT$)

$$\frac{\textsf{x} \notin PV(\Gamma) \quad FV(\textsf{e}) \subseteq A}{\Gamma \vdash_A \{P[\textsf{e}/\textsf{x}]\}\textsf{x:=e}\{P\}}$$

($SEQUENCE$)

$$\frac{\Gamma \vdash_{A_1} \{P_1\}C_1\{P_2\} \quad \Gamma \vdash_{A_2} \{P_2\}C_2\{P_3\}}{\Gamma \vdash_{A_1 \cup A_2} \{P_1\}C_1 \,;\, C_2\{P_3\}}$$

($CONDITIONAL$)

$$\frac{\Gamma \vdash_{A_1} \{P \wedge B\}C_1\{Q\} \quad \Gamma \vdash_{A_2} \{P \wedge \neg B\}C_2\{Q\}}{\Gamma \vdash_{A_1 \cup A_2} \{P\}\textsf{if } B \textsf{ then } C_1 \textsf{ else } C_2\{Q\}}$$

($LOOP$)

$$\frac{\Gamma \vdash_A \{P \wedge B\} C \{P\}}{\Gamma \vdash_A \{P\} \mathsf{while}\ B\ \mathsf{do}\ C \{P \wedge \neg B\}}$$

($CONSEQUENCE$)

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

($AUXILIARY$)

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

($CONJUNCTION$)

$$\frac{\Gamma \vdash_{A_1} \{P_1\} C \{Q_1\} \quad \Gamma \vdash_{A_2} \{P_2\} C \{Q_2\}}{\Gamma \vdash_{A_1 \cup A_2} \{P_1 \wedge P_2\} C \{Q_1 \wedge Q_2\}}$$

($LOOKUP$)

$$\frac{\mathsf{x} \notin PV(\Gamma) \cup FV(\mathsf{e}, \mathsf{e'})}{\Gamma \vdash_A \{P[\mathsf{e'}/\mathsf{x}] \wedge \mathsf{e} \mapsto \mathsf{e'}\} \mathsf{x} := [\mathsf{e}] \{P \wedge \mathsf{e} \mapsto \mathsf{e'}\}}$$

($UPDATE$)

$$\frac{}{\Gamma \vdash_A \{\mathsf{e} \mapsto -\} [\mathsf{e}] := \mathsf{e'} \{\mathsf{e} \mapsto \mathsf{e'}\}}$$

($ALLOCATION$)

$$\frac{\mathsf{x} \notin FV(\mathsf{e}_1, \ldots, \mathsf{e}_n) \cup PV(\Gamma)}{\Gamma \vdash_A \{\mathsf{emp}\} \mathsf{x} := \mathsf{cons}((\mathsf{e}_1, \ldots, \mathsf{e}_n)) \{\mathsf{x} \mapsto \mathsf{e}_1, \ldots, \mathsf{e}_n\}}$$

($DISPOSAL$)

$$\frac{}{\Gamma \vdash_A \{\mathsf{e} \mapsto -\} \mathsf{dispose}(\mathsf{e}) \{\mathsf{emp}\}}$$

($FRAME$)

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

($PARALLEL$)

$$\frac{\Gamma \vdash_{A_1} \{P_1\} C_1 \{Q_1\} \quad \Gamma \vdash_{A_2} \{P_2\} C_2 \{Q_2\} \quad mod(C_1) \cap A_2 = mod(C_2) \cap A_1 = \emptyset}{\Gamma \vdash_{A_1 \cup A_2} \{P_1 * P_2\} C_1 \parallel C_2 \{Q_1 * Q_2\}}$$

($CRITICAL\ REGION$)

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

(*LOCAL RESOURCE*)

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

(*RENAMING*)

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

We remind the reader to the difference between the (*RENAMING*) rule presented above and the rule presented in [4]. Note that the renaming rule presented here implies the correspondent rule in [4].

2.6 Examples

In this section, we give some examples of derivable program's specifications for the rules presented before.

We start by the operations of pop and push on a stack. And we show that we can perform, in parallel, both operations on the same stack.

Example 5.

The operation of pop in a stack has the following specification that rely in one variable,

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

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

We protect the variable y by the resource st, so we do not need to rely on it. Note that the specification is well-formed.

Consider the following specification.

$$\begin{array}{lll}
\vdash_{\{\mathsf{x1,z,y}\}} & \{\mathsf{emp} * \exists_{a,b}\ \mathsf{z} \mapsto a, b * stack(b)\} \\
& \mathsf{y{:=}z;} \\
& \{\mathsf{emp} * \exists_{a,b}\ \mathsf{y} \mapsto a, b * stack(b)\} \\
& \mathsf{x1{:=}y;} \\
& \{\mathsf{emp} * \exists_{a,b}\ \mathsf{x1} \mapsto a * \mathsf{y{+}1} \mapsto b * stack(b))\} \\
& \quad \{\exists_b\ b = b \wedge \mathsf{y} + 1 \mapsto b\} \\
& \quad \mathsf{z{:=}[y{+}1];} \\
& \quad \{\exists_b\ \mathsf{z} = b \wedge \mathsf{y} + 1 \mapsto b\} \\
& \{\mathsf{emp} * \exists_a\ \mathsf{x1} \mapsto a * \mathsf{y{+}1} \mapsto \mathsf{z} * stack(\mathsf{z}))\} \\
& \quad \{\mathsf{y} + 1 \mapsto \mathsf{z}\} \\
& \quad \mathsf{dispose(y{+}1)} \\
& \quad \{\mathsf{emp}\} \\
& \{(\exists_a\ \mathsf{x1} \mapsto a) * (stack(\mathsf{z}))\}
\end{array}$$

By the (*CRITICAL REGION*), we have

$$\begin{array}{lll}
st(\mathsf{z},\mathsf{y}) : stack(z) & \vdash_{\{\mathsf{x1}\}} & \{\mathsf{emp}\} \\
& & pop(); \\
& & \{\ \exists_a\ \mathsf{x1} \mapsto a\} \\
& & \mathsf{dispose(x1)} \\
& & \{\mathsf{emp}\}
\end{array}$$

The push operator in a stack has the following specification,

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

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

As before, we note that the specification is well-formed. And the specification above is obtained, from the following specification and the $(CRITICAL\ REGION)$.

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

Noting that $mod(push(\mathsf{x2})) \cap \{\mathsf{x1}\} = \emptyset$ and $mod(pop) \cap \{\mathsf{x2}\} = \emptyset$, we can apply the $(PARALLEL)$ rule and we have the following specification

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

The next example shows that we can create a simple binary semaphore for two threads. This example is inspired in the solutions presented for the critical region problem in [1, Section 3].

In contrast to the solution obtained in [1, Section 3], we obtain a simpler solution for the critical region problem. The program that we obtain is more simple, because we can use the command with r when B do C. This example is similar to an example proposed in [7, Section 4].

Example 6.

We have the following specifications for the thread p:

$$se(\mathsf{wantp},\mathsf{wantq}) : S \vdash_{\emptyset} \{\mathsf{emp}\} lockp() \{\mathsf{emp}\},$$
$$se(\mathsf{wantp},\mathsf{wantq}) : S \vdash_{\emptyset} \{\mathsf{emp}\} releasep() \{\mathsf{emp}\},$$

where

- $S = \mathsf{wantp} + \mathsf{wantq} \leq 1 \wedge (\mathsf{wantp} = 0 \vee \mathsf{wantq} = 0) \wedge \mathsf{emp}$,
- $lockp() =$ with se when $\mathsf{wantq} = 0$ do wantp:=1,
- $releasep() =$ with se do wantp:=0.

Consider the next well-formed specification of programs.

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

and

$$\begin{array}{ll} \vdash_{\{\mathsf{wantp},\mathsf{wantq}\}} & \{\mathtt{emp} * S\} \\ & \{0 + \mathsf{wantq} \leq 1 \wedge \mathtt{emp}\} \\ & \mathsf{wantp}{:=}0 \\ & \{\mathsf{wantp} = 0 \wedge \mathsf{wantp} + \mathsf{wantq} \leq 1 \wedge \mathtt{emp}\} \\ & \{\mathtt{emp} * S\} \end{array}$$

Applying the $(CRITICAL\ REGION)$ we obtain the desired specifications. Considering the analogous programs and derivations for the thread q we obtain:

- $se(\mathsf{wantp}, \mathsf{wantq}) : S \vdash_{\emptyset} \{\mathtt{emp}\} lockq() \{\mathtt{emp}\}$
- $se(\mathsf{wantp}, \mathsf{wantq}) : S \vdash_{\emptyset} \{\mathtt{emp}\} releaseq() \{\mathtt{emp}\}$,

where $lockq() = \mathsf{with}\ se\ \mathsf{when}\ \mathsf{wantp} = 0\ \mathsf{do}\ \mathsf{wantq}{:=}1$ and $releaseq() = \mathsf{with}\ se\ \mathsf{do}\ \mathsf{wantq}{:=}0$.

Using the $(PARALLEL)$ rule, we obtain the next specification

$$se(\mathsf{wantp}, \mathsf{wantq}) : S \vdash_{\emptyset} \{\mathtt{emp}\} (lockp() \,;\, releasep()) \parallel (lockq() \,;\, releaseq()) \{\mathtt{emp}\}$$

Now, we give an informal reason to explain why the resource, se, and the operations, $lock$ and $release$, are a solution to the critical region problem. In particular, we discuss the properties of mutual exclusion, free from deadlock, and free from starvation. We do not formalize this discussion, because Concurrent Separation Logic is not suitable for this questions.

Consider the following program

$$\begin{array}{c|c|c} lockp() & \parallel & lockq() \\ \text{C.R.} & \parallel & \text{C.R.} \\ releasep() & \parallel & releaseq() \end{array}$$

Note that when the thread p (q) enter the critical region (C.R.), it changes the value of the variable wantp (wantq, respectively) to 1. The mutual exclusion follows from the invariant $\mathsf{wantp} + \mathsf{wantq} \leq 1$.

The execution of this program is free from deadlock, because the invariant impose that one of the control variables $\mathsf{wantp}, \mathsf{wantq}$ have the value 0.

Assuming that the parallel execution is fair, we also have that the program is free from starvation. Because each thread automatically releases the semaphore after computing the critical region.

In the last example, we view the multiplication of matrices. We illustrate the problem by multiplying a line from one matrix with a column from the other matrix. Moreover we show that we can perform the multiplication of two lines with two column in parallel. To compute the multiplication of two matrix is enough to compute the operation for each line and column in parallel.

Example 7.

Let $i_1, j_1 \in \mathbb{N}$ and lin_{i_1}, col_{j_1} the resources defined in the example 3. Consider the following derivation.

$\vdash_{\{X,k,Y,k,m,r_1,l_1,a_1,b_1\}} \{lin_{i_1}(X,k) * col_{j_1}(Y,k,m)\}$

r_1:=0 ; l_1:=0;

$\{(lin_{i_1}(X,k) * col_{j_1}(Y,k,m)) \wedge r_1 = 0 \wedge l_1 = 0\}$

$\{(lin_{i_1}(X,k) * col_{j_1}(Y,k,m)) \wedge r_1 = \sum_{i=0}^{l_1} a_{i_1,i} b_{i,j_1} \wedge l_1 \leq k\}$

while $l_1 < k$ do (

$\{(lin_{i_1}(X,k) * col_{j_1}(Y,k,m)) \wedge r_1 = \sum_{i=0}^{l_1} a_{i_1,i} b_{i,j_1} \wedge l_1 \leq k \wedge l_1 < k\}$

a_1:=$[X + (i_1 - 1)k + l_1]$; b_1:=$[Y + (j_1 - 1) + kl_1]$;

$\{(lin_{i_1}(X,k) * col_{j_1}(Y,k,m)) \wedge r_1 + a_1 b_1 = \sum_{i=0}^{l_1+1} a_{i_1,i} b_{i,j_1} \wedge l_1 \leq k \wedge l_1 < k\}$

r_1:=$r_1 + a_1 b_1$;

$\{(lin_{i_1}(X,k) * col_{j_1}(Y,k,m)) \wedge r_1 = \sum_{i=0}^{l_1+1} a_{i_1,i} b_{i,j_1} \wedge l_1 \leq k \wedge l_1 < k\}$

l_1:=$l_1 + 1$

$\{(lin_{i_1}(X,k) * col_{j_1}(Y,k,m)) \wedge r_1 = \sum_{i=0}^{l_1} a_{i_1,i} b_{i,j_1} \wedge l_1 - 1 \leq k \wedge l_1 - 1 < k\}$

$\{(lin_{i_1}(X,k) * col_{j_1}(Y,k,m)) \wedge r_1 = \sum_{i=0}^{l_1} a_{i_1,i} b_{i,j_1} \wedge l_1 \leq k\}$

)

$\{(lin_{i_1}(X,k) * col_{j_1}(Y,k,m)) \wedge r_1 = \sum_{i=0}^{l_1} a_{i_1,i} b_{i,j_1} \wedge l_1 \leq k \wedge l_1 \geq k\}$

$\{(lin_{i_1}(X,k) * col_{j_1}(Y,k,m)) \wedge r_1 = \sum_{i=0}^{k} a_{i_1,i} b_{i,j_1}\}$

Denoting by $multi_{i_1,j_1}$ the program

r_1:=0 ; l_1:=0 ; while $l_1 < k$ do (a_1:=$[X + (i_1 - 1)k + l]$; b_1:=$[Y + (j_1 - 1) + kl]$; r_1:=$r_1 + a_1 b_1$; l_1:=$l_1 + 1$)

And applying the critical region rule, we have the specification above

$$lin_{i_1}(X,k), col_{j_1}(Y,k,m) \vdash_{A_1} \{\texttt{emp}\}\text{with } lin_{i_1} \text{ do with } col_{j_1} \text{ do } multi_{i_1,j_1}\{r_1 = \sum_{i=0}^{k} a_{i_1,i} b_{i,j_1} \wedge \texttt{emp}\},$$

where $A_1 = \{k, r_1, l_1, a_1, b_1\}$.

On the same way, we obtain the following specification

$$lin_{i_2}(X,k), col_{j_2}(Y,k,m) \vdash_{A_2} \{\texttt{emp}\}\text{with } lin_{i_2} \text{ do with } col_{j_2} \text{ do } multi_{i_2,j_2}\{r_2 = \sum_{i=0}^{k} a_{i_2,i} b_{i,j_2} \wedge \texttt{emp}\},$$

where $A_2 = \{k, r_2, l_2, a_2, b_2\}$.

Note that $mod(multi_{i_1,j_1}) = \{r_1, l_1, a_1, b_1\}$ and $mod(multi_{i_2,j_2}) = \{r_2, l_2, a_2, b_2\}$.

Then $A_1 \cap mod(multi_{i_2,j_2}) = \emptyset$, $A_2 \cap mod(multi_{i_1,j_1}) = \emptyset$ and by the $(PARALLEL)$ rule, we conclude that

$$lin_{i_1}(X,k), col_{j_1}(Y,k,m), lin_{i_2}(X,k), col_{j_2}(Y,k,m) \vdash_{A_1 \cup A_2} \{\mathtt{emp}\}$$
$$\mathsf{with}\ lin_{i_1}\ \mathsf{do\ with}\ col_{j_1}\ \mathsf{do}\ multi_{i_1,j_1} \| \mathsf{with}\ lin_{i_2}\ \mathsf{do\ with}\ col_{j_2}\ \mathsf{do}\ multi_{i_2,j_2}$$
$$\{r_1 = \sum_{i=0}^{k} a_{i_1,i} b_{i,j_1} \wedge r_2 = \sum_{i=0}^{k} a_{i_2,i} b_{i,j_2} \wedge \mathtt{emp}\}.$$

We finish this analyze by noting that we can use the $(LOCAL\ RESOURCE)$ rule and obtain the next specification

$$\vdash_{\{X,Y,n,m,k,r_1,l_1,a_1,b_1,r_2,l_2,a_2,b_2\}} \{mat(X,n,k) * mat(Y,k,m)\}$$
$$\mathsf{resource}\ lin_{i_1}\ \mathsf{in\ resource}\ lin_{i_2}\ \mathsf{in\ resource}\ col_{j_1}\ \mathsf{in\ resource}\ col_{j_2}\ \mathsf{in}$$
$$\mathsf{with}\ lin_{i_1}\ \mathsf{do\ with}\ col_{j_1}\ \mathsf{do}\ multi_{i_1,j_1} \| \mathsf{with}\ lin_{i_2}\ \mathsf{do\ with}\ col_{j_2}\ \mathsf{do}\ multi_{i_2,j_2}$$
$$\{mat(X,n,k) * mat(Y,k,m) \wedge r_1 = \sum_{i=0}^{k} a_{i_1,i} b_{i,j_1} \wedge r_2 = \sum_{i=0}^{k} a_{i_2,i} b_{i,j_2}\}.$$

3 Operational Semantics

Now, the operational semantics is introduced. The soundness of Concurrent Separation Logic will be proved using this operational semantics.

The evaluation of expressions, Boolean expressions and assertions are recapped, as it is done in [17, 13, 14]. Next, the language of programs is augmented with a command that expresses the execution of the critical region with the resource acquired. This extension follows the propose done by Vafeiadis [15].

The resource configuration, expressing the ownership of resources, is added to the usual definition of program's state. Then the transitions of programs and the notion of validity are presented. In the following, some results about the transitions of programs are proved, e.g. safety monotonicity, frame property, and the behave of resource configurations along an execution is specified.

3.1 Expressions and Boolean Expressions evaluation

We represent the value assignment to each variable by the function $s : \mathbf{Var} \to \mathbf{Val}$, which it is denoted by *storage*. The set of all storages is denoted by $\mathcal{S}$.

We write $s[x_1 : v_1 \mid x_2 : v_2 \mid \ldots \mid x_n : v_n]$ to represent the modified storage such that $s[x_1 : v_1 \mid x_2 : v_2 \mid \ldots \mid x_n : v_n](y) = s(y)$, if $y \notin \{x_1, x_2, \ldots, x_n\}$ and $s[x_1 : v_1 \mid x_2 : v_2 \mid \ldots \mid x_n : v_n](x_i) = v_i$, where $i = 1, 2, \ldots, n$.

Next, we define how the expressions and boolean expressions are evaluated.

Definition 18. *Let $s \in \mathcal{S}$ and $e \in \boldsymbol{Exp}$. Inductively, the evaluation of e is defined by:*

- $[\![x]\!]_s = s(x)$,
- $[\![n]\!]_s = n$,
- $[\![e_1 + e_2]\!]_s = [\![e_1]\!]_s + [\![e_2]\!]_s$,
- $[\![e_1 - e_2]\!]_s = [\![e_1]\!]_s - [\![e_2]\!]_s$,
- $[\![e_1 \times e_2]\!]_s = [\![e_1]\!]_s \times [\![e_2]\!]_s$.

Definition 19. *Let $s \in \mathcal{S}$ and $B \in \boldsymbol{Bool}$. Inductively, the evaluation of B is defined by:*

- $[\![\mathit{true}]\!]_s = \mathit{true}$,
- $[\![\mathit{false}]\!]_s = \mathit{false}$,
- *if* $[\![e_1]\!]_s = [\![e_2]\!]_s$, *then* $[\![e_1 = e_2]\!]_s = \mathit{true}$, *else* $[\![e_1 = e_2]\!]_s = \mathit{false}$,
- *if* $[\![e_1]\!]_s < [\![e_2]\!]_s$, *then* $[\![e_1 < e_2]\!]_s = \mathit{true}$, *else* $[\![e_1 < e_2]\!]_s = \mathit{false}$,
- $[\![B_1 \wedge B_2]\!]_s = [\![B_2]\!]_s \wedge [\![B_1]\!]_s$,
- $[\![B_1 \vee B_2]\!]_s = [\![B_1]\!]_s \vee [\![B_2]\!]_s$,
- $[\![\neg B]\!]_s = \neg[\![B]\!]_s$.

Usually, we write $s(e)$ for $[\![e]\!]_s$ and $s(B)$ for $[\![B]\!]_s$.

Next, we see that the evaluation of expressions or Boolean expressions only depends on their free variables.

Proposition 1. *Let* $e \in \mathbf{Exp}$, $B \in \mathbf{Bool}$ *and* $s, s' \in \mathcal{S}$.

- *If* $s(x) = s'(x)$, *for every* $x \in FV(e)$, *then* $s(e) = s'(e)$,
- *If* $s(x) = s'(x)$, *for every* $x \in FV(B)$, *then* $s(B) = s'(B)$.

Proof. First, we prove by induction on the structure of the expressions.

Let $s, s' \in \mathcal{S}$ and $e \in \mathbf{Exp}$ such that $s(x) = s'(x)$, for every $x \in FV(e)$.

Suppose that $e = x$. Then $x \in FV(e)$ and

$$s(e) = s(x) = s'(x) = s'(e).$$

Suppose that $e = n$, $n \in \mathbf{Val}$. Then

$$s(e) = n = s'(e).$$

Suppose that $e = e_1 + e_2$. Then $FV(e) = FV(e_1) \cup FV(e_2)$.

By induction hypothesis, we know that $s(e_1) = s'(e_1)$ and $s(e_2) = s'(e_2)$. Hence

$$s(e) = s(e_1) + s(e_2) = s'(e_1) + s'(e_2) = s'(e).$$

The cases $e_1 - e_2$ and $e_1 \times e_2$ are similar to the previous case. The proof for the expressions is complete.

Next, we prove by induction on the structure of the Boolean expressions.

Let $s, s' \in \mathcal{S}$ and $B \in \mathbf{Bool}$ such that $s(x) = s'(x)$, for every $x \in FV(B)$.

Suppose that $B = \mathtt{true}$ or $B = \mathtt{false}$. Then

$$s(B) = s'(B).$$

Suppose that $B = (e_1 = e_2)$. Then $FV(B) = FV(e_1) \cup FV(e_2)$.

Using the proposition for the expressions, we know that

$$s(e_1) = s'(e_1) \quad s(e_2) = s'(e_2).$$

Note that $s(e_1) = s(e_2)$ if and only if $s'(e_1) = s'(e_2)$. Therefore

$$s(B) = s'(B).$$

The case $B = (e_1 \leq e_2)$ is similar to the previous one.

Suppose that $B = B_1 \wedge B_2$. Then $FV(B) = FV(B_1) \cup FV(B_2)$.

By induction hypothesis, we know that

$$s(B_1) = s'(B_1) \quad s(B_2) = s'(B_2).$$

Therefore

$$s(B) = s(B_1) \wedge s(B_2) = s'(B_1) \wedge s'(B_2) = s'(B).$$

The cases $B_1 \vee B_2$ and $B = \neg B'$ are proved in the same way. □

3.2 Assertions evaluation

The set of *locations* is denoted by **Loc**. And each location correspond to a natural number, i.e. $\mathbf{Loc} \subseteq \mathbb{N}$.

The partial function $h : \mathbf{Loc} \to \mathbf{Val}$ is used to represent the memory, which it is denoted by *heap*. We represent the set of heaps by $\mathcal{H}$.

Next, we define some operations over the set of heaps.

Definition 20. *Let $h, h_1, h_2 \in \mathcal{H}$.*

- *If $dom(h_1) \cap dom(h_2) = \emptyset$, we say that h_1 and h_2 are* disjoints heaps *and we write that $h_1 \perp h_2$.*
- *We say that h_1 is a* sub-heap *of h_2 and we write that $h_1 \subseteq h_2$, if $dom(h_1) \subseteq dom(h_2)$ and $h_1(x) = h_2(x)$, for every $x \in dom(h_1)$.*
- *If $h_1 \perp h_2$, we denote by $h_1 \uplus h_2$ the unique heap such that $dom(h_1 \uplus h_2) = dom(h_1) \cup dom(h_2)$ and $h_1 \uplus h_2(x) = h_i(x)$, for $x \in dom(h_i)$ and $i = 1, 2$.*
- *If $h_2 \subseteq h_1$, we write $h_1 \setminus h_2$ to represent the unique heap such that $dom(h_1 \setminus h_2) = dom(h_1) \setminus dom(h_2)$ and $h_1 \setminus h_2(x) = h_1(x)$, for $x \in dom(h_1) \setminus dom(h_2)$.*
- *If $A \subseteq dom(h)$, we write $h \setminus A$ to denote the unique heap obtained by restrict the domain of h by $dom(h) \setminus A$.*

As before, we use the notation $h[l_1 : v_1 \mid l_2 : v_2 \mid \ldots \mid l_n : v_n]$ to represent a extension or modification on the heap.

