Owicki-Gries Reasoning for Weak Memory Models

Ori Lahav and Viktor Vafeiadis

Max Planck Institute for Software Systems (MPI-SWS)

Abstract. We show that even in the absence of auxiliary variables, the well-known Owicki-Gries method for verifying concurrent programs is unsound for weak memory models. By strengthening its non-interference check, however, we obtain OGRA, a program logic that is sound for reasoning about programs in the release-acquire fragment of the C11 memory model. We demonstrate the usefulness of this logic by applying it to several challenging examples, ranging from small litmus tests to an implementation of the RCU synchronization primitives.

1 Introduction

In 1976, Owicki and Gries [10] introduced a proof system for reasoning about concurrent programs, which formed the basis of rely/guarantee reasoning. Their system includes the usual Hoare logic rules for sequential programs, a rule for introducing auxiliary variables, and the following parallel composition rule:

$$\frac{\{P_1\}\, c_1\, \{Q_1\} \quad \{P_2\}\, c_2\, \{Q_2\} \quad \text{the two proofs are non-interfering}}{\{P_1 \wedge P_2\}\, c_1 \parallel c_2\, \{Q_1 \wedge Q_2\}}$$

This rule allows one to compose two verified programs into a verified concurrent program that assumes both preconditions and ensures both postconditions. The soundness of this rule requires that the two proofs are *non-interfering*, namely that every assertion R in the one proof is stable under any $\{P\}x := e$ (guarded) assignment in the other and vice versa; i.e., for every such pair, $R \wedge P \vdash R[e/x]$.

The Owicki-Gries system (OG) assumes a fairly simple but unrealistic concurrency model: *sequential consistency* (SC) [7]. This is essential: OG is complete for verifying concurrent programs under SC [12], and is therefore unsound under a weakly consistent memory semantics, such as TSO [9]. Auxiliary variables are instrumental in achieving completeness—without them, OG is blatantly incomplete; e.g., it cannot verify that $\{x = 0\}\, x \overset{\mathsf{at}}{:=} x + 1 \parallel x \overset{\mathsf{at}}{:=} x + 1\, \{x = 2\}$ (where "$\overset{\mathsf{at}}{:=}$" denotes atomic assignment).

Nevertheless, many useful OG proofs do not use auxiliary variables, and one might wonder whether such proofs are sound under weak memory models. This is sadly not the case. Figure 1 presents an OG proof that a certain program cannot return $a = b = 0$ whereas under all known weak memory models it can in fact do so. Intuitively speaking, the proof is invalid under weak memory because the two threads may have different views of memory before executing each command. Thus, when the second thread terminates, the first thread may perform $a := y$ reading $y = 0$ and storing 0 in a, thereby

Due to space limits, supplementary material including full proofs and further examples is available at: `http://plv.mpi-sws.org/ogra/`.

$$
\begin{array}{l}
\{x = 0 \wedge y = 0 \wedge a \neq 0\} \\
\quad \{a \neq 0\} \quad \| \quad \{\top\} \\
\quad x := 1; \quad \| \quad y := 1; \\
\quad \{x \neq 0\} \quad \| \quad \{y \neq 0\} \\
\quad a := y \quad \| \quad b := x \\
\quad \{x \neq 0\} \quad \| \quad \{y \neq 0 \wedge (a \neq 0 \vee b = x)\} \\
\qquad \{a \neq 0 \vee b \neq 0\}
\end{array}
$$

Non-interference checks are trivial. For example,

$$y \neq 0 \wedge (a \neq 0 \vee b = x) \wedge a \neq 0 \vdash y \neq 0 \wedge (a \neq 0 \vee b = 1)$$

and

$$y \neq 0 \wedge (a \neq 0 \vee b = x) \wedge x \neq 0 \vdash y \neq 0 \wedge (y \neq 0 \vee b = x)$$

show stability of the last assertion of thread II under $\{a \neq 0\}x := 1$ and $\{x \neq 0\}a := y$.

Fig. 1. OG proof that the "store buffering" program cannot return $a = b = 0$. This can also be proved in the restricted OG system with one (stable) global invariant [11]. Note that OG's "indivisible assignments" condition (3.1) is met: assignments mention at most one shared location.

invalidating the second thread's last assertion. We note that $y = 0$ was also readable by the second thread, albeit at an earlier point (before the $y := 1$ assignment). This is no accident, and this observation is essential for soundness of our proposed alternative.

In this paper we identify a stronger non-interference criterion that does not assume SC semantics. Thus, while considering the effect of an assignment $\{P\}x := y$ in thread I on the validity of an assertion R in thread II, one does not get to assume that R holds for the view of thread I while reading y. In fact, in some executions, the value read for y might even be inconsistent with R. Instead, the only allowed assumption is that some assertion that held not later than R in thread II was true while reading y. Thus our condition for checking stability of R under $\{P\}x := y$ is that $R \wedge P \vdash R[v/x]$ for every value v of y that is consistent with P and some non-later assertion of thread II.

We show that OG with our stronger non-interference criterion is sound under the release-acquire (RA) fragment of the C11 memory model [6], which exhibits a good balance between performance and mathematical sanity (see, e.g., [16,17]). Soundness under TSO follows, as TSO behaviors are all observable under RA (see [1]). Formalizing the aforementioned intuitions into a soundness proof for RA executions is far from trivial. Indeed, RA is defined axiomatically without an operational semantics and without the notion of a state. As a basis for the soundness proof, we introduce such a notion and study the properties of sequences of states observed by different threads.

We believe that the results of this paper may provide new insights for understanding weak memory models, as well as a *simple* and *useful* method for proving partial correctness of concurrent programs. We demonstrate the applicability of our logic (which we call OGRA) with several challenging examples, ranging from small litmus tests to an implementation of the read-copy-update (RCU) synchronization primitives [3]. We also provide support for fence instructions by implementing them as RMWs to an otherwise unused location and for a simple class of auxiliary variables, namely *ghost values*.

Related Work. Aiming to understand and verify high-performance realistic concurrent programs, program logics for weak memory models have recently received a lot of attention (see, e.g., [4,13,18,16,14]). Most of these logics concern the TSO memory model. Only two—RSL [18] and GPS [16]—can handle RA, but have a fairly complex foundation being based on separation logic. The most advanced of the two logics, GPS, has been used (with considerable ingenuity) to verify the RCU synchronization primitives [15], but simpler examples such as "read-read coherence" seem to be beyond its power (see Fig. 8). Finally, Cohen [2] studies an alternative memory model under which OG reasoning can be performed at the execution level.

2 Preliminaries

In this section, we present a simplified programming language, whose semantics adheres to that of the release-acquire fragment of C11's memory model [1]. We assume a finite set of locations $\mathsf{Loc} = \{\nu_1, \ldots, \nu_M\}$, a finite set Val of values with a distinguished value $0 \in \mathsf{Val}$, and any standard interpreted language for expressions containing at least all locations and values. We use x, y, z as metavariables for locations, v for values, e for expressions, and denote by $e(x_1, \ldots, x_n)$ an expression in which $x_1, \ldots, x_n$ are the only mentioned locations. The language's commands are given by the following grammar:

$$c ::= \texttt{skip} \mid \texttt{if}\ e(x)\ \texttt{then}\ c\ \texttt{else}\ c \mid \texttt{while}\ e(x)\ \texttt{do}\ c \mid c\,;c \mid c \parallel c \mid x := v \mid x := e(y) \mid x \overset{y,z}{:=} e(y,z) \mid x \overset{\mathsf{at}}{:=} e(x)$$

To keep the presentation simple, expressions in assignments are limited to mention at most two locations, and those in conditionals and loops mention one location. Assignments of expression mentioning two locations also specify the order in which these locations should be read (if one of them is local, this has no observable effect). The command $x \overset{\mathsf{at}}{:=} e(x)$ is an atomic assignment corresponding to a primitive read-modify-write (RMW) instruction and, as such, mentions only one location.[1]

Now, as in the C11 formalization, the semantics of a program is defined to be its set of consistent executions [1]. An execution G is a triple $\langle A, L, E\rangle$ where:

- $A \subseteq \mathbb{N}$ is a finite set of *nodes*. We identify G with this set, e.g., when writing $a \in G$.
- L is a function assigning a *label* to each node, where a label is either $\langle \mathtt{S}\rangle$ ("Skip"), a triple of the form $\langle \mathtt{R}, x, v_r\rangle$ ("Read"), a triple of the form $\langle \mathtt{W}, x, v_w\rangle$ ("Write"), or a quadruple of the form $\langle \mathtt{U}, x, v_r, v_w\rangle$ ("Update"). For $\mathtt{T} \in \{\mathtt{S}, \mathtt{R}, \mathtt{W}, \mathtt{U}\}$, we denote by $G.\mathtt{T}$ the set of nodes $a \in A$ for which $\mathtt{T}$ is the first entry of $L(a)$, while $G.\mathtt{T}_x$ denotes the set of $a \in G.\mathtt{T}$ for which x is the second entry of $L(a)$. In addition, L induces the partial functions $G.loc : A \to \mathsf{Loc}$, $G.val_r : A \to \mathsf{Val}$, and $G.val_w : A \to \mathsf{Val}$ that respectively return (when applicable) the x, v_r and v_w components of a node.
- $E \subseteq (A \times A) \cup (A \times A \times \mathsf{Loc})$ is a set of *edges*, such that for every triple $\langle a, b, x\rangle \in E$ (*reads-from* edge) we have $a \in G.\mathtt{W}_x \cup G.\mathtt{U}_x$, $b \in G.\mathtt{S} \cup G.\mathtt{R}_x \cup G.\mathtt{U}_x$, and $G.val_w(a) = G.val_r(b)$ whenever $b \notin G.\mathtt{S}$.[2] The subset $E \cap (A \times A)$ is denoted by $G.po$ (*program order*), and $G.E_x$ denotes the set $\{\langle a, b\rangle \in A \times A \mid \langle a, b, x\rangle \in E\}$ (x-*reads-from*) for every $x \in \mathsf{Loc}$. Finally, $G.E_{all}$ denotes the set $G.po \cup \bigcup_{x \in \mathsf{Loc}} E_x$.

For all these notations, we often omit the "$G.$" prefix when it is clear from the context. Given an execution $G = \langle A, L, E\rangle$ and a set E' of edges we write $G \cup E'$ for the triple $\langle A, L, E \cup E'\rangle$ and $G \setminus E'$ for $\langle A, L, E \setminus E'\rangle$.

Definition 1. A node a in an execution G is *initial* (*terminal*) in G if $\langle b, a\rangle \notin E_{all}$ ($\langle a, b\rangle \notin E_{all}$) for every $b \in G$. An edge $\langle a, b\rangle \in po$ is *initial* (*terminal*) in G if a is initial (b is terminal) in G.

Definition 2. Let $G = \langle A, L, E\rangle$ and $G' = \langle A', L', E'\rangle$ be two executions with disjoint sets of nodes.

[1] Unlike usual OG [10], our assignments can mention more than one shared variable. In fact, our formal development does not differentiate between local and shared variables.

[2] Reads-from edges $\langle a, b, x\rangle$ with $b \in G.\mathtt{S}$ are used for defining visible states (see Definition 7).

$$
\begin{array}{ll}
[\![\texttt{skip}]\!] & = \mathcal{SG} \\
[\![\texttt{if } e(x) \texttt{ then } c_1 \texttt{ else } c_2]\!] & = \bigcup\{\mathcal{RG}(x,v); [\![c_i]\!] \mid v \in \mathsf{Val}, i \in \{1,2\}, [\![e]\!](v) = 0 \textit{ iff } i = 2\} \\
[\![\texttt{while } e(x) \texttt{ do } c]\!] & = \bigcup_{n \geq 0} (\bigcup\{\mathcal{RG}(x,v) \mid v \in \mathsf{Val}, [\![e]\!](v) \neq 0\}; [\![c]\!])^n; \\
 & \quad \bigcup\{\mathcal{RG}(x,v) \mid v \in \mathsf{Val}, [\![e]\!](v) = 0\} \\
[\![c_1; c_2]\!] & = [\![c_1]\!]\,; [\![c_2]\!] \\
[\![c_1 \parallel c_2]\!] & = \mathcal{SG}; ([\![c_1]\!] \parallel [\![c_2]\!]); \mathcal{SG} \\
[\![x := v]\!] & = \mathcal{WG}(x,v) \\
[\![x := e(y)]\!] & = \bigcup\{\mathcal{RG}(y,v); \mathcal{WG}(x, [\![e]\!](v)) \mid v \in \mathsf{Val}\} \\
[\![x \overset{y,z}{:=} e(y,z)]\!] & = \bigcup\{\mathcal{RG}(y,v_y); \mathcal{RG}(z,v_z); \mathcal{WG}(x, [\![e]\!](v_y,v_z)) \mid v_y, v_z \in \mathsf{Val}\} \\
[\![x \overset{\mathsf{at}}{:=} e(x)]\!] & = \bigcup\{\mathcal{UG}(x,v, [\![e]\!](v)) \mid v \in \mathsf{Val}\}
\end{array}
$$

Fig. 2. Mapping of commands to sets of executions.

- The execution $G \parallel G'$ is given by $\langle A \cup A', E \cup E', L \cup L' \rangle$.
- The execution $G; G'$ is given by $(G \parallel G') \cup (O \times I)$, where O is the set of terminal nodes of G, and I is the set of initial nodes of G'.
- Given $n \geq 0$, G^n is inductively defined by $G^0 = \langle \emptyset, \emptyset, \emptyset \rangle$ and $G^{n+1} = G^n; G$.

The above operations are extended to sets of executions in the obvious way (e.g., $\mathcal{G}; \mathcal{G}' = \{G; G' \mid G \in \mathcal{G}, G' \in \mathcal{G}', G; G' \text{ is defined}\}$).

Definition 3. Given $x \in \mathsf{Loc}$ and $v \in \mathsf{Val}$, an $\langle x, v \rangle$-*read gadget* is any execution of the form $\langle \{a\}, \{a \mapsto \langle \mathtt{R}, x, v \rangle\}, \emptyset \rangle$. $\langle x, v \rangle$-*write gadgets*, $\langle x, v_r, v_w \rangle$-*update gadgets* and *skip gadgets* are defined similarly. $\mathcal{RG}(x,v)$, $\mathcal{WG}(x,v)$, $\mathcal{UG}(x,v_r,v_w)$ and $\mathcal{SG}$ denote, respectively, the sets of all $\langle x, v \rangle$-read gadgets, all $\langle x, v \rangle$-write gadgets, all $\langle x, v_r, v_w \rangle$-update gadgets, and all skip gadgets.

Using these definitions, the mapping of commands to (sets of) executions is given in Fig. 2. Note that every execution $G \in [\![c]\!]$ for some command c satisfies $G.E_{all} = G.po$, and has a unique initial node that can reach any node, and a unique terminal node that can be reached from any node. We refer to such executions as *plain*. However, many of these executions are nonsensical as they can, for instance, read values never written in the program. We restrict our attention to *consistent* executions, as defined next.

Definition 4. A relation R is called a *modification order* for a location $x \in \mathsf{Loc}$ in an execution G if the following hold: (i) R is a total strict order on $\mathtt{W}_x \cup \mathtt{U}_x$; (ii) if $\langle a, b \rangle \in E^*_{all}$ then $\langle b, a \rangle \notin R$; (iii) if $\langle a, b \rangle \in E^+_{all}$ and $\langle c, b \rangle \in E_x$ then $\langle c, a \rangle \notin R$; and (iv) if $\langle a, b \rangle, \langle b, c \rangle \in R$ and $c \in \mathtt{U}$ then $\langle a, c \rangle \notin E_x$.