In the following definition, we see how the assertions are evaluated.

Definition 21. *Let $s \in \mathcal{S}$, $h \in \mathcal{H}$ and $P \in \boldsymbol{Astn}$. Inductively, we define the relation $s, h \models P$ as follows:*

- $s, h \models B$, *if* $s(B) = \mathtt{true}$,
- $s, h \models \neg P$, *if* $s, h \not\models P$,
- $s, h \models P_1 \wedge P_2$, *if* $s, h \models P_1$ *and* $s, h \models P_2$,
- $s, h \models P_1 \vee P_2$, *if* $s, h \models P_1$ *or* $s, h \models P_2$,
- $s, h \models P_1 \Rightarrow P_2$, *if* $s, h \not\models P_1$ *or* $s, h \models P_2$,
- $s, h \models \forall x P$, *if for every* $v \in \boldsymbol{Val}$, $s[x : v], h \models P$,
- $s, h \models \exists x P$, *if there exists* $v \in \boldsymbol{Val}$ *such that* $s[x : v], h \models P$,
- $s, h \models \mathtt{emp}$, *if* $dom(h) = \emptyset$,
- $s, h \models e \mapsto e'$, *if* $dom(h) = \{s(e)\}$ *and* $h(s(e)) = s(e')$,
- $s, h \models P_1 * P_2$, *if there exist* h_1, h_2 *such that* $h = h_1 \uplus h_2$, $s, h_1 \models P_1$ *and* $s, h_2 \models P_2$,
- $s, h \models P_1 -\!\!* P_2$, *if for every* h' *such that* $h \perp h'$ *and* $s, h' \models P_1$ *we have that* $s, h \uplus h' \models P_2$.

We say that is P is valid and we write $\models P$, if for every $s \in \mathcal{S}$ and $h \in \mathcal{H}$ we have that $s, h \models P$.

For a given heap and storage, the precise assertions uniquely determine the subheap that verify it. This notion is formalized in the next definition.

Definition 22. *Let $P \in \boldsymbol{Astn}$. We say that P is* precise *if for every $h \in \mathcal{H}$ and $s \in \mathcal{S}$, there is at maximum one sub heap $h' \subseteq h$ such that $s, h' \models P$.*

In the proposition below, we stated that the evaluation of assertions only depends in their free variables.

Proposition 2. *Let $P \in \boldsymbol{Astn}$, $s, s' \in \mathcal{S}$ and $h \in \mathcal{H}$. If $s(x) = s'(x)$, for every $x \in FV(P)$, then*

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

Proof. The prove is done by induction on the structure of the assertions.

Let $s, s' \in \mathcal{S}$, $h \in \mathcal{H}$ and $P \in \mathbf{Astn}$ such that $s(x) = s'(x)$, for every $x \in FV(P)$.

Suppose that $P = B$. Then $FV(P) = FV(B)$. By the proposition 1, we know that

$$s(B) = s'(B).$$

If $s, h \models P$, we have that $s(B) = \texttt{true}$. Hence $s', h \models P$. The converse is analogous.

Suppose that $P = \neg P'$. Then $FV(P) = FV(P')$.

By induction hypothesis, we know that

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

Then

$$s, h \models \neg P' \text{ iff } s, h \not\models P' \text{ iff } s', h \not\models P' \text{ iff } s', h \models \neg P'$$

Suppose that $P = P_1 \wedge P_2$. Then $FV(P) = FV(P_1) \cup FV(P_2)$.

By induction hypothesis, we know, for $i = 1, 2$, that

$$s, h \models P_i \text{ iff } s', h \models P_i.$$

Then

$$s, h \models P_1 \wedge P_2 \text{ iff } s, h \models P_1 \wedge s, h \models P_2 \text{ iff } s', h \models P_1 \wedge s', h \models P_2 \text{ iff } s', h \models P_1 \wedge P_2$$

The cases $P_1 \vee P_2$ and $P_1 \Rightarrow P_2$ are similar to the previous case.

Suppose that $P = \forall x P'$. Then $FV(P) = FV(P') \setminus \{x\}$.

By the definition

$$s, h \models P \text{ iff } s[x : v], h \models P',$$

for every $v \in \mathbf{Val}$.

Note that $s[x : v](y) = s'[x : v](y)$, for every $y \in FV(P')$. Then, by the induction hypothesis

$$s[x : v], h \models P' \text{ iff } s'[x : v], h \models P',$$

for every $v \in \mathbf{Val}$. Therefore

$$s, h \models P \text{ iff } s', h \models P,$$

The case $\exists x P$ is identical to the previous case.

Suppose that $P = \texttt{emp}$. Then

$$s, h \models \texttt{emp} \text{ iff } dom(h) = \emptyset \text{ iff } s', h \models \texttt{emp}.$$

Suppose that $P = e \mapsto e'$. Then $FV(P) = FV(e, e')$.

If $s, h \models e \mapsto e'$, then $dom(h) = \{s(e)\}$ and $h(s(e)) = s(e')$. By proposition 1, we know that

$$s(e) = s'(e) \quad s(e') = s'(e')$$

Hence $dom(h) = \{s'(e)\}$ and $h(s'(e)) = s'(e')$. Therefore

$$s', h \models e \mapsto e'.$$

The other direction is similar.

Suppose that $P = P_1 * P_2$. Then $FV(P) = FV(P_1) \cup FV(P_2)$.

If $s, h \models P_1 * P_2$, then there are h_1, h_2 such that $h = h_1 \uplus h_2$ and

$$s, h_1 \models P_1 \quad s, h_2 \models P_2.$$

By induction hypothesis, we have that $s, h_i \models P_i$ if and only if $s', h_i \models P_i$, for $i = 1, 2$. Then

$$s', h \models P_1 * P_2.$$

Analogous, if $s', h \models P_1 * P_2$, then $s, h \models P_1 * P_2$.
Suppose that $P = P_1 -\!\!* P_2$. Then $FV(P) = FV(P_1) \cup FV(P_2)$.
If $s, h \models P_1 -\!\!* P_2$, then for every $h' \perp h$ such that $s, h' \models P_1$, we have that $s, h \uplus h' \models P_2$.
By induction hypothesis, we have that

$$s, h' \models P_1 \text{ iff } s', h' \models P_1$$

and

$$s, h \uplus h' \models P_2 \text{ iff } s', h \uplus h' \models P_2.$$

Then for every $h' \perp h$ such that $s', h' \models P_1$, we have that $s, h' \models P_1$ and $s, h \uplus h' \models P_2$. Hence, by induction hypothesis $s', h \uplus h' \models P_2$
Therefore

$$s', h \models P_1 -\!\!* P_2.$$

The other direction is similar. □

The next proposition is proved in [14, Proposition 3].

Proposition 3. *Let $P \in \boldsymbol{Astn}$, $x_1, x_2, \ldots, x_n \in \boldsymbol{Var}$, $v_1, v_2, \ldots, v_n \in \boldsymbol{Exp}$ and $s \in \mathcal{S}$. Then, for every $h \in \mathcal{H}$*

$$s, h \models P[v_1/x_1, v_2/x_2, \ldots, v_n/x_n] \quad \textit{iff} \quad s[x_1 : s(v_1) \mid x_2 : s(v_2) \mid \ldots \mid x_n : s(v_n)], h \models P.$$

3.3 Extended Programming Language

We extend the set of commands with the command within r do C, following the propose done in [15]. Intuitively, this command represents the execution of C with the resource r acquired. The set of extended commands is denoted by $\mathcal{C}$.

We extend the definitions given before for commands in the following way:

- $FV(\text{within } r \text{ do } C) = FV(C)$;
- $mod(\text{within } r \text{ do } C) = mod(C)$;
- $Res(\text{within } r \text{ do } C) = Res(C) \cup \{r\}$;
- $(\text{within } r \text{ do } C) \backslash X = \text{within } r \text{ do } (C \backslash X)$, where X is a set of auxiliary variables for within r do C;
- $(\text{within } \hat{r} \text{ do } C)[r'/r] = \text{within } \hat{r}[r'/r] \text{ do } (C[r'/r])$, where $r' \notin Res(\text{within } r \text{ do } C)$.

For every command in the extended language, we can associate a set of locked resources. This set is formed by the resources that have been acquired during the execution, and did not release yet.

Definition 23. *Let $C \in \mathcal{C}$. The set of* resources locked *by C is denoted by $Locked(C)$ and it is inductively defined by:*

- $Locked(C_1 \,;\, C_2) = Locked(C_1)$,
- $Locked(C_1 \parallel C_2) = Locked(C_1) \cup Locked(C_2)$,
- $Locked(\textit{within } r \textit{ do } C) = Locked(C) \cup \{r\}$,
- $Locked(\textit{resource } r \textit{ in } C) = Locked(C) \setminus \{r\}$,
- $Locked(C) = \emptyset$, *otherwise.*

3.4 Transitions of Programs

To express the ownership of resources, we introduce the *resource configurations*, that will be part of the state of a program.

Let $O, L, D \subseteq Res$. We say that $\rho = (O, L, D)$ is a *resource configuration* if O, L and D are disjoint pairwise. We represent the set of all resource configurations by $\mathcal{O}$.

In the resource configuration the set O represents the resources owned by the program, the set L represents the resources locked by the environment and the set D represents the resources available.

We fix the following notation for resource configurations.

Definition 24. *Let $\rho = (O, L, D), \rho' \in \mathcal{O}$ and $r \in \textbf{\textit{Res}}$.*

- *If $r \in O \cup L \cup D$, we write that $r \in \rho$, else $r \notin \rho$.*
- *If $\rho' = (O', L', D')$ and $O' = O \setminus \{r\}$, $L' = L \setminus \{r\}$ and $D' = D \setminus \{r\}$, then we write that $\rho' = \rho \setminus \{r\}$.*

Except when remarked, we will represent the set of resources in a resource configuration with the same superscript, e.g. $\rho' = (O', L', D')$.

The *state of a program* is given by (s, h, ρ), where $s \in \mathcal{S}$, $h \in \mathcal{H}$ and $\rho \in \mathcal{O}$, or by the abort state, abort.

The *transitions* in the operational semantic are represented by $\rightarrow_p$, where $\rightarrow_p \subseteq (\mathcal{C} \times \mathcal{S} \times \mathcal{H} \times \mathcal{O}) \times ((\mathcal{C} \times \mathcal{S} \times \mathcal{H} \times \mathcal{O}) \cup \{\mathsf{abort}\})$. This relation is defined by the following rules:

$$\frac{s(\mathsf{e}) = v}{\mathsf{x{:=}e}, (s, h, \rho) \rightarrow_p \mathsf{skip}, (s[\mathsf{x} : v], h, \rho)} \ (ASSIGN)$$

$$\frac{s(B) = \mathtt{true}}{\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, (s, h, \rho) \rightarrow_p C_1, (s, h, \rho)} \ (IF1)$$

$$\frac{s(B) = \mathtt{false}}{\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, (s, h, \rho) \rightarrow_p C_2, (s, h, \rho)} \ (IF2)$$

$$\frac{}{\mathsf{skip}; C_2, (s, h, \rho) \rightarrow_p C_2, (s, h, \rho)} \ (SEQ1) \qquad \frac{C_1, (s, h, \rho) \rightarrow_p C_1', (s', h', \rho')}{C_1 \,;\, C_2, (s, h, \rho) \rightarrow_p C_1' \,;\, C_2, (s', h', \rho')} \ (SEQ2)$$

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

$$\frac{s(\mathsf{e}) \in dom(h) \quad h(s(\mathsf{e})) = v}{\mathsf{x{:=}[e]}, (s, h, \rho) \rightarrow_p \mathsf{skip}, (s[\mathsf{x} : v], h, \rho)} \ (READ) \qquad \frac{s(\mathsf{e}) = l \in dom(h) \quad s(\mathsf{e'}) = v}{\mathsf{[e]{:=}e'}, (s, h, \rho) \rightarrow_p \mathsf{skip}, (s, h[l : v], \rho)} \ (WRI)$$

$$\frac{s(\mathsf{e}) = l \in dom(h)}{\mathsf{dispose(e)}, (s, h, \rho) \rightarrow_p \mathsf{skip}, (s, h \setminus \{l\}, \rho)} \ (FREE)$$

$$\frac{l \notin dom(h) \quad s(\mathsf{e}) = v}{\mathsf{x{:=}cons(e)}, (s, h, \rho) \rightarrow_p \mathsf{skip}, (s[\mathsf{x} : l], h[l : v], \rho)} \ (ALL)$$

$$\frac{C_1, (s, h, \rho) \rightarrow_p C_1', (s', h', \rho')}{C_1 \parallel C_2, (s, h, \rho) \rightarrow_p C_1' \parallel C_2, (s', h', \rho')} \ (PAR1)$$

$$\frac{C_2, (s, h, \rho) \rightarrow_p C_2', (s', h', \rho')}{C_1 \parallel C_2, (s, h, \rho) \rightarrow_p C_1 \parallel C_2', (s', h', \rho')} \ (PAR2) \qquad \frac{}{\mathsf{skip} \parallel \mathsf{skip}, (s, h, \rho) \rightarrow_p \mathsf{skip}, (s, h, \rho)} \ (PAR3)$$

$$\frac{r \notin \rho}{\mathsf{resource}\ r\ \mathsf{in}\ \mathsf{skip}, (s, h, \rho) \rightarrow_p \mathsf{skip}, (s, h, \rho)} \ (RES0)$$

$$\frac{r \notin \rho = (O, L, D) \quad r \in Locked(C) \quad C, (s, h, (O \cup \{r\}, L, D)) \rightarrow_p C', (s', h', \rho')}{\mathsf{resource}\ r\ \mathsf{in}\ C, (s, h, \rho) \rightarrow_p \mathsf{resource}\ r\ \mathsf{in}\ C', (s', h', \rho' \setminus \{r\})} \ (RES1)$$

$$\frac{r \notin \rho = (O, L, D) \quad r \notin Locked(C) \quad C, (s, h, (O, L, D \cup \{r\})) \rightarrow_p C', (s', h', \rho')}{\mathsf{resource}\ r\ \mathsf{in}\ C, (s, h, \rho) \rightarrow_p \mathsf{resource}\ r\ \mathsf{in}\ C', (s', h', \rho' \setminus \{r\})}\ (RES2)$$

$$\frac{\rho = (O, L, D \cup \{r\}) \quad \rho' = (O \cup \{r\}, L, D) \quad s(B) = \mathtt{true}}{\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ C, (s, h, \rho) \rightarrow_p \mathsf{within}\ r\ \mathsf{do}\ C, (s, h, \rho')}\ (WITH0)$$

$$\frac{r \in O \quad C, (s, h, (O \setminus \{r\}, L, D)) \rightarrow_p C', (s', h', (O', L', D'))}{\mathsf{within}\ r\ \mathsf{do}\ C, (s, h, (O, L, D)) \rightarrow_p \mathsf{within}\ r\ \mathsf{do}\ C', (s', h', (O' \cup \{r\}, L', D')}\ (WITH1)$$

$$\frac{\rho = (O \cup \{r\}, L, D) \quad \rho' = (O, L, D \cup \{r\})}{\mathsf{within}\ r\ \mathsf{do\ skip}, (s, h, \rho) \rightarrow_p \mathsf{skip}, (s, h, \rho')}\ (WITH2)$$

We use a simple version of the allocation command, because the allocation of multiple spaces do not add any difficult to our study.

In addition to the transition presented above, there is some execution that abort. Next, we define the transitions to the abort state.

$$\frac{C_1, (s, h, \rho) \rightarrow_p \mathsf{abort}}{C_1\ ; C_2, (s, h, \rho) \rightarrow_p \mathsf{abort}}\ (SEQA) \quad \frac{s(\mathsf{e}) \notin dom(h)}{\mathsf{dispose(e)}, (s, h, \rho) \rightarrow_p \mathsf{abort}}\ (FREEA)$$

$$\frac{s(\mathsf{e}) \notin dom(h)}{\mathsf{x{:=}[e]}, (s, h, \rho) \rightarrow_p \mathsf{abort}}\ (READA) \quad \frac{s(\mathsf{e}) \notin dom(h)}{\mathsf{[e]{:=}e'}, (s, h, \rho) \rightarrow_p \mathsf{abort}}\ (WRIA)$$

$$\frac{C_1, (s, h, \rho) \rightarrow_p \mathsf{abort}}{C_1 \parallel C_2, (s, h, \rho) \rightarrow_p \mathsf{abort}}\ (PAR1A) \quad \frac{C_2, (s, h, \rho) \rightarrow_p \mathsf{abort}}{C_1 \parallel C_2, (s, h, \rho) \rightarrow_p \mathsf{abort}}\ (PAR2A)$$

$$\frac{r \in Locked(C) \quad C, (s, h, (O \cup \{r\}, L, D)) \rightarrow_p \mathsf{abort}}{\mathsf{resource}\ r\ \mathsf{in}\ C, (s, h, (O, L, D)) \rightarrow_p \mathsf{abort}}\ (RESA1)$$

$$\frac{r \notin Locked(C) \quad C, (s, h, (O, L, D \cup \{r\})) \rightarrow_p \mathsf{abort}}{\mathsf{resource}\ r\ \mathsf{in}\ C, (s, h, (O, L, D)) \rightarrow_p \mathsf{abort}}\ (RESA2)$$

$$\frac{r \in \rho}{\mathsf{resource}\ r\ \mathsf{in}\ C, (s, h, \rho) \rightarrow_p \mathsf{abort}}\ (RESA) \quad \frac{r \notin \rho}{\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ C, (s, h, \rho) \rightarrow_p \mathsf{abort}}\ (WITHA)$$

$$\frac{C, (s, h, \rho \setminus \{r\}) \rightarrow_p \mathsf{abort}}{\mathsf{within}\ r\ \mathsf{do}\ C, (s, h, \rho) \rightarrow_p \mathsf{abort}}\ (WITHA1)$$

$$\frac{r \notin O}{\mathsf{within}\ r\ \mathsf{do}\ C, (s, h, (O, L, D)) \rightarrow_p \mathsf{abort}}\ (WITHA2)$$

First, we see that all transitions respect the resource configuration, i.e. the set of resources obtained after a transition are disjoint pairwise. Moreover, we see that every transition of program do not add or delete resource from the resource context, and the resources locked by the environment are not altered.

Proposition 4. *Let* $C, C' \in \mathcal{C}$, $s, s' \in \mathcal{S}$, $h, h' \in \mathcal{H}$, $O, O', L, L', D, D' \subseteq \boldsymbol{Res}$. *If* $(O, L, D) \in \mathcal{O}$ *and*

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

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

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

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

We prove by induction on the rules of $\rightarrow_p$.

The rules $(ASSIGN)$, $(IF1)$, $(IF2)$, $(SEQ1)$, $(LOOP)$, $(READ)$, (WRI), (ALL), $(FREE)$, $(PAR3)$, $(RES0)$ do not change the resource configuration. Therefore the conclusion is immediate.

If the transition is given by $(SEQ2)$. We have $C = C_1 \,;C_2$ and $C' = C_1' \,;C_2$ such that

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

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

The rules $(PAR1)$ and $(PAR2)$ are analogous to the previous rule.

If the transition is given by $(RES1)$. We have $C = \mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}$, $C' = \mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}'$, $r \notin (O, L, D)$ and $r \in \mathit{Locked}(\tilde{C})$ such that

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

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

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

From $r \notin L$, it follows that $r \notin L''$. Hence $L = L'$.

Because $r \notin (O, L, D)$, we know that

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

We still need to check that $(O', L', D') \in \mathcal{O}$. We know that $O' = O'' \setminus \{r\}$ and $D' = D'' \setminus \{r\}$. Then

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

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

The rule $(RES2)$ is analogous to the previous rule.

If the transition is given by $(WITH0)$. We have $D' = D \setminus \{r\}$, $O' = O \cup \{r\}$, $L = L'$ and $r \in D$. Then

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

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

If the transition is given by $(WITH1)$. We have $C = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}$, $C' = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}'$ and $r \in O$ such that

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

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 the next

$$\begin{gathered} O' \cap L' = (O'' \cup \{r\}) \cap L'' = (O'' \cap L'') \cup (\{r\} \cap L) = \emptyset, \\ O' \cap D' = (O'' \cup \{r\}) \cap D'' = (O'' \cap D'') \cup (\{r\} \cap D'') = \emptyset, \\ L' \cap D' = L'' \cap D'' = \emptyset. \end{gathered}$$

If the transition is given by $(WITH2)$. We have $r \in O$, $L = L'$, $D' = D \cup \{r\}$ and $O' = O \setminus \{r\}$. Consider the following

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

Then $(O', L', D') \in \mathcal{O}$. We also see that $L = L'$ and

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

□

3.5 Validity

Now, we define the *validity* of a program's specification in the operational semantic.

For $k \in \mathbb{N}_0$. We denote by $\rightarrow_p^k$ the composition of k transitions.

Definition 25. *We write $\Gamma \models \{P\}C\{Q\}$, if for every state $s \in \mathcal{S}$ and $h \in \mathcal{H}$ such that $s, h \models P * inv(\Gamma)$, we have that*

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

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

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

Next, we present some examples of specification valid in the operational semantics.

Example 8. *The following specifications of programs are valid*

- $r(x, y) : x = y \wedge \textsf{emp} \models \{\textsf{emp}\}\textsf{x:=x+1} \parallel \textsf{y:=y+1}\ \{\textsf{emp}\}$
- $\models \{x = 0\}\textsf{x:=x+1} \parallel \textsf{x:=x+1}\{x = 2\}$
- $buf(z, \textsf{full}) : R \models \{\textsf{emp}\}(\textsf{x:=cons(-)}\ ;\ PUT(x)) \parallel (GET(y)\ ;\ \textsf{dispose(y)})\{\textsf{emp}\}$,
  where $R = (\textsf{full} = 1 \wedge z \mapsto -) \vee (\textsf{full} = 0 \wedge \textsf{emp})$, $PUT(x) =$ with buf when full $= 0$ do z:=x ; full:=1 and $GET(y) =$ with buf when full $= 1$ do y:=z ; full:=0.

Note that, the first two examples are not derivable in Concurrent Separation Logic, at least without the use of additional resources and auxiliary variables. And the last specification is proved in the [4] for Concurrent Separation Logic, so this example will be a consequence of the soundness prove.

Next, we see that the transitions of commands only depends on their free variables and do not change other variables.

Proposition 5. *Let $C \in \mathcal{C}$, $X \subseteq \textbf{\textit{Var}}$, $s, s' \in \mathcal{S}$, $h \in \mathcal{H}$ and $\rho \in \mathcal{O}$ such that $FV(C) \subseteq X$.*

- *If $s(x) = s'(x)$, for every $x \in FV(C)$, and $C, (s, h, \rho) \rightarrow_p$ abort, then*

$$C, (s', h, \rho) \rightarrow_p \textsf{abort}.$$

- *If $s(x) = s'(x)$, for every $x \in X$, and $C, (s, h, \rho) \rightarrow_p C', (s_1, h', \rho')$, then there exists $s_1' \in \mathcal{S}$ such that $s_1(x) = s_1'(x)$, for every $x \in X$, and*

$$C, (s', h, \rho) \rightarrow_p C', (s_1', h', \rho').$$

Proof. The prove is done by induction on the rule of the transitions. First, the transition to the abort state.

Let $s, s' \in \mathcal{S}$, $h \in \mathcal{H}$, $\rho \in \mathcal{O}$ and $C \in \mathcal{C}$ such that $s(x) = s'(x)$, for every $x \in FV(C)$, and

$$C, (s, h, \rho) \rightarrow_p \textsf{abort}.$$

Suppose that the transition is given by $(FREEA)$. Then $C = \textsf{dispose(e)}$, $FV(C) = FV(\textsf{e})$ and

$$s(\textsf{e}) \notin dom(h).$$

Because $s(x) = s'(x)$, for every $x \in FV(\textsf{e})$, we have by proposition 1 that

$$s(\textsf{e}) = s'(\textsf{e}).$$

Therefore $s'(\mathsf{e}) \notin dom(h)$ and

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

The transitions $(READA)$ and $(WRIA)$ are similar to the previous case.

Suppose that the transition is given by $(SEQA)$. Then $C = C_1 \,;\, C_2$, $FV(C) = FV(C_1) \cup FV(C_2)$ and

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

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

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

The cases $(PAR1A)$, $(PAR2A)$, $((RESA1))$, $(RESA2)$ and $(WITHA1)$ are identical.

The transitions $(RESA)$, $(WITHA)$ and $(WITHA2)$ are independent from the storage and the conclusion is trivial

Next, we prove the second part of the proposition.

Let $X \subseteq \mathbf{Var}$, $s, s_1, s' \in \mathcal{S}$, $h, h' \in \mathcal{H}$, $\rho, \rho' \in \mathcal{O}$ and $C, C' \in \mathcal{C}$ such that $FV(C) \subseteq X$, $s(x) = s'(x)$, for every $x \in X$, and

$$C, (s, h, \rho) \to_p C', (s_1, h', \rho').$$

We'll show that there exists s'_1 such that $s_1(x) = s'_1(x)$, for every $x \in X$, and

$$C, (s', h, \rho) \to_p C', (s'_1, h', \rho').$$

The transitions given by the rules $(SEQ1)$, $(LOOP)$, $(PAR3)$, $(RES0)$ and $(WITH2)$ are independent from the storage and do not change it. Therefore, taking $s'_1 = s_1$, the conclusion is immediate.

Suppose that the transition is given by $(ASSIGN)$. Then $C = \mathsf{x{:=}e}$, $C' = \mathsf{skip}$, $s_1 = s[\mathsf{x} : v]$, $h' = h$ and $\rho' = \rho$, where $v = s(\mathsf{e})$. And $FV(C) = \{\mathsf{x}\} \cup FV(\mathsf{e})$.

From the proposition 1, we know that $s'(\mathsf{e}) = s(\mathsf{e}) = v$. Taking $s'_1 = s'[\mathsf{x} : v]$, we have that

$$C, (s', h, \rho) \to_p C', (s'_1, h', \rho').$$

Moreover, for every $y \in X$, we have the following

$$s'_1(y) = s'[\mathsf{x} : v](y) = s[\mathsf{x} : v](y) = s_1(y).$$

The cases $(IF1)$, $(IF2)$, $(READ)$, (WRI), (ALL), $(FREE)$ and $(WITH0)$ are analogous to the previous case.

Suppose that the transition is given by $(SEQ2)$. Then $C = C_1 \,;\, C_2$, $C' = C'_1 \,;\, C_2$ and

$$C_1, (s, h, \rho) \to_p C'_1, (s_1, h', \rho').$$

Because $FV(C_1) \subseteq FV(C)$, we can apply the induction hypothesis and conclude that there is s'_1 such that $s'_1(x) = s_1(x)$, for every $x \in X$ and

$$C_1, (s', h, \rho) \to_p C'_1, (s'_1, h', \rho').$$

Applying the $(SEQ2)$ rule, we conclude that

$$C, (s', h, \rho) \to_p C', (s'_1, h', \rho').$$

The cases $(RES1)$, $(RES2)$, $(WITH1)$, $(PAR1)$ and $(PAR2)$ are analogous to the previous case. □

3.6 Safety monotonicity and Frame Property

Next, we prove the safety monotonicity and the frame property for the operational semantic presented here. In the work of Separation Logic, this results were introduced to prove the soundness of the frame rule.

Proposition 6. *Let $C \in \mathcal{C}$, $s \in \mathcal{S}$, $h, h_F \in \mathcal{H}$ and $\rho \in \mathcal{O}$ such that $h \perp h_F$. If $C, (s, h, \rho) \not\rightarrow_p$ abort then*

$$C, (s, h \uplus h_F, \rho) \not\rightarrow_p \textit{abort}.$$

Proof. Let $C \in \mathcal{C}$, $s \in \mathcal{S}$, $h, h_F \in \mathcal{H}$ and $\rho \in \mathcal{O}$ such that $h \perp h_F$ and

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

We'll prove the proposition by induction on the relation, $\rightarrow_p$.

Suppose that the transition to the abort state is given by $(RESA)$, $(WITHA)$ or $(WITHA2)$. Then the transitions does not depended on the heap and we have that

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

Suppose that is given by $(FREEA)$, $(READA)$ or $(WRIA)$. Then $s(\mathsf{e}) \notin dom(h \uplus h_F)$.
It follows that $s(\mathsf{e}) \notin dom(h)$ and

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

Suppose that is given by $(SEQA)$ or $(PAR1A)$. Then $C = C_1 \,;\, C_2$ or $C = C_1 \parallel C_2$ and

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

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

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

The remaining cases $(PAR2A)$, $(RESA1)$, $(RESA2)$ and $(WITHA1)$ are similar to the previous case. □

The frame property is stated in the following proposition.

Proposition 7. *Let $C, C' \in \mathcal{C}$, $s, s' \in \mathcal{S}$, $h, h', h_F \in \mathcal{H}$ and $\rho, \rho' \in \mathcal{O}$ such that $h \perp h_F$.*
If $C, (s, h, \rho') \not\rightarrow_p$ abort and $C, (s, h \uplus h_F, \rho) \rightarrow_p C', (s', h', \rho')$, then $h_F \subseteq h'$ and

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

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

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

We prove the proposition by induction on the rules of $\rightarrow_p$. Next, we consider the different transition rules.

Suppose that the transition is given by $(ASSIGN)$.
We have $C = \mathsf{x{:=}e}$, $C' = \mathsf{skip}$, $s(\mathsf{e}) = v$, $s' = s[\mathsf{x} : v]$, $h' = h \uplus h_F$ and $\rho' = \rho$.
Then $h_F \subseteq h'$ and $h' \setminus h_F = h$.
Using the $(ASSIGN)$ rule, we get that

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

Suppose that the transition is given by $(IF1)$.
We have $C = \mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2$, $C' = C_1$, $s(B) = \mathtt{true}$, $s' = s$, $h' = h \uplus h_F$ and $\rho' = \rho$.

Then $h_F \subseteq h'$ and $h' \setminus h_F = h$.
The rule $(IF1)$ gives that

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

The case $(IF2)$ is analogous to the previous case.

Suppose that the transition is given by $(SEQ1)$.
We have $C = \mathsf{skip}\,;C_2$, $C' = C_2$, $s' = s$, $h' = h \uplus h_F$ and $\rho' = \rho$.
Then $h_F \subseteq h'$ and $h' \setminus h_F = h$.
By $(SEQ1)$, we get that

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

Suppose that the transition is given by $(SEQ2)$.
We have $C = C_1\,;C_2$ 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

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

Suppose that the transition is given by $(READ)$.
We have $C = \mathsf{x{:=}[e]}$, $C' = \mathsf{skip}$, $s(\mathsf{e}) \in dom(h \uplus h_F)$, $(h \uplus h_F)(s(\mathsf{e})) = v$, $s' = s[\mathsf{x} : v]$, $h' = h \uplus h_F$ and $\rho' = \rho$.
Then $h_F \subseteq h'$ and $h' \setminus h_F = h$.
If $s(\mathsf{e}) \notin dom(h)$, then $C, (s, h, \rho) \rightarrow \mathsf{abort}$. Hence

$$s(\mathsf{e}) \in dom(h).$$

From the previous fact and $(h \uplus h_F)(s(\mathsf{e})) = v$, we know that $h(s(\mathsf{e})) = v$. Applying the rule $(READ)$, we conclude that

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

Suppose that the transition is given by (WRI).
We have $C = \mathsf{[e]{:=}e'}$, $C' = \mathsf{skip}$, $s(\mathsf{e}) = l \in dom(h \uplus h_F)$, $s(\mathsf{e'}) = v$, $s' = s$, $h' = (h \uplus h_F)[l : v]$ and $\rho' = \rho$.
As before, we know that $s(\mathsf{e}) \in dom(h)$. Hence, we rewrite the heap in the following expression

$$h' = h[l : v] \uplus h_F.$$

Then $h_F \subseteq h'$ and $h' \setminus h_F = h[l : v]$.
From $s(\mathsf{e}) = l \in dom(h)$, $s(\mathsf{e'}) = v$ and the rule (WRI), we conclude that

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

Suppose that the transition is given by $(FREE)$.
We have $C = \mathsf{dispose(e)}$, $C' = \mathsf{skip}$, $s(\mathsf{e}) = l \in dom(h \uplus h_F)$, $s' = s$, $h' = (h \uplus h_F) \setminus \{l\}$ and $\rho' = \rho$.
Like in the previous cases, we know that $s(\mathsf{e}) \in dom(h)$. Hence

$$h' = (h \setminus \{l\}) \uplus h_F.$$

Then $h_F \subseteq h'$ and $h' \setminus h_F = h \setminus \{l\}$.

Moreover, we have that

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

Suppose that the transition is given by (ALL).

We have $C = \mathsf{x{:=}cons(e)}$, $C' = \mathsf{skip}$, $l \notin dom(h \uplus h_F)$, $s(\mathsf{e}) = v$, $s' = s[\mathsf{x} : l]$, $h' = (h \uplus h_F)[l : v]$ and $\rho' = \rho$.

We have $l \notin dom(h)$, because $l \notin dom(h \uplus h_F)$. Note that, we can rewrite the heap in the expression

$$h' = h[l : v] \uplus h_F.$$

Then $h_F \subseteq h'$ and $h' \setminus h_F = h[l : v]$.

From $l \notin dom(h)$, we can apply the rule (ALL) and obtain that

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

The cases $(PAR1)$, $(PAR2)$, $(RES1)$, $(RES2)$ and $(WITH1)$ are similar to the case $(SEQ2)$.

The cases $(LOOP)$, $(PAR3)$, $(RES0)$, $(WITH0)$ and $(WITH2)$ are similar to the case $(SEQ1)$. □

Next, we prove that adding some of the resources locked by the environment to the resources owned by the execution do not introduce new transition to abort. This result will be used to prove the soundness of the parallel rule. This result can be seen as an analogous to the safety monotonicity with respect to resource configurations.

Proposition 8. *Let $C \in \mathcal{C}$, $s \in \mathcal{S}$, $h \in \mathcal{H}$ and $(O_1 \cup O_2, L, D), (O_1, L \cup O_2, D) \in \mathcal{O}$.*
If $C, (s, h, (O_1, L \cup O_2, D)) \not\rightarrow_p$ abort, then

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

Proof. We'll prove the contra-position by induction on the rules of $\rightarrow_p$.

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

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

If the transition to the abort state is given by $(FREEA)$, $(READA)$ or $(WRIA)$.

Note that the transition is independent from the resource configuration. Then

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

If the transition to the abort state is given by $(SEQA)$. We have

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

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

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

The cases $(PAR1A)$ and $(PAR2A)$ are similar to the previous case.

If the transition to the abort state is given by $(RESA)$ or $(WITHA)$.

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 to the abort state is given by $(RESA1)$. We have $C = \mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}$, $r \in Locked(\tilde{C})$ and

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

By induction hypothesis, we have that $\tilde{C}, (s, h, (O_1 \cup \{r\}, L \cup O_2, D)) \rightarrow_p \mathsf{abort}$. Hence

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

The case $(RESA2)$ is analogous to the previous case

If the transition to the abort state is given by $(WITHA1)$. We have $C = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}$ and

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

By induction hypothesis, we have that $\tilde{C}, (s, h, (O_1, L \cup O_2, D) \setminus \{r\}) \to_p \mathsf{abort}$. Therefore

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

If the transition to the abort state is given by $(WITHA2)$. We have $r \notin O_1 \cup O_2$. Then $r \notin O_1$ and

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

□

In the next proposition, we see that passing some resources from the set of owned resources to the set of locked resources do not affect the execution of program, providing that the execution do not abort for the smaller set of owned resources. This result can be seen as the analogous to the frame property for resource configurations.

Proposition 9. *Let $C, C' \in \mathcal{C}$, $s, s' \in \mathcal{S}$, $h, h' \in \mathcal{H}$ and $(O_1 \cup O_2, L, D), (O_1, L \cup O_2, D) \in \mathcal{O}$. If $C, (s, h, (O_1, L \cup O_2, D)) \not\to_p$ abort and $C, (s, h, (O_1 \cup O_2, L, D)) \to_p C', (s', h', (O', L, D'))$, then $O_2 \subseteq O'$ and*

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

Proof. Let $C, C', s, s', h, h', O_1, L, O_2, D, O', D'$ such 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 rules of transition above.

If the transition is given by one of the following rules: $(ASSIGN)$, $(IF1)$, $(IF2)$, $(SEQ1)$, $(LOOP)$, $(READ)$, (WRI), $(FREE)$, (ALL) or $(PAR3)$. Note that the transitions do not depend on the resource configuration. Then, the conclusion is immediate.

If the transition is given by $(WITH0)$.

We have $C = \mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}$, $C' = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}$, $r \in D$, $s(B) = \mathtt{true}$, $s' = s$, $h' = h$, $O' = O_1 \cup O_2 \cup \{r\}$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 $(WITH1)$.

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

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

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

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

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

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

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

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

If the transition is given by $(WITH2)$.

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

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

If the transition is given by $(RES0)$.

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

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

If the transition is given by $(RES1)$.

We have $C = \mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}$, $C' = \mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}'$, $r \notin (O_1 \cup O_2, L, D)$, $r \in Locked(\tilde{C})$ and

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

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

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

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

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

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

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

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

The case $(RES2)$ is similar to the previous case.

If the transition is given by $(SEQ2)$.

We have $C = C_1 \,;\, C_2$, $C' = C_1' \,;\, C_2$ and

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

We have $C_1, (s, h, (O_1, L \cup O_2, D)) \not\to_p \mathsf{abort}$, because $C, (s, h, (O_1, L \cup O_2, D)) \not\to_p \mathsf{abort}$.

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

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

Therefore

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

The cases $(PAR1)$ and $(PAR2)$ are similar to the previous case. □

To study the renaming rule we define the rename of resources inside a resource context.

Definition 26. *Let $r, r' \in \boldsymbol{Res}$ and $\rho = (O, L, D) \in \mathcal{O}$ such that $r' \notin \rho$. The substitution of r by r' on ρ is denoted by $\rho[r'/r]$, and it is given by*

$$\rho[r'/r] = (O[r'/r], L[r'/r], D[r'/r]),$$

where $A[r'/r] = \{\hat{r}[r'/r] : \hat{r} \in A\}$.

The next proposition ensures that renaming resources do not introduce transitions to abort.

Proposition 10. *Let $C \in \mathcal{C}$, $s \in \mathcal{S}$, $h \in \mathcal{H}$, $\rho \in \mathcal{O}$ and $r, r' \in \textbf{\textit{Res}}$ such that $r' \notin Res(C)$ and $r' \notin \rho$. If $C[r'/r], (s, h, \rho[r'/r]) \not\to_p$ abort then*

$$C, (s, h, \rho) \not\to_p \textit{abort}.$$

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

We will prove, by induction on $\to_p$, that if $C, (s, h, \rho) \to_p \mathsf{abort}$, then

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

Suppose that the transition to the abort state is given by $(FREEA)$, $(READA)$ or $(WRIA)$. Note that the transitions is independent from the resource configuration and $C[r'/r] = C$. Then

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

Suppose that the transition is given by $(SEQA)$. Then $C = C_1 \,;\, C_2$ and

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

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

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

Hence

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

The cases $(PAR1)$, $(PAR2)$ are identical to the previous case.

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

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

The cases $(WITHA)$ and $(WITHA2)$ are analogous to the case before.

Suppose that the transition is given by $(RESA1)$. 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

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

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

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

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

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

Therefore

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

The case $(RESA2)$ is analogous to the previous case.

Suppose that the transition is given by $(WITHA1)$. We have $C = \mathsf{within}\ \hat{r}\ \mathsf{do}\ \tilde{C}$ and

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

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

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

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

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

□

In the next propositions, we see that the auxiliary variables do not influence executions. First, we define the number of assignment to auxiliary variables inside a command.

Definition 27. *Let $C \in \mathcal{C}$ and $X \subseteq$ **Var**, such that X is a set of auxiliary variables for C. We denote by $l(C)$ the number of assignments to auxiliary variables, and it is defined inductively by:*

- *if* $x \in X$, *then* $l(x{:=}e) = 1$,
- $l(C_1 \,;\, C_2) = l(\textsf{if } B \textsf{ then } C_1 \textsf{ else } C_2) = l(C_1 \parallel C_2) = l(C_1) + l(C_2)$,
- $l(\textsf{while } B \textsf{ do } C) = l(\textsf{resource } r \textsf{ in } C) = l(\textsf{with } r \textsf{ when } B \textsf{ do } C) = l(C)$,
- $l(C) = 0$, *otherwise.*

In the proposition below, we see that we can suppress the auxiliary variables from a command and there is not new transitions to abort.

Proposition 11. *Let $C \in \mathcal{C}$, $s \in \mathcal{S}$, $h \in \mathcal{H}$, $\rho \in \mathcal{O}$ and $X \subseteq$ **Var** such that X is a set of auxiliary variables for C. If $C, (s, h, \rho) \not\to_p$ abort, then*

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

Proof. We prove the proposition by induction on the structure of C.

Let $C \in \mathcal{C}$, $s \in \mathcal{S}$, $h \in \mathcal{H}$, $\rho \in \mathcal{O}$, $X \subseteq \mathbf{Var}$ such that X is a set of auxiliary variables for C and

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

Let $C = \textsf{skip}$. We have $C \setminus X = \textsf{skip}$ and

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

Let $C = \textsf{x:=e}$. Then $C \setminus X = \textsf{skip}$ or $C \setminus X = C$. In both cases, we have that

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

Let $C = \textsf{x:=[e]} \mid \textsf{x:=cons(e)} \mid \textsf{dispose(e)}$. We have $C \setminus X = C$ and

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

Let $C = \textsf{if } B \textsf{ then } C_1 \textsf{ else } C_2$. Then $C \setminus X = \textsf{if } B \textsf{ then } C_1 \setminus X \textsf{ else } C_2 \setminus X$ and

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

Let $C = C_1 \,;\, C_2$. We have $C \setminus X = C_1 \setminus X \,;\, C_2 \setminus X$ and $C_1, (s, h, \rho) \not\to_p \textsf{abort}$.
Note that X is a set of auxiliary variables for C_1. By induction hypothesis, we obtain that

$$C_1 \setminus X, (s, h, \rho) \not\to_p \textsf{abort}.$$

Therefore

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

Let $C = \textsf{while } B \textsf{ do } \tilde{C}$. We have $C \setminus X = \textsf{while } B \textsf{ do } \tilde{C} \setminus X$ and

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

Let $C = C_1 \parallel C_2$. We have $C \setminus X = C_1 \setminus X \parallel C_2 \setminus X$,

$$C_1, (s, h, \rho) \not\to_p \textsf{abort} \quad \text{and} \quad C_2, (s, h, \rho) \not\to_p \textsf{abort}.$$

Note that X is a set of auxiliary variables for C_1 and C_2. By induction hypothesis we have that

$$C_1 \setminus X, (s, h, \rho) \not\to_p \textsf{abort} \quad \text{and} \quad C_2 \setminus X, (s, h, \rho) \not\to_p \textsf{abort}.$$

Therefore

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

Let $C = \mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}$. We have $C \setminus X = \mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C} \setminus X$ and $r \in \rho$. Therefore

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

Let $C = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}$. We have $C \setminus X = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C} \setminus X$, $r \in O$ and

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

Note that X is a set of auxiliary variables for $\tilde{C}$. Using the induction hypothesis, we conclude that

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

Therefore

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

Let $C = \mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}$. We have $C \setminus X = \mathsf{resource}\ r\ \mathsf{in}\ \tilde{C} \setminus X$, $r \notin \rho$,

- if $r \in \mathit{Locked}(\tilde{C})$, then $\tilde{C}, (s, h, (O \cup \{r\}, L, D)) \not\rightarrow_p \mathsf{abort}$, and
- if $r \notin \mathit{Locked}(\tilde{C})$, then $\tilde{C}, (s, h, (O, L, D \cup \{r\})) \not\rightarrow_p \mathsf{abort}$.

Note that X is a set of auxiliary variables for $\tilde{C}$. By induction hypothesis, we have that

- if $r \in \mathit{Locked}(\tilde{C})$, then $\tilde{C} \setminus X, (s, h, (O \cup \{r\}, L, D)) \not\rightarrow_p \mathsf{abort}$, and
- if $r \notin \mathit{Locked}(\tilde{C})$, then $\tilde{C} \setminus X, (s, h, (O, L, D \cup \{r\})) \not\rightarrow_p \mathsf{abort}$.

It is easy to see that $\mathit{Locked}(\tilde{C} \setminus X) = \mathit{Locked}(\tilde{C})$. Therefore

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

□

In the next proposition, we reinforce the idea that the suppression of auxiliary variables do not change the executions. Moreover, we prove that only the set of auxiliary variables is different in an execution with auxiliary variables and an execution without auxiliary variables.

Proposition 12. *Let $C, C' \in \mathcal{C}$, $s, s' \in \mathcal{S}$, $h, h' \in \mathcal{H}$, $\rho, \rho' \in \mathcal{O}$, $l \in \mathbb{N}_0$ and $X \subseteq$ **Var** such that X is a set of auxiliary variables for C and $l = l(C)$. If $C \setminus X, (s, h, \rho) \rightarrow_p C', (s', h', \rho')$, then there exist C'', s'', l'' and $k \leq l + 1$ such that*

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

where $s''(y) = s'(y)$, for every $y \notin X$, $C' = C'' \setminus X$, $l'' = l(C'')$ and $l'' \leq 2l - (k - 1)$.

Proof. We prove the proposition by induction on the rules of $\rightarrow_p$.

Let $C, C' \in \mathcal{C}$, $s, s' \in \mathcal{S}$, $h, h' \in \mathcal{H}$, $\rho, \rho' \in \mathcal{O}$, $l \in \mathbb{N}_0$ and $X \subseteq$ **Var** such that X is a set of auxiliary variables for C, $l = l(C)$ and

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

Suppose that the transition is given by $(ASSIGN)$, (WRI), $(FREE)$, (WRI) or (ALL). Then $C = C \setminus X$ and $C' = \mathsf{skip}$.

Considering $C'' = \mathsf{skip}$, $s'' = s'$, $l'' = 0$ and $k = 1$. We have the following transition

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

Moreover, we have $s''(y) = s'(y)$, for every $y \notin X$, $C' = C'' \setminus X$, $l'' = l(C'')$ and $l'' \leq 2l - (k - 1)$.

Suppose that the transition is given by $(IF1)$. Then $C = \mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2$, $C' = C_1 \setminus X$ and $l = l_1 + l_2$, where $l_i = l(C_i)$, for $i = 1, 2$.

Considering $C'' = C_1$, $s'' = s'$, $l'' = l_1$ and $k = 1$. We have the following transition

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

Moreover, we have $s''(y) = s'(y)$, for every $y \notin X$, $C' = C'' \setminus X$, $l'' = l_1 = l(C_1) = l(C'')$ and $l'' = l_1 \leq 2(l_1 + l_2) - (k-1) = 2l - (k-1)$.

If the transition is given by $(IF2)$, the proof is analogous to the previous case.

Suppose that the transition is given by $(LOOP)$. Then $C = \mathsf{while}\ B\ \mathsf{do}\ \tilde{C}$, $C' = \mathsf{if}\ B\ \mathsf{then}\ \tilde{C} \setminus X\ ;\ \mathsf{while}\ B\ \mathsf{do}\ \tilde{C} \setminus X\ \mathsf{else\ skip}$.

Considering $C'' = \mathsf{if}\ B\ \mathsf{then}\ \tilde{C}\ ;\ \mathsf{while}\ B\ \mathsf{do}\ \tilde{C}\ \mathsf{else\ skip}$, $s'' = s'$, $l'' = 2l$ and $k = 1$. We have the following transition

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

Moreover, we have $s''(y) = s'(y)$, for every $y \notin X$, $C' = C'' \setminus X$, $l'' = 2l = 2l(\tilde{C}) = l(C'')$ and $l'' = 2l \leq 2l - (k-1)$.