Definition 5. An execution $G = \langle A, L, E \rangle$ is called:

- *complete* if for every $b \in \mathtt{R} \cup \mathtt{U}$, we have $\langle a, b \rangle \in E_{loc(b)}$ for some $a \in \mathtt{W} \cup \mathtt{U}$.
- *coherent* if E_{all} is acyclic, and there is a modification order in G for each $x \in \mathsf{Loc}$.
- *consistent* if $G \cup E'$ is complete and coherent for some $E' \subseteq A \times A \times \mathsf{Loc}$.

To illustrate these definitions, Fig. 3 depicts a consistent non-SC execution of the "store buffering" program of Fig. 1 together with the implicit variable initializations.

While our notations are slightly different, the axiomatic semantics presented above corresponds to the semantics of C11 programs (see [1]) in which all locations are atomic, reads are acquire reads, writes are release writes, and updates are acquire-release RMWs. In addition, we do not allow reads from uninitialized locations. C11's "happens-before" relation corresponds to our E^+_{all}.

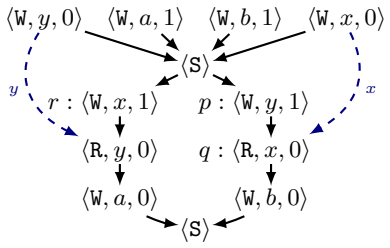


Fig. 3. Ignoring the dashed edges, this graph G is an initialized execution of the "store buffering" program (i.e., $G \in \mathcal{WG}(\top); [\![c]\!]$, Def. 8). G is consistent as it can be extended with the set E' of the two dashed reads-from edges.

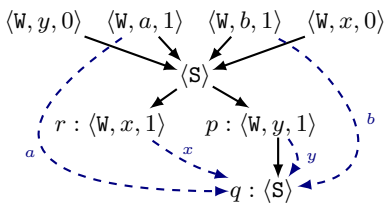


Fig. 4. Ignoring the dashed edges, we have the snapshot of $G \cup E'$ of Fig. 3 at $\langle p, q\rangle$ with respect to $\{r\}$. Adding the dashed edges results in a coherent execution; so the state $\{x \mapsto 1, y \mapsto 1, a \mapsto 1, b \mapsto 1\}$ is visible at $\langle p, q\rangle$ in $G \cup E'$.

3 An Owicki-Gries Proof System for Release-Acquire

In this section, we present OGRA—our logic for reasoning about concurrent programs under release-acquire. As usual, the basic constructs are *Hoare triples* of the form $\{P\}\, c\, \{Q\}$, where P and Q are assertions and c is a command. To define validity of such a triple (in the absence of usual operational semantics), we formalize the notion of a visible state, taken to be a function from Loc to Val.

Definition 6. A *snapshot* of an execution $G = \langle A, L, E\rangle$ at an edge $\langle a, b\rangle \in po$ with respect to a set $B \subseteq A$ of nodes, denoted by $\mathcal{S}(G, \langle a, b\rangle, B)$, is the execution $\langle A' \uplus \{b\}, L|_{A'} \cup \{b \mapsto \langle \mathtt{S}\rangle\}, E|_{A'} \cup \{\langle a, b\rangle\}\rangle$, where:
- $A' = \{a' \in A \setminus \{b\} \mid \exists c \in B \cup \{a\}.\ \langle a', c\rangle \in E^*_{all}\}$ and
- $E|_{A'} = E \cap ((A' \times A') \cup (A' \times A' \times \mathsf{Loc}))$.

Definition 7. Let G be an execution, and let $\langle a, b\rangle \in po$.
- A function $D : \mathsf{Loc} \to \mathbb{N}$ is called a $\langle G, \langle a, b\rangle\rangle$-*reader* of a state $\sigma : \mathsf{Loc} \to \mathsf{Val}$ if $D(x) \in \mathtt{W}_x \cup \mathtt{U}_x$ and $val_w(D(x)) = \sigma(x)$ for every $x \in \mathsf{Loc}$, and the execution $\mathcal{S}(G, \langle a, b\rangle, D[\mathsf{Loc}]) \cup \{\langle D(x), b, x\rangle \mid x \in \mathsf{Loc}\}$ is coherent.
- A state σ is called *visible* at $\langle a, b\rangle$ in G if there is a $\langle G, \langle a, b\rangle\rangle$-reader of σ.
- An assertion P *holds* at $\langle a, b\rangle$ in G if $\sigma \models P$ for every state σ visible at $\langle a, b\rangle$ in G.

In essence, the snapshot restricts the execution to the edge $\langle a, b\rangle$, all nodes in B, and all prior nodes and edges, and replaces the label of b by a skip. For a state to be visible at $\langle a, b\rangle$, additional reads-from edges should be added. For an example, see Fig. 4.

Definition 8. For a state σ, let $\mathcal{WG}(\sigma)$ be $\mathcal{WG}(\nu_1, \sigma(\nu_1)) \parallel \ldots \parallel \mathcal{WG}(\nu_M, \sigma(\nu_M))$, the set of all σ-initializations. Given an assertion P, $\mathcal{WG}(P) = \bigcup\{\mathcal{WG}(\sigma) \mid \sigma \models P\}$.

An execution G is called *initialized* if $G = (G_1; G_2) \cup E$ for some $G_1 \in \mathcal{WG}(\top)$, plain execution G_2, and set $E \subseteq A_1 \times A_2 \times \mathsf{Loc}$ of edges. It can be shown that if G is coherent and initialized, then at least one state is visible at every program order edge.

Definition 9. A Hoare triple $\{P\}\, c\, \{Q\}$ is *valid* if Q holds at the terminal edge of $G \cup E'$ in $G \cup E'$ for every execution $G = \langle A, L, E\rangle$ in $\mathcal{WG}(P); [\![c]\!]; \mathcal{SG}$ and set $E' \subseteq A \times A \times \mathsf{Loc}$, such that $G \cup E'$ is a complete and coherent execution.

OG-style reasoning is often judged as non-compositional because it refers to non-interference of proof outlines that cannot be checked based solely on the two input Hoare triples. A straightforward remedy is to use a *rely/guarantee-style presentation* of OG, that permits compositional reasoning. In this case, the rely component, denoted by $\mathcal{R}$, consists of a set of assertions that are assumed to be stable under assignments performed by other threads. In turn, the guarantee component, denoted by $\mathcal{G}$, is a set of *guarded assignments*, that is assignments together with their immediate preconditions. Roughly speaking, a validity of an OG judgment $\mathcal{R}; \mathcal{G} \Vdash \{P\}\, c\, \{Q\}$ amounts to: "every terminating run of c starting from a state in P ends in a state in Q, and performs only assignments in $\mathcal{G}$, where each of which is performed while satisfying its guard; and moreover, the above holds in parallel to any run of a program c', provided that the assertions in $\mathcal{R}$ are stable under each of the assignments performed by c'."

Now, as demonstrated in the introduction, reasoning under RA requires a richer rely condition, as stability of an assertion in thread I under a guarded assignment of the form $\{P\}x := e(y)$ in thread II should be checked for all values readable for y in some non-later point of thread I. Similarly, stability under $\{P\}x \overset{y,z}{:=} e(y,z)$ should cover all values readable for y and z in two non-later points. Hence, we take $\mathcal{R}$ to consist of pairs of assertions, where the first component of each pair describes the current state and the second summarizes all non-later states. This leads us to the following definitions.

Definition 10. An *OG judgment* $\mathcal{R}; \mathcal{G} \Vdash \{P\}\, c\, \{Q\}$ extends a Hoare triple with two extra components:

- A finite set $\mathcal{R}$ of pairs of the form $R \nearrow C$, where R and C are assertions. We write $\mathcal{R}^R$ for $\bigvee\{R \mid R \nearrow \_ \in \mathcal{R}\}$ and $\mathcal{R}^C$ for $\bigwedge\{C \mid \_ \nearrow C \in \mathcal{R}\}$. We also write $\mathcal{R} \leq \mathcal{R}'$ for such sets if for every $R \nearrow C \in \mathcal{R}$ there exists C' such that $R \nearrow C' \in \mathcal{R}'$ and $C \vdash C'$.
- A finite set $\mathcal{G}$ of *guarded assignments*, i.e., pairs of the form $\{R\}c$, where R is an assertion and c is an assignment command. We write $\mathcal{G} \leq \mathcal{G}'$ for such sets if for every $\{R\}c \in \mathcal{G}$ there exists R' such that $\{R'\}c \in \mathcal{G}'$ and $R \vdash R'$.

Definition 11. A pair $R \nearrow C$ is *stable* under $\{P\}c$ if one of the following holds:

- c has the form $x := v$ and $R \wedge P \vdash R[v/x]$;
- c has the form $x := e(y)$ and $R \wedge P \vdash R[[\![e]\!](v_y)/x]$ for every $v_y \in \mathsf{Val}$ such that $C \wedge P \nvdash y \neq v_y$ (i.e., for every $v_y \in \mathsf{Val}$ such that $C \wedge P \wedge y = v_y$ is satisfiable);
- c has the form $x \overset{y,z}{:=} e(y,z)$ and $R \wedge P \vdash R[[\![e]\!](v_y, v_z)/x]$ for every $v_y, v_z \in \mathsf{Val}$, such that $C \wedge P \nvdash y \neq v_y$ and $C \wedge P \nvdash z \neq v_z$; or
- c has the form $x \overset{\mathsf{at}}{:=} e(x)$ and $R \wedge P \vdash R[e/x]$.

The proof system for deriving OGRA's judgments is given in Fig. 5. The rules are essentially those of Owicki and Gries [10] with minor adjustments due to our rely/guarantee style presentation and the more complex form of the $\mathcal{R}$ component. (To assist the reader, the supplementary material includes a similar presentation of usual OG.) Typically, we require the preconditions and postconditions to be included in $\mathcal{R}$, and make sure their second components keep track of (at least) all non-later assertions: for example, all the assignment rules require $\{P \nearrow P, Q \nearrow (P \vee Q)\} \leq \mathcal{R}$.

The rule for parallel composition (PAR) allows composing non-interfering judgments. Its precondition is the conjunction of the preconditions of the threads, while its

$$\text{(CONSEQ)}\ \frac{\mathcal{R};\mathcal{G} \Vdash \{P\}\,c\,\{Q\} \quad P' \vdash P \quad Q \vdash Q' \quad \mathcal{R} \le \mathcal{R}' \quad \mathcal{G} \le \mathcal{G}'}{\mathcal{R}';\mathcal{G}' \Vdash \{P'\}\,c\,\{Q'\}} \qquad \text{(SEQ)}\ \frac{\mathcal{R}_1;\mathcal{G}_1 \Vdash \{P\}\,c_1\,\{R\} \quad \mathcal{R}_2;\mathcal{G}_2 \Vdash \{R\}\,c_2\,\{Q\} \quad \mathcal{R}_1^R \vdash \mathcal{R}_2^C}{\mathcal{R}_1 \cup \mathcal{R}_2;\mathcal{G}_1 \cup \mathcal{G}_2 \Vdash \{P\}\,c_1;c_2\,\{Q\}}$$

$$\text{(SKIP)}\ \frac{\{P \nearrow P\} \le \mathcal{R}}{\mathcal{R};\emptyset \Vdash \{P\}\,\texttt{skip}\,\{P\}} \qquad \text{(PAR)}\ \frac{\mathcal{R}_1;\mathcal{G}_1 \Vdash \{P_1\}\,c_1\,\{Q_1\} \quad \mathcal{R}_2;\mathcal{G}_2 \Vdash \{P_2\}\,c_2\,\{Q_2\} \quad Q_1 \wedge Q_2 \vdash Q \quad \mathcal{R}_1;\mathcal{G}_1 \text{ and } \mathcal{R}_2;\mathcal{G}_2 \text{ are non-interfering}}{\mathcal{R}_1 \cup \mathcal{R}_2 \cup \{Q \nearrow (\mathcal{R}_1^R \vee \mathcal{R}_2^R \vee Q)\};\mathcal{G}_1 \cup \mathcal{G}_2 \Vdash \{P_1 \wedge P_2\}\,c_1 \parallel c_2\,\{Q\}}$$

$$\text{(ASSN}_0\text{)}\ \frac{P \vdash Q[v/x] \quad \{P \nearrow P, Q \nearrow (P \vee Q)\} \le \mathcal{R}}{\mathcal{R};\{\{P\}x := v\} \Vdash \{P\}\,x := v\,\{Q\}} \qquad \text{(ASSN}_1\text{)}\ \frac{P \vdash Q[e(y)/x] \quad \{P \nearrow P, Q \nearrow (P \vee Q)\} \le \mathcal{R}}{\mathcal{R};\{\{P\}x := e(y)\} \Vdash \{P\}\,x := e(y)\,\{Q\}}$$

$$\text{(ASSN}_2\text{)}\ \frac{P \vdash Q[e(y,z)/x] \quad \{P \nearrow P, Q \nearrow (P \vee Q)\} \le \mathcal{R} \quad \{(P \wedge (y = v)) \nearrow P \mid v \in \mathsf{Val}\} \le \mathcal{R}}{\mathcal{R};\{\{P\}x \overset{y,z}{:=} e(y,z)\} \Vdash \{P\}\,x \overset{y,z}{:=} e(y,z)\,\{Q\}} \qquad \text{(ASSN}_{at}\text{)}\ \frac{P \vdash Q[e(x)/x] \quad \{P \nearrow P, Q \nearrow (P \vee Q)\} \le \mathcal{R}}{\mathcal{R};\{\{P\}x \overset{\mathsf{at}}{:=} e(x)\} \Vdash \{P\}\,x \overset{\mathsf{at}}{:=} e(x)\,\{Q\}}$$

$$\text{(ITE)}\ \frac{\{P \nearrow P\} \le \mathcal{R} \quad P \vdash \mathcal{R}^C \quad \mathcal{R};\mathcal{G} \Vdash \{P \wedge (e(x) \neq 0)\}\,c_1\,\{Q\} \quad \mathcal{R};\mathcal{G} \Vdash \{P \wedge (e(x) = 0)\}\,c_2\,\{Q\}}{\mathcal{R};\mathcal{G} \Vdash \{P\}\,\texttt{if}\ e(x)\ \texttt{then}\ c_1\ \texttt{else}\ c_2\,\{Q\}} \qquad \text{(WHILE)}\ \frac{P \nearrow \_ \in \mathcal{R} \quad \mathcal{R}^R \vdash \mathcal{R}^C \quad P \wedge (e(x) = 0) \vdash Q \quad \mathcal{R};\mathcal{G} \Vdash \{P \wedge (e(x) \neq 0)\}\,c\,\{P\}}{\mathcal{R} \cup \{Q \nearrow (\mathcal{R}^R \vee Q)\};\mathcal{G} \Vdash \{P\}\,\texttt{while}\ e(x)\ \texttt{do}\ c\,\{Q\}}$$

Fig. 5. Owicki-Gries proof system for release-acquire.

postcondition, Q, is any stable assertion implied by the conjunction of the thread postconditions. (The asymmetry is because of the second components of the $\mathcal{R}$ entries: the states prior to the end of the parallel compositions are the union of those of each thread, and hence the stability of Q does not necessarily follow from that of Q_1 and Q_2.) Non-interference is checked for every rely condition of one thread and guarded assignment in the guarantee component of the other:

Definition 12. $\mathcal{R}_1;\mathcal{G}_1$ and $\mathcal{R}_2;\mathcal{G}_2$ are *non-interfering* if every $R \nearrow C \in \mathcal{R}_i$ is stable under every $\{P\}c \in \mathcal{G}_j$ for $i \neq j$.

The consequence rule (CONSEQ) allows strengthening the precondition ($P' \vdash P$), weakening the postcondition ($Q \vdash Q'$), increasing the set of assertions required to be stable ($\mathcal{R} \le \mathcal{R}'$), and increasing the set of allowed guarded assignments ($\mathcal{G} \le \mathcal{G}'$).

The sequential composition rule (SEQ) collects the assertions and allowed assignments of both commands, and checks that $\mathcal{R}_1^R \vdash \mathcal{R}_2^C$. This ensures that stability of c_2's assertions would take into account all the states of c_1, that now become previous states.