Suppose that the transition is given by $(SEQ2)$. Then $C = C_1\ ;\ C_2$, $C' = C_1'\ ;\ C_2 \setminus X$ and $l = l_1 + l_2$ such that $l_i = l(C_i)$, for $i = 1, 2$, and

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

Note that X is a set of auxiliary variables for C_1. By induction hypothesis we know that there are C_1'', s_1'', l_1'' and $k_1 \leq l_1 + 1$ such that

$$C_1, (s, h, \rho) \rightarrow_p^{k_1} C_1'', (s_1'', h', \rho'),$$

where $s_1''(y) = s'(y)$, for every $y \notin X$, $C_1' = C_1'' \setminus X$, $l_1'' = l(C_1'')$ and $l_1'' = 2l_1 - (k-1)$.

Considering $C'' = C_1''\ ;\ C_2$, $k = k_1 \leq l + 1$, $s'' = s_1''$ and $l'' = l_1'' + l_2$. Applying the rule $(SEQ2)$ to the transitions above, we have the following transitions

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

Moreover, we have $s''(y) = s_1''(y) = s'(y)$, for every $y \notin X$, $C' = C_1'\ ;\ C_2 \setminus X = C_1'' \setminus X\ ;\ C_2 \setminus X = C'' \setminus X$, $l'' = l_1'' + l_2 = l(C_1'') + l(C_2) = l(C'')$ and $l'' = l_1'' + l_2 \leq 2l_1 + 2l_2 - (k-1) = 2l - (k-1)$.

The cases $(PAR1)$, $(PAR2)$, $(RES1)$, $(RES2)$ and $(WITH1)$ are similar to the previous case.

Suppose that the transition is given by $(WITH0)$. We have $C = \mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}$ and $C' = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C} \setminus X$.

Considering $C'' = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}$, $k = 1$, $s'' = s'$ and $l'' = l$. We have the following transition

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

Moreover, we have $s''(y) = s'(y)$, for every $y \notin X$, $C' = C'' \setminus X$, $l'' = l(\tilde{C}) = l(C'')$ and $l'' \leq 2l - (k-1)$.

Before we finish the proof of the proposition, we prove the following lemma.

Lemma 1. *Let $C \in \mathcal{C}$, $s \in \mathcal{S}$, $h \in \mathcal{H}$, $\rho \in \mathcal{O}$ and X a set of auxiliary variables for C. If $C \setminus X = \mathsf{skip}$, then there exists s' such that $s'(y) = s(y)$, for every $y \notin X$, and*

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

Proof. From $C \setminus X = \mathsf{skip}$, it follows that $C = \mathsf{skip}$ or $C = \mathsf{x{:=}e}$, for some $\mathsf{x} \in X$.

Suppose that $C = \mathsf{skip}$. Then $l(C) = 0$. The conclusion is immediate, by taking $s' = s$.

Suppose that $C = \mathsf{x{:=}e}$, for some $\mathsf{x} \in X$. Then $l(C) = 1$. Consider $s' = s[\mathsf{x} : v]$, where $v = s(\mathsf{e})$.

We have $s'(y) = s(y)$, for every $y \notin X$, and

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

□

Suppose that the transition is given by $(SEQ1)$. Then $C = C_1 \,;\, C_2$, $C_1 \setminus X = \mathsf{skip}$, $C' = C_2 \setminus X$, $s' = s$, $h' = h$ and $\rho' = \rho$.

Using the transition of the lemma above together with the rule $(SEQ2)$, and the rule $(SEQ1)$, we know that there is s'' such that $s''(y) = s(y)$, for every $y \notin X$, and

$$C, (s, h, \rho) \to_p^{l(C_1)} \mathsf{skip} \,;\, C_2, (s'', h, \rho) \to_p C_2, (s'', h, \rho).$$

Considering $C'' = C_2$, $k = l(C_1) + 1$ and $l'' = l(C_2)$. We have

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

Moreover, we have $s''(y) = s(y) = s'(y)$, for every $y \notin X$, $C' = C'' \setminus X$, $l'' = l(C'')$ and $l'' = l(C_2) \leq l(C_1) + 2l(C_2) = 2l - (k-1)$.

Suppose that the transition is given by $(PAR3)$. Then $C = C_1 \parallel C_2$, $C_1 \setminus X = C_2 \setminus X = \mathsf{skip}$, $C' = \mathsf{skip}$, $s' = s$, $h' = h$ and $\rho' = \rho$.

From the lemma above we know that exist transition for the commands C_1 and C_2. Applying the rules $(PAR1)$ and $(PAR2)$ to the transitions obtained from the lemma, then the rule $(PAR3)$ we know that there is s'' such that $s'(y) = s(y)$, for every $y \notin X$, and

$$C, (s, h, \rho) \to_p^{l(C_1)+l(C_2)} \mathsf{skip} \parallel \mathsf{skip}, (s'', h, \rho) \to_p \mathsf{skip}, (s'', h, \rho).$$

Considering $C'' = \mathsf{skip}$, $k = l(C) + 1$ and $l'' = 0$. We have the following transitions

$$C, (s, h, \rho) \to_p^k \mathsf{skip}, (s'', h, \rho).$$

Moreover, we have $s''(y) = s(y) = s'(y)$, for every $y \notin X$, $C' = C'' \setminus X$, $l'' = l(C'')$ and $l'' \leq 2l - (k-1)$.

Using the lemma above, the cases $(WITH2)$ and $(RES0)$ are analogous to the previous cases. □

3.7 Reachable commands

In the extended programming language, there is some command that can not appear in any execution of commands considered by Concurrent Separation Logic. We start this section by defining the reachable commands.

Definition 28. *We say that $C \in \mathcal{C}$ is* reachable*, if there exist $C' \in \boldsymbol{Com}$, $k \geq 0$, $s, s' \in \mathcal{S}$, $h, h' \in \mathcal{H}$ and $\rho, \rho' \in \mathcal{O}$ such that*

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

and $C', (s', h', \rho') \not\to_p^j \mathsf{abort}$, for every $j \leq k$.

Next, we give some examples of reachable commands, and we illustrate some difficulties to handle the definition above. This difficulties motivate the subsequent study done in this section.

Example 9.

- *If $C \in \boldsymbol{Com}$, then it is reachable.*
- *$\mathsf{within}\ r\ \mathsf{do}\ \mathsf{skip}$ is a reachable command, because*

  $$\mathsf{with}\ r\ \mathsf{do}\ \mathsf{skip}, (s, h, (\emptyset, \emptyset, \{r\})) \to_p \mathsf{within}\ r\ \mathsf{do}\ \mathsf{skip}, (s, h, (\{r\}, \emptyset, \emptyset)),$$

  and $\mathsf{with}\ r\ \mathsf{do}\ \mathsf{skip} \in \boldsymbol{Com}$.
- *Intuitively, the command $\mathsf{within}\ r\ \mathsf{do}\ \mathsf{within}\ r\ \mathsf{do}\ \mathsf{skip}$ should not be reachable. However to prove this we need to consider all the possible executions. Next, we introduce some notions and results that will help us to check which commands are not reachable.*

As the example above show the definition of reachable is not very suitable to use. To overcome this difficulty we introduce, as done in [16], the notion of user commands and well-formed commands.

The set of *user commands* is represent by the following function.

Definition 29. *The function* $usr\_cmd : \mathcal{C} \to \{\mathtt{true}, \mathtt{false}\}$ *is defined by:*

- $usr\_cmd(C_1 \,;\, C_2) = usr\_cmd(C_1) \wedge usr\_cmd(C_2)$,
- $usr\_cmd(\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2) = usr\_cmd(C_1) \wedge usr\_cmd(C_2)$,
- $usr\_cmd(\mathsf{while}\ B\ \mathsf{do}\ C) = usr\_cmd(C)$,
- $usr\_cmd(\mathsf{resource}\ r\ \mathsf{in}\ C) = usr\_cmd(C)$,
- $usr\_cmd(\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ C) = usr\_cmd(C)$
- $usr\_cmd(C_1 \parallel C_2) = usr\_cmd(C_1) \wedge usr\_cmd(C_2)$,
- $usr\_cmd(\mathsf{within}\ r\ \mathsf{do}\ C) = \mathtt{false}$,
- $usr\_cmd(\alpha) = \mathtt{true}$, *otherwise.*

It is easy to check that $usr\_cmd(C) = \mathtt{true}$ if and only if $C \in \mathbf{Com}$.

The set of *well formed* commands is represent by the following function.

Definition 30. *The function* $wf\_cmd : \mathcal{C} \to \{\mathtt{true}, \mathtt{false}\}$ *is defined by:*

- $wf\_cmd(C_1 \,;\, C_2) = wf\_cmd(C_1) \wedge usr\_cmd(C_2)$,
- $wf\_cmd(\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2) = usr\_cmd(C_1) \wedge usr\_cmd(C_2)$,
- $wf\_cmd(\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ C) = wf\_cmd(\mathsf{while}\ B\ \mathsf{do}\ C) = usr\_cmd(C)$,
- $wf\_cmd(\mathsf{resource}\ r\ \mathsf{in}\ C) = wf\_cmd(C)$,
- $wf\_cmd(C_1 \parallel C_2) = wf\_cmd(C_1) \wedge wf\_cmd(C_2) \wedge (Locked(C_1) \cap Locked(C_2) = \emptyset)$,
- $wf\_cmd(\mathsf{within}\ r\ \mathsf{do}\ C) = wf\_cmd(C) \wedge r \notin Locked(C)$,
- $wf\_cmd(C) = \mathtt{true}$, *otherwise.*

We write $C \in wf\_cmd$, if $wf\_cmd(C) = \mathtt{true}$.

In the following result we will see that a reachable command is a well-formed command. First, we see that the user commands are well-formed.

Proposition 13. *Let* $C \in \mathcal{C}$. *If* $usr\_cmd(C) = \mathtt{true}$, *then* $C \in wf\_cmd$ *and* $Locked(C) = \emptyset$.

Proof. Prove by induction on the structure of C.

Let $C \in \mathcal{C}$ such that $usr\_cmd(C) = \mathtt{true}$.
Let $C = \mathsf{skip} \mid \mathsf{x{:=}e} \mid \mathsf{x{:=}[e]} \mid \mathsf{[e]{:=}e'} \mid \mathsf{x{:=}cons(e)} \mid \mathsf{dispose(e)}$.
We have that $C \in wf\_cmd$ and $Locked(C) = \emptyset$.

Let $C = \mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2$. Then $usr\_cmd(C_1) = \mathtt{true}$ and $usr\_cmd(C_2) = \mathtt{true}$.
By the definition $C \in wf\_cmd$ and $Locked(C) = \emptyset$.

Let $C = C_1 \,;\, C_2$. Then $usr\_cmd(C_1) = \mathtt{true}$ and $usr\_cmd(C_2) = \mathtt{true}$.
Applying the induction hypothesis, we get that $C_1 \in wf\_cmd$ and $Locked(C_1) = \emptyset$.
Therefore $C \in wf\_cmd$ and $Locked(C) = \emptyset$.

Let $C = \mathsf{while}\ B\ \mathsf{do}\ \tilde{C} \mid \mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}$. Then $usr\_cmd(\tilde{C}) = \mathtt{true}$.
By the definition $C \in wf\_cmd$ and $Locked(C) = \emptyset$.

Let $C = \mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}$. Then $usr\_cmd(\tilde{C}) = \mathtt{true}$.
Applying the induction hypothesis, we get that $\tilde{C} \in wf\_cmd$ and $Locked(\tilde{C}) = \emptyset$.
Therefore $C \in wf\_cmd$ and $Locked(C) = \emptyset$.

Let $C = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}$. Then $usr\_cmd(C) = \mathtt{false}$.

Let $C = C_1 \parallel C_2$. Then $usr\_cmd(C_1) = \mathtt{true}$ and $usr\_cmd(C_2) = \mathtt{true}$.
By the induction hypothesis $C_1 \in wf\_cmd$, $C_2 \in wf\_cmd$, $Locked(C_1) = \emptyset$ and $Locked(C_2) = \emptyset$.
Therefore $C \in wf\_cmd$ and $Locked(C) = \emptyset$. □

Next, we see that the transitions on the operational semantics preserve the well-form of commands.

Proposition 14. *Let $C, C' \in \mathcal{C}$, $s, s' \in \mathcal{S}$, $h, h' \in \mathcal{H}$ and $\rho, \rho' \in \mathcal{O}$. If $C \in wf\_cmd$, $C, (s, h, \rho) \not\rightarrow_p$ abort and $C, (s, h, \rho) \rightarrow_p C', (s', h', \rho')$, then $C' \in wf\_cmd$.*

Proof. Prove by induction on the rules of $\rightarrow_p$:
Let $C, C' \in \mathcal{C}$, $s, s' \in \mathcal{S}$, $h, h' \in \mathcal{H}$ and $\rho, \rho' \in \mathcal{O}$ such that $C \in wf\_cmd$, $C, (s, h, \rho) \not\rightarrow_p \mathsf{abort}$ and

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

Suppose that the transition is given by $(ASSIGN)$, $(READ)$, (WRI), (ALL), $(FREE)$, $(RES0)$, $(WITH2)$ or $(PAR3)$. We have $C' = \mathsf{skip}$. Then $C' \in wf\_cmd$.

Suppose that the transition is given by $(IF1)$. We have $C = \mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2$, $C' = C_1$.
From $C \in wf\_cmd$, we know that $user\_cmd(C') = \mathtt{true}$.
Therefore by proposition 13, we have that $C' \in wf\_cmd$.

The cases $(IF2)$ and $(SEQ1)$ are analogous to the previous case.

Suppose that the transition is given by $(SEQ2)$. We have $C = C_1 \,;\, C_2$, $C' = C_1' \,;\, C_2$ and

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

From $C \in wf\_cmd$, we know that $C_1 \in wf\_cmd$ and $user\_cmd(C_2) = \mathtt{true}$.
From $C, (s, h, \rho) \not\rightarrow_p \mathsf{abort}$, we also know that

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

Applying the induction hypothesis, we get that $C_1' \in wf\_cmd$. Therefore $C' \in wf\_cmd$.

Suppose that the transition is given by $(LOOP)$.
We have $C = \mathsf{while}\ B\ \mathsf{do}\ \tilde{C}$ and $C' = \mathsf{if}\ B\ \mathsf{then}\ \tilde{C}\ ;\ \mathsf{while}\ B\ \mathsf{do}\ \tilde{C}\ \mathsf{else}\ \mathsf{skip}$.
From $C \in wf\_cmd$, we know that $user\_cmd(\tilde{C}) = \mathtt{true}$. Therefore

$$\begin{aligned} wf\_cmd(C') &= usr\_cmd(\tilde{C};\ while\ B\ do\ \tilde{C}\ ) \wedge usr\_cmd(\mathsf{skip}) \\ &= usr\_cmd(\tilde{C}) \wedge usr\_cmd(while\ B\ do\ \tilde{C}) \\ &= usr\_cmd(\tilde{C}) = \mathtt{true}. \end{aligned}$$

Suppose that the transition is given by $(RES1)$. We have $r \notin \rho$, $C = \mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}$, $C' = \mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}'$, $r \in Locked(\tilde{C})$ and

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

where $\rho'' \setminus \{r\} = \rho'$. From $C \in wf\_cmd$, we know that $\tilde{C} \in wf\_cmd$.
We also know that $\tilde{C}, (s, h, (O \cup \{r\}, L, D)) \not\rightarrow_p \mathsf{abort}$, because $C, (s, h, \rho) \not\rightarrow_p \mathsf{abort}$.
Applying the induction hypothesis, we obtain that $\tilde{C}' \in wf\_cmd$. Therefore $C' \in wf\_cmd$.

The case $(RES2)$ is analogous to the previous case.

Suppose that the transition is given by $(WITH0)$. We have $C = \mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}$ and $C' = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}$.
From $C \in wf\_cmd$, it follows $user\_cmd(\tilde{C}) = \mathtt{true}$.
By proposition 13, we know that $\tilde{C} \in wf\_cmd$ and $Locked(\tilde{C}) = \emptyset$. Therefore $C' \in wf\_cmd$.

Suppose that the transition is given by $(WITH1)$. We have $r \in O \cap O'$, $C = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}$, $C' = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}'$ and

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

From $C \in wf\_cmd$, we know that $\tilde{C} \in wf\_cmd$ and $r \notin Locked(\tilde{C})$.

We also know that $\tilde{C}, (s, h, (O \setminus \{r\}, L, D)) \not\rightarrow_p \mathsf{abort}$, because $C, (s, h, (O, L, D)) \rightarrow_p \mathsf{abort}$.

Applying the induction hypothesis, we conclude that $\tilde{C}' \in wf\_cmd$.

To conclude that $C' \in wf\_cmd$, we need to check that $r \notin Locked(\tilde{C}')$.

Having in mind that $r \notin (O \setminus \{r\}, L, D))$, we conclude that the resource r is not acquired by the transition $\tilde{C}, (s, h, (O \setminus \{r\}, L, D)) \rightarrow_p \tilde{C}', (s', h', (O' \setminus \{r\}, L', D'))$.

Hence $r \notin Locked(\tilde{C}')$ and $C' \in wf\_cmd$.

Suppose that the transition is given by $(PAR1)$. We have $C = C_1 \parallel C_2$, $C' = C_1' \parallel C_2$ and

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

From $C \in wf\_cmd$, we know that $C_1 \in wf\_cmd$, $C_2 \in wf\_cmd$ and $Locked(C_1) \cap Locked(C_2) = \emptyset$.

Note that $C_1, (s, h, (O, L, D)) \not\rightarrow_p \mathsf{abort}$, because $C, (s, h, (O, L, D)) \rightarrow_p \mathsf{abort}$.

Then, by induction hypothesis, we conclude that $C_1' \in wf\_cmd$.

To prove that $C' \in wf\_cmd$, we need to check that $Locked(C_1') \cap Locked(C_2) = \emptyset$.

Suppose that there is r such that $r \in Locked(C_1') \cap Locked(C_2)$.

We have that $r \notin Locked(C_1)$, because $Locked(C_1) \cap Locked(C_2) = \emptyset$. Then the resource r need to be acquired by the transition

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

It can only acquire the resource r if $r \in D$. By the rule $(WITHA2)$, if $r \in D$ and $r \in Locked(C_2)$, then $C_2, (s, h, (O, L, D)) \rightarrow_p \mathsf{abort}$.

Hence, if $r \in Locked(C_1') \cap Locked(C_2)$, then $C, (s, h, (O, L, D)) \rightarrow_p \mathsf{abort}$.

The previous show that $Locked(C_1') \cap Locked(C_2) = \emptyset$. Therefore $C' \in wf\_cmd$.

The case $(PAR2)$ is analogous to the previous case. □

By induction on the number of transitions and the previous propositions, we have the following corollary. This corollary gives a necessary condition to the reachable commands.

Corollary 1. *If C is reachable, then $C \in wf\_cmd$.*

Using the corollary above, we revisit the motivational example presented at the begin of this section.

Example 10. *The command $\mathsf{within}\ r\ \mathsf{do}\ \mathsf{within}\ r\ \mathsf{do}\ \mathsf{skip}$ is not well-formed. Then, by the corollary 1, it is not reachable.*

The next proposition shows how the resource configuration is modified by a transition.

Proposition 15. *Let $C \in wf\_cmd$, $C' \in \mathcal{C}$, $s, s' \in \mathcal{S}$, $h, h' \in \mathcal{H}$ and $\rho = (O, L, D), \rho' = (O', L', D') \in \mathcal{O}$. If $C, (s, h, \rho) \not\rightarrow_p \mathsf{abort}$ and $C, (s, h, \rho) \rightarrow_p C', (s', h', \rho')$, then*

$$O' = (O \cup (Locked(C') \setminus Locked(C))) \setminus (Locked(C) \setminus Locked(C')).$$

Proof. Using the proposition 14, we start by noting that $C' \in wf\_cmd$.

Let $C, C' \in wf\_cmd$, $s, s' \in \mathcal{S}$, $h, h' \in \mathcal{H}$ and $\rho = (O, L, D), \rho' = (O', L', D') \in \mathcal{O}$. If $C, (s, h, \rho) \not\rightarrow_p \mathsf{abort}$ and

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

We prove the proposition by induction on the rules for the transition above.

Suppose that the transition is given by $(ASSIGN)$, $(LOOP)$, $(READ)$, (WRI), (ALL), $(FREE)$, $(PAR3)$ or $(RES0)$. We have $Locked(C) = \emptyset$, $Locked(C') = \emptyset$ and $\rho' = \rho$.

Then the conclusion is immediate.

Suppose that the transition is given by $(IF1)$. We have $C = \mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2$, $C' = C_1$ and $\rho' = \rho$.

From $C \in wf\_cmd$, we know that $usr\_cmd(C_1) = \texttt{true}$ and $Locked(C) = Locked(C') = \emptyset$. Then the conclusion is obvious.

The case $(IF2)$ is analogous to the previous case.

Suppose that the transition is given by $(SEQ1)$. We have $C = \mathsf{skip}\,;C_2$, $C' = C_2$ and $\rho' = \rho$. From $C \in wf\_cmd$, we know that $usr\_cmd(C_2) = \texttt{true}$ and $Locked(C') = Locked(C) = \emptyset$. Then the conclusion is trivial.

Suppose that the transition is given by $(SEQ2)$. We have $C = C_1\,;C_2$, $C' = C_1'\,;C_2$ and

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

From $C, (s, h, \rho) \not\rightarrow_p \mathsf{abort}$, we know

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

Applying the induction hypothesis, we get that

$$O' = (O \cup (Locked(C_1') \setminus Locked(C_1))) \setminus (Locked(C_1) \setminus Locked(C_1')).$$

The conclusion is valid, because $Locked(C) = Locked(C_1)$ and $Locked(C') = Locked(C_1')$.

Suppose that the transition is given by $(PAR1)$. We have $C = C_1' \parallel C_2$, $C = C_1' \parallel C_2$ and

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

Note that $C_1, (s, h, \rho) \not\rightarrow_p \mathsf{abort}$, because $C, (s, h, \rho) \not\rightarrow_p \mathsf{abort}$. And that $C_1 \in wf\_cmd$, because $C \in wf\_cmd$. Using the induction hypothesis, we have that

$$O' = (O \cup (Locked(C_1') \setminus Locked(C_1))) \setminus (Locked(C_1) \setminus Locked(C_1')).$$

We know that $Locked(C_1) \cap Locked(C_2) = \emptyset$ and $Locked(C_1') \cap Locked(C_2) = \emptyset$, because the commands C and C' are well-formed.

By $Locked(C) = Locked(C_1) \cup Locked(C_2)$, $Locked(C') = Locked(C_1') \cup Locked(C_2')$ and the equalities above, we have that

$$O' = (O \cup (Locked(C') \setminus Locked(C))) \setminus (Locked(C) \setminus Locked(C')).$$

The case $(PAR2)$ is analogous to the previous case.

Suppose that the transition is given by $(RES1)$. We have $r \notin \rho$, $C = \mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}$, $C' = \mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}'$, $r \in Locked(\tilde{C})$ and

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

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

Note that $\tilde{C}, (s, h, (O \cup \{r\}, L, D)) \not\rightarrow_p \mathsf{abort}$. Using the induction hypothesis, we obtain that

$$O'' = (O \cup \{r\} \cup (Locked(\tilde{C}') \setminus Locked(\tilde{C}))) \setminus (Locked(\tilde{C}) \setminus Locked(\tilde{C}')).$$

We have the following equalities of sets:

$$(A \cup B) \setminus C = (A \setminus C) \cup (B \setminus C),$$
$$(A \setminus B) \setminus C = (A \setminus C) \setminus (B \setminus C)$$

Therefore

$$O' = O'' \setminus \{r\} = ((O \cup \{r\} \cup (Locked(\tilde{C}') \setminus Locked(\tilde{C}))) \setminus \{r\}) \setminus ((Locked(\tilde{C}) \setminus Locked(\tilde{C}')) \setminus \{r\}).$$

From $r \notin O$, we obtain that

$$O' = (O \cup ((Locked(\tilde{C}') \setminus \{r\}) \setminus (Locked(\tilde{C}) \setminus \{r\}))) \setminus ((Locked(\tilde{C}) \setminus \{r\}) \setminus (Locked(\tilde{C}') \setminus \{r\})).$$

Note that $Locked(C) = Locked(\tilde{C}) \setminus \{r\}$ and $Locked(C') = Locked(\tilde{C}') \setminus \{r\}$. Hence

$$O' = (O \cup (Locked(C') \setminus Locked(C))) \setminus (Locked(C) \setminus Locked(C')).$$

The case $(RES2)$ is analogous to the previous case.