The next interesting rule is ASSN_2 concerning assignments with expressions reading two variables. The rule requires that the value of the first variable being read (y) is stable assuming P also holds. This check is needed because of the way we interpret assertions as snapshot reads differs from the way that programs read the variables (one at a time): the stability check ensures that the difference is not observable. Note that the stability of y is trivial in case that there are no assignments to it in other threads.

Finally, the rules for conditionals and while-loops are standard: as with the SEQ rule, we require that the second component of $\mathcal{R}$ has taken into account all earlier states, and include the initial precondition in the set of stable assertions.

We can now state our main theorem, namely the soundness of OGRA.

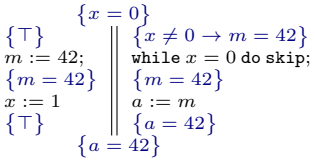


Fig. 6. Proof outline for a simple message passing idiom.

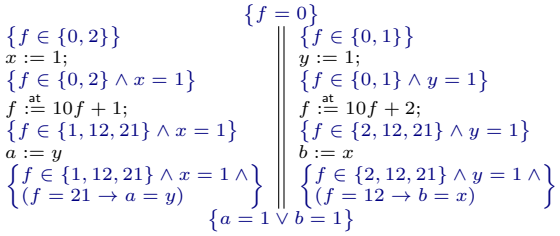


Fig. 7. Proof outline for "store buffering" with fences.

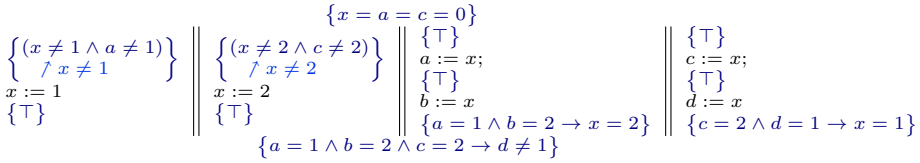


Fig. 8. Proof outline for read-read coherence test (example CoRR2 in [8]).

Theorem 1. *If $\mathcal{R}; \mathcal{G} \Vdash \{P\}\, c\, \{Q\}$ is derivable, then $\{P\}\, c\, \{Q\}$ is valid.*

Before proving this theorem, we provide a few example derivations. The derivations are presented in a proof outline fashion. For each thread, the set $\mathcal{R}$ consists of all the assertions in its proof outline, with the second component being $\top$ (all values are possible) unless mentioned otherwise. The set $\mathcal{G}$ consists of all the assignments in the proof outline guarded by their immediate preconditions.

Our first example, shown in Fig. 6, is a simple message passing idiom. Thread I initializes a message m to 42 and then raises a flag x; thread II waits for x to have a non-zero value and then reads m, which should have value 42. To prove this, thread II assumes the invariant $x \neq 0 \to m = 42$ that holds initially and is stable.

Our next example, shown in Fig. 7, is a variant of the "store buffering" program (see Fig. 1) that uses fences to restore sequential consistency. Fence instructions are implemented as RMWs to a distinguished location f. The RA semantics enforces the corresponding update nodes to be linearly ordered by E^*_{all}, so this implementation imposes a synchronization between every pair of fences. These fences are stronger than C11's SC fences, as they restore full SC when placed between every pair of consecutive instructions. While any atomic assignment to f will have this effect, we choose commands that record the exact order in which the fences are linearized. By referring to this order in the proof, we can easily show that the outcome $a = b = 0$ is not possible.

Our third example, shown in Fig. 8, is a coherence test, demonstrating that threads cannot observe writes to the same location happen in different orders. The program consists of two independent writes to x and two readers: the goal is to prove that the first reader cannot read the one write and then the other, while the second reads them in the reverse order. The key to showing this are the assertions at the end of the reader threads saying that the value of x cannot change after both assignments have been observed. For these assertions to be stable, the writers correspondingly assert that the assignments to x happen before the corresponding reader observes x to have that value. Formally, the precondition of the $x := 1$ assignment is $(x \neq 1 \wedge a \neq 1) \nearrow x \neq 1$. This is stable under the $a := x$ assignment because 1 is not a readable value for x (we have: $x \neq 1 \not\vdash x \neq v$ iff $v \neq 1$).

3.1 Soundness Proof

We present the main steps in the proof of Theorem 1. Annotations play a crucial role. An *annotation* is a function that assigns an assertion to every pair in $\mathbb{N} \times \mathbb{N}$. An annotation Θ is *valid* for an execution G if $\Theta(\langle a, b\rangle)$ holds at $\langle a, b\rangle$ in G for every $\langle a, b\rangle \in po$.

The proof consists of two parts. First, we show that derivability of a judgment $\mathcal{R}; \mathcal{G} \Vdash \{P\}\, c\, \{Q\}$ allows us to construct annotations of executions of c, that are *locally valid* and *stable*, as defined below. Then, we prove that such annotations, for complete and coherent executions, must also be valid. Theorem 1 is obtained as a corollary.

Definition 13. An annotation Θ is *locally valid* for an execution G if the following hold for every $\langle a, b\rangle \in po$, where $P = \bigwedge_{\langle a', a\rangle \in po} \Theta(\langle a', a\rangle)$ and $Q = \Theta(\langle a, b\rangle)$:

- If $L(a) = \langle \mathtt{S}\rangle$ and a is not initial then $P \vdash Q$.
- If $L(a) = \langle \mathtt{R}, x, v\rangle$ then $P \wedge (x = v) \vdash Q$.
- If $L(a) = \langle \mathtt{W}, x, v\rangle$ then either $P \vdash Q[v/x]$, or there is a unique node a' such that $\langle a', a\rangle \in po$, and we have $a' \in \mathtt{R}$ and $P \wedge (loc(a') = val_r(a')) \vdash Q[v/x]$.
- If $L(a) = \langle \mathtt{U}, x, v_r, v_w\rangle$ then $P \wedge (x = v_r) \vdash Q[v_w/x]$.

Definition 14. Let G be an execution. An edge $\langle b_1, b_2\rangle \in po$ is called *G-before* an edge $\langle a_1, a_2\rangle \in po$ if either $\langle b_1, b_2\rangle = \langle a_1, a_2\rangle$ or $\langle b_2, a_1\rangle \in po^*$.

Definition 15. Let G be an execution. A node $c \in G$ *interferes* with $\langle a, b\rangle \in po$ in G for an annotation Θ if the following hold:

- $\langle c, a\rangle \notin po^*$ and $\langle b, c\rangle \notin po^*$ (c is *parallel* to $\langle a, b\rangle$ in G).
- For all $c' \in \mathtt{R}$ with $\langle c', c\rangle \in po$ and $\langle c', a\rangle \notin po^*$, we have $\Theta(\langle a', b'\rangle) \wedge \Theta(\langle c', c\rangle) \not\vdash loc(c') \neq val_r(c')$ for some $\langle a', b'\rangle \in po$ such that $\langle a', b'\rangle$ is G-before $\langle a, b\rangle$ and $\langle b', c'\rangle \notin po^*$.

Definition 16. An annotation Θ is *stable* for an execution G if the following hold for every $\langle a, b\rangle \in po$ and node $c \in \mathtt{W} \cup \mathtt{U}$ that interferes with $\langle a, b\rangle$ in G for Θ, where $R = \Theta(\langle a, b\rangle)$ and $P = \bigwedge_{\langle c', c\rangle \in po} \Theta(\langle c', c\rangle)$:

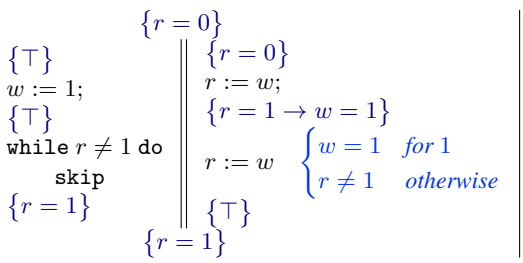
- If $L(c) = \langle \mathtt{W}, x, v\rangle$ then $P \wedge R \vdash R[v/x]$.
- If $L(c) = \langle \mathtt{U}, x, v_r, v_w\rangle$ then $P \wedge (x = v_r) \wedge R \vdash R[v_w/x]$.

Definition 17. A Hoare triple $\{P\}\, c\, \{Q\}$ is *safe* if for every $G \in \mathcal{SG}; [\![c]\!]; \mathcal{SG}$, there is an annotation Θ that is locally valid and stable for G, and assigns some assertion P', such that $P \vdash P'$, to the initial edge of G, and some assertion Q', such that $Q' \vdash Q$, to its terminal edge.

Theorem 2. *If $\mathcal{R}; \mathcal{G} \Vdash \{P\}\, c\, \{Q\}$ is derivable for some $\mathcal{R}, \mathcal{G}$, then $\{P\}\, c\, \{Q\}$ is safe.*

Proof (Outline). Call a judgment $\mathcal{R}; \mathcal{G} \Vdash \{P\}\, c\, \{Q\}$ *good* if for every execution $G \in \mathcal{SG}; [\![c]\!]; \mathcal{SG}$, there exists an annotation Θ that satisfies the conditions given in Definition 17, as well as the following ones:

- $\mathcal{R}$ covers Θ for G, i.e., for every $\langle a_1, a_2\rangle \in po$, there exist $P_1 \nearrow C_1, \ldots, P_n \nearrow C_n \in \mathcal{R}$ such that $\bigwedge P_i \dashv\vdash \Theta(\langle a_1, a_2\rangle)$ and $\Theta(\langle b_1, b_2\rangle) \vdash \bigwedge C_i$ for every $\langle b_1, b_2\rangle \in po$ that is G-before $\langle a_1, a_2\rangle$ (in particular, for $\langle b_1, b_2\rangle = \langle a_1, a_2\rangle$).



Main non-interference checks:

$r = 1$ under $\{r = 0\} r := w$
$r = 1$ under $\{w = 1\} r := w$
$r = 1$ under $\{r \neq 1\} r := w$
$r = 1 \to w = 1$ under $\{\top\} w := 1$
$w = 1$ under $\{\top\} w := 1$
All the checks are trivial.

Fig. 9. Simplified RCU example illustrating the use of the stronger assignment rule.

- $\mathcal{G}$ covers Θ for G, i.e., for every $a_2 \in \mathtt{W} \cup \mathtt{U}$, there exist an edge $\langle a_1, a_2 \rangle \in po$ and an assertion P', such that $\Theta(\langle a_1, a_2 \rangle) \vdash P'$, and one of the following holds:
  - $L(a_2) = \langle \mathtt{W}, x, v \rangle$ and $\{P'\} x := v \in \mathcal{G}$.
  - $L(a_2) = \langle \mathtt{W}, x, v \rangle$, $L(a_1) = \langle \mathtt{R}, y, v_y \rangle$, and $\{P'\} x := e(y) \in \mathcal{G}$ for some expression $e(y)$ such that $[\![e]\!](v_y) = v$.
  - $L(a_2) = \langle \mathtt{W}, x, v \rangle$, $L(a_1) = \langle \mathtt{R}, z, v_z \rangle$, $\Theta(\langle a_1, a_2 \rangle) \vdash y = v_y$ for some $v_y \in \mathsf{Val}$, and $\{P'\} x \overset{y,z}{:=} e(y, z) \in \mathcal{G}$ for some expression $e(y, z)$ such that $[\![e]\!](v_y, v_z) = v$.
  - $L(a_2) = \langle \mathtt{U}, x, v_r, v_w \rangle$ and $\{P'\} x \overset{\mathsf{at}}{:=} e(x) \in \mathcal{G}$ for some expression $e(x)$ such that $[\![e]\!](v_r) = v_w$.

Next, by induction on the derivation, one shows that every derivable judgment $\mathcal{R}; \mathcal{G} \Vdash \{P\}\, c\, \{Q\}$ is good, and so $\{P\}\, c\, \{Q\}$ is safe. The non-interference condition is needed for showing that two annotations of executions G_1 and G_2 can be joined to a stable annotation of the parallel composition of G_1 and G_2. □

It remains to establish the link from safety of a Hoare triple to its validity.

Theorem 3. *Let G be a complete coherent initialized execution. If an annotation Θ is locally valid and stable for G, then it is valid for G.*

The proof (given in the full version of this paper) requires analyzing the relations between states that are visible on consecutive edges and parallel edges in the RA memory model. An alternative equivalent formulation of coherence, based on a new "write-before" relation, is particularly useful for this task.

3.2 A Stronger Assignment Rule

Consider the program shown in Fig. 9, which contains an idiom found in the RCU implementation (verified in the supplementary material). Thread II reads w and writes its value to r twice, while thread I sets w to 1 and then waits for r to become 1. The challenge is to show that after thread I reads $r = 1$, the value of r does not change; i.e. that $r = 1$ is stable under the $r := w$ assignments. For the first $r := w$ assignment, this is easy because its precondition is inconsistent with $r = 1$. For the second assignment, however, there is not much we can do. Stability requires us to consider *any value* for w readable at some point by thread I. Our idea is to do a case split on the value that w reads. If w reads the value 1, then it writes $r := 1$, and so $r = 1$ is unaffected. If w reads a different value, then from the assignment's precondition, we can derive $r \neq 1$, which contradicts the $r = 1$ assertion.

$$\begin{array}{c}\{\top\}\\ x := 2; \;\|\; y := 2;\\ y := 1 \;\|\; x := 1\\ \{x \neq 2 \vee y \neq 2\}\end{array}$$

Fig. 10. Auxiliary variables are necessary under SC.

$$\begin{array}{c}\{x = \langle 0,0\rangle\}\\ \{x \in \{\langle 0,0\rangle, \langle 1,2\rangle\}\} \;\|\; \{x \in \{\langle 0,0\rangle, \langle 1,1\rangle\}\}\\ x \overset{\mathsf{at}}{:=} \langle x_{\mathrm{fst}} + 1, x_{\mathrm{snd}} + 1\rangle \;\|\; x \overset{\mathsf{at}}{:=} \langle x_{\mathrm{fst}} + 1, x_{\mathrm{snd}} + 2\rangle\\ \{x \in \{\langle 1,1\rangle, \langle 2,3\rangle\}\} \;\|\; \{x \in \{\langle 1,2\rangle, \langle 2,3\rangle\}\}\\ \{x = \langle 2,3\rangle\}\end{array}$$

Fig. 11. Verification of the parallel increment example.

To support such case splits, we provide the following stronger assignment rule. For simplicity, we consider only assignments of the form $x := e(y)$.

$$(\textsc{assn}'_1)\ \frac{\begin{array}{ll} P \vdash Q[e(y)/x] & \{P \nearrow P, Q \nearrow (P \vee Q)\} \leq \mathcal{R}\\ \text{For every } v \in \mathsf{Val}\text{:} & P \wedge (y = v) \vdash P_v \qquad \{P_v \nearrow P\} \leq \mathcal{R}\end{array}}{\mathcal{R}; \{\{P_v\} x := e(y) \mid v \in \mathsf{Val}\} \Vdash \{P\}\, x := e(y)\, \{Q\}}$$

The previous assignment rule is an instance of this rule by taking $P_v = P$ for all v.

4 Discussion and Further Research

While OGRA's non-interference condition appears to be restrictive, we note that it is unsound for weaker memory models, such as C11's relaxed accesses because it can prove, e.g., message passing, see Fig. 6. We also observe that OGRA's non-interference check coincides with the standard OG one for assignments of values ($x := v$) and atomic assignments ($x \overset{\mathsf{at}}{:=} e(x)$). Moreover, the non-interference check is irrelevant for assignments to variables that do not occur in the proof outlines of other threads. Therefore, standard OG (without auxiliary variables) is sound under RA provided that all $x := e(y)$ and $x \overset{y,z}{:=} e(y,z)$ assignments write to variables that do not appear in the proof outlines of other threads. Figures 6 and 7 provide two such cases in point. In addition, this entails, for instance, that the program in Fig. 10 cannot be verified in standard OG without auxiliary variables, as $x = 2 \wedge y = 2$ is a possible outcome for this program under RA.