Suppose that the transition is given by $(WITH0)$.
We have $C = \mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}$, $C' = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}$, $\rho = (O, L, D \cup \{r\})$ and $\rho' = (O \cup \{r\}, L, D)$.
From $C \in wf\_cmd$, we know $\tilde{C} \in usr\_cmd$. Hence, $Locked(C') = \{r\}$ and $Locked(C) = \emptyset$. The conclusion is straightforward.

Suppose that the transition is given by $(WITH1)$. We have $r \notin (O, L, D)$, $\rho = (O \cup \{r\}, L, D)$, $\rho' = (O' \cup \{r\}, L', D')$, $C = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}$, $C' = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}'$ and

$$\tilde{C}, (s, h, (O, L, D)) \rightarrow_p \tilde{C}', (s', h', (O', L', D')).$$

Note that $\tilde{C}, (s, h, (O, L, D)) \not\rightarrow_p \mathsf{abort}$. Using the induction hypothesis, we have that

$$O' = (O \cup (Locked(\tilde{C}') \setminus Locked(\tilde{C}))) \setminus (Locked(\tilde{C}) \setminus Locked(\tilde{C}')).$$

Using that $Locked(C) = Locked(\tilde{C}) \cup \{r\}$ and $Locked(C') = Locked(\tilde{C}') \cup \{r\}$, we conclude that

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

Suppose that the transition is given by $(WITH2)$. We have $C = \mathsf{within}\ r\ \mathsf{do}\ \mathsf{skip}$, $C' = \mathsf{skip}$, $\rho = (O \cup \{r\}, L, D)$ and $\rho' = (O', L', D' \cup \{r\})$.
The conclusion follows from $Locked(C) = \{r\}$ and $Locked(C') = \emptyset$. □

For any execution that start with every resource available, in the next proposition we give the explicit formula to the resource configuration at any time of that execution.

Proposition 16. *Let $k \geq 0$, $C, C' \in \mathcal{C}$, $s, s' \in \mathcal{S}$, $h, h' \in \mathcal{H}$, $\rho' \in \mathcal{O}$ such that $C \in \boldsymbol{Com}$ and 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 $k \geq 0$, $C, C' \in \mathcal{C}$, $s, s' \in \mathcal{S}$, $h, h' \in \mathcal{H}$, $\rho' \in \mathcal{O}$ such that $C \in \mathbf{Com}$, C' is reachable from C and

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

The prove is done by induction on k.
Let $k = 0$, it is immediately from $usr\_cmd(C) = \mathtt{true}$ 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, we have that

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

Using the proposition 15, we know that

$$O' = (Locked(C'') \cup (Locked(C') \setminus Locked(C''))) \setminus (Locked(C'') \setminus Locked(C')).$$

Next we see that $O' = Locked(C')$. Let $r \in Locked(C')$.
Suppose that $r \in Locked(C'')$. Then

$$r \in Locked(C'') \setminus (Locked(C'') \setminus Locked(C')) \subseteq O'.$$

Suppose that $r \notin Locked(C'')$. Then

$$r \in (Locked(C') \setminus Locked(C'')) \setminus (Locked(C'') \setminus Locked(C')) \subseteq O'.$$

Therefore

$$Locked(C') \subseteq O'.$$

Let $r \in O'$.
If $r \in Locked(C'')$, then $r \in Locked(C')$.
If $r \notin Locked(C'')$, then $r \in Locked(C')$.
Hence

$$Locked(C') = O'.$$

By proposition 4, we conclude that

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

□

4 Extended operational semantics

In this section, the operational semantics is extended with the environment transition. The environment transition allows the environment to change the heap associated to global properties and to change the value assignment to variables not protected by the rely-set and/or resources acquired. The notion of validity for the extended operational semantic is introduced, the notion of safety for the next n transition.

Finally the soundness of Concurrent Separation Logic with respect the operational semantics is proved.

4.1 Environment Transition

The transformation on the environment is defined by the following relation.

Definition 31. *Let $s, s' \in \mathcal{S}$, $h, h' \in \mathcal{H}$, $(O, L, D), (O', L', D') \in \mathcal{O}$ and $A \subseteq \textbf{\textit{Var}}$.*

We say that $(s, h, (O, L, D))$ is transformed by the environment to $(s', h', (O', L', D'))$ 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*
- *$O' = O$ and $L' \cup D' = L \cup D$, and*
- *$h' = h$.*

The transformation on the environment defined above naturally defines the relation $\overset{A}{\rightsquigarrow} \subseteq (\mathcal{S} \times \mathcal{H} \times \mathcal{O}) \times (\mathcal{S} \times \mathcal{H} \times \mathcal{O})$. The next proposition is immediate from the definition.

Proposition 17. *Let $A', A \subseteq \textbf{\textit{Var}}$. The relation $\overset{A}{\rightsquigarrow}$ is an equivalence relation.*

If $A' \subseteq A$ and $(s, h, \rho) \overset{A}{\rightsquigarrow} (s', h', \rho')$, then $(s, h, \rho) \overset{A'}{\rightsquigarrow} (s', h', \rho')$.

Let Γ be a well-formed resource context and $A \subseteq \textbf{Var}$.

The environment transition denoted by the relation $\xrightarrow{A,\Gamma}_e \subseteq (\mathcal{C} \times \mathcal{S} \times \mathcal{H} \times \mathcal{O}) \times (\mathcal{C} \times \mathcal{S} \times \mathcal{H} \times \mathcal{O})$, it is defined by the following rule.

Let $h_G, h'_G \in \mathcal{H}$, $C \in \mathcal{C}$, $s, s' \in \mathcal{S}$, $h, h' \in \mathcal{H}$, $\rho, \rho' \in \mathcal{O}$ and $A' = A \cup \bigcup_{r \in Locked(C)} PV(r)$. 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

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

Using the rule of the environment we can:

- change the storage, except inside the rely-set A and the variables protected by the resources acquired at the moment;
- interchange the resources locked by the environment and available; and
- modified the subheap that it is uniquely determined by the resources available.

We extend the transitions on the operational semantic with the environment transition, and we define the following relation $\xrightarrow{A,\Gamma}\subseteq (\mathcal{C}\times\mathcal{S}\times\mathcal{H}\times\mathcal{O})\times((\mathcal{C}\times\mathcal{S}\times\mathcal{H}\times\mathcal{O})\cup\{\mathsf{abort}\})$. This new relation is given by

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

4.2 Safety

We start by defining the set of variables that the next transition can change.

Definition 32. *Let $C\in\mathcal{C}$. The set of variables possible to* change *by C in the next transition is denoted by chng and it is given by:*

- $chng(\mathsf{x:=E})=chng(\mathsf{x:=[E]})=chng(\mathsf{x:=cons(E)})=\{\mathsf{x}\}$,
- $chng(C_1\,;C_2)=chng(C_1)$,
- $chng(\mathsf{within}\ r\ \mathsf{do}\ C)=chng(\mathsf{resource}\ r\ \mathsf{in}\ C)=chng(C)$,
- $chng(C_1\parallel C_2)=chng(C_1)\cup chng(C_2)$,
- $chng(C)=\emptyset$, *otherwise.*

Now, we introduce the notion of safety in the next n transitions.

Definition 33. *Let $C\in\mathcal{C}$, $s\in\mathcal{S}$, $h\in\mathcal{H}$, $\rho\in\mathcal{O}$, Γ a well-formed resource context , $A\subseteq$ **Var** and $Q\in$ **Astn**. We say that:*

- $Safe_0(C,s,h,\rho,\Gamma,Q,A)$ *is always valid.*
- $Safe_{n+1}(C,s,h,\rho,\Gamma,Q,A)$ *is valid if:*

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

  (ii) The next transition of C does not abort for (s,h,ρ), i.e.

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

  (iii) The next transition of C does not change variables protected by resources not owned, i.e.

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

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

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

  then there exist h' and h'_G such that $\hat{h}=h'\uplus h'_G$, $Safe_n(C',s',h',\rho',\Gamma,Q,A)$ is valid and

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

The next theorem is proved by induction on the number of program's transitions.

Theorem 1. *Let $C \in \boldsymbol{Com}$, $P, Q \in \boldsymbol{Astn}$, Γ a well-formed resource context and $A \subseteq \boldsymbol{Var}$. If for every s, h 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*

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

If a specification is safe for n transitions, then it is also valid for less transitions. This is stated in the proposition below.

Proposition 18. *Let $C \in \mathcal{C}$, $s \in \mathcal{S}$, $h \in \mathcal{H}$, $\rho \in \mathcal{O}$, Γ a well-formed resource context, $Q \in \boldsymbol{Astn}$, $A \subseteq \boldsymbol{Var}$, $n \in \mathbb{N}_0$ and $k \leq n$. If $Safe_n(C, s, h, \rho, \Gamma, Q, A)$ is valid, then $Safe_k(C, s, h, \rho, \Gamma, Q, A)$ is valid.*

Proof. Let $C \in \mathcal{C}$, $s \in \mathcal{S}$, $h \in \mathcal{H}$, $\rho \in \mathcal{O}$, Γ a well-formed resource context, $Q \in \mathbf{Astn}$, $A \subseteq \mathbf{Var}$, $n \in \mathbb{N}_0$ and $k \leq n$ such that $Safe_n(C, s, h, \rho, \Gamma, Q, A)$ is valid.

If $k = 0$, it is trivial.

If $k > 0$, the first three conditions are immediate because $n > 0$.

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

Consider the reflexive environment transition, we conclude that $Safe_{n-1}(C, s, h, \rho, \Gamma, Q, A)$ is valid.

Repeating the argument before, we obtain that $Safe_k(C, s, h, \rho, \Gamma, Q, A)$ is valid. □

4.3 Safety's properties

In this section, we give technical results that will be used in the last section, when we prove the soundness of Concurrent Separation Logic.

For the skip rule, we have the following result.

Proposition 19. *Let $s \in \mathcal{S}$, $h \in \mathcal{H}$, $\rho \in \mathcal{O}$, Γ a well-formed resource context, $Q \in \boldsymbol{Astn}$ 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 $s \in \mathcal{S}$, $h \in \mathcal{H}$, $\rho \in \mathcal{O}$, Γ a well-formed resource context, $Q \in \mathbf{Astn}$ and $A \subseteq \mathbf{Var}$ such that $FV(Q) \subseteq A$ and $s, h \models Q$.

We prove the proposition by induction on n.

Let $n = 0$. It is trivially valid.

Let $n = k + 1$.

From $s, h \models Q$, we obtain the condition (i) of $Safe_n(\mathsf{skip}, s, h, \rho, \Gamma, Q, A)$.

The command skip never aborts, then the property (ii) of $Safe_n(\mathsf{skip}, s, h, \rho, \Gamma, Q, A)$ is immediate.

From $chng(\mathsf{skip}) = \emptyset$, we have the property (iii) of $Safe_n(\mathsf{skip}, s, h, \rho, \Gamma, Q, A)$.

It remains to prove the condition (iv) of $Safe_n(\mathsf{skip}, s, h, \rho, \Gamma, Q, A)$.

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

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

The only possible transitions to the command skip are given by environments transitions. Then $C' = \mathsf{skip}$,

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

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.

Note that $s(x) = s'(x)$, for every $x \in FV(Q) \subseteq A$. By proposition 2, we have that

$$s', h \models Q.$$

Therefore by induction hypothesis, $Safe_k(\mathsf{skip}, s', h, \rho', \Gamma, Q, A)$ is valid. □

The result below will be used to prove the soundness of the sequence rule.

Proposition 20. *Let $C_1 ; C_2 \in wf\_cmd$, $s \in \mathcal{S}$, $h \in \mathcal{H}$, $\rho \in \mathcal{O}$, $R, Q \in \boldsymbol{Astn}$, $A_1, A_2 \subseteq \boldsymbol{Var}$, $n \in \mathbb{N}_0$ and Γ a well-formed resource context such that $\rho = (Locked(C_1), L, D)$, $L \cup D = Res(\Gamma) \setminus Locked(C_1)$, $FV(R) \subseteq A_1$, and $FV(Q) \subseteq A_2$.*

If, for every $s', h' \models R$ and $L' \cup D' = Res(\Gamma)$, $Safe_n(C_2, s', h', (\emptyset, L', D'), \Gamma, Q, A_2)$ is valid and $Safe_n(C_1, s, h, \rho, \Gamma, R, A_1)$ is valid, then $Safe_n(C_1 ; C_2, s, h, \rho, \Gamma, Q, A_1 \cup A_2)$ is valid.

Proof. Let $C_1 ; C_2 \in wf\_cmd$, $s \in \mathcal{S}$, $h \in \mathcal{H}$, $\rho \in \mathcal{O}$, $R, Q \in \mathbf{Astn}$, $A_1, A_2 \subseteq \mathbf{Var}$, $n \in \mathbb{N}_0$ and Γ is a well-formed resource context such that $\rho = (Locked(C_1), L, D)$, $L \cup D = Res(\Gamma) \setminus Locked(C_1)$, $FV(R) \subseteq A_1$, $FV(Q) \subseteq A_2$, $Safe_n(C_1, s, h, \rho, \Gamma, R, A_1)$ is valid and, for every $s', h' \models R$ and $L' \cup D' = Res(\Gamma)$, $Safe_n(C_2, s', h', (\emptyset, L', D'), \Gamma, Q, A_2)$ is valid.

We start by noting that $Locked(C) = Locked(C_1)$. From $C_1 ; C_2 \in wf\_cmd$, we know that $C_1 \in wf\_cmd$ and $C_2 \in \mathbf{Com}$. Then $Locked(C_2) = \emptyset$.

We prove the proposition by induction on n.

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

The first property of $Safe_n(C_1 ; C_2, s, h, \rho, \Gamma, Q, A_1 \cup A_2)$ is immediate, because $C_1 ; C_2 \neq \mathsf{skip}$.

From $Safe_n(C_1, s, h, \rho, \Gamma, R, A_1)$, we know that

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

Then

$$C_1 ; C_2, (s, h, \rho) \not\to_p \mathsf{abort}.$$

So the property (ii) of $Safe_n(C_1 ; C_2, s, h, \rho, \Gamma, Q, A_1 \cup A_2)$ is verified.

We have $chng(C_1 ; C_2) = chng(C_1)$.

It follows from $Safe_n(C_1, s, h, \rho, \Gamma, R, A_1)$, that

$$chng(C_1 ; C_2) \cap \bigcup_{r \in L \cup D} PV(r) = chng(C_1) \cap \bigcup_{r \in L \cup D} PV(r) = \emptyset.$$

The previous establish the property (iii) of $Safe_n(C_1 ; C_2, s, h, \rho, \Gamma, Q, A_1 \cup A_2)$.

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

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

Below we consider the possible transitions.

Suppose that the transition is given by $(SEQ1)$. We have $C' = C_2$, $C_1 = \mathsf{skip}$, $s' = s$, $h' = h \uplus h_G$ and $\rho' = \rho = (\emptyset, L, D)$, where $L \cup D = Res(\Gamma)$.

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

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

From $Safe_n(\mathsf{skip}, s, h, \rho, \Gamma, R, A_1)$, we know

$$s, h \models R.$$

By the hypothesis, we have that $Safe_n(C_2, s, h, (\emptyset, L, D), \Gamma, Q, A_2)$ is valid.

From $\overset{A_2}{\rightsquigarrow} \supseteq \overset{A_1 \cup A_2}{\rightsquigarrow}$ and proposition 18, we conclude that $Safe_k(C_2, s, h, (\emptyset, L, D), \Gamma, Q, A_1 \cup A_2)$ is valid.

Suppose that the transition is given by $(SEQ2)$. We have $C' = C'_1 ; C_2$ and

$$C_1, (s, h \uplus h_G, \rho) \to_p C'_1, (s', h', \rho').$$

From $Safe_n(C_1, s, h, (Locked(C), L, D), \Gamma, R, A_1)$ be valid and the previous transition, we know that there is $h'_G \subseteq h'$ such that $Safe_k(C'_1, s', h' \setminus h'_G, \rho', \Gamma, R, A_1)$ is valid and

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

By the proposition 15, we obtain that

$$\rho' = (Locked(C_1'), L', D'),$$

where $L' \cup D' = Res(\Gamma) \setminus Locked(C_1')$.

From proposition 18, we have that $Safe_k(C_2, s', h', (\emptyset, L', D'), \Gamma, Q, A_2)$ is valid, for every $s', h' \models R$ and $L' \cup D' = res(\Gamma)$.

Hence, by induction, we conclude that $Safe_k(C_1' \,;\, C_2, s', h' \setminus h'_G, \rho', \Gamma, Q, A_1 \cup A_2)$ is valid.

Suppose that the transition is given by (ENV). Let $A' = A_1 \cup A_2 \cup \bigcup_{r \in Locked(C)} PV(r)$.

We have $C' = C_1 \,;\, C_2$,

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

and there exists $h'_G \subseteq h'$ such that $h' = h \uplus h'_G$ and

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

It is sufficient to check that $Safe_k(C_1 \,;\, C_2, s', h, \rho', \Gamma, Q, A_1 \cup A_2)$ is valid.

From $A_1 \cup \bigcup_{r \in Locked(C_1)} PV(r) \subseteq A_1 \cup A_2 \cup \bigcup_{r \in Locked(C)} PV(r)$, we have the following environment transformation

$$(s, h, \rho) \overset{A_1'}{\rightsquigarrow} (s', h, \rho'),$$

where $A_1' = A_1 \cup \bigcup_{r \in Locked(C_1)} PV(r)$.

Considering the same h_G and h'_G, as before, we have the following environment transition

$$C_1, (s, h \uplus h_G, \rho) \xrightarrow{A_1, \Gamma}_e C_1, (s', h \uplus h'_G, \rho').$$

Using that $Safe_n(C_1, s, h, \rho, \Gamma, R, A_1)$ is valid and the environment transition above, we obtain that $Safe_k(C_1, s', h, \rho', \Gamma, R, A_1)$ is valid.

Note that $\rho' = (Locked(C_1), L', D')$ and $L' \cup D' = Res(\Gamma) \setminus Locked(C_1)$.

By the induction hypothesis, we conclude that $Safe_k(C_1 \,;\, C_2, s', h, \rho', \Gamma, Q, A_1 \cup A_2)$ is valid. □

To prove the soundness of the parallelism's rule, we'll use the next proposition.

Proposition 21. *Let $n \in \mathbb{N}_0$, $s \in \mathcal{S}$, $C_1 \parallel C_2 \in wf\_cmd$, $h, h_1, h_2 \in \mathcal{H}$, $\rho, \rho_1, \rho_2 \in \mathcal{O}$, $Q_1, Q_2 \in$ **Astn** and $A_1, A_2 \subseteq$ **Var** such that $FV(Q_1) \subseteq A_1$, $FV(Q_2) \subseteq A_2$, $\rho = (O_1 \cup O_2, L, D)$, $\rho_1 = (O_1, L \cup O_2, D)$, $\rho_2 = (O_2, L \cup O_1, D)$, and $h = h_1 \uplus h_2$.*

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, and $A_1 \cap mod(C_2) = A_2 \cap mod(C_1) = \emptyset$, then $Safe_n(C_1 \parallel C_2, s, h, \rho, \Gamma, Q_1 * Q_2, A_1 \cup A_2)$ is valid.

Proof. Let $n \in \mathbb{N}_0$, $s \in \mathcal{S}$, $C_1 \parallel C_2 \in wf\_cmd$, $h, h_1, h_2 \in \mathcal{H}$, $\rho, \rho_1, \rho_2 \in \mathcal{O}$, $Q_1, Q_2 \in$ **Astn** and $A_1, A_2 \subseteq$ **Var** such that $\rho = (O_1 \cup O_2, L, D)$, $\rho_1 = (O_1, L \cup O_2, D)$, $\rho_2 = (O_2, L \cup O_1, D)$, $FV(Q_1) \subseteq A_1$, $FV(Q_2) \subseteq A_2$, $h = h_1 \uplus h_2$ and the next expression are valid

$$Safe_{k+1}(C_1, s, h_1, \rho_1, \Gamma, Q_1, A_1), \tag{1}$$

$$Safe_{k+1}(C_2, s, h_2, \rho_2, \Gamma, Q_2, A_2). \tag{2}$$

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

Let $n = k + 1$.

From $C_1 \parallel C_2 \neq \mathsf{skip}$, the condition (i) of $Safe_n(C_1 \parallel C_2, s, h, \rho, \Gamma, Q_1 * Q_2, A_1 \cup A_2)$ is immediate.

Applying the safety monotonicity (proposition 6) to the assumptions (1) and (2), we see that

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

By the proposition 8,

$$C_1, (s, h, \rho) \not\to_p \mathsf{abort} \quad \text{and} \quad C_2, (s, h, \rho) \not\to \mathsf{abort}.$$

Therefore

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

The previous prove that the condition (ii) of $Safe_n(C_1 \parallel C_2, s, h, \rho, \Gamma, Q_1 * Q_2, A_1 \cup A_2)$ is respected.

By the assumptions (1) and (2), we have that

$$chng(C_1) \cap \bigcup_{r \in L \cup O_2 \cup D} PV(r) = \emptyset \quad \text{and} \quad chng(C_2) \cap \bigcup_{r \in L \cup O_1 \cup D} PV(r) = \emptyset.$$

Then

$$(chng(C_1) \cup chng(C_2)) \cap \bigcup_{r \in L \cup D} PV(r) = \emptyset.$$

The condition (iii) of $Safe_n(C_1 \parallel C_2, s, h, \rho, \Gamma, Q_1 * Q_2, A_1 \cup A_2)$ follows from the previous statements and $chng(C) = chng(C_1) \cup chng(C_2)$.

Next, we prove the condition (iv) of $Safe_n(C_1 \parallel C_2, s, h, \rho, \Gamma, Q_1 * Q_2, A_1 \cup A_2)$. Let h_G, C', s', h' and ρ' such that $s, h_G \models \circledast_{r \in D} \Gamma(r)$ and

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

Below, we consider the four possible transitions.

Suppose that the transition is given by $(PAR1)$. We have that $C' = C'_1 \parallel C_2$ and

$$C_1, (s, h_1 \uplus h_2 \uplus h_G, \rho) \rightarrow_p C'_1, (s', h', \rho').$$

From the assumption (1), we know that $C_1, (s, h_1, \rho_1) \not\rightarrow_p \mathsf{abort}$.

By the safety monotonicity (proposition 6), for every $h_F \perp h$, we have that $C_1, (s, h_1 \uplus h_F, \rho_1) \not\rightarrow_p \mathsf{abort}$.

Using the proposition 9, we get that $O_2 \subseteq O'$ and

$$C_1, (s, h_1 \uplus h_2 \uplus h_G, \rho_1) \rightarrow_p C'_1, (s', h', (O' \setminus O_2, L \cup O_2, D')).$$

By the frame property (proposition 7), we have that $h_2 \subseteq h'$ and

$$C_1, (s, h_1 \uplus h_G, \rho_1) \rightarrow_p C'_1, (s', h' \setminus h_2, (O' \setminus O_2, L \cup O_2, D')).$$

Consider $\rho'_1 = (O' \setminus O_2, L \cup O_2, D')$. By (1) be valid, we know that there are h'_1 and h'_G such that $h' = h'_1 \uplus h_2 \uplus h'_G$, $Safe_k(C'_1, s', h'_1, \rho'_1, \Gamma, Q_1, A_1)$ is valid and

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

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

First, we see that we have the following environment transformation

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

where $\rho'_2 = (O_2, L \cup (O' \setminus O_2), D')$ and $A'_2 = A_2 \cup \bigcup_{r \in Locked(C_2)} PV(r)$.

We have that $s(x) = s'(x)$, for every $x \in A_2$, because $mod(C_1) \cap A_2 = \emptyset$.

Note that $Locked(C_2) \subseteq O_2$, otherwise we obtain by the rule $(WITHA2)$ that $C_2, (s, h, \rho_2) \rightarrow_p \mathsf{abort}$.

From property (iii) of (1) and $Locked(C_2) \subseteq O_2$, we have that $s(x) = s'(x)$ for every $x \in A'_2$.

It is straightforward that $O_2 = O_2$ and $h_2 = h_2$.

From the proposition 4, we know that $O_1 \cup D = (O' \setminus O_2) \cup D'$. Then

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

Hence we can consider the following environment transition

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

By (2), we conclude that $Safe_k(C_2, s', h_2, \rho'_2, \Gamma, Q_2, A_2)$ is valid.

To apply the induction hypothesis, we need to check that ρ', ρ'_1, ρ'_2 respect the conditions in the proposition. This is immediate from $O' = (O' \setminus O_2) \cup O_2$.

By the induction hypothesis, we conclude that $Safe_k(C'_1 \parallel C_2, s', h'_1 \uplus h_2, \rho', \Gamma, Q_1 * Q_2, A_1 \cup A_2)$.

The case $(PAR2)$ is analogous to the previous case.

Suppose that the transition is given by $(PAR3)$. 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

$$s, h_1 \models Q_1 \quad \text{and} \quad s, h_2 \models Q_2.$$

Then

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

From the proposition 19, we conclude that $Safe_k(\mathsf{skip}, s, h_1 \uplus h_2, \rho, \Gamma, Q_1 * Q_2, A_1 \cup A_2)$ is valid.

Suppose that the transition is given by (ENV). 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 start by proving that

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

where $\rho'_1 = (O_1, L' \cup O_2, D')$ and $A'_1 = A_1 \cup \bigcup_{r \in Locked(C_1)} PV(r)$.

We know that $s(x) = s'(x)$, for every $x \in A'_1 \subseteq A'$. It is straightforward that $O_1 = O_1$ and $h_1 = h_1$.

From $L \cup D = L' \cup D'$, we infer that $L \cup O_2 \cup D = L' \cup O_2 \cup D'$. Then we have the environment transformation above.

From the environment transformation above, $s, h_G \models \circledast_{r \in D} \Gamma(r)$, $s', h'_G \models \circledast_{r \in D'} \Gamma(r)$ and (1), 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 $\rho'_2 = (O_2, L' \cup O_1, D')$.

Note that the new resources configurations ρ', ρ'_1 and ρ'_2 respect the condition in the proposition.

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

For the critical region, we have the following auxiliary result.

Proposition 22. *Let $C \in wf\_cmd$, $s \in \mathcal{S}$, $h \in \mathcal{H}$, $\rho \in \mathcal{O}$, Γ a well-formed resource context, $Q, R \in$ **Astn**, $A, X \subseteq$ **Var**, $r \in$ **Res** and $n \in \mathbb{N}_0$ such that $\Gamma' = \Gamma, r(X) : R$ is a well-formed resource context, $\rho = (O, L, D)$, $r \in O$ and $FV(Q) \subseteq A$.*

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

Proof. Let $C \in wf\_cmd$, $s \in \mathcal{S}$, $h \in \mathcal{H}$, $\rho \in \mathcal{O}$, Γ a well-formed resource context, $Q, R \in \mathbf{Astn}$, $A, X \subseteq \mathbf{Var}$ and $r \in \mathbf{Res}$ such that $\Gamma' = \Gamma, r(X) : R$ is a well-formed resource context, $\rho = (O, L, D)$, $r \in O$, $FV(Q) \subseteq A$ and $Safe_n(C, s, h, \rho \setminus \{r\}, \Gamma, Q * R, A \cup X)$ is valid.

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

For $n = 0$, it is trivial.

Let $n = k + 1$.

Because $\mathsf{within}\ r\ \mathsf{do}\ C \neq \mathsf{skip}$, we have the property (i) of $Safe_n(\mathsf{within}\ r\ \mathsf{do}\ C, s, h, \rho, \Gamma', Q, A)$.

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

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

From the previous and $r \in O$, we obtain that

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

Hence, the property (ii) of $Safe_n(\mathsf{within}\ r\ \mathsf{do}\ C, s, h, \rho, \Gamma', Q, A)$ is respected.

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

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

Then, we have the condition (iii) of $Safe_n(\mathsf{within}\ r\ \mathsf{do}\ C, s, h, \rho, \Gamma', Q, A)$.

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

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

Next, we consider the possible transitions.

Suppose that the transition is given by $(WITH1)$. We have $C' = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}$, $r \in O \cap O'$ and

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

Taking in account that $Safe_n(C, s, h, \rho \setminus \{r\}, \Gamma, Q * R, A \cup X)$ is valid, we know that there are h'_G and 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

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

Applying the induction hypothesis to $Safe_k(\tilde{C}, s', h'_L, \rho' \setminus \{r\}, \Gamma, Q * R, A \cup X)$, 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 $(WITH2)$.

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 $Safe_n(\mathsf{skip}, s, h, \rho, \Gamma, Q * R, A \cup X)$ be valid, we have that

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

Then there exists $h_R \subseteq h$ such that

$$s, h_R \models R \quad \text{and} \quad s, h \setminus h_R \models Q.$$

Let $h'_G = h_G \uplus h_R$. Then

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

By the proposition 19, $FV(Q) \subseteq A$ 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. The conclusion follows from $h' \setminus h'_G = h \setminus h_R$.

Suppose that the transition is given by (ENV). 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

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

Noting that $A' = A \cup X \cup \bigcup_{\hat{r} \in Locked(C)} PV(\hat{r})$, we can consider the following 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 $Safe_n(C, s, h, \rho \setminus \{r\}, \Gamma, Q * R, A \cup X)$, we have that $Safe_k(C, s', h, \rho' \setminus \{r\}, \Gamma, Q * R, A \cup X)$ is valid.
Therefore, by the induction hypothesis, $Safe_k(\mathsf{within}\ r\ \mathsf{do}\ C, s', h, \rho', \Gamma', Q, A)$ is valid. □

The following auxiliary proposition deals with the local resource rule.

Proposition 23. *Let $C \in wf\_cmd$, $s \in \mathcal{S}$, $h \in \mathcal{H}$, Γ a well-formed resource context, $Q, R \in \boldsymbol{Astn}$, $A, X \subseteq \boldsymbol{Var}$, $r \in \boldsymbol{Res}$, $n \in \mathbb{N}_0$ and $\rho = (O, L, D) \in \mathcal{O}$ such that $r \notin \rho$, $FV(Q) \subseteq A$ and $\Gamma' = \Gamma, r(X) : R$ is a well-formed resource context. We have the following statements:*

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

Proof. Let C, s, h, ρ, Γ, Q, R, A, X, r, n as stated in the proposition such that $r \notin \rho$, $FV(Q) \subseteq A$ and $\Gamma' = \Gamma, r(X) : R$ is a well-formed resource context. Consider the next statements:

$P(n)$ is 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.

$Q(n)$ is 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.

This prove has three parts. First, we note that $P(O) \wedge Q(0)$ is true. Next, we prove 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 $n \geq 0$. This three steps prove the proposition.

It is trivial that $P(0)$ and $Q(0)$ are true.

Next, we prove that $P(n) \wedge Q(n)$ implies $Q(n+1)$.
Suppose that $r \notin Locked(C)$, $h_R \perp h$, $s, h_R \models R$ and $Safe_{n+1}(C, s, h, (O, L, D \cup \{r\}), \Gamma', Q, A)$ is valid.
The property (i) 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 $\mathsf{resource}\ r\ \mathsf{in}\ C \neq \mathsf{skip}$.
Suppose that

$$\mathsf{resource}\ r\ \mathsf{in}\ C, (s, h \uplus h_R, \rho) \rightarrow_p \mathsf{abort}.$$

Then, we have one of the following:

- $r \in \rho$, or,

- $r \in Locked(C)$ and $C, (s, h \uplus h_R, (O \cup \{r\}, L, D)) \to_p \mathsf{abort}$, or,
- $r \notin Locked(C)$ and $C, (s, h \uplus h_R, (O, L, D \cup \{r\})) \to_p \mathsf{abort}$.

The first two cases are contradictory with the hypothesis $r \notin \rho$ and $r \notin Locked(C)$.
Hence, it must be the third case. So, we have that

$$C, (s, h \uplus h_R, (O, L, D \cup \{r\})) \to \mathsf{abort}.$$

However, from $Safe_{n+1}(C, s, h, (O, L, D \cup \{r\}), \Gamma', Q, A)$ be valid and proposition 6, we have that

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

We reached an absurd, so the property (ii) of $Safe_{n+1}(\mathsf{resource}\ r\ \mathsf{in}\ C, s, h \uplus h_R, \rho, \Gamma, Q * R, A \cup X)$ is valid.
Observe that

$$chng(C) = chng(\mathsf{resource}\ r\ \mathsf{in}\ C).$$

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, we obtain the property (iii) of $Safe_{n+1}(\mathsf{resource}\ r\ \mathsf{in}\ C, s, h \uplus h_R, \rho, \Gamma, Q * R, A \cup X)$.
Let h_G, C', s', h' and ρ' such that $s, h_G \models \circledast_{\hat{r} \in D} \Gamma(\hat{r})$ and

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

Next, we study the possible transitions.

Suppose that the transition is given by $(RES0)$. We have that $C = C' = \mathsf{skip}$, $s' = s$, $h' = h \uplus h_R \uplus h_G$ and $\rho' = \rho$.
Taking $h'_G = h_G$. We know that $h'_G \subseteq h'$ and

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

From $Safe_{n+1}(\mathsf{skip}, s, h, (O, L, D \cup \{r\}), \Gamma', Q, A)$ be valid, we have that

$$s, h \models Q.$$

Moreover, we have

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

By the proposition 19, we conclude that $Safe_n(\mathsf{skip}, s, h \uplus h_R, \Gamma, Q * R, A \cup X)$ is valid.
The transition $(RES1)$ is not possible, because $r \notin Locked(C)$.
Suppose that the transition is given by $(RES2)$. 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

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

To complete the analysis of this transition, we need to study two different cases: $r \in O''$ or $r \in D''$ (note that $r \notin L''$, because the proposition 4).

First, we suppose that $r \in O''$. We have that $D'' = D'$ and

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

From the proposition 15, we also have that $r \in \mathit{Locked}(\tilde{C})$.

From $r \in O''$, we can apply the hypothesis $P(n)$ to $\mathit{Safe}_n(\tilde{C}, s', h' \setminus h'_G, \rho'', \Gamma', Q, A)$ and obtain that $\mathit{Safe}_n(\mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}, s', h' \setminus h'_G, \rho', \Gamma, Q * R, A \cup X)$ is valid.

To the other case, we suppose that $r \in D''$. Note that

$$\circledast_{\hat{r} \in D''} \Gamma'(\hat{r}) = R * \left(\circledast_{\hat{r} \in D'} \Gamma(\hat{r}) \right).$$

Then there exists $h'_R \subseteq h'_G$ such that $s', h'_R \models R$ and

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

From the proposition 15, it follows that $r \notin \mathit{Locked}(\tilde{C})$. Note that

$$h' = h' \setminus h'_G \uplus h'_R \uplus h'_G \setminus h'_R.$$

Therefore $\mathit{Safe}_n(\mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}, s', h' \setminus h'_G \uplus h'_R, \rho', \Gamma, Q * R, A \cup X)$ is valid, by $Q(n)$.

Hence there exists $h'_G \subseteq h'$ such that $\mathit{Safe}_n(\mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}, s', h' \setminus h'_G, \rho', \Gamma, Q * R, A \cup X)$ is valid and

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

Suppose that the transition is given by (ENV). Let $A' = A \cup X \cup \bigcup_{\hat{r} \in \mathit{Locked}(\mathsf{resource}\ r\ \mathsf{in}\ C)} PV(\hat{r})$.

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

From $r \notin \mathit{Locked}(C)$, we have that $\mathit{Locked}(C) = \mathit{Locked}(\mathsf{resource}\ r\ \mathsf{in}\ C)$ 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 the proposition 2, 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

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

and $(O', L', D' \cup \{r\}) \setminus \{r\} = \rho'$.

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 see that $Safe_n(\mathsf{resource}\ r\ \mathsf{in}\ C, s', h \uplus h_R, \rho', \Gamma, Q * R, A \cup X)$ is valid.

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.

The properties (i), (ii) and (iii) of $Safe_{n+1}(\mathsf{resource}\ r\ \mathsf{in}\ C, s, h, (O, L, D), \Gamma, Q * R, A \cup X)$ are obtained in a similar way that before.

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

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

Next, we study the possible transitions.

The transition $(RES0)$ can not occur because $r \in Locked(C)$.

Suppose that the transition is given by $(RES1)$. 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 $\rho' = \rho'' \setminus \{r\}$.

Because $r \notin D$, we know that

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

Considering the transition above, it follows from $Safe_{n+1}(C, s, h, (O \cup \{r\}, L, D), \Gamma', Q, A)$ 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

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

Like in the previous step, we have to study two cases $r \in O''$ or $r \in D''$.

First, we suppose that $r \in O''$. We have $D'' = D'$ and

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

By the proposition 15, we know that $r \in Locked(\tilde{C})$.

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

For the other case, we suppose that $r \in D''$. Note that

$$\circledast_{\hat{r} \in D''} \Gamma'(\hat{r}) = R * \circledast_{\hat{r} \in D'} \Gamma(\hat{r}).$$

Hence, we know that there exists $h'_R \subseteq h'_G$ such that $s', h'_R \models R$ and

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

From the proposition 15, it follows $r \notin Locked(\tilde{C})$. And note that

$$h' = h' \setminus h'_G \uplus h'_R \uplus h'_G \setminus h'_R.$$

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

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

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

The transition $(RES2)$ can not happen because $r \in Locked(C)$.

Suppose that the transition is given by (ENV). 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

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

Let $A'' = A \cup \bigcup_{\hat{r} \in Locked(C)} PV(\hat{r})$. Note that $A'' = A'$, because $Locked(\mathsf{resource}\ r\ \mathsf{in}\ C) \cup \{r\} = Locked(C)$ and $PV(r) = X$.

Therefore

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

Moreover, we know that

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

Then, we have the following environment transition

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

and $(O' \cup \{r\}, L', D') \setminus \{r\} = \rho'$.

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

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

The soundness of the frame rule follows from the next proposition.

Proposition 24. *Let $C \in wf\_cmd$, $s \in \mathcal{S}$, $h, h_R \in \mathcal{H}$, $\rho \in \mathcal{O}$, $Q, R \in \boldsymbol{Astn}$, Γ a well-formed resource context, $A \subseteq \boldsymbol{Var}$ such that $h \perp h_R$ and $s, h_R \models R$. 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. We prove the proposition by induction on n.

Let $C \in wf\_cmd$, $s \in \mathcal{S}$, $h, h_R \in \mathcal{H}$, $\rho \in \mathcal{O}$, $Q, R \in \mathbf{Astn}$, Γ a well-formed resource context, $A \subseteq \mathbf{Var}$ such that $h \perp h_R$, $s, h_R \models R$, $mod(C) \cap FV(R) = \emptyset$ and $Safe_n(C, s, h, \rho, \Gamma, Q, A)$ is valid.

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

Suppose that $C = \mathsf{skip}$, otherwise it is immediate. From $Safe_n(C, s, h, \rho, \Gamma, Q, A)$, we have that $s, h \models Q$. Hence

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

By $Safe_n(C, s, h, \rho, \Gamma, Q, A)$ and proposition 6, we have that

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

Again by $Safe_n(C, s, h, \rho, \Gamma, Q, A)$, we know the following

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

Until now, we proved the properties (i), (ii) and (iii) of $Safe_n(C, s, h \uplus h_R, \rho, \Gamma, Q * R, A \cup FV(R))$.

Next, we prove the property (iv). 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

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

Next, we consider two cases for the transition.

Suppose that it is a program transition, i.e. $\xrightarrow{A \cup FV(R), \Gamma} = \rightarrow_p$.

By the frame property, we have that 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 $Safe_n(C, s, h, \rho, \Gamma, Q, A)$ be valid 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

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

Applying the induction hypothesis, we obtain that $Safe_k(C', s', h' \setminus h'_G, \rho', \Gamma, Q * R, A \cup FV(R))$ is valid.

Suppose that the transition is an environment transition, i.e. $\xrightarrow{A \cup FV(R), \Gamma} = \xrightarrow{A \cup FV(R), \Gamma}_e$.
Let $A' = A \cup FV(R) \cup \bigcup_{r \in Locked(C)} PV(r)$. We have

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

and there exists $h'_G \subseteq h'$ such that $h' = h \uplus h_R \uplus h'_G$ and

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

Let $A'' = A \cup \bigcup_{r \in Locked(C)} PV(r)$. Note that $A'' \subseteq A'$. Then

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

From $Safe_n(C, s, h, \rho, \Gamma, Q, A)$ and the corresponded transition of the environment to the transformation above, we have that $Safe_k(C, s', h, \rho', \Gamma, Q, A)$ is valid.
From $FV(R) \subseteq A'$, we know that

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

By the induction hypothesis, we conclude that $Safe_k(C, s', h \uplus h_R, \rho', \Gamma, Q * R, A \cup FV(R))$ is valid. □

Before, we give the auxiliary result for the renaming rule, we complete the analysis started in the proposition 10. In the next proposition, we state that the executions are independent of the resources names.

Proposition 25. *Let $C, C' \in wf\_cmd$, $s, s' \in \mathcal{S}$, $h, h' \in \mathcal{H}$, $\rho, \rho' \in \mathcal{O}$, $A \subseteq$ **Var**, Γ, Γ' well-formed resource contexts and $r, r' \in$ **Res** such that $r' \notin Res(C)$, $r' \notin \rho$, $r' \notin \Gamma$ and $\Gamma' = \Gamma[r'/r]$.*

If $C, (s, h, \rho) \xrightarrow{A, \Gamma} C', (s', h', \rho')$, then

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

Proof. The prove is done by induction on the rule of $\xrightarrow{A, \Gamma}$.
Let $C, C' \in wf\_cmd$, $s, s' \in \mathcal{S}$, $h, h' \in \mathcal{H}$, $\rho, \rho' \in \mathcal{O}$, $A \subseteq$ **Var**, Γ, Γ' well-formed resource contexts and $r, r' \in$ **Res** such that $\Gamma' = \Gamma[r'/r]$, $r' \notin Res(C)$, $r' \notin \rho$, $r' \notin \Gamma$ and

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

Suppose that the transition is given by $(ASSIGN)$, $(IF1)$, $(IF2)$, $(SEQ1)$, $(LOOP)$, $(READ)$, (WRI), (ALL), $(FREE)$ or $(PAR3)$. Then the transition neither depend in the resources names nor in the resource context. So, it is trivial that

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

Suppose that the transition is given by $(SEQ2)$, $(PAR1)$ or $(PAR2)$. Using the induction hypothesis, it is straightforward that

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

Suppose that the transition is given by $(RES0)$. Then $C = \mathsf{resource}\ \hat{r}\ \mathsf{in}\ \mathsf{skip}$, $C' = \mathsf{skip}$ and $\hat{r} \notin \rho$. We note that $C[r'/r] = \mathsf{resource}\ \hat{r}[r'/r]\ \mathsf{in}\ \mathsf{skip}$, $C'[r'/r] = \mathsf{skip}$ and $\hat{r}[r'/r] \notin \rho[r'/r]$. Therefore

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

Suppose that the transition is given by $(RES1)$. Then $C = \mathsf{resource}\ \hat{r}\ \mathsf{in}\ \tilde{C}$, $C' = \mathsf{resource}\ \hat{r}\ \mathsf{in}\ \tilde{C}'$, $\hat{r} \notin \rho$, $\hat{r} \in Locked(C)$ and

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

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

From $r' \notin Res(C)$, we know that $\hat{r}$ is different from r' and $r' \notin Res(\tilde{C})$.

As before, we note that $C[r'/r] = \mathsf{resource}\ \hat{r}[r'/r]\ \mathsf{in}\ \tilde{C}[r'/r]$, $C'[r'/r] = \mathsf{resource}\ \hat{r}[r'/r]\ \mathsf{in}\ \tilde{C}'[r'/r]$ and $\hat{r}[r'/r] \notin \rho[r'/r]$.

From $r \notin \rho$ and $\hat{r}$ be different from r', we have that

$$r' \notin (O \cup \{\hat{r}\}, L, D).$$

We also know that $\tilde{C} \in wf\_cmd$, because $C \in wf\_cmd$. By the induction hypothesis, we have the following transition

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

As noted in the proposition 10, we know that $(\rho'' \setminus \{\hat{r}\})[r'/r] = \rho''[r'/r] \setminus \{\hat{r}[r'/r]\}$ and

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

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

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

The cases $(RES2)$ and $(WITH1)$ are analogous to the previous case.

Suppose that the transition is given by $(WITH0)$. Then $s(B) = \mathsf{true}$, $C = \mathsf{with}\ \hat{r}\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}$, $C' = \mathsf{within}\ \hat{r}\ \mathsf{do}\ \tilde{C}$, $\rho = (O, L, D \cup \{\hat{r}\})$ and $\rho' = (O \cup \{\hat{r}\}, L, D)$.

If $\hat{r}$ is different from r, the conclusion is immediate. Then, we 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

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

The case $(WITH2)$ is analogous to the previous case.

Suppose that the transition is given by (ENV). Let $A' = A \cup \bigcup_{\hat{r} \in Locked(C)} PV(\hat{r})$. We have $C' = C$ and there exists $\hat{h}$ such that $h = \hat{h} \uplus h_G$, $h' = \hat{h} \uplus h'_G$,

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

and

$$(s, \hat{h}, \rho) \overset{A'}{\rightsquigarrow} (s', \hat{h}, \rho').$$

Let $A'' = A \cup \bigcup_{\hat{r} \in Locked(C[r'/r])} PV(\hat{r})$. Note that $A'' = A'$. It is obvious that $C'[r'/r] = C[r'/r]$,

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

and

$$(s, \hat{h}, \rho[r'/r]) \overset{A''}{\rightsquigarrow} (s', \hat{h}, \rho'[r'/r]).$$

Therefore

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

□

The soundness of the rename rule follows from the next proposition.

Proposition 26. *Let $C \in wf\_cmd$, $s \in \mathcal{S}$, $h \in \mathcal{H}$, $\rho = (O, L, D) \in \mathcal{O}$, $A \subseteq \boldsymbol{Var}$, Γ a well-formed resource context and $r, r' \in \boldsymbol{Res}$ 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 $Safe_n(C, s, h, \rho, \Gamma, Q, A)$ is valid.

Proof. We prove the proposition by induction on n. Let $C \in wf\_cmd$, $s \in \mathcal{S}$, $h \in \mathcal{H}$, $\rho = (O, L, D) \in \mathcal{O}$, $A \subseteq \mathbf{Var}$, Γ a well-formed resource context and $r, r' \in \mathbf{Res}$ 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 $n = k + 1$.

If $C = \mathsf{skip}$, then $C[r'/r] = \mathsf{skip}$. By $Safe_n(C[r'/r], s, h, \rho[r'/r], \Gamma[r'/r], Q, A)$, we have that

$$s, h \models Q.$$

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

From $Safe_n(C[r'/r], s, h, \rho[r'/r], \Gamma[r'/r], Q, A)$ be valid and the proposition 10, we have that

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

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 $C' \in wf\_cmd$, because the proposition 14. By the proposition 25, 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 previous transition, we have that 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, we have that $Safe_k(C', s', h' \setminus h'_G, \rho', \Gamma, Q, A)$ is valid. □

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

Proposition 27. *Let $C \in wf\_cmd$, $s, s' \in \mathcal{S}$, $h \in \mathcal{H}$, $\rho, \rho' \in \mathcal{O}$, $Q \in \boldsymbol{Astn}$, $A, X \subseteq \boldsymbol{Var}$, Γ a well-formed resource context and $l \in \mathbb{N}_0$ such that $\rho' = (Locked(C), Res(\Gamma) \setminus Locked(C), \emptyset)$, X is a set of auxiliary variables for C, $l = l(C)$, $FV(Q) \cap X = \emptyset$, $FV(Q) \subseteq A$ and $X \cap PV(\Gamma) = \emptyset$.*

If $Safe_{3n+2^n l}(C, s, h, \rho', \Gamma, Q, A \cup X)$ 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, where $\rho = (Locked(C \setminus X), L, D)$ such that $L \cup D = Res(\Gamma) \setminus Locked(C \setminus X)$.

Proof. Let $C \in wf\_cmd$, $s, s' \in \mathcal{S}$, $h \in \mathcal{H}$, $\rho, \rho' \in \mathcal{O}$, $Q \in \mathbf{Astn}$, $A, X \subseteq \mathbf{Var}$, Γ a well-formed resource context and $l \in \mathbb{N}_0$ such that $\rho' = (Locked(C), Res(\Gamma) \setminus Locked(C), \emptyset)$, X is a set of auxiliary variables for C, $l = l(C)$, $FV(Q) \cap X = \emptyset$, $FV(Q) \subseteq A$, $X \cap PV(\Gamma) = \emptyset$, $Safe_{3n+2^n l}(C, s, h, \rho, \Gamma, Q, A \cup X)$ is valid and $s'(x) = s(x)$, for every $x \notin X$.