OG's auxiliary variables, in general, are unsound under weak memory because they can be used to record the exact thread interleavings and establish completeness under SC [12]. A simple form of auxiliary state, which we call *ghost values*, however, *is* sound. The idea is as follows: given a program c, one may choose a domain G of "ghost" values, together with a function $\alpha : \mathsf{G} \to \mathsf{Val}$, and obtain a program c' by substituting each expression $e(x_1, \ldots, x_n)$ in c by an expression $e'(x_1, \ldots, x_n)$ such that $\alpha([\![e']\!](g_1, \ldots, g_n)) = [\![e]\!](\alpha(g_1), \ldots, \alpha(g_n))$ for all $g_1, \ldots, g_n \in \mathsf{G}$. The validity of $\{P'\}\, c'\, \{Q'\}$ entails the validity of $\{P\}\, c\, \{Q\}$, provided that the following hold:

- If a state satisfies P then some corresponding ghost state satisfies P';
- If a state does not satisfy Q then any corresponding ghost state does not satisfy Q';

where a ghost state $\sigma' : \mathsf{Loc} \to \mathsf{G}$ *corresponds* to a state $\sigma : \mathsf{Loc} \to \mathsf{Val}$ iff $\alpha(\sigma'(x)) = \sigma(x)$ for every $x \in \mathsf{Loc}$. This solution suffices, for instance, to reason about the parallel increment example, as shown in Fig. 11. There we took $\mathsf{G} = \mathsf{Val} \times \mathbb{N}$, with α being the first projection mapping. The second component tracks which of the assignments has already happened (0: none, 1: the first thread, 2: the second thread, otherwise: both). As a result, we obtain the validity of $\{x = 0\}\, x \overset{\mathsf{at}}{:=} x + 1 \parallel x \overset{\mathsf{at}}{:=} x + 1\, \{x = 2\}$.

Analyzing soundness of other restricted forms of auxiliary variables is left for future work. Such extensions seem to be a prerequisite for obtaining a program logic that is both sound and complete under RA. Automation of proof search is another future goal. Our initial experiments show that, at least for the examples in this paper, HSF [5] is successful in automatically finding proofs in OGRA.

Acknowledgments. We would like to thank the ICALP'15 reviewers for their feedback. This work was supported by EC FET project ADVENT (308830).

References

1. Batty, M., Owens, S., Sarkar, S., Sewell, P., Weber, T.: Mathematizing C++ concurrency. In: POPL 2011. pp. 55–66. ACM (2011)
2. Cohen, E.: Coherent causal memory. CoRR abs/1404.2187 (2014)
3. Desnoyers, M., McKenney, P.E., Stern, A.S., Dagenais, M.R., Walpole, J.: User-level implementations of read-copy update. IEEE Trans. Parallel Distrib. Syst. 23(2), 375–382 (2012)
4. Ferreira, R., Feng, X., Shao, Z.: Parameterized memory models and concurrent separation logic. In: Gordon, A.D. (ed.) ESOP 2010. LNCS, vol. 6012, pp. 267–286. Springer, Heidelberg (2010)
5. Grebenshchikov, S., Lopes, N.P., Popeea, C., Rybalchenko, A.: Synthesizing software verifiers from proof rules. In: PLDI 2012. pp. 405–416. ACM (2012)
6. ISO/IEC 14882:2011: Programming language C++ (2011)
7. Lamport, L.: How to make a multiprocessor computer that correctly executes multiprocess programs. IEEE Trans. Computers 28(9), 690–691 (1979)
8. Maranget, L., Sarkar, S., Sewell, P.: A tutorial introduction to the ARM and POWER relaxed memory models. http://www.cl.cam.ac.uk/˜pes20/ppc-supplemental/test7.pdf (2012)
9. Owens, S., Sarkar, S., Sewell, P.: A better x86 memory model: x86-TSO. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) TPHOLs 2009. LNCS, vol. 5674, pp. 391–407. Springer, Heidelberg (2009)
10. Owicki, S., Gries, D.: An axiomatic proof technique for parallel programs I. Acta Informatica 6(4), 319–340 (1976)
11. Owicki, S., Gries, D.: Verifying properties of parallel programs: An axiomatic approach. Commun. ACM 19(5), 279–285 (May 1976)
12. Owicki, S.S.: Axiomatic Proof Techniques for Parallel Programs. Ph.D. thesis, Cornell University, Ithaca, NY, USA (1975)
13. Ridge, T.: A rely-guarantee proof system for x86-TSO. In: Leavens, G.T., O'Hearn, P.W., Rajamani, S.K. (eds.) VSTTE 2010. LNCS, vol. 6217, pp. 55–70. Springer, Heidelberg (2010)
14. Sieczkowski, F., Svendsen, K., Birkedal, L., Pichon-Pharabod, J.: A separation logic for fictional sequential consistency. In: Vitek, J. (ed.) ESOP 2015. LNCS, vol. 9032, pp. 736–761. Springer, Heidelberg (2015)
15. Tassarotti, J., Dreyer, D., Vafeiadis, V.: Verifying read-copy-update in a logic for weak memory. In: PLDI 2015. ACM (2015)
16. Turon, A., Vafeiadis, V., Dreyer, D.: GPS: Navigating weak memory with ghosts, protocols, and separation. In: OOPSLA 2014. pp. 691–707. ACM (2014)
17. Vafeiadis, V., Balabonski, T., Chakraborty, S., Morisset, R., Zappa Nardelli, F.: Common compiler optimisations are invalid in the C11 memory model and what we can do about it. In: POPL 2015. pp. 209–220. ACM (2015)
18. Vafeiadis, V., Narayan, C.: Relaxed separation logic: A program logic for C11 concurrency. In: OOPSLA 2013. pp. 867–884. ACM (2013)

A Appendix: Additional Examples

This appendix includes additional examples illustrating the use of OG to reason about programs under the RA memory model.

A.1 Another Coherence Example

Our first example is an another example testing read-read coherence: two threads write two different values to x in parallel while a third thread reads x three times. The goal is to prove that the third thread cannot first read the value written by the first thread, then the value written by the second thread, and then again the value written by the first thread. The code and the proof outline are given below:

$$
\begin{array}{l}
\{a = 0\} \\
x := 0 \\
\{a = 0 \wedge x = 0\} \\
\left\{\begin{array}{l}(x \neq 1 \wedge a \neq 1) \\ \quad \curvearrowright x \neq 1\end{array}\right\} \\
x := 1 \\
\{\top\}
\end{array}
\;\Big\|\;
\begin{array}{l}
\{\top\} \\
x := 2 \\
\{\top\}
\end{array}
\;\Big\|\;
\begin{array}{l}
\{\top\} \\
a := x; \\
\{\top\} \\
b := x; \\
\{a = 1 \wedge b = 2 \rightarrow x = 2\} \\
c := x \\
\{a = 1 \wedge b = 2 \rightarrow c = 2\}
\end{array}
$$
$$\{a = 1 \wedge b = 2 \rightarrow c = 2\}$$

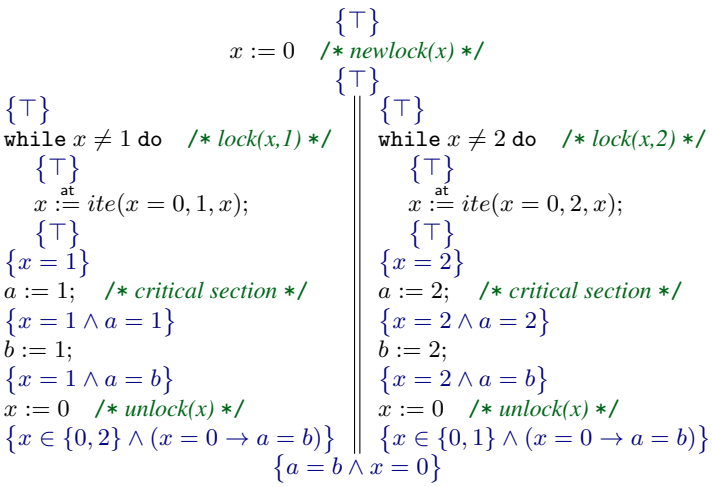
The proof of this algorithm is similar to that of the CoRR2 example. Note that the assertion of the first thread $(x \neq 1 \wedge a \neq 1) \curvearrowright x \neq 1$ is stable under the assignment $a := x$.

A.2 A Simple Spinlock

Our next example is a simple spinlock. The lock itself is represented by location x that stores the identifier of the thread that has acquired the lock or 0 whenever the lock is free. Acquiring the lock is implemented using using the test-and-set atomic instruction, which we encode as an atomic assignment of an if-then-else expression: $ite(e, e', e'')$ tests the condition e and returns e' if the condition holds and e'' otherwise. Releasing the lock simply its value to 0.

To demonstrate that the lock implementation provides mutual exclusion, we use a simple critical section that sets two other locations a and b to the same value one at a

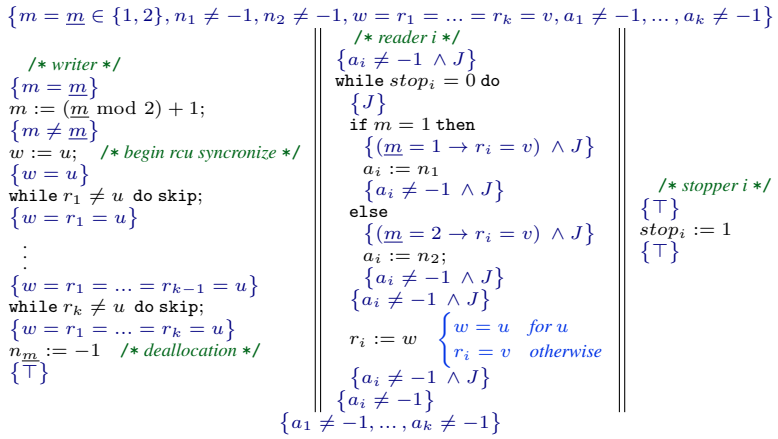
time. At the end we prove that both a and b have the same value.



A.3 Read-Copy-Update

Read-copy-update (RCU) is a synchronization mechanism used in Linux that allows efficient synchronization between a writer thread and multiple reader threads. When the writer wants to update part of the shared data structure, it creates a local copy of the updated part, updates it, and then atomically replaces the old part of the data structure with the new updated part. This ensures that any concurrent reader will either see the old version or the new one: they will never see a mixture of the two versions. There is, however, one important subtlety: when can the writer delete (deallocate) the old version of the data structure? Clearly, it cannot do that immediately because there may be readers still accessing it. What it does instead is to wait for all the readers to finish reading, at which point the old version is no longer accessible.

The following code is based on an implementation of the quiescent-state-based user-mode RCU implementation of Desnoyers et al. [3]. We consider a very simple shared data structure, that consists of two containers n_1, n_2 (that store, e.g., linked lists or arrays), and one index variable m, with the property that n_m represents the updated container. This structure is accessed concurrently by k reader threads, that constantly read n_m, until they are terminated by a stopper thread. We consider a scenario in which the writer thread wants to flip the current container $n_{\underline{m}}$ ($\underline{m}$ is either 1 or 2) and deallocate $n_{\underline{m}}$ (which we model by the assignment $n_{\underline{m}} := -1$). Thus it flips m, and then synchronizes with all the concurrent readers. Synchronization happens by the writer storing a new value u (with $u \neq v$) in the writer's RCU synchronization location w, and waiting for each of the k readers to acknowledge that they are aware of this new value. When a reader acknowledges the writer's update to w (by setting its RCU synchronization location r_i to match the value of w), it notifies the writer that it is no longer accessing $n_{\underline{m}}$. Only then the writer can deallocate $n_{\underline{m}}$. We verify the safety of this mechanism, by proving that the readers never access a deallocated value: they always return $a_i \neq -1$.



where:

$$\begin{aligned} J \triangleq\ & m \in \{1,2\} \wedge n_{(\underline{m} \bmod 2)+1} \neq -1 \wedge w \in \{u,v\} \wedge \\ & (w = u \rightarrow m \neq \underline{m}) \wedge (r_i = v \rightarrow n_{\underline{m}} \neq -1) \wedge \\ & (w = v \rightarrow r_i = v) \end{aligned}$$

The proof is similar to the ones of the previous simpler examples. To handle the $r_i := w$ assignment, we make use of the stronger assignment rule, given in Section 3.2.

B Introducing "Write-Before"

In this appendix we provide equivalent definition of coherence (Definition 5), that is not based on modification orders. In fact, we recognize the exact conditions, that ensure the existence of modification orders, without actually providing them. This alternative formulation makes it easier to reason about coherence preserving transformations, that are needed to prove the soundness of the proposed program logic (see Theorem 5).

Notation 1. Given $R \subseteq A \times A$, we denote its reflexive closure by $R^{=}$, and for $a \in A$, the relation $(R \cap ((A \setminus \{a\}) \times (A \setminus \{a\})))^*$ (R-paths avoiding a) is denoted by $R^{*/a}$.

Definition 18. Let G be an execution, and let $x \in \mathsf{Loc}$. The relation $G.wb_x$ (*x-write-before*) is defined by $\langle a, b\rangle \in G.wb_x$ if $a, b \in \mathtt{W}_x \cup \mathtt{U}_x$, $a \neq b$, and there exists a pair $\langle a', b'\rangle \in E_{all}^{+}$, such that $\langle a', a\rangle \in E_x^{*/b}$ and $\langle b, b'\rangle \in E_x^{=}$.

Definition 19. An execution is *wb-coherent* if E_{all} is acyclic, as well as wb_x for each $x \in \mathsf{Loc}$.

Proposition 1. *Let G be a wb-coherent execution, and let $x \in \mathsf{Loc}$.*

1. *E_x is an injective relation, and $E_x \cap (A \times \mathtt{U})$ is a functional relation.*
2. *If $\langle a, b\rangle, \langle b, c\rangle \in E_x^*$, then $a = b = c$ or $\langle a, c\rangle \notin E_x^{*/b}$.*
3. *If $\langle a, c\rangle, \langle b, c\rangle \in E_x^*$, then either $a = b$, $\langle a, b\rangle \in E_x^+$ or $\langle b, a\rangle \in E_x^+$.*
4. *If $a \in \mathtt{W} \cup \mathtt{U}$ and $\langle c, a\rangle \in E_x^{*/b}$, then $\langle c, b\rangle \in wb_x^+$ iff $\langle a, b\rangle \in wb_x^+$.*

Proof. Items $1-3$ easily follow from our definitions. We prove 4. Assume that $a \in \mathtt{W} \cup \mathtt{U}$ and $\langle c, a\rangle \in E_x^{*/b}$. If $a = c$, then the claim obviously holds. Suppose otherwise. Then, since $\langle c, a\rangle \in E_x^{*/b}$, we also have $a \neq b$ and $c \neq b$.

($\Rightarrow$) Suppose that $\langle c, b\rangle \in wb_x^+$. Since G is wb-coherent, this implies that $\langle b, c\rangle \notin wb_x^*$, and so $\langle b, c\rangle \notin E_x^*$. If $\langle a, b\rangle \in E_x^+$, then $\langle a, b\rangle \in wb_x$, and we are done. Suppose otherwise. We first show that $\langle b, a\rangle \notin E_x^*$. Indeed, otherwise, we have $\langle b, a\rangle, \langle c, a\rangle \in E_x^*$ and $\langle b, c\rangle \notin E_x^*$, and hence $\langle c, b\rangle \in E_x^+$. But, $\langle c, b\rangle \in E_x^+$, $\langle b, a\rangle \in E_x^*$, and $a \neq b$ together imply that $\langle c, a\rangle \notin E_x^{*/b}$.
Now, let $c = b_0, b_1, \ldots, b_n, b_{n+1} = b$ such that $\langle b_i, b_{i+1}\rangle \in wb_x$ for every $0 \leq i \leq n$. Let $j = \max\{i \geq 0 \mid \langle b_i, a\rangle \in E_x^*\}$. Then, $0 \leq j \leq n$ (since $\langle c, a\rangle \in E_x^*$ and $\langle b, a\rangle \notin E_x^*$). Since $\langle b_j, a\rangle \in E_x^*$ and $\langle b_{j+1}, a\rangle \notin E_x^*$, we have $\langle b_j, a\rangle \in E_x^{*/b_{j+1}}$. Since $\langle b_j, b_j + 1\rangle \in wb_x$, it follows that $\langle a, b_{j+1}\rangle \in wb_x$. Hence, we have $\langle a, b\rangle \in wb_x^+$.