We remind that $Locked(C) = Locked(C \setminus X)$.

We will prove the proposition by induction on n.

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

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

If $C = \mathsf{skip}$, then $l = 0$. From $Safe_{3n+2^n l}(C, s, h, \rho, \Gamma, Q, A \cup X)$, it is immediate that

$$s, h \models Q.$$

Because $s(y) = s'(y)$, for every $y \in FV(Q)$, we have that

$$s', h \models Q.$$

If $C = \mathsf{x{:=}e}$, $\mathsf{x} \in X$, then $l = 1$. Consider the transition given by $(ASSIGN)$

$$C, (s, h, \rho) \rightarrow_p \mathsf{skip}, (s[\mathsf{x} : v], h, \rho),$$

where $v = s(\mathsf{e})$.

Then, we know that $Safe_{3n+2^n l-1}(\mathsf{skip}, s[\mathsf{x} : v], h, \rho, \Gamma, Q, A \cup X)$ is valid. Therefore

$$s[\mathsf{x} : v], h \models Q.$$

Note that $s[\mathsf{x} : v](y) = s'(y)$, for every $y \in FV(Q)$. Then we have that

$$s', h \models Q.$$

The analysis above show that the property (i) of $Safe_n(C \setminus X, s', h, \rho, \Gamma, Q, A)$ is true.

From $Safe_{3n+2^n l}(C, s, h, \rho', \Gamma, Q, A \cup X)$, we know that

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

By proposition 11, we have that

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

Having in account that $s(x) = s'(x)$, for every $x \in FV(C \setminus X) = FV(C) \setminus X$ and the proposition 5, we know that

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

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

Note that

$$chng(C \setminus X) = chng(C) \setminus X.$$

Again by $Safe_{3n+2^n l}(C, s, h, \rho', \Gamma, Q, A \cup X)$, we have that

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

The property (iii) of $Safe_n(C \setminus X, s', h, \rho, \Gamma, Q, A)$ is establish by the previous inequalities of sets.
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

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

First, we note that we have the following environment transition

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

Now, we suppose that the transition in the execution of $C \setminus X$ is a program transition, i.e. $\xrightarrow{A,\Gamma} = \rightarrow_p$.

From $X \cap FV(C \setminus X) = \emptyset$ and the proposition 5, we know that there is $\hat{s}'$ such that $\hat{s}'(x) = \hat{s}(x)$, for every $x \notin X$, and

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

Using the proposition 12, we know that there exist C'' and $j \leq l + 1$ such that $\hat{C} = C'' \setminus X$

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

where $\hat{s}'(x) = \hat{s}''(x)$, for every $x \notin X$, and $l'' \leq 2l - (j - 1)$, where $l'' = l(C'')$.
Considering the following $(j + 1)$-transitions

$$C, (s, h, \rho') \xrightarrow{A \cup X, \Gamma}^{j+1} C'', (\hat{s}'', \hat{h}, \hat{\rho}).$$

From $Safe_{3n+2^n l}(C, s, h, \rho', \Gamma, Q, A \cup X)$ be valid, we conclude that there exists $h'_G \subseteq \hat{h}$ such that $Safe_{n+2^n l-j-1}(C'', \hat{s}'', \hat{h} \setminus h'_G, \hat{\rho}, \Gamma, Q, A \cup X)$ is valid and

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

From $\hat{s}''(x) = \hat{s}(x)$, for every $x \notin X$, $X \cap PV(\Gamma) = \emptyset$ and the proposition 2, we obtain that

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

By the proposition 4, we have that $\hat{\rho} = (Locked(C''), \hat{L}, \hat{D})$, where $L \cup D = Res(\Gamma) \setminus Locked(C'')$.
Let $\hat{\rho}' = (Locked(C''), Res(\Gamma) \setminus Locked(C''), \emptyset)$ and consider the environment transition below

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

Hence, $Safe_{3n+2^n l-(j+2)}(C'', \hat{s}'', \hat{h} \setminus h'_G, \hat{\rho}', \Gamma, Q, A \cup X)$ is valid.
From $l'' \leq 2l - (j - 1)$, we know that

$$2^k l'' \leq 2^n l - 2^k (j - 1).$$

Because $(j - 1) \leq 2^k (j - 1)$, for every $k \geq 0$ and $j \geq 1$, we obtain that

$$2^k l'' \leq 2^n l - (j - 1).$$

Note that $2^n l - (j - 1) \geq 2^k l''$ is equivalent to

$$3n + 2^n l - (j + 2) \geq 3k + 2^k l''.$$

Using the proposition 18, we see that $Safe_{3k+2^k l''}(C'', \hat{s}'', \hat{h} \setminus h'_G, \hat{\rho}', \Gamma, Q, A \cup X)$ is valid.
By the induction hypothesis, we conclude that $Safe_k(C'' \setminus X, \hat{s}, \hat{h} \setminus h'_G, \hat{\rho}, \Gamma, Q, A)$ is valid.

To finish, we suppose that the transition is an environment transition,i.e. $\xrightarrow{A,\Gamma} = \xrightarrow{A,\Gamma}_e$.
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 $\hat{\rho} = (Locked(C \setminus X), L, D)$, such that$L \cup D = Res(\Gamma) \setminus Locked(C \setminus X)$.
Let $\hat{s}' \in \mathcal{S}$, such that $\hat{s}'(x) = \hat{s}(x)$, if $x \notin X$, and $\hat{s}'(x) = s(x)$, if $x \in X$.
We have the following environment transformation

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

where $A'' = A \cup X \cup \bigcup_{r \in Locked(C)} PV(r)$.
Then, we can consider the environment transition

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

By the proposition 18 and the previous transition, we obtain that $Safe_{3k+2^{kl}}(C, \hat{s}', h, \rho', \Gamma, Q, A \cup X)$ is valid.
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. □

4.4 Soundness

In this section, we finally prove the soundness of Concurrent Separation Logic with respect to the operational semantics.

Theorem 2. *If $\Gamma \vdash_A \{P\}C\{Q\}$ is a derivable well-formed specification, then $\Gamma \models \{P\}C\{Q\}$.*

The proof of this result is a immediate consequence of the next theorem and the theorem 1.

Theorem 3. *Let $C \in \boldsymbol{Com}$, $P, Q \in \boldsymbol{Astn}$, Γ a well-formed resource context and $A \subseteq \boldsymbol{Var}$.*
If $\Gamma \vdash_A \{P\}C\{Q\}$ is a derivable well-formed specification, then for every $s \in \mathcal{S}$, $h \in \mathcal{H}$ 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 $L \cup D = Res(\Gamma)$.

Proof. Let $C \in \mathbf{Com}$, $P, Q \in \mathbf{Astn}$, Γ a well-formed resource context, $A \subseteq \mathbf{Var}$, $s \in \mathcal{S}$ and $h \in \mathcal{H}$ such that $\Gamma \vdash_A \{P\}C\{Q\}$ is a derivable well-formed specification and $s, h \models P$. We will prove the proposition by induction on the inference rules.

$(SKIP)$

We have that $C = \mathsf{skip}$, $Q = P$ and $FV(P) \subseteq A$. By the proposition 19 and $s, h \models P$, we know that $Safe_n(\mathsf{skip}, s, h, (\emptyset, L, D), \Gamma, P, A)$ is valid, for every $n \in \mathbb{N}_0$, where $L \cup D = Res(\Gamma)$.

$(ASSIGNMENT)$

We have $C = \mathsf{x{:=}e}$, $\mathsf{x} \notin PV(\Gamma)$, $\{\mathsf{x}\} \cup FV(Q) \cup FV(\mathsf{e}) \subseteq A$ and $P = Q[\mathsf{e/x}]$. We will prove by induction on n that, if $s, h \models Q[\mathsf{e/x}]$ and $\mathsf{x} \notin PV(\Gamma)$, then $Safe_n(\mathsf{x{:=}e}, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is valid, where $L \cup D = Res(\Gamma)$.
For $n = 0$, it is trivial. Let $n = k + 1$.
The property (i) of $Safe_n(\mathsf{x{:=}e}, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is immediate, because $\mathsf{x{:=}e} \neq \mathsf{skip}$.
The property (ii) of $Safe_n(\mathsf{x{:=}e}, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is valid, because the command does not abort.

We have that $chng(\mathsf{x}:=\mathsf{e}) = \{\mathsf{x}\}$. From $\mathsf{x} \notin PV(\Gamma)$, it follows that

$$chng(\mathsf{x}:=\mathsf{e}) \cap \bigcup_{r \in L \cup D} PV(r) = \emptyset.$$

Hence the property (iii) of $Safe_n(\mathsf{x}:=\mathsf{e}, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is valid.

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

$$\mathsf{x}:=\mathsf{e}, (s, h \uplus h_G, (\emptyset, L, D)) \xrightarrow{A,\Gamma} C', (s', h', \rho').$$

Next we consider the possible transitions.

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

We have that $C' = \mathsf{skip}$, $s' = s[\mathsf{x} : v]$, $h' = h \uplus h_G$ and $\rho' = (\emptyset, L, D)$, where $v = s(\mathsf{e})$. From $\mathsf{x} \notin PV(\Gamma)$, we have that

$$s[\mathsf{x} : v], h_G \models \circledast_{r \in D} \Gamma(r).$$

We just need to check that $Safe_k(\mathsf{skip}, s[\mathsf{x} : v], h, (\emptyset, L, D), \Gamma, Q, A)$ is valid.

By the proposition 3 and $s, h \models Q[\mathsf{e}/\mathsf{x}]$, we know that

$$s[\mathsf{x} : v], h \models Q.$$

From $FV(Q) \subseteq A$ and the proposition 19, we have that $Safe_k(\mathsf{skip}, s[\mathsf{x} : v], h, (\emptyset, L, D), \Gamma, Q, A)$ is valid.

If the transition is given by (ENV).

We have $C' = \mathsf{x}:=\mathsf{e}$,

$$(s, h, (\emptyset, L, D)) \overset{A}{\rightsquigarrow} (s', h, \rho')$$

and there exists $h'_G \subseteq h'$ such that $h' = h \uplus h'_G$ and

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

From $\{\mathsf{x}\} \cup FV(Q) \cup FV(\mathsf{e}) \subseteq A$, we know that

$$s', h \models Q[\mathsf{e}/\mathsf{x}].$$

Moreover $\rho' = (\emptyset, L', D')$, where $L' \cup D' = L \cup D = res(\Gamma)$.

Therefore, by induction hypothesis we have that $Safe_k(\mathsf{x}:=\mathsf{e}, s', h, \rho', \Gamma, Q, A)$ is valid.

$(LOOKUP)$

We have that $C = \mathsf{x}:=[\mathsf{e}]$, $Q = R \wedge \mathsf{e} \mapsto \mathsf{e}'$, $P = R[\mathsf{e}'/\mathsf{x}] \wedge \mathsf{e} \mapsto \mathsf{e}'$, $\mathsf{x} \notin FV(\mathsf{e}, \mathsf{e}')$, $\mathsf{x} \notin PV(\Gamma)$ and $\{\mathsf{x}\} \cup FV(R) \cup FV(\mathsf{e}, \mathsf{e}') \subseteq A$.

We prove by induction on n that, if $s, h \models R[\mathsf{e}'/\mathsf{x}] \wedge \mathsf{e} \mapsto \mathsf{e}'$, $\mathsf{x} \notin FV(\mathsf{e}, \mathsf{e}')$ and $\mathsf{x} \notin PV(\Gamma)$, then $Safe_n(\mathsf{x}:=[\mathsf{e}], s, h, (\emptyset, L, D), \Gamma, Q, A)$ is valid, where $L \cup D = Res(\Gamma)$.

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

The property (i) of $Safe_n(\mathsf{x}:=[\mathsf{e}], s, h, (\emptyset, L, D), \Gamma, Q, A)$ is immediate, because $\mathsf{x}:=[\mathsf{e}] \neq \mathsf{skip}$.

From $s, h \models \mathsf{e} \mapsto \mathsf{e}'$, it follows the property (ii) of $Safe_n(\mathsf{x}:=[\mathsf{e}], s, h, (\emptyset, L, D), \Gamma, Q, A)$.

We have that $chng(\mathsf{x}:=[\mathsf{e}]) = \{\mathsf{x}\}$. Because $\mathsf{x} \notin PV(\Gamma)$, we have that

$$chng(\mathsf{x}:=[\mathsf{e}]) \cap \bigcup_{r \in L \cup D} PV(r) = \emptyset.$$

The previous establish the property (iii) of $Safe_n(\mathsf{x}:=[\mathsf{e}], s, h, (\emptyset, L, D), \Gamma, Q, A)$.

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

$$\mathsf{x}:=[\mathsf{e}], (s, h \uplus h_G, (\emptyset, L, D)) \xrightarrow{A,\Gamma} C', (s', h', \rho').$$

Next we consider the possible transitions.

If the transition is given by $(READ)$.
We have $C' = \mathsf{skip}$, $s' = s[\mathsf{x} : v']$, $h' = h \uplus h_G$ and $\rho' = (\emptyset, L, D)$, where $v' = s(\mathsf{e'})$.
From $\mathsf{x} \notin PV(\Gamma)$, we know that

$$s[\mathsf{x} : v'], h_G \models \circledast_{r \in D} \Gamma(r).$$

We just need to check that $Safe_k(\mathsf{skip}, s[\mathsf{x} : v'], h, (\emptyset, L, D), \Gamma, Q, A)$.
By proposition 3 and $s, h \models R[\mathsf{e'}/\mathsf{x}] \wedge \mathsf{e} \mapsto \mathsf{e'}$, we have that

$$s[\mathsf{x} : v'], h \models R \wedge \mathsf{e} \mapsto \mathsf{e'}.$$

Therefore by the proposition 19, $Safe_k(\mathsf{skip}, s[\mathsf{x} : v'], h, (\emptyset, L, D), \Gamma, Q, A)$ is valid.

If the transition is given by (ENV). We have $C' = \mathsf{x}{:=}[\mathsf{e}]$,

$$(s, h, (\emptyset, L, D)) \overset{A}{\rightsquigarrow} (s', h, \rho')$$

and there exists $h'_G \subseteq h'$ such that $h' = h \uplus h'_G$ and

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

From $FV(R) \cup \{\mathsf{e}, \mathsf{e'}, \mathsf{x}\} \subseteq A$, we have that

$$s', h \models R[\mathsf{e'}/\mathsf{x}] \wedge \mathsf{e} \mapsto \mathsf{e'}.$$

Note that $\rho' = (\emptyset, L', D')$, where $L' \cup D' = L \cup D = Res(\Gamma)$.
Therefore by induction hypothesis, we have that $Safe_k(\mathsf{x}{:=}[\mathsf{e}], s', h, \rho', \Gamma, R \wedge \mathsf{e} \mapsto \mathsf{e'}, A)$ is valid.

$(UPDATE)$

We have that $C = [\mathsf{e}]{:=}\mathsf{e'}$, $Q = \mathsf{e} \mapsto \mathsf{e'}$, $P = \mathsf{e} \mapsto -$ and $FV(\mathsf{e}, \mathsf{e'}) \subseteq A$.
We prove by induction on n that, if $s, h \models \mathsf{e} \mapsto -$, then $Safe_n([\mathsf{e}]{:=}\mathsf{e'}, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is valid, where $L \cup D = Res(\Gamma)$.
For $n = 0$, it is trivial. Let $n = k + 1$.
The property (i) of $Safe_n([\mathsf{e}]{:=}\mathsf{e'}, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is immediate, because $[\mathsf{e}]{:=}\mathsf{e'} \neq \mathsf{skip}$.
From $s, h \models \mathsf{e} \mapsto -$, it follows the property (ii) of $Safe_n([\mathsf{e}]{:=}\mathsf{e'}, s, h, (\emptyset, L, D), \Gamma, Q, A)$.
We have that $chng([\mathsf{e}]{:=}\mathsf{e'}) = \emptyset$. Then, the property (iii) of $Safe_n([\mathsf{e}]{:=}\mathsf{e'}, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is immediate.
Let h_G, C', s', h' and ρ' such that $h_G \perp h$, $s, h_G \models \circledast_{r \in D} \Gamma(r)$ and

$$[\mathsf{e}]{:=}\mathsf{e'}, (s, h \uplus h_G, (\emptyset, L, D)) \xrightarrow{A,\Gamma} C', (s', h', \rho').$$

Above, we consider the possible transitions.

If the transition is given by (WRI). We have $C' = \mathsf{skip}$, $s' = s$, $h' = (h \uplus h_G)[s(\mathsf{e}) : s(\mathsf{e'})]$ and $\rho' = (\emptyset, L, D)$.
Because $s, h \models \mathsf{e} \mapsto -$, we know that $s(\mathsf{e}) \in dom(h)$ and we can rewrite

$$h' = h[s(\mathsf{e}) : s(\mathsf{e'})] \uplus h_G.$$

Taking $h'_G = h_G$. Then

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

We just need to check that $Safe_k(\mathsf{skip}, s, h[s(\mathsf{e}) : s(\mathsf{e'})], (\emptyset, L, D), \Gamma, Q, A)$.
We have that

$$s, h[s(\mathsf{e}) : s(\mathsf{e'})] \models \mathsf{e} \mapsto \mathsf{e'}.$$

Therefore, by the proposition 19, $Safe_k(\mathsf{skip}, s, h[s(\mathsf{e}) : s(\mathsf{e'})], (\emptyset, L, D), \Gamma, Q, A)$ is valid.

If the transition is given by (ENV). We have $C' = [\mathsf{e}]{:=}\mathsf{e'}$,

$$(s, h, (\emptyset, L, D)) \overset{A}{\rightsquigarrow} (s', h, \rho')$$

and there exists $h'_G \subseteq h'$ such that $h' = h \uplus h'_G$ and

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

It is sufficient to check that $Safe_k([\mathsf{e}]{:=}\mathsf{e'}, s', h, \rho', \Gamma, Q, A)$ is valid.

From $FV(\mathsf{e}) \subseteq A$, we get that $s', h \models \mathsf{e} \mapsto -$.

And note that $\rho' = (\emptyset, L', D')$, where $L' \cup D' = L \cup D = Res(\Gamma)$.

Then, it follows from the induction hypothesis that $Safe_k([\mathsf{e}]{:=}\mathsf{e'}, s', h, \rho', \Gamma, Q, A)$ is valid.

$(ALLOCATION)$

We have $C = \mathsf{x}{:=}\mathsf{cons}(\mathsf{e})$, $P = \mathsf{emp}$, $Q = \mathsf{x} \mapsto \mathsf{e}$, $\mathsf{x} \notin PV(\Gamma)$, $\mathsf{x} \notin FV(\mathsf{e})$ and $FV(\mathsf{e}) \cup \{\mathsf{x}\} \subseteq A$.

We prove by induction on n that, if $s, h \models \mathsf{emp}$, then $Safe_n(\mathsf{x}{:=}\mathsf{cons}(\mathsf{e}), s, h, (\emptyset, L, D), \Gamma, Q, A)$ is valid, where $L \cup D = Res(\Gamma)$.

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

The property (i) of $Safe_n(\mathsf{x}{:=}\mathsf{cons}(\mathsf{e}), s, h, (\emptyset, L, D), \Gamma, Q, A)$ is verified, because $\mathsf{x}{:=}\mathsf{cons}(\mathsf{e}) \neq \mathsf{skip}$.

The property (ii) of $Safe_n(\mathsf{x}{:=}\mathsf{cons}(\mathsf{e}), s, h, (\emptyset, L, D), \Gamma, Q, A)$ is trivial, because the command does not abort.

We have that $chng(\mathsf{x}{:=}\mathsf{cons}(\mathsf{e})) = \{\mathsf{x}\}$. Because $\mathsf{x} \notin PV(\Gamma)$, we know that

$$chng(\mathsf{x}{:=}\mathsf{cons}(\mathsf{e})) \cap \bigcup_{r \in L \cup D} PV(r) = \emptyset.$$

The previous establish the property (iii) of $Safe_n(\mathsf{x}{:=}\mathsf{cons}(\mathsf{e}), s, h, (\emptyset, L, D), \Gamma, Q, A)$.

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

$$\mathsf{x}{:=}\mathsf{cons}(\mathsf{e}), (s, h \uplus h_G, (\emptyset, L, D)) \xrightarrow{A,\Gamma} C', (s', h', \rho').$$

Next, we consider the possible transitions.

If the transition is given by (ALL). We have $C' = \mathsf{skip}$, $s' = s[\mathsf{x} : l]$, $h' = (h \uplus h_G)[l : s(\mathsf{e})]$ and $\rho' = (\emptyset, L, D)$, such that $l \notin dom(h \uplus h_G)$.

Rewriting the heap, we see that

$$h' = h[l : s(\mathsf{e})] \uplus h_G.$$

Taking $h'_G = h_G$. Because $\mathsf{x} \notin PV(\Gamma)$, we have that

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

We need to check that $Safe_k(\mathsf{skip}, s', h', (\emptyset, L, D), \Gamma, Q, A)$ is valid.

It is easy to see that

$$s', h[l : s(\mathsf{e})] \models \mathsf{x} \mapsto \mathsf{e}.$$

Therefore by proposition 19, we obtain that $Safe_k(\mathsf{skip}, s', h', (\emptyset, L, D), \Gamma, Q, A)$ is valid.

If the transition is given by (ENV). We have $C' = \mathsf{x}{:=}\mathsf{cons}(\mathsf{e})$,

$$(s, h, (\emptyset, L, D)) \overset{A}{\rightsquigarrow} (s', h, \rho')$$

and there exists $h'_G \subseteq h'$ such that $h' = h \uplus h'_G$ and

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

It is enough to check that $Safe_k(\mathsf{x}{:=}\mathsf{cons}(\mathsf{e}), s', h, \rho', \Gamma, Q, A)$ is valid. This follows from the induction hypothesis, because $s', h \models \mathsf{emp}$ and $\rho' = (\emptyset, L', D')$, where $L' \cup D' = L \cup D = Res(\Gamma)$.

$(DISPOSAL)$

We have $C = \mathsf{dispose(e)}$, $P = \mathsf{e} \mapsto -$, $Q = \mathsf{emp}$ and $FV(\mathsf{e}) \subseteq A$.

We prove by induction on n that, if $s, h \models P$, then $Safe_n(\mathsf{dispose(e)}, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is valid, where $L \cup D = Res(\Gamma)$.

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

The property (i) of $Safe_n(\mathsf{dispose(e)}, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is verified, because $\mathsf{dispose(e)} \neq \mathsf{skip}$.

From $s, h \models \mathsf{e} \mapsto -$, it follows the property (ii) of $Safe_n(\mathsf{dispose(e)}, s, h, (\emptyset, L, D), \Gamma, Q, A)$.

By $chng(\mathsf{dispose(e)}) = \emptyset$, we have the property (iii) of $Safe_n(\mathsf{dispose(e)}, s, h, (\emptyset, L, D), \Gamma, Q, A)$.

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

$$\mathsf{dispose(e)}, (s, h \uplus h_G, (\emptyset, L, D)) \xrightarrow{A,\Gamma} C', (s', h', \rho').$$

Next, we consider the possible transitions.

If the transition is given by $(FREE)$. We have $C' = \mathsf{skip}$, $s' = s$, $h' = (h \uplus h_G) \setminus \{s(\mathsf{e})\}$ and $\rho' = (\emptyset, L, D)$.

From $s, h \models \mathsf{e} \mapsto -$, we know that $\{s(\mathsf{e})\} = dom(h)$ and we can rewrite the heap in the following expression

$$h' = h \setminus \{s(\mathsf{e})\} \uplus h_G.$$

Consider $h'_G = h_G$. Then

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

We need to check that $Safe_k(\mathsf{skip}, s, h \setminus \{s(e)\}, (\emptyset, L, D), \Gamma, Q, A)$ is valid.

From $s, h \models \mathsf{e} \mapsto -$, we have that

$$s, h \setminus \{s(\mathsf{e})\} \models \mathsf{emp}.$$

Therefore by proposition 19, we conclude that $Safe_k(\mathsf{skip}, s, h \setminus \{s(\mathsf{e})\}, (\emptyset, L, D), \Gamma, Q, A)$ is valid.