($\Leftarrow$) Suppose that $\langle a, b\rangle \in wb_x^+$. Since $\langle c, a\rangle \in E_x^{*/b}$, $a \neq c$, and $a \in \mathtt{W} \cup \mathtt{U}$, we have $a, c \in \mathtt{W}_x \cup \mathtt{U}_x$ and $\langle c, a\rangle \in E_x^+$, and so, by definition, $\langle c, a\rangle \in wb_x$. It follows that $\langle c, b\rangle \in wb_x^+$. □

Next, we prove the equivalence of this model to the C11 formulation.

Lemma 1 (Successor-preserving linear extension). *Let $\langle A, \leq\rangle$ be a partially ordered set. Let $S : A \rightharpoonup A$ be a partial injective function, such that $a < b$ iff $S(a) \leq b$ for every $a \in dom(S)$ and $b \in A$. Then, $\leq$ can be extended to a total order $\preceq$, such that $a \prec b$ iff $S(a) \preceq b$ for every $a \in dom(S)$ and $b \in A$.*

Proof. Let $S^* : A \to A$ be the (total) function defined by $S^*(a) = S^k(a)$ where k is the maximal non-negative integer for which $S^k(a)$ is defined. Let A' be the image of S^*, and $\leq'$ be the restriction of $\leq$ to A'. Extend $\leq'$ to a total order $\leq^*$ on A'. Define $\preceq$ by $a \preceq b$ if $b = S^k(a)$ for some $k \geq 0$ or $S^*(a) <^* S^*(b)$. It is straightforward to verify that $\preceq$ has all required properties. □

Theorem 4. *A complete execution G is wb-coherent iff it is coherent.*

Proof. Let $G = \langle A, L, E\rangle$ be a complete execution. Suppose that G is wb-coherent. By definition E_{all} is acyclic. Let $x \in \mathsf{Loc}$. We show that there exists a modification order for x in G. Let $S = E_x \cap (A \times \mathtt{U})$. By Proposition 1, S defines an invective partial function, and $\langle a, b\rangle \in wb_x^+$ iff $\langle S(a), b\rangle \in wb_x^*$ for every $a \in dom(S)$ and $b \in \mathtt{W}_x \cup \mathtt{U}_x$. Therefore, by Lemma 1, wb_x^* can be extended to a total order $\preceq$ on $\mathtt{W}_x \cup \mathtt{U}_x$, such that $a \prec b$ iff $S(a) \preceq b$ for every $a \in dom(S)$ and $b \in \mathtt{W}_x \cup \mathtt{U}_x$. It is straightforward to verify that $\prec$ is a modification order for x in G.

For the converse, suppose that G is coherent. Let $x \in \mathsf{Loc}$, and let mo_x be a modification order for x in G. We show that $wb_x \subseteq mo_x$, and thus, since mo_x is an order relation, wb_x is acyclic. Let $\langle a, b\rangle \in wb_x$. Then, $a, b \in \mathtt{W}_x \cup \mathtt{U}_x$, $a \neq b$, and there exists a pair $\langle c, d\rangle \in E_{all}^+$, such that $\langle c, a\rangle \in E_x^{*/b}$, and $d = b$ or $\langle b, d\rangle \in E_x$. Let $c = c_1, \ldots, c_n = a$ be a b-avoiding path from c to a in E_x. By induction on i, we show that every node c_i along this path has $\langle c_i, b\rangle \in mo_x$. First, for $c_0 = c$, since $\langle c, d\rangle \in E_{all}^+$, and $d = b$ or $\langle b, d\rangle \in E_x$, we have that $\langle b, c\rangle \notin mo_x$, and since mo_x is total and $b \neq c$, we have $\langle c, b\rangle \in mo_x$. Now, suppose that $\langle c_i, b\rangle \in mo_x$ for some $1 \leq i \leq n-1$. Since $c_{i+1} \in \mathtt{U}$ and $\langle c_i, c_{i+1}\rangle \in E_x$, it cannot be the case that $\langle b, c_{i+1}\rangle \in mo_x$. It follows that $\langle c_{i+1}, b\rangle \in mo_x$. □

C Appendix: Full Proofs

In this appendix we provide full proofs of some of the results above (to be consulted at the discretion of program committee members). Note that we use the alternative coherence characterization given in Appendix B. Additional definitions, notations and lemmas are included as well.

Notation 2. Let $G = \langle A, L, E\rangle$ be an execution. Given a set $B \subseteq A$, we denote by $G \cap B$ the triple $\langle B, L|_B, E \cap ((B \times B) \cup (B \times B \times \mathsf{Loc}))\rangle$. $G \setminus B$ denotes the triple $G \cap (A \setminus B)$. Given $a \in A$ and label l, $G[a \mapsto l]$ stands for the triple $\langle A, L[a \mapsto l], E\rangle$.

Notation 3. For a plain execution G, we denote by $G.i$ and $G.o$ the initial and terminal nodes of G (respectively).

Proof (of Theorem 2). Call a judgment $\mathcal{R}; \mathcal{G} \Vdash \{P\}\, c\, \{Q\}$ *good* if for every execution $G \in \mathcal{SG}; [\![c]\!]; \mathcal{SG}$, there exists an annotation Θ that satisfies the following conditions:

- Θ is locally valid and stable for G.
- Θ assigns some assertion P', such that $P \vdash P'$, to the initial edge of G, and some assertion Q', such that $Q' \vdash Q$, to the terminal edge of G.
- $\mathcal{R}$ covers Θ for G, i.e., for every $\langle a_1, a_2\rangle \in po$, there exist $P_1 \nearrow C_1, \ldots, P_n \nearrow C_n \in \mathcal{R}$ such that $\bigwedge P_i \dashv\vdash \Theta(\langle a_1, a_2\rangle)$ and $\Theta(\langle b_1, b_2\rangle) \vdash \bigwedge C_i$ for every $\langle b_1, b_2\rangle \in po$ that is G-before $\langle a_1, a_2\rangle$ (in particular, for $\langle b_1, b_2\rangle = \langle a_1, a_2\rangle$).
- $\mathcal{G}$ covers Θ for G, i.e., for every $a_2 \in \mathtt{W} \cup \mathtt{U}$, there exist an edge $\langle a_1, a_2\rangle \in po$ and an assertion P', such that $\Theta(\langle a_1, a_2\rangle) \vdash P'$, and one of the following holds:
  - $L(a_2) = \langle \mathtt{W}, x, v\rangle$ and $\{P'\}x := v \in \mathcal{G}$.
  - $L(a_2) = \langle \mathtt{W}, x, v\rangle$, $L(a_1) = \langle \mathtt{R}, y, v_y\rangle$, and $\{P'\}x := e(y) \in \mathcal{G}$ for some expression $e(y)$ such that $[\![e]\!](v_y) = v$.
  - $L(a_2) = \langle \mathtt{W}, x, v\rangle$, $L(a_1) = \langle \mathtt{R}, z, v_z\rangle$, $\Theta(\langle a_1, a_2\rangle) \vdash y = v_y$ for some $v_y \in \mathsf{Val}$, and $\{P'\}x \overset{y,z}{:=} e(y, z) \in \mathcal{G}$ for some expression $e(y, z)$ such that $[\![e]\!](v_y, v_z) = v$.
  - $L(a_2) = \langle \mathtt{U}, x, v_r, v_w\rangle$ and $\{P'\}x \overset{\mathsf{at}}{:=} e(x) \in \mathcal{G}$ for some expression $e(x)$ such that $[\![e]\!](v_r) = v_w$.

We use induction on the derivation to show that every derivable judgment $\mathcal{H} = \mathcal{R}; \mathcal{G} \Vdash \{P\}\, c\, \{Q\}$ is good. In particular, it follows that $\{P\}\, c\, \{Q\}$ is safe. In each of the cases, we show how to build an annotation Θ for arbitrary $G \in \mathcal{SG}; [\![c]\!]; \mathcal{SG}$. It is straightforward to verify that these annotations satisfy the conditions above. Note that we only provide the relevant part of Θ (i.e., its values for $G.po$).

(SKIP) Suppose that $\{P \nearrow P\} \leq \mathcal{R}$. We show that $\mathcal{R}; \emptyset \Vdash \{P\}\, \mathtt{skip}\, \{Q\}$ is good. Let $G = \langle A, L, E\rangle$ be an execution in $\mathcal{SG}; [\![\mathtt{skip}]\!]; \mathcal{SG}$. Then, there exist distinct $a, b, c \in \mathbb{N}$, such that $A = \{a, b, c\}$, $L = \{a \mapsto \langle \mathtt{S}\rangle, b \mapsto \langle \mathtt{S}\rangle, c \mapsto \langle \mathtt{S}\rangle\}$, and $E = \{\langle a, b\rangle, \langle b, c\rangle\}$. Let $\Theta = \{\langle a, b\rangle \mapsto P, \langle b, c\rangle \mapsto P\}$.

(SEQ) Suppose that $\mathcal{H}_1 = \mathcal{R}_1; \mathcal{G}_1 \Vdash \{P\}\, c_1\, \{R\}$ and $\mathcal{H}_2 = \mathcal{R}_2; \mathcal{G}_2 \Vdash \{R\}\, c_2\, \{Q\}$ are good, and that $\mathcal{R}_1^R \vdash \mathcal{R}_2^C$. We show that $\mathcal{R}_1 \cup \mathcal{R}_2; \mathcal{G}_1 \cup \mathcal{G}_2 \Vdash \{P\}\, c_1; c_2\, \{Q\}$ is good. Let $G = \langle A, L, E\rangle$ be an execution in $\mathcal{SG}; [\![c_1; c_2]\!]; \mathcal{SG}$. Then, there exist distinct $a, b \in \mathbb{N}$ and executions $G_1 = \langle A_1, L_1, E_1\rangle$ in $[\![c_1]\!]$ and $G_2 = \langle A_2, L_2, E_2\rangle$

in $[\![c_2]\!]$, such that $A = \{a, b\} \uplus A_1 \uplus A_2$, $L = \{a \mapsto \langle \mathtt{S} \rangle, b \mapsto \langle \mathtt{S} \rangle\} \cup L_1 \cup L_2$, and $E = \{\langle a, G_1.i \rangle, \langle G_1.o, G_2.i \rangle, \langle G_2.o, b \rangle\} \cup E_1 \cup E_2$. Let $A'_1 = \{a, b\} \cup A_1$, $A'_2 = \{a, b\} \cup A_2$, $G'_1 = (G \cap A'_1) \cup \{\langle G_1.o, b \rangle\}$, and $G'_2 = (G \cap A'_2) \cup \{\langle a, G_2.i \rangle\}$. Then, $G'_1 \in \mathcal{SG}; [\![c_1]\!]; \mathcal{SG}$ and $G'_2 \in \mathcal{SG}; [\![c_2]\!]; \mathcal{SG}$. Since $\mathcal{H}_1$ and $\mathcal{H}_2$ are good, there exist annotations Θ_1 and Θ_2 that satisfy the required conditions for G'_1 and G'_2 (respectively). Let $\Theta = \Theta_1|_{E_1 \cup \{\langle a, G_1.i \rangle\}} \cup \Theta_2|_{E_2 \cup \{\langle G_2.o, b \rangle\}} \cup \{\langle G_1.o, G_2.i \rangle \mapsto \Theta_1(\langle G_1.o, b \rangle)\}$.

(ASSN_0) Suppose that $P \vdash Q[v/x]$ and $\{P \nearrow P, Q \nearrow (P \vee Q)\} \leq \mathcal{R}$. We show that $\mathcal{R}; \{\{P\} x := v\} \Vdash \{P\}\, x := v\, \{Q\}$ is good. Let $G = \langle A, L, E \rangle$ be an execution in $\mathcal{SG}; [\![x := v]\!]; \mathcal{SG}$. Then, there exist distinct $a, b, c \in \mathbb{N}$, such that $A = \{a, b, c\}$, $L = \{a \mapsto \langle \mathtt{S} \rangle, b \mapsto \langle \mathtt{W}, x, v \rangle, c \mapsto \langle \mathtt{S} \rangle\}$, and $E = \{\langle a, b \rangle, \langle b, c \rangle\}$. Let $\Theta = \{\langle a, b \rangle \mapsto P, \langle b, c \rangle \mapsto Q\}$.

(ASSN_1) Suppose that $P \vdash Q[e(y)/x]$ and $\{P \nearrow P, Q \nearrow (P \vee Q)\} \leq \mathcal{R}$. We show that $\mathcal{R}; \{\{P\} x := e(y)\} \Vdash \{P\}\, x := e(y)\, \{Q\}$ is good. Let $G = \langle A, L, E \rangle$ be an execution in $\mathcal{SG}; [\![x := e(y)]\!]; \mathcal{SG}$. Then, there exist $v_y \in \mathsf{Val}$ and distinct $a, b_y, b, c \in \mathbb{N}$, such that $A = \{a, b_y, b, c\}$, $L = \{a \mapsto \langle \mathtt{S} \rangle, b_y \mapsto \langle \mathtt{R}, y, v_y \rangle, b \mapsto \langle \mathtt{W}, x, [\![e]\!](v_y) \rangle, c \mapsto \langle \mathtt{S} \rangle\}$, and $E = \{\langle a, b_y \rangle, \langle b_y, b \rangle, \langle b, c \rangle\}$. Let $\Theta = \{\langle a, b_y \rangle \mapsto P, \langle b_y, b \rangle \mapsto P, \langle b, c \rangle \mapsto Q\}$.

(ASSN_2) Suppose that $P \vdash Q[e(y,z)/x]$, and $\{P \nearrow P, Q \nearrow (P \vee Q)\} \cup \{(P \wedge (y = v)) \nearrow P \mid v \in \mathsf{Val}\} \leq \mathcal{R}$. We show that $\mathcal{R}; \{\{P\} x \overset{y,z}{:=} e(y,z)\} \Vdash \{P\}\, x \overset{y,z}{:=} e(y,z)\, \{Q\}$ is good. Let $G = \langle A, L, E \rangle$ be an execution in $\mathcal{SG}; [\![x \overset{y,z}{:=} e(y,z)]\!]; \mathcal{SG}$. Then, there exist $v_y, v_z \in \mathsf{Val}$ and distinct $a, b_y, b_z, b, c \in \mathbb{N}$, such that $A = \{a, b_y, b_z, b, c\}$, $L = \{a \mapsto \langle \mathtt{S} \rangle, b_y \mapsto \langle \mathtt{R}, y, v_y \rangle, b_z \mapsto \langle \mathtt{R}, z, v_z \rangle, b \mapsto \langle \mathtt{W}, x, [\![e]\!](v_y, v_z) \rangle, c \mapsto \langle \mathtt{S} \rangle\}$, and $E = \{\langle a, b_y \rangle, \langle b_y, b_z \rangle, \langle b_z, b \rangle, \langle b, c \rangle\}$. Let $\Theta = \{\langle a, b_y \rangle \mapsto P, \langle b_y, b_z \rangle \mapsto P \wedge (y = v_y), \langle b_z, b \rangle \mapsto P \wedge (y = v_y), \langle b, c \rangle \mapsto Q\}$.