If the transition is given by (ENV). We have $C' = \mathsf{dispose(e)}$,

$$(s, h, (\emptyset, L, D)) \overset{A}{\rightsquigarrow} (s', h, \rho')$$

and there exists $h'_G \subseteq h'$ such that $h' = h \uplus h'_G$ and

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

We just need to check that $Safe_k(\mathsf{dispose(e)}, s', h, \rho', \Gamma, Q, A)$ is valid. This follows from the induction hypothesis, because $s', h \models \mathsf{e} \mapsto -$ and $\rho' = (\emptyset, L', D')$, where $L' \cup D' = L \cup D = Res(\Gamma)$.

$(SEQUENCE)$

We have $C = C_1 \,;\, C_2$ and $A = A_1 \cup A_2$. By induction on the inference rules, there is $R \in \mathbf{Astn}$ such that

- if $s', h' \models P$, then $Safe_n(C_1, s', h', (\emptyset, L, D), \Gamma, R, A_1)$ is valid, where $L \cup D = res(\Gamma)$ and $n \geq 0$.
- if $s', h' \models R$, then $Safe_n(C_2, s', h', (\emptyset, L, D), \Gamma, Q, A_2)$ is valid, where $L \cup D = res(\Gamma)$ and $n \geq 0$.

The conclusion is immediate from proposition 20, observing that $Locked(C_1) = \emptyset$.

$(CONDITIONAL)$

We have that $C = \mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2$, $FV(P, Q) \subseteq A_1 \cap A_2$ and $A = A_1 \cup A_2$. And by induction on the inference rules,

- if $s', h' \models P \wedge B$, then $Safe_n(C_1, s', h', (\emptyset, L, D), \Gamma, Q, A_1)$ is valid, where $L \cup D = res(\Gamma)$ and $n \geq 0$; and

- if $s', h' \models P \wedge \neg B$, then $Safe_n(C_2, s', h', (\emptyset, L, D), \Gamma, Q, A_2)$ is valid, where $L \cup D = res(\Gamma)$ and $n \geq 0$.

We prove by induction on n that, if $s, h \models P$, then $Safe_n(\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is valid, where $L \cup D = res(\Gamma)$.

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

The property (i) of $Safe_n(\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is valid, because $C \neq \mathsf{skip}$.

The property (ii) of $Safe_n(\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is respected, because the command does not abort.

We have that $chng(\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2) = \emptyset$.

Then the property (iii) of $Safe_n(\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is immediate.

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

$$\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, (s, h \uplus h_G, (\emptyset, L, D)) \xrightarrow{A,\Gamma} C', (s', h', \rho').$$

Next, we consider the possible transitions.

If the transition is given by $(IF1)$. We have $C' = C_1$, $s, h \models B$, $s' = s$, $h' = h \uplus h_G$ and $\rho' = (\emptyset, L, D)$.

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

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

We just need to check that $Safe_k(C_1, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is valid.

We know that $s, h \models P \wedge B$. Hence $Safe_k(C_1, s, h, (\emptyset, L, D), \Gamma, Q, A_1)$ is valid. From $A_1 \subseteq A$, we know that

$$\overset{A_1}{\rightsquigarrow} \supseteq \overset{A}{\rightsquigarrow}.$$

Therefore $Safe_k(C_1, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is valid.

For the transition $(IF2)$ we use an identical argument.

If the transition is given by (ENV). We have $C' = \mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2$,

$$(s, h, (\emptyset, L, D)) \overset{A}{\rightsquigarrow} (s', h, \rho')$$

and there exists $h'_G \subseteq h'$ such that $h' = h \uplus h'_G$ and

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

From $FV(P) \subseteq A$, $s, h \models P$ and the proposition 2, we know that

$$s', h \models P.$$

Then by the induction hypothesis, we obtain that $Safe_k(C, s', h, \rho', \Gamma, Q, A)$ is valid.

$(LOOP)$

We have $C = \mathsf{while}\ B\ \mathsf{do}\ \tilde{C}$ and $FV(P) \cup FV(B) \subseteq A$.

By induction on inference rules, if $s', h' \models P \wedge B$, then $Safe_n(\tilde{C}, s', h', (\emptyset, L, D), \Gamma, P, A)$ is valid, where $L \cup D = res(\Gamma)$ and $n \geq 0$.

We will prove by induction on n that, if $s, h \models P$, then $Safe_n(\mathsf{while}\ B\ \mathsf{do}\ \tilde{C}, s, h, (\emptyset, L, D), \Gamma, P \wedge \neg B, A)$ is valid, where $L \cup D = res(\Gamma)$.

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

From $\mathsf{while}\ B\ \mathsf{do}\ \tilde{C} \neq \mathsf{skip}$, we get the property (i) of $Safe_n(\mathsf{while}\ B\ \mathsf{do}\ \tilde{C}, s, h, (\emptyset, L, D), \Gamma, P \wedge \neg B, A)$.

The property (ii) of $Safe_n(\mathsf{while}\ B\ \mathsf{do}\ \tilde{C}, s, h, (\emptyset, L, D), \Gamma, P \wedge \neg B, A)$ is verified, because the command does not abort.

The property (iii) of $Safe_n(\mathsf{while}\ B\ \mathsf{do}\ \tilde{C}, s, h, (\emptyset, L, D), \Gamma, P \wedge \neg B, A)$ is immediate, because we know that $chng(\mathsf{while}\ B\ \mathsf{do}\ \tilde{C}) = \emptyset$.

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

$$\mathsf{while}\ B\ \mathsf{do}\ \tilde{C}, (s, h \uplus h_G, (\emptyset, L, D)) \xrightarrow{A,\Gamma} C', (s', h', \rho').$$

Next, we consider the possible transitions.

If the transition is given by $(LOOP)$. We have $C' = \mathsf{if}\ B\ \mathsf{then}\ \tilde{C}\ ;\mathsf{while}\ B\ \mathsf{do}\ \tilde{C}\ \mathsf{else\ skip}$, $s' = s$, $h' = h \uplus h_G$ and $\rho' = (\emptyset, L, D)$.

Considering $h'_G = h_G$, we know that

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

We just need to check that $Safe_k(\mathsf{if}\ B\ \mathsf{then}\ \tilde{C}\ ;\mathsf{while}\ B\ \mathsf{do}\ \tilde{C}\ \mathsf{else\ skip}, s, h, (\emptyset, L, D), \Gamma, P \wedge \neg B, A)$ is valid.

By induction on the inference rules, if $s, h \models P \wedge B$, then $Safe_k(\tilde{C}, s, h, (\emptyset, L, D), \Gamma, P, A)$ is valid.

By induction on n, we know that if $s, h \models P$, then $Safe_k(\mathsf{while}\ B\ \mathsf{do}\ \tilde{C}, s, h, (\emptyset, L, D), \Gamma, P \wedge \neg B, A)$ is valid.

Noting that $Locked(\tilde{C}) = \emptyset$, we can apply the proposition 20 to conclude that if $s, h \models P \wedge B$, then $Safe_k(\tilde{C}\ ;\mathsf{while}\ B\ \mathsf{do}\ \tilde{C}, s, h, (\emptyset, L, D), \Gamma, P \wedge \neg B, A)$.

From proposition 19, we obtain that if $s, h \models P \wedge \neg B$, then $Safe_k(\mathsf{skip}, s, h, (\emptyset, L, D), \Gamma, P \wedge \neg B, A)$ is valid.

Repeating the argument used in the $(CONDITIONAL)$ rule, we obtain that if $s, h \models P$, then $Safe_k(\mathsf{if}\ B\ \mathsf{then}\ \tilde{C}\ ;\mathsf{while}\ B\ \mathsf{do}\ \tilde{C}\ \mathsf{else\ skip}, s, h, (\emptyset, L, D), \Gamma, P \wedge \neg B, A)$ is valid.

If the transition is given by (ENV). We have $C' = \mathsf{while}\ B\ \mathsf{do}\ \tilde{C}$,

$$(s, h, (\emptyset, L, D)) \overset{A}{\rightsquigarrow} (s', h, \rho')$$

and there exists $h'_G \subseteq h'$ such that $h' = h \uplus h'_G$ and

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

From $FV(P) \subseteq A$, $s, h \models P$ and the proposition 2, it follows that

$$s', h \models P.$$

Therefore by induction on n we have that $Safe_k(\mathsf{while}\ B\ \mathsf{do}\ \tilde{C}, s', h, (\emptyset, L', D'), \Gamma, Q, A)$ is valid.

$(PARALLEL)$

We have $C = C_1 \parallel C_2$, $P = P_1 * P_2$, $Q = Q_1 * Q_2$, $A = A_1 \cup A_2$, $FV(P_1, Q_1) \subseteq A_1$, $FV(P_2, Q_2) \subseteq A_2$ and $A_1 \cap mod(C_2) = A_2 \cap mod(C_1) = \emptyset$.

By induction on the inference rules,

- if $s', h' \models P_1$, then $Safe_n(C_1, s', h', (\emptyset, L, D), \Gamma, Q_1, A_1)$, where $L \cup D = res(\Gamma)$ and $n \geq 0$.
- if $s', h' \models P_2$, then $Safe_n(C_2, s', h', (\emptyset, L, D), \Gamma, Q_2, A_2)$, where $L \cup D = res(\Gamma)$ and $n \geq 0$.

From $s, h \models P$, we know that there are h_1, h_2 such that $h = h_1 \uplus h_2$,

$$s, h_1 \models P_1 \quad s, h_2 \models P_2.$$

Hence $Safe_n(C_1, s, h_1, (\emptyset, L, D), \Gamma, Q_1, A_1)$ and $Safe_n(C_2, s, h_2, (\emptyset, L, D), \Gamma, Q_2, A_2)$ are valid.

Applying the proposition 21, we get that $Safe_n(C_1 \parallel C_2, s, h, (\emptyset, L, D), \Gamma, Q_1 * Q_2, A_1 \cup A_2)$ is valid.

$(CRITICAL\ REGION)$

We have $C = \mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}$, $FV(P,Q) \subseteq A$ and $\Gamma = \Gamma', r(X) : R$ is a well-formed resource context.

By induction on the inference rules, if $s', h' \models (P \wedge B) * R$, $L' \cup D' = res(\Gamma')$ and $n \geq 0$ then $Safe_n(\tilde{C}, s', h', (\emptyset, L', D'), \Gamma', Q * R, A \cup X)$ is valid.

We will prove by induction on n that, if $s, h \models P$ and $L \cup D = res(\Gamma) = res(\Gamma') \cup \{r\}$, then $Safe_n(\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is valid.

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

The property (i) of $Safe_n(\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is verified, because $C \neq \mathsf{skip}$.

We have the property (ii) of $Safe_n(\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}, s, h, (\emptyset, L, D), \Gamma, Q, A)$, because $r \in L \cup D$.

We know that $chng(\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}) = \emptyset$.

Then the property (iii) of $Safe_n(\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is immediate.

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

$$\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}, (s, h \uplus h_G, (\emptyset, L, D)) \xrightarrow{A,\Gamma} C', (s', h', \rho').$$

Next, we consider the possible transitions.

If the transition is given by $(WITH0)$. We have $C' = \mathsf{within}\ r\ \mathsf{do}\ \tilde{C}$, $r \in D$, $s' = s$, $h' = h \uplus h_G$, $\rho' = (\{r\}, L, D \setminus \{r\})$ and $s, h \uplus h_G \models B$.

From $r \in D$ and $s, h_G \models \circledast_{\hat{r} \in D} \Gamma(\hat{r})$, we know that there exist h_R and h'_G such that $h_G = h_R \uplus h'_G$ and

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

We just need to check that $Safe_k(\mathsf{within}\ r\ \mathsf{do}\ \tilde{C}, s, h \uplus h_R, (\{r\}, L, D \setminus \{r\}), \Gamma, Q, A)$ is valid.

We have that

$$s, h \uplus h_R \models (P \wedge B) * R.$$

Hence $Safe_k(\tilde{C}, s, h, (\emptyset, L, D \setminus \{r\}), \Gamma', Q * R, A \cup X)$ is valid, because $L \cup D \setminus \{r\} = Res(\Gamma')$.

By the proposition 22, we conclude that $Safe_k(\mathsf{within}\ r\ \mathsf{do}\ \tilde{C}, s, h \uplus h_R, (\{r\}, L, D \setminus \{r\}), \Gamma, Q, A)$ is valid.

If the transition is given by (ENV). We have $C' = \mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}$,

$$(s, h, (\emptyset, L, D)) \overset{A}{\rightsquigarrow} (s', h, (\emptyset, L', D'))$$

and there exists $h'_G \subseteq h'$ such that $h' = h \uplus h'_G$ and

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

Note that $L' \cup D' = L \cup D = Res(\Gamma) \cup \{r\}$.

It is sufficient to prove that $Safe_k(\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}, s', h, (\emptyset, L', D'), \Gamma, Q, A)$ is valid.

From $FV(P) \subseteq A$, $s, h \models P$ and the proposition 2, it follows that

$$s', h \models P.$$

Therefore by induction hypothesis on n, $Safe_k(\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}, s', h, (\emptyset, L', D'), \Gamma, Q, A)$ is valid.

$(LOCAL\ RESOURCE)$

We have $C = \mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}$ and $\Gamma' = \Gamma, r(X) : R$ is a well-formed resource context.

By induction on the inference rules, if $s', h' \models P$, then $Safe_n(\tilde{C}, s', h', (\emptyset, L', D'), \Gamma', Q, A)$ is valid, where $n \geq 0$ and $L \cup D = res(\Gamma) \cup \{r\}$.

We prove by induction on n that, if $s, h \models P * R$, then $Safe_n(\mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}, s, h, \rho, \Gamma, Q * R, A \cup X)$ is valid, where $\rho = (\emptyset, L, D)$ and $L \cup D = res(\Gamma)$.

From $s, h \models P * R$, there exists $h_R \subseteq h$ such that

$$s, h_R \models R, \qquad s, h \setminus h_R \models P.$$

Hence $Safe_n(\tilde{C}, s, h \setminus h_R, (\emptyset, L, D \cup \{r\}), \Gamma', Q, A)$ is valid.

Note that $r \notin Locked(\tilde{C})$, because $Locked(\tilde{C}) = \emptyset$.

By the proposition 23, we conclude that $Safe_n(\mathsf{resource}\ r\ \mathsf{in}\ \tilde{C}, s, h, (\emptyset, L, D), \Gamma, Q * R, A \cup X)$ is valid.

$(RENAMING)$

We have $r' \notin Res(C)$ and $r' \notin Res(\Gamma)$. And by induction hypothesis on the inference rules, if $s', h' \models P$, then $Safe_n(C[r'/r], s', h', (\emptyset, L, D), \Gamma[r'/r], Q, A)$ is valid, where $L \cup D = res(\Gamma[r'/r])$ and $n \geq 0$.

From $s, h \models P$, we know that $Safe_n(C[r'/r], s', h', (\emptyset, L, D)[r'/r], \Gamma[r'/r], Q, A)$ is valid for every $n \geq 0$ and $L \cup D[r'/r] = res(\Gamma)[r'/r]$.

Note that

$$L \cup D[r'/r] = res(\Gamma)[r'/r] \ \text{ iff } \ L \cup D = res(\Gamma).$$

By the proposition 26, we conclude that $Safe_n(C, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is valid, for every $n \geq 0$ and $L \cup D = res(\Gamma)$.

$(FRAME)$

By induction on the inference rules, if $s', h' \models P$, then $Safe_n(C, s', h', (\emptyset, L, D), \Gamma, Q, A)$ is valid, where $L \cup D = res(\Gamma)$ and $n \geq 0$.

Let s, h, R such that $s, h \models P * R$ and $mod(C) \cap FV(R) = \emptyset$. There is a $h_R \subseteq h$ such that $s, h_R \models R$ and $s, h \setminus h_R \models P$.

By induction on the inference rules, we have that $Safe_n(C, s, h \setminus h_R, (\emptyset, L, D), \Gamma, Q, A)$ is valid.

By proposition 24, we conclude that $Safe_n(C, s, h \uplus h_R, (\emptyset, L, D), \Gamma, Q * R, A \cup FV(R))$ is valid.

$(CONSEQUENCE)$

We have $A' \subseteq A$, $\models P \Rightarrow P'$ and $\models Q' \Rightarrow Q$.

By induction on the inference rules, if $s', h' \models P'$, then $Safe_n(C, s', h', (\emptyset, L, D), \Gamma, Q', A')$ is valid, where $L \cup D = res(\Gamma)$ and $n \geq 0$.

From $s, h \models P$ and $\models P \Rightarrow P'$, we know that $s, h \models P'$.

Hence $Safe_n(C, s, h, (\emptyset, L, D), \Gamma, Q, A')$ is valid.

First, we note that $Safe_n(C, s, h, (\emptyset, L, D), \Gamma, Q, A)$ is valid, because

$$\overset{A}{\leftsquigarrow} \subseteq \overset{A'}{\leftsquigarrow}.$$

From $\models Q' \Rightarrow Q$, it follows that $Safe_n(C, s, h, (\emptyset, L, D), \Gamma, Q', A)$ is valid.

$(AUXILIARY)$

We have X is a auxiliary set for C, $X \cap FV(P, Q) = \emptyset$ and $X \cap PV(\Gamma) = \emptyset$.

By induction on the inference rules, if $s', h' \models P$, then $Safe_n(C, s', h', (\emptyset, L', D'), \Gamma, Q, A \cup X)$ is valid, where $L' \cup D' = res(\Gamma)$ and $n \geq 0$.

From $s, h \models P$, we know that $Safe_{3n+2^n l}(C, s, h, (\emptyset, res(\Gamma), \emptyset), \Gamma, Q, A \cup X)$ is valid, for every $n \geq 0$, where $l = l(C)$.

Applying the proposition 27, we get that $Safe_n(C \setminus X, s, h, (\emptyset, L, D), \Gamma, Q, A)$, for every $n \geq 0$, where $L \cup D = res(\Gamma)$.

$(CONJUNCTION)$

By induction on the inference rules, if $s', h' \models P_i$, then $Safe_n(C, s', h', (\emptyset, L', D'), \Gamma, Q_i, A_i)$ is valid, where $L' \cup D' = res(\Gamma)$ and $n \geq 0$, for $i = 1, 2$.

From $s, h \models P_1 \wedge P_2$, we know that $s, h \models P_1$ and $s, h \models P_2$.

Hence $Safe_n(C, s, h, (\emptyset, L', D'), \Gamma, Q_i, A_i)$ is valid, where $L' \cup D' = res(\Gamma)$, $n \geq 0$ and $i = 1, 2$.

Next, we prove by induction on the n that if $Safe_n(C, s, h, (\emptyset, L, D), \Gamma, Q_i, A_i)$ is valid, for $i = 1, 2$, then $Safe_n(C, s, h, (\emptyset, L, D), \Gamma, Q_1 \wedge Q_2, A_1 \cup A_2)$ is valid.

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

If $C = \mathsf{skip}$, then $s, h \models Q_i$, for $i = 1, 2$. Hence $s, h \models Q_1 \wedge Q_2$.

The property (i) of $Safe_n(C, s, h, (\emptyset, L, D), \Gamma, Q_1 \wedge Q_2, A_1 \cup A_2)$ is verified.

From $Safe_n(C, s, h, (\emptyset, L, D), \Gamma, Q_i, A_i)$, $i = 1, 2$, we know that

$$C, (s, h, (\emptyset, L, D)) \not\rightarrow_p \mathsf{abort}.$$

Hence we have the property (ii) of $Safe_n(C, s, h, (\emptyset, L, D), \Gamma, Q_1 \wedge Q_2, A_1 \cup A_2)$.

From $Safe_n(C, s, h, (\emptyset, L, D), \Gamma, Q_i, A_i)$, $i = 1, 2$, we know that

$$chng(\mathsf{with}\ r\ \mathsf{when}\ B\ \mathsf{do}\ \tilde{C}) \cap \bigcup_{r \in L \cup D} PV(r) = \emptyset.$$

Then we have the property (iii) of $Safe_n(C, s, h, (\emptyset, L, D), \Gamma, Q_1 \wedge Q_2, A_1 \cup A_2)$.

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

$$C, (s, h \uplus h_G, (\emptyset, L, D)) \xrightarrow{A_1 \cup A_2, \Gamma} C', (s', h', \rho').$$

Next we consider the possible transitions.

If the transition is a transition of the program, i.e. $\xrightarrow{A_1 \cup A_2, \Gamma} = \rightarrow_p$.

For $i = 1, 2$. By $Safe_n(C, s, h, (\emptyset, L, D), \Gamma, Q_i, A_i)$, we know that there is $h'_{G_i} \subseteq h'$ such that $Safe_k(C', s', h' \setminus h'_{G_i}, \rho', \Gamma, Q_i, A_i)$ is valid and

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

Using that $\circledast_{r \in D'} \Gamma(r)$ is precise, we have that $h'_{G_1} = h'_{G_2}$. Consider $h'_G = h'_{G_1}$. We have that

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

From the induction hypothesis on n, we also have that $Safe_k(C', s', h' \setminus h'_G, \rho', \Gamma, Q_1 \wedge Q_2, A_1 \cup A_2)$ is valid.

If the transition is given by (ENV). Let $A' = A_1 \cup A_2 \cup \bigcup_{r \in Locked(C)} PV(r)$, $A'_1 = A_1 \cup \bigcup_{r \in Locked(C)} PV(r)$ and $A'_2 = A_2 \cup \bigcup_{r \in Locked(C)} PV(r)$.

We have that $C' = C$,

$$(s, h, (\emptyset, L, D)) \overset{A'}{\rightsquigarrow} (s', h, (\emptyset, L', D'))$$

and there exists $h'_G \subseteq h'$ such that $h' = h \uplus h'_G$ and

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

It is sufficient to prove that $Safe_k(C', s', h, \rho', \Gamma, Q_1 \wedge Q_2, A_1 \cup A_2)$ is valid.

Having in mind that $\overset{A'}{\rightsquigarrow} \subseteq \overset{A'_i}{\rightsquigarrow}$, for $i = 1, 2$. We have that $Safe_k(C', s', h' \setminus h'_{G_i}, \rho', \Gamma, Q_i, A_i)$ is valid, for $i = 1, 2$.

Therefore by induction hypothesis on n, we have that $Safe_k(C', s', h, \rho', \Gamma, Q_1 \wedge Q_2, A_1 \cup A_2)$ is valid. $\square$

5 Conclusion

In this work, we presented a structural operational semantics, which is widely understood, and we prove the soundness of Concurrent Separation Logic with respect to this operational semantics. Moreover, we showed that if a program is partially correct in Concurrent Separation Logic, then this program can be safely integrated in any environment, which respects the variables protected by the rely-set and resources.

We believe that the present text can be useful to prove the soundness of new extensions to Concurrent Separation Logic. We predict that the comparison between the environment transition used by Vafeiadis [16] and the environment transition introduced here can be relevant to understand the differences between RGSep and Concurrent Separation Logic.

Brookes proved the equivalence between Concurrent Separation Logic and Syntactic Control Interference Separation Logic [5]. The operational semantics presented here can be adapted to express fractional permissions of variables. Furthermore, the soundness of Syntactic Control Interference Separation Logic with respect to a structural operational semantics should follows by adapting the environment transition and the results presented here.

References

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

[2] J. Boyland. Checking interference with fractional permissions. In R. Cousot, editor, *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. *Electronic Notes in Theoretical Computer Science*, 276:5–28, 2011.

[5] S. Brookes. Syntactic control of interference and concurrent separation logic. *Electronic Notes in Theoretical Computer Science*, 286:87–102, 2012.

[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 L. Fribourg, editor, *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 A. K. Chandra, D. Wotschke, E. P. Friedman, and M. A. Harrison, editors, *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 J. Field and M. Hicks, editors, *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] J. C. Reynolds. An introduction to separation logic. `http://www.cs.cmu.edu/~jcr/copenhagen08.pdf`, 2008.

[15] V. Vafeiadis. Concurrent separation logic and operational semantics. *Electronic Notes in Theoretical Computer Science*, 276:335–351, 2011.

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

[17] G. Winskel. *The formal semantics of programming languages - an introduction*. Foundation of computing series. MIT Press, 1993.