(ASSN_{at}) Suppose that $P \vdash Q[e(x)/x]$ and $\{P \nearrow P, Q \nearrow (P \vee Q)\} \leq \mathcal{R}$. We show that $\mathcal{R}; \{\{P\} x \overset{\mathsf{at}}{:=} e(x)\} \Vdash \{P\}\, x \overset{\mathsf{at}}{:=} e(x)\, \{Q\}$ is good. Let $G = \langle A, L, E \rangle$ be an execution in $\mathcal{SG}; [\![x \overset{\mathsf{at}}{:=} e(x)]\!]; \mathcal{SG}$. Then, there exist $v \in \mathsf{Val}$ and distinct $a, b, c \in \mathbb{N}$, such that $A = \{a, b, c\}$, $L = \{a \mapsto \langle \mathtt{S} \rangle, b \mapsto \langle \mathtt{U}, x, v, [\![e]\!](v) \rangle, c \mapsto \langle \mathtt{S} \rangle\}$, and $E = \{\langle a, b \rangle, \langle b, c \rangle\}$. Let $\Theta = \{\langle a, b \rangle \mapsto P, \langle b, c \rangle \mapsto Q\}$.

(ITE) Let $c = \mathtt{if}\ e(x)\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2$. Suppose that $\mathcal{H}_1 = \mathcal{R}; \mathcal{G} \Vdash \{P \wedge (e(x) \neq 0)\}\, c_1\, \{Q\}$ and $\mathcal{H}_2 = \mathcal{R}; \mathcal{G} \Vdash \{P \wedge (e(x) = 0)\}\, c_2\, \{Q\}$ are good, $\{P \nearrow P\} \leq \mathcal{R}$, and $P \vdash \mathcal{R}^C$. We show that $\mathcal{R}; \mathcal{G} \Vdash \{P\}\, c\, \{Q\}$ is good. Let $G = \langle A, L, E \rangle$ be an execution in $\mathcal{SG}; [\![c]\!]; \mathcal{SG}$. Then, $G \in \mathcal{SG}; \mathcal{RG}(x, v); [\![c_k]\!]; \mathcal{SG}$ for some $k \in \{1, 2\}$ and $v \in \mathsf{Val}$ such that $[\![e]\!](v) \neq 0$ iff $k = 1$. Thus there exist distinct $a, b, c \in \mathbb{N}$, and execution $G_k = \langle A_k, L_k, E_k \rangle$ in $[\![c_k]\!]$, such that $A = \{a, b, c\} \uplus A_k$, $L = \{a \mapsto \langle \mathtt{S} \rangle, b \mapsto \langle \mathtt{R}, x, v \rangle, c \mapsto \langle \mathtt{S} \rangle\} \cup L_k$, and $E = \{\{\langle a, b \rangle, \langle b, G_k.i \rangle, \langle G_k.o, c \rangle\} \cup E_k$. Let $G' = (G \setminus \{b\}) \cup \{\langle a, G_k.i \rangle\}$. Then, $G' \in \mathcal{SG}; [\![c_k]\!]; \mathcal{SG}$. Since $\mathcal{H}_k$ is good, there exists an annotation Θ' that satisfies the required conditions for G'. Let $\Theta = \Theta'[\langle a, b \rangle \mapsto P, \langle b, G_k.i \rangle \mapsto \Theta'(\langle a, G_k.i \rangle)]$.

(WHILE) Let $c' = \mathtt{while}\ e(x)\ \mathtt{do}\ c$. Suppose that $\mathcal{H} = \mathcal{R}; \mathcal{G} \Vdash \{P \wedge (e(x) \neq 0)\}\, c\, \{P\}$ is good, and that $P \wedge (e(x) = 0) \vdash Q$, $\mathcal{R}^R \vdash \mathcal{R}^C$, and $P \nearrow \_ \in \mathcal{R}$. We show that $\mathcal{R} \cup \{Q \nearrow (\mathcal{R}^R \vee Q)\}; \mathcal{G} \Vdash \{P\}\, c'\, \{Q\}$

is good. Let $G = \langle A, L, E\rangle$ be an execution in $\mathcal{SG}; [\![c']\!]; \mathcal{SG}$. Then, there exist $n \geq 0$, distinct $a, a_1, \ldots, a_n, a_{n+1}, b \in \mathbb{N}$, executions $G_1 = \langle A_1, L_1, E_1\rangle, \ldots, G_n = \langle A_n, L_n, E_n\rangle$ in $[\![c]\!]$, $v_1, \ldots, v_{n+1} \in \mathsf{Val}$ satisfying $[\![e]\!](v_i) \neq 0$ for $1 \leq i \leq n$, and $[\![e]\!](v_{n+1}) = 0$, such that $A = \{a, a_1, \ldots, a_{n+1}, b\} \uplus \biguplus A_j$, $L = \{a \mapsto \langle \mathtt{S}\rangle, b \mapsto \langle \mathtt{S}\rangle\} \cup \{a_i \mapsto \langle \mathtt{R}, x, v_i\rangle \mid 1 \leq i \leq n+1\} \cup \bigcup L_j$, and $E = \{\langle a, a_1\rangle, \langle a_{n+1}, b\rangle\} \cup \{\langle a_j, G_j.i\rangle \mid 1 \leq j \leq n\} \cup \{\langle G_j.o, a_{j+1}\rangle \mid 1 \leq j \leq n\} \cup \bigcup E_j$. For $1 \leq j \leq n$, let $H_j = G \cap (A_j \cup \{a_j, a_{j+1}\})[a_j \mapsto \langle \mathtt{S}\rangle, a_{j+1} \mapsto \langle \mathtt{S}\rangle]$. Then, $H_j \in \mathcal{SG}; [\![c]\!]; \mathcal{SG}$ for $1 \leq j \leq n$. Since $\mathcal{H}$ is good, there exist annotations $\Theta_1, \ldots, \Theta_n$ that satisfy the required conditions for $H_1, \ldots, H_n$ (respectively). Let $\Theta = \bigcup \Theta_j|_{H_j.po} \cup \{\langle a, a_1\rangle \mapsto P, \langle a_{n+1}, b\rangle \mapsto Q\}$.

(PAR) Suppose that $\mathcal{H}_1 = \mathcal{R}_1; \mathcal{G}_1 \Vdash \{P_1\}\, c_1\, \{Q_1\}$ and $\mathcal{H}_2 = \mathcal{R}_2; \mathcal{G}_2 \Vdash \{P_2\}\, c_2\, \{Q_2\}$ are good, $\mathcal{R}_1; \mathcal{G}_1$ and $\mathcal{R}_2; \mathcal{G}_2$ are non-interfering, and $Q_1 \wedge Q_2 \vdash Q$. We show that $\mathcal{R}_1 \cup \mathcal{R}_2 \cup \{Q\!\uparrow\!(\mathcal{R}_1^R \vee \mathcal{R}_2^R \vee Q)\}; \mathcal{G}_1 \cup \mathcal{G}_2 \Vdash \{P_1 \wedge P_2\}\, c_1 \parallel c_2\, \{Q\}$ is good. Let $G = \langle A, L, E\rangle$ be an execution in $\mathcal{SG}; [\![c_1 \parallel c_2]\!]; \mathcal{SG}$. Then, there exist distinct $a, a', b, b' \in \mathbb{N}$, and executions $G_1 = \langle A_1, L_1, E_1\rangle$ in $[\![c_1]\!]$ and $G_2 = \langle A_2, L_2, E_2\rangle$ in $[\![c_2]\!]$, such that $A = \{a, a', b', b\} \uplus A_1 \uplus A_2$, $L = \{a \mapsto \langle \mathtt{S}\rangle, a' \mapsto \langle \mathtt{S}\rangle, b' \mapsto \langle \mathtt{S}\rangle, b \mapsto \langle \mathtt{S}\rangle\} \cup L_1 \cup L_2$, and $E = \{\langle a, a'\rangle, \langle a', G_1.i\rangle, \langle a', G_2.i\rangle, \langle G_1.o, b'\rangle, \langle G_2.o, b'\rangle, \langle b', b\rangle\} \cup E_1 \cup E_2$. For $k = 1, 2$, let $H_k = G \cap (A_k \cup \{a', b'\})$. Then, $H_k \in \mathcal{SG}; [\![c_k]\!]; \mathcal{SG}$ for $k = 1, 2$. Since $\mathcal{H}_1$ and $\mathcal{H}_2$ are good, there exist annotations Θ_1 and Θ_2 that satisfy the required conditions for H_1 and H_2 (respectively). Let $\Theta = \Theta_1|_{H_1.po} \cup \Theta_2|_{H_2.po} \cup \{\langle a, a'\rangle \mapsto P_1 \wedge P_2, \langle b', b\rangle \mapsto Q\}$. We only show here that Θ is stable for G. It is straightforward to verify that Θ satisfies the other conditions for G. Let $\langle d', d\rangle \in po$ and $R = \Theta(\langle d', d\rangle)$. Let $c \in \mathtt{W} \cup \mathtt{U}$, $x = loc(c)$, $v = val_w(c)$, and $P_c = \bigwedge_{\langle c', c\rangle \in G.po} \Theta(\langle c', c\rangle)$. Suppose that c interferes with $\langle d', d\rangle$ in G for Θ. We show that $P_c \wedge R \vdash R[v/x]$ if $c \in \mathtt{W}$, and that $P_c \wedge (x = val_r(c)) \wedge R \vdash R[v/x]$ if $c \in \mathtt{U}$. Since $\langle c, d'\rangle \notin G.po^*$ and $\langle d, c\rangle \notin G.po^*$, we have $c \in A_1 \cup A_2$ and $\langle d', d\rangle \in H_1.po \cup H_2.po$. Suppose, w.l.o.g., that $c \in A_1$. Now, if $\langle d', d\rangle \in H_1.po$, then we are done using the fact that Θ_1 is stable for H_1. Assume now that $\langle d', d\rangle \in H_2.po$. Since $\mathcal{R}_2$ covers Θ_2 for H_2, there exist $R_1\!\uparrow\!C_1, \ldots, R_n\!\uparrow\!C_n \in \mathcal{R}_2$, such that $\bigwedge R_i \dashv\vdash R$, and $\Theta_2(\langle d'_0, d_0\rangle) \vdash \bigwedge C_i$ for every $\langle d'_0, d_0\rangle \in po$ that is H_2-before $\langle d', d\rangle$. Consider the two options:

$c \in \mathtt{W}$. Since $\mathcal{G}_1$ covers Θ_1 for H_1, there exist edge $\langle c', c\rangle \in H_1.po$ and assertion P'_c such that $\Theta_1(\langle c', c\rangle) \vdash P'_c$, and the following hold:

- $\{P'_c\}x := v \in \mathcal{G}_1$. Since $\mathcal{R}_1; \mathcal{G}_1$ and $\mathcal{R}_2; \mathcal{G}_2$ are non-interfering, we have $P'_c \wedge R_i \vdash R_i[v/x]$ for every $1 \leq i \leq n$, and so $P_c \wedge R \vdash R[v/x]$.
- The exists an expression $e(y)$, such that $\{P'_c\}x := e(y) \in \mathcal{G}_1$, $c' \in \mathtt{R}_y$, and $[\![e]\!](val_r(c')) = v$. Let $v_y = val_r(c')$. Since $c' \in \mathtt{R}$, we have $\langle c', d'\rangle \notin G.po^*$. Thus since c interferes with $\langle d', d\rangle$ in G for Θ, $\Theta(\langle d'_0, d_0\rangle) \wedge \Theta(\langle c', c\rangle) \not\vdash y \neq v_y$ for some $\langle d'_0, d_0\rangle \in G.po$ such that $\langle d'_0, d_0\rangle$ is G-before $\langle d', d\rangle$ and $\langle d_0, c'\rangle \notin G.po^*$. It follows that $\langle d'_0, d_0\rangle \in H_2.po$, and $\langle d'_0, d_0\rangle$ is H_2-before $\langle d', d\rangle$. Therefore, $\Theta_2(\langle d'_0, d_0\rangle) \vdash \bigwedge C_i$. Hence, $C_i \wedge P'_c \not\vdash y \neq v_y$ for every $1 \leq i \leq n$.

Since $[\![e]\!](v_y) = v$ and $\mathcal{R}_1; \mathcal{G}_1$ and $\mathcal{R}_2; \mathcal{G}_2$ are non-interfering, this implies that $P'_c \wedge R_i \vdash R_i[v/x]$ for every $1 \leq i \leq n$. Therefore, $P_c \wedge R \vdash R[v/x]$.

– The exists an expression $e(y,z)$ and value $v_y \in \mathsf{Val}$, such that $\{P'_c\}x \overset{y,z}{:=} e(y,z) \in \mathcal{G}_1$, $\Theta_1(\langle c', c\rangle) \vdash y = v_y$, $c' \in \mathtt{R}_z$, and $[\![e]\!](v_y, val_r(c')) = v$. Let $v_y = val_r(c')$. As in the previous case, it follows that $C_i \wedge P'_c \not\vdash z \neq v_z$ for every $1 \leq i \leq n$. Now, if $P_c \wedge R$ is inconsistent, then obviously $P_c \wedge R \vdash R[v/x]$. Otherwise, $P_c \wedge C_i$ is consistent as well for every $1 \leq i \leq n$ (since we have $R \vdash C_i$, as $\langle d', d\rangle$ is H_2-before itself). Then, since $P_c \vdash y = v_y$ and $P_c \vdash P'_c$, we have also $C_i \wedge P'_c \not\vdash y \neq v_y$ for every $1 \leq i \leq n$. Since $[\![e]\!](v_y, v_z) = v$ and $\mathcal{R}_1; \mathcal{G}_1$ and $\mathcal{R}_2; \mathcal{G}_2$ are non-interfering, this implies that $P'_c \wedge R_i \vdash R_i[v/x]$ for every $1 \leq i \leq n$. Therefore, $P_c \wedge R \vdash R[v/x]$.

$c \in \mathtt{U}$. Let $v_r = val_r(c)$. Since $\mathcal{G}_1$ covers Θ_1 for H_1, there exist edge $\langle c', c\rangle \in H_1.po$ and assertion P'_c such that $\Theta_1(\langle c', c\rangle) \vdash P'_c$, and $\{P'_c\}x \overset{\mathsf{at}}{:=} e(x) \in \mathcal{G}_1$ for some expression $e(x)$, such that $[\![e]\!](v_r) = v$. Since $\mathcal{R}_1; \mathcal{G}_1$ and $\mathcal{R}_2; \mathcal{G}_2$ are non-interfering, we have $P'_c \wedge R_i \vdash R_i[e(x)/x]$ for every $1 \leq i \leq n$. Since $[\![e]\!](v_r) = v$, this implies that $P'_c \wedge R_i \wedge (x = v_r) \vdash R_i[v/x]$ for every $1 \leq i \leq n$. Therefore, $P_c \wedge R \wedge (x = v_r) \vdash R[v/x]$.

(CONSEQ) Immediately follows from our definitions (keeping the same annotation).

(ASSN'_1) Suppose that $P \vdash Q[e(y)/x]$, $\{P \upharpoonright P, Q \upharpoonright (P \vee Q)\} \leq \mathcal{R}$, and $P \wedge (y = v) \vdash P_v$ and $\{P_v \upharpoonright P\} \leq \mathcal{R}$ for every $v \in \mathsf{Val}$. We show that $\mathcal{R}; \{\{P_v\}x := e(y) \mid v \in \mathsf{Val}\} \Vdash \{P\}\, x := e(y)\, \{Q\}$ is good. Let $G = \langle A, L, E\rangle$ be an execution in $\mathcal{SG}; [\![x := e(y)]\!]; \mathcal{SG}$. Then, there exist $v_y \in \mathsf{Val}$ and distinct $a, b_y, b, c \in \mathbb{N}$, such that $A = \{a, b_y, b, c\}$, $L = \{a \mapsto \langle \mathtt{S}\rangle, b_y \mapsto \langle \mathtt{R}, y, v_y\rangle, b \mapsto \langle \mathtt{W}, x, [\![e]\!](v_y)\rangle, c \mapsto \langle \mathtt{S}\rangle\}$, and $E = \{\langle a, b_y\rangle, \langle b_y, b\rangle, \langle b, c\rangle\}$. Let $\Theta = \{\langle a, b_y\rangle \mapsto P, \langle b_y, b\rangle \mapsto P \wedge P_{v_y}, \langle b, c\rangle \mapsto Q\}$. It is easy to verify that Θ satisfies the required conditions for G. □

Definition 20. An execution $G = \langle A, L, E\rangle$ *extends* an execution $G' = \langle A', L', E'\rangle$ if $A' \subseteq A$, $G'.E_x \subseteq G.E_x$ for every $x \in \mathsf{Loc}$, $G'.po \subseteq G.E_{all}$, and $L'(a) \in \{\langle \mathtt{S}\rangle, L(a)\}$ for every $a \in A'$.

Theorem 5 (Coherence Preserving Transformations). *Let $G = \langle A, L, E\rangle$ be a coherent execution.*

(*a*) *If G extends an execution G', then G' is coherent.*
(*b*) *Let $a, c \in A$ such that $\langle a, c\rangle \in E^+_{all}$. Let G' be an execution obtained from G by adding a new node b labeled with $\langle \mathcal{S}\rangle$, and edges $\langle a, b\rangle, \langle b, c\rangle$. Then, G' is coherent.*
(*c*) *Let $a \in \mathcal{S}$ and $x \in \mathsf{Loc}$. Assume that $\langle b, a\rangle \in E^*_{all}$ for some $b \in \mathtt{W}_x \cup \mathtt{U}_x$. Then, $G' = G \cup \{\langle a_x, a, x\rangle\}$ is coherent for some $a_x \in \mathtt{W}_x \cup \mathtt{U}_x$ such that $\langle a_x, a\rangle \in E^*_{all}$.*
(*d*) *Let $a \in \mathcal{S}$, $b, c \in A$, and $x \in \mathsf{Loc}$. Suppose that $\langle c, a\rangle, \langle a, b\rangle \in E^*_{all}$, and $\langle c, b\rangle \in E_x$. Then, $G' = G \cup \{\langle c, a, x\rangle\}$ is coherent.*
(*e*) *Let $a, b \in A$, $c \in \mathcal{S}$, and $x \in \mathsf{Loc}$. Suppose that $\langle a, b\rangle, \langle b, c\rangle \in E_x$. Then, $G' = (G[b \mapsto \langle \mathcal{S}\rangle] \setminus \{\langle b, d, y\rangle \mid d \in A, y \in \mathsf{Loc}\}) \cup \{\langle b, c\rangle, \langle a, c, x\rangle\}$ is coherent.*
(*f*) *Let $a, c \in A$, $b \in \mathcal{S}$, and $x \in \mathsf{Loc}$. Suppose that: $\langle c, a\rangle \in E_x$; $\langle b, d\rangle \in po$ iff $d = a$; and $\langle d, b\rangle \in E_y$ implies $\langle d, a\rangle \in E_y$ for every $d \in A$ and $y \in \mathsf{Loc}$. Then, $G' = G \cup \{\langle c, b, x\rangle\}$ is coherent.*

Proof. Item (a) is easy to prove. We prove Items (b) to (f)::

(b) Let $O = \{\langle o, b\rangle \mid \langle o, a\rangle \in G.E^*_{all}\} \cup \{\langle b, o\rangle \mid \langle c, o\rangle \in G.E^*_{all}\}$. Clearly, we have $G'.E^+_{all} \subseteq G.E^+_{all} \cup O$. It follows that $G'.E_{all}$ is acyclic. It is also easy to see that $G.wb_x = G'.wb_x$ for every $x \in \mathsf{Loc}$.

(c) Let $B = \{b \in \mathtt{W}_x \cup \mathtt{U}_x \mid \exists c \in A.\ \langle c, a\rangle \in E^+_{all} \wedge \langle c, b\rangle \in E^*_x\}$. Our assumption ensures that B is non-empty. Since G is coherent, B is partially ordered by wb^*_x. Let m be a wb^*_x-maximal element in B. Let $C = \{c \in A \mid \langle c, a\rangle \in E^+_{all}, \langle c, m\rangle \in E^*_x\}$. Since $m \in B$, we have that C is non-empty. By Proposition 1, C is linearly ordered by E^*_{all}. Choose a_x to be the E^*_{all}-maximal element of C. We show that $G' = G \cup \{\langle a_x, a, x\rangle\}$ is coherent. First, note that $G'.E^+_{all} = G.E^+_{all}$ (since $\langle a_x, a\rangle \in G.E^+_{all}$). Hence, $G'.E_{all}$ is acyclic. This also implies that $G'.wb_y = G.wb_y$ for every $y \neq x$. Next, we show that $G'.wb_x$ is acyclic as well. Let $B' = \{b' \in \mathtt{W}_x \cup \mathtt{U}_x \mid \exists c \in A.\ \langle c, a\rangle \in E^+_{all} \wedge \langle c, b'\rangle \in G.E^{*/a_x}_x\}$. Then $B' \subseteq B$, and the following hold:

- If $\langle a_x, b'\rangle \in G.wb^+_x$ for some $b' \in B'$, then $\langle b', m\rangle \in G.E^*_x$.
  Proof. Assume that $\langle a_x, b'\rangle \in G.wb^+_x$ for some $b' \in B'$. Suppose for contradiction that $\langle b', m\rangle \notin G.E^*_x$. Then, $\langle a_x, m\rangle \in G.E^{*/b'}_x$. By Proposition 1, $\langle m, b'\rangle \in G.wb^+_x$. But, since $b' \in B' \subseteq B$, this contradicts the $G.wb_x$-maximality of m in B.
- $G'.wb_x \subseteq G.wb_x \cup \{\langle b', a_x\rangle \mid b' \in B'\}$.
  Proof. Let $\langle a_1, a_2\rangle \in G'.wb_x$. Then, $a_1, a_2 \in \mathtt{W}_x \cup \mathtt{U}_x$, $a_1 \neq a_2$, and there exists a pair $\langle a'_1, a'_2\rangle \in G'.E^+_{all}$, such that $\langle a'_1, a_1\rangle \in G'.E^{*/a_2}_x$, and $\langle a_2, a'_2\rangle \in G'.E^=_x$. Since $G'.E^+_{all} = G.E^+_{all}$, we have $\langle a'_1, a'_2\rangle \in G.E^+_{all}$ as well. Since $a \in \mathtt{S}$, we also have $\langle a'_1, a_1\rangle \in G.E^{*/a_2}_x$. Now, if $\langle a_2, a'_2\rangle \in G.E^=_x$, then $\langle a_1, a_2\rangle \in G.wb_x$, and we are done. Otherwise, $a_2 = a_x$ and $a'_2 = a$. In this case $\langle a'_1, a\rangle \in G.E^+_{all}$ and $\langle a'_1, a_1\rangle \in G.E^{*/a_x}_x$, so $a_1 \in B'$.

  Now, suppose for contradiction that $G'.wb_x$ contains a cycle. Since $G.wb_x$ is acyclic, this cycle must include an edge $\langle b', a_x\rangle \in G'.wb_x \setminus G.wb_x$ for some $b' \in B'$. W.l.o.g., we can assume it contains exactly one such edge, and thus $\langle a_x, b'\rangle \in G.wb^+_x$. Since $b' \in B'$, there exists $c \in A$ such that $\langle c, a\rangle \in E^+_{all}$ and $\langle c, b'\rangle \in G.E^{*/a_x}_x$. Since $\langle a_x, b'\rangle \in G.wb^+_x$, we have $\langle b', m\rangle \in G.E^*_x$. Hence, we also have $\langle c, m\rangle \in G.E^*_x$, and so $c \in C$. Since C is linearly ordered by E^*_{all} and $c \neq a_x$, the E^*_{all}-maximality of a_x entails that $\langle c, a_x\rangle \in E^+_{all}$. But, this implies that $\langle b', a_x\rangle \in G.wb_x$ (as $\langle c, a_x\rangle \in G.E^+_{all}$ and $\langle c, b'\rangle \in G.E^{*/a_x}_x$), contradicting our assumption.

(d) Clearly, $G'.E^+_{all} = G.E^+_{all}$. Since $G.E_{all}$ is acyclic, this implies that $G'.E_{all}$ is acyclic. Additionally, this implies that $G'.wb_y = G.wb_y$ (and so $G'.wb_y$ is acyclic) for every $y \neq x$. We show that $G'.wb_x \subseteq G.wb_x$ (and so, since $G.wb_x$ is acyclic, so is $G'.wb_x$). Let $\langle a_1, a_2\rangle \in G'.wb_x$. Then, $a_1, a_2 \in \mathtt{W}_x \cup \mathtt{U}_x$, $a_1 \neq a_2$, and there exists a pair $\langle a'_1, a'_2\rangle \in G'.E^+_{all}$, such that $\langle a'_1, a_1\rangle \in G'.E^{*/a_2}_x$, and $\langle a_2, a'_2\rangle \in G'.E^=_x$. Since $G'.E^+_{all} = G.E^+_{all}$, we have $\langle a'_1, a'_2\rangle \in G.E^+_{all}$ as well. Since $a \in \mathtt{S}$, we also have $\langle a'_1, a_1\rangle \in G.E^{*/a_2}_x$. Now, if $\langle a_2, a'_2\rangle \in G.E^=_x$, then $\langle a_1, a_2\rangle \in G.wb_x$, and we are done. Otherwise, $a_2 = c$ and $a'_2 = a$. Then,

$\langle a_2, b\rangle \in G.E_x$, and $\langle a'_1, b\rangle \in G.E^+_{all}$ (since $\langle a'_1, a\rangle \in G.E^+_{all}$ and $\langle a, b\rangle \in G.E^*_{all}$). It follows that $\langle a_1, a_2\rangle \in G.wb_x$.

(e) We have $G'.E^+_{all} \subseteq G.E^+_{all}$, and thus it remains to prove that $G'.wb_x$ is acyclic. We first prove that $G'.wb_x \subseteq G.wb_x \cup \{\langle b', a\rangle \mid b' \in B\}$, where $B = \{b' \in G \mid \langle b', b\rangle \in G.wb_x\}$. Let $\langle a_1, a_2\rangle \in G'.wb_x$. Then, $a_1, a_2 \in G'.\mathtt{W}_x \cup G'.\mathtt{U}_x$, $a_1 \neq a_2$, and there exists a pair $\langle a'_1, a'_2\rangle \in G'.E^+_{all}$, such that $\langle a'_1, a_1\rangle \in G'.E^{*/a_2}_x$, and $\langle a_2, a'_2\rangle \in G'.E^=_x$. Since $G'.E^+_{all} \subseteq G.E^+_{all}$, we have $\langle a'_1, a'_2\rangle \in G.E^+_{all}$ as well. Since $c \in \mathtt{S}$, we also have $\langle a'_1, a_1\rangle \in G.E^{*/a_2}_x$. Now, if $\langle a_2, a'_2\rangle \in G.E^=_x$, then $\langle a_1, a_2\rangle \in G.wb_x$, and we are done. Otherwise, $a_2 = a$ and $a'_2 = c$. Since $b \in G'.\mathtt{S}$, we have $b \neq a_1$ and $\langle a'_1, a_1\rangle \in G.E^{*/b}_x$. Hence, $\langle a_1, b\rangle \in G.wb_x$ (since $\langle a'_1, a'_2\rangle \in G.E^+_{all}$, $\langle a'_1, a_1\rangle \in G.E^{*/b}_x$, and $\langle b, c\rangle \in G.E_x$). Now, suppose for contradiction that $G'.wb_x$ contains a cycle. Since $G.wb_x$ is acyclic, this cycle must include an edge $\langle b', a\rangle \in G'.wb_x \setminus G.wb_x$ for some $b' \in B$. W.l.o.g., we can assume it contains exactly one such edge, and thus $\langle a, b'\rangle \in G.wb^+_x$. Since $b' \in B$, we have $\langle b', b\rangle \in G.wb_x$, and in particular $b' \neq b$. Since $\langle a, b'\rangle \in G.wb^+_x$ and $\langle a, b\rangle \in E^{*/b'}_x$, by Proposition 1, $\langle b, b'\rangle \in G.wb^+_x$. But, this implies that $G.wb_x$ is cyclic.

(f) First, we have $G'.E^+_{all} \subseteq G.E^+_{all} \cup \{\langle d, b\rangle \mid d \neq b, \langle d, a\rangle \in G.E^+_{all}\}$, and therefore, $G'.E_{all}$ is acyclic. Indeed, any non-empty path in $G'.E_{all}$ that does not use the edge $\langle c, b, x\rangle$ is a non-empty path in $G.E_{all}$. If it does use this edge, we can assume it uses it exactly once, and either it continues to a, or ends in b. In the first case, a corresponding non-empty path in $G.E_{all}$ can use the edge $\langle c, a, x\rangle$. In the latter, let $d \in G$ be the first node in this path. Then, $\langle d, c\rangle \in G.E^*_{all}$, and hence $\langle d, a\rangle \in G.E^+_{all}$. Suppose for contradiction that $d = b$. Then this path must go first to a, and continue with a path from a to c in $G.E_{all}$. But, $\langle c, a\rangle \in G.E_{all}$, and this contradicts the fact that G is coherent.

Next, let $y \in \mathsf{Loc}$. We show that $G'.wb_y \subseteq G.wb_y$ (and so, since $G.wb_y$ is acyclic, so is $G'.wb_y$). Let $\langle a_1, a_2\rangle \in G'.wb_y$. Then, $a_1, a_2 \in \mathtt{W}_y \cup \mathtt{U}_y$, $a_1 \neq a_2$, and there exists a pair $\langle a'_1, a'_2\rangle \in G'.E^+_{all}$, such that $\langle a'_1, a_1\rangle \in G'.E^{*/a_2}_y$, and $\langle a_2, a'_2\rangle \in G'.E^=_y$. Since $b \in \mathtt{S}$, we have $\langle a'_1, a_1\rangle \in G.E^{*/a_2}_y$ as well. Consider two cases:

- $a'_2 \neq b$. In this case, $\langle a'_1, a'_2\rangle \in G.E^+_{all}$ as well. Further, $\langle a_2, a'_2\rangle \in G'.E_y$ iff $\langle a_2, a'_2\rangle \in G.E_y$. Hence, $\langle a_2, a'_2\rangle \in G.E^=_y$, and therefore, $\langle a_1, a_2\rangle \in G.wb_y$.
- $a'_2 = b$. In this case, since $\langle a'_1, a'_2\rangle \in G'.E^+_{all}$, either $\langle a'_1, a'_2\rangle \in G.E^+_{all}$, or $\langle a'_1, a\rangle \in G.E^+_{all}$. In the first case we have $\langle a'_1, a\rangle \in G.E^+_{all}$ as well (since $\langle b, a\rangle \in G.po$). Now, since $b \in \mathtt{S}$ and $a_2 \in \mathtt{W} \cup \mathtt{U}$, $b \neq a_2$, and so $\langle a_2, b\rangle \in G'.E_y$. It follows that $\langle a_2, a\rangle \in G.E_y$. Therefore, we have $\langle a_1, a_2\rangle \in G.wb_y$ (since $\langle a'_1, a\rangle \in G.E^+_{all}$, $\langle a'_1, a_1\rangle \in G.E^{*/a_2}_y$, and $\langle a_2, a\rangle \in G.E_y$) in this case as well. □

Lemma 2. *Let $G = \langle A, L, E\rangle$ be an execution, $\langle a, b\rangle \in po$, and σ be a state that is visible at $\langle a, b\rangle$ in G. Let $c \in A \setminus \{a, b\}$. If $c \notin \mathtt{W} \cup \mathtt{U}$, $val_w(c) \neq \sigma(loc(c))$, or $\langle b, c\rangle \in E^*_{all}$, then σ is visible at $\langle a, b\rangle$ in $G \setminus \{c\}$.*

Proof. Let D be a $\langle G, \langle a, b\rangle\rangle$-reader of σ. We show that $D(x) \neq c$ for every $x \in \mathsf{Loc}$. Using Theorem 5(a), this entails that D is also a $\langle G \setminus \{c\}, \langle a, b\rangle\rangle$-reader of σ. Let $x \in \mathsf{Loc}$. First, if $c \notin \mathtt{W} \cup \mathtt{U}$ or $val_w(c) \neq \sigma(loc(c))$, then by definition $D(x) \neq c$. Suppose now that $\langle b, c\rangle \in E^*_{all}$, but $D(x) = c$. Let $H = \mathcal{S}(G, \langle a, b\rangle, D[\mathsf{Loc}]) \cup \{\langle D(y), b, y\rangle \mid y \in \mathsf{Loc}\}$. Since D is a $\langle G, \langle a, b\rangle\rangle$-reader, H is coherent. But, $\langle b, c\rangle \in H.E^*_{all}$ and $\langle c, b\rangle \in H.E_{all}$, and so $H.E^+_{all}$ is cyclic. □

Definition 21. Let $G = \langle A, L, E\rangle$ be an execution. An execution G' is called a *prefix* of G if $G' = G \cap A'$ for some $A' \subseteq A$ that is downwards closed with respect to $G.E^*_{all}$ (i.e., if $b \in A'$ and $\langle a, b\rangle \in G.E^*_{all}$ then $a \in A'$).

Lemma 3. *Let G' be a prefix of an execution G. If a state σ is visible at $\langle a, b\rangle \in G'.po$ in G', then it is also visible at $\langle a, b\rangle$ in G.*

Proof. Let D be a $\langle G', \langle a, b\rangle\rangle$-reader of σ. Then, $D(x) \in G'.\mathtt{W}_x \cup G'.\mathtt{U}_x$ and $val_w(D(x)) = \sigma(x)$ for every $x \in \mathsf{Loc}$; and $\mathcal{S}(G', \langle a, b\rangle, D[\mathsf{Loc}]) \cup \{\langle D(x), b, x\rangle \mid x \in \mathsf{Loc}\}$ is coherent. Since $D[\mathsf{Loc}] \cup \{a\} \subseteq G'$, we have $\mathcal{S}(G', \langle a, b\rangle, D[\mathsf{Loc}]) = \mathcal{S}(G, \langle a, b\rangle, D[\mathsf{Loc}])$. Hence D is a $\langle G, \langle a, b\rangle\rangle$-reader of σ as well. □

Lemma 4. *Let $G = \langle A, L, E\rangle$ be an execution. Let $\langle a, b\rangle \in po$ and $B \subseteq A$. Suppose that a state σ is visible at $\langle a, b\rangle$ in $\mathcal{S}(G, \langle a, b\rangle, B)$. Then:*

- *σ is visible at $\langle a, b\rangle$ in G.*
- *Let $c \in A \setminus (B \cup \{a, b\})$ such that $\langle c, d\rangle \in G.E^+_{all}$ implies that $d = b$. Then, σ is visible at $\langle a, b\rangle$ in $G \setminus \{c\}$.*

Proof. Let $G' = \mathcal{S}(G, \langle a, b\rangle, B)$. Let D be a $\langle G', \langle a, b\rangle\rangle$-reader of σ. Then, $D(x) \in G'.\mathtt{W}_x \cup G'.\mathtt{U}_x$ and $val_w(D(x)) = \sigma(x)$ for every $x \in \mathsf{Loc}$; and $\mathcal{S}(G', \langle a, b\rangle, D[\mathsf{Loc}]) \cup \{\langle D(x), b, x\rangle \mid x \in \mathsf{Loc}\}$ is coherent. It is easy to verify that $\mathcal{S}(G, \langle a, b\rangle, D[\mathsf{Loc}]) = \mathcal{S}(G', \langle a, b\rangle, D[\mathsf{Loc}])$, as well as $\mathcal{S}(G \setminus \{c\}, \langle a, b\rangle, D[\mathsf{Loc}]) = \mathcal{S}(G', \langle a, b\rangle, D[\mathsf{Loc}])$. It follows that D is a $\langle G, \langle a, b\rangle\rangle$-reader of σ, as well as a $\langle G \setminus \{c\}, \langle a, b\rangle\rangle$-reader of σ. □

Lemma 5. *Let G be an execution, and let $\langle a, b\rangle \in po$. Let $D : \mathsf{Loc} \to \mathbb{N}$, such that $D(x) \in \mathtt{W}_x \cup \mathtt{U}_x$ for every $x \in \mathsf{Loc}$. Let H be an extension of $\mathcal{S}(G, \langle a, b\rangle, D[\mathsf{Loc}])$ such that $\langle D(x), b\rangle \in H.E_x$ for every $x \in \mathsf{Loc}$. If H is coherent, then the state $\sigma = \lambda x \in \mathsf{Loc}.\ val_w(D(x))$ is visible at $\langle a, b\rangle$ in G.*

Proof. Since H is coherent, by Theorem 5(a), so is $\mathcal{S}(G, \langle a, b\rangle, D[\mathsf{Loc}]) \cup \{\langle D(x), b, x\rangle \mid x \in \mathsf{Loc}\}$. It follows that D is a $\langle G, \langle a, b\rangle\rangle$-reader of σ, and so σ is visible at $\langle a, b\rangle$ in G. □

Lemma 6. *Let $G = \langle A, L, E\rangle$ be a coherent initialized execution, and let $\langle a, b\rangle \in po$. Then, there exists a state σ that is visible at $\langle a, b\rangle$ in G.*

Proof. The case that $\langle a, b\rangle$ is initial is easy. Suppose otherwise. Since G is coherent, by Theorem 5(a), so is $H = \mathcal{S}(G, \langle a, b\rangle, \emptyset)$. By repeatedly applying Theorem 5(c) on the (initialized) execution H, the node b, and each $x \in \mathsf{Loc}$, we obtain a function $D : \mathsf{Loc} \to \mathbb{N}$, such that $D(x) \in \mathtt{W}_x \cup \mathtt{U}_x$ for every $x \in \mathsf{Loc}$,

and $H' = H \cup \{\langle D(x), b, x\rangle \mid x \in \mathsf{Loc}\}$ is coherent. H' is an an extension of $\mathcal{S}(G, \langle a, b\rangle, D[\mathsf{Loc}])$, and $\langle D(x), b\rangle \in H'.E_x$ for every $x \in \mathsf{Loc}$. By Lemma 5, $\sigma = \lambda x \in \mathsf{Loc}.\ val_w(D(x))$ is visible at $\langle a, b\rangle$ in G. □

Lemma 7. *Let $G = \langle A, L, E\rangle$ be an initialized execution.*

(*a*) *Let $\langle a, b\rangle \in po$, such that $\langle c, a\rangle \in E^*_{all}$ for every $c \in A \setminus \{b\}$, and $G \setminus \{b\}$ is complete. Let $x_a = loc(a)$ (undefined if $a \in \mathtt{S}$). Let σ be a state visible at $\langle a, b\rangle$ in G. Then the following hold:*
- *If $a \in \mathtt{S}$, then σ is visible at $\langle a', a\rangle$ in $G \setminus \{b\}$ for every $a' \in A$ with $\langle a', a\rangle \in po$.*
- *If $a \in \mathtt{R}$, then $\sigma(x_a) = val_r(a)$, and σ is visible at $\langle a', a\rangle$ in $G \setminus \{b\}$ for every $a' \in A$ with $\langle a', a\rangle \in po$.*
- *If $a \in \mathtt{W}$, then $\sigma(x_a) = val_w(a)$, and some x_a-variant σ' of σ is visible at $\langle a', a\rangle$ in $G \setminus \{b\}$ for every $a' \in A$ with $\langle a', a\rangle \in po$.*[3]
- *If $a \in \mathtt{U}$, then $\sigma(x_a) = val_w(a)$, and $\sigma[x_a \mapsto val_r(a)]$ is visible at $\langle a', a\rangle$ in $G \setminus \{b\}$ for every $a' \in A$ with $\langle a', a\rangle \in po$.*

(*b*) *Let $\langle a', a\rangle \in po$ be a non-initial edge, and let $b \in A \setminus \{a', a\}$, such that $\langle b, d\rangle \notin E_{all}$ for every $d \neq a$, and $G \setminus \{a\}$ is complete. Suppose that a state σ is visible at $\langle a', a\rangle$ in G, but not in $G \setminus \{b\}$. Then, there exist a prefix G' of G that does not include a, and some $v \in \mathsf{Val}$ such that the following hold:*
(a) $\sigma[loc(b) \mapsto v]$ is visible at $\langle a', a\rangle$ in $G \setminus \{b\}$.
(b) $\sigma[loc(b) \mapsto v]$ is visible at $\langle b', b\rangle$ in G' for every $b' \in A$ with $\langle b', b\rangle \in G.po$.
(c) If $b \in \mathtt{U}$, then $v = val_r(b)$.

(*c*) *Suppose that G is coherent. Let $\langle a_1, a_2\rangle, \langle c_1, c_2\rangle \in po$, such that $\langle a_2, c_1\rangle \notin E^*_{all}$ and c_1 is not initial. Then, there exists an edge $\langle b_1, b_2\rangle \in po$ such that $\langle b_1, b_2\rangle$ is G-before $\langle a_1, a_2\rangle$, $\langle b_2, c_1\rangle \notin E^*_{all}$, and every state that is visible at $\langle c_1, c_2\rangle$ in $\mathcal{S}(G, \langle c_1, c_2\rangle, \emptyset)$ is also visible at $\langle b_1, b_2\rangle$ in $\mathcal{S}(G, \langle b_1, b_2\rangle, \{c_1\})$.*

Proof. (a) Let D be a $\langle G, \langle a, b\rangle\rangle$-reader of σ. Then, the execution $H = \mathcal{S}(G, \langle a, b\rangle, D[\mathsf{Loc}]) \cup \{\langle D(x), b, x\rangle \mid x \in \mathsf{Loc}\}$ is coherent. Since D is a $\langle G, \langle a, b\rangle\rangle$-reader of σ, H is coherent, and, if $a \notin \mathtt{S}$, $D(x_a) \in \mathtt{W}_{x_a} \cup \mathtt{U}_{x_a}$. We first show that the following hold:

(i) If $a \in \mathtt{W} \cup \mathtt{U}$, then $D(x_a) = a$.

Proof. Suppose for contradiction that $a \in \mathtt{W} \cup \mathtt{U}$, but $D(x_a) \neq a$. Then, $\langle D(x_a), a\rangle \in H.E^+_{all}$, and so $\langle D(x_a), a\rangle \in H.wb_{x_a}$. However, we also have $\langle a, D(x_a)\rangle \in H.wb_{x_a}$ (since $\langle a, b\rangle \in H.E_{all}$ and $\langle D(x_a), b\rangle \in H.E_{x_a}$). This contradicts the fact that H is coherent.

(ii) If $a \in \mathtt{R}$ and $\langle a_0, a\rangle \in E_{x_a}$, then $D(x_a) = a_0$.

Proof. Assume that $a \in \mathtt{R}$ and $\langle a_0, a\rangle \in E_{x_a}$. Since $a \in \mathtt{R}$, we have $a \neq D(x_a)$, and so $\langle D(x_a), a\rangle \in H.E^+_{all}$. Suppose for contradiction that $D(x_a) \neq a_0$. Hence, $\langle D(x_a), a_0\rangle \in H.wb_{x_a}$ (since $\langle D(x_a), a\rangle \in H.E^+_{all}$ and $\langle a_0, a\rangle \in H.E_{x_a}$), and also $\langle a_0, D(x_a)\rangle \in H.wb_{x_a}$ (since $\langle a_0, a\rangle \in H.E_{all}$, $\langle a, b\rangle \in H.E_{all}$, and $\langle D(x_a), b\rangle \in H.E_{x_a}$). Again, this contradicts the fact that H is coherent.

[3] By *x-variant* of a σ we mean a state σ' such that $\sigma' = \sigma[x \mapsto v]$ for some $v \in \mathsf{Val}$.

Now, let $H_s = H[a \mapsto \langle \mathtt{S} \rangle] \setminus \{\langle a, d, x\rangle \mid d \in A, x \in \mathsf{Loc}\}$. Let $X = \{x \in loc \mid D(x) \neq a\}$. Now, by repeatedly applying Theorem 5(d) for each location $x \in X$ and node $D(x)$ starting from H_s, we obtain that $H' = H_s \cup \{\langle D(x), a, x\rangle \mid x \in X\}$ is coherent. Next, consider the four cases:

- $a \in G.\mathtt{S}$. In this case, $X = \mathsf{Loc}$. Let $a' \in A$ with $\langle a', a\rangle \in po$. H' is an extension of $\mathcal{S}(G \setminus \{b\}, \langle a', a\rangle, D[\mathsf{Loc}])$ and $\langle D(x), a\rangle \in H'.E_x$ for every $x \in \mathsf{Loc}$. By Lemma 5, σ is visible at $\langle a', a\rangle$ in $G \setminus \{b\}$.
- $a \in G.\mathtt{R}$. As in the previous case, one obtains that σ is visible at $\langle a', a\rangle$ in $G \setminus \{b\}$ for every $a' \in A$ with $\langle a', a\rangle \in po$. It remains to show that $\sigma(x_a) = val_r(a)$. Since $G \setminus \{b\}$ is complete, $\langle a_0, a\rangle \in G.E_{x_a}$ for some $a_0 \in A$. By (ii) above, $D(x_a) = a_0$. Hence, $\sigma(x_a) = val_w(D(x_a)) = val_w(a_0) = val_r(a)$.
- $a \in G.\mathtt{W}$. In this case, by (i) above, $D(x_a) = a$, and so $\sigma(x_a) = val_w(a)$, and $X = \mathsf{Loc} \setminus \{x_a\}$. It remains to show that some x_a-variant of σ is visible at $\langle a', a\rangle$ in $G \setminus \{b\}$ for every $a' \in A$ with $\langle a', a\rangle \in po$. If a is initial, then the claim vacuously holds. Assume otherwise. Then, H' is initialized. By Theorem 5(c), there exists some $a_0 \in \mathtt{W}_{x_a} \cup \mathtt{U}_{x_a}$, such that $\langle a_0, a\rangle \in H'.E^*_{all}$ and $H'' = H' \cup \{\langle a_0, a, x_a\rangle\}$ is coherent. Let $D' = D[x_a \mapsto a_0]$, and $a' \in A$ with $\langle a', a\rangle \in po$. Then, H'' is an extension of $\mathcal{S}(G \setminus \{b\}, \langle a', a\rangle, D'[\mathsf{Loc}])$ and $\langle D'(x), a\rangle \in H''.E_x$ for every $x \in \mathsf{Loc}$. By Lemma 5, $\sigma[x_a \mapsto val_w(a_0)]$ is visible at $\langle a', a\rangle$ in $G \setminus \{b\}$.
- $a \in G.\mathtt{U}$. As in the previous case, $D(x_a) = a$, and so $\sigma(x_a) = val_w(a)$, and $X = \mathsf{Loc} \setminus \{x_a\}$. Since $G \setminus \{b\}$ is complete, we have $\langle a_0, a\rangle \in H'.E_{x_a}$ for some $a_0 \in A$. Let $D' = D[x_a \mapsto a_0]$, and $a' \in A$ with $\langle a', a\rangle \in po$. Then, H' is an extension of $\mathcal{S}(G \setminus \{b\}, \langle a', a\rangle, D'[\mathsf{Loc}])$ and $\langle D'(x), a\rangle \in H'.E_x$ for every $x \in \mathsf{Loc}$. By Lemma 5, $\sigma[x_a \mapsto val_w(a_0)] = \sigma[x_a \mapsto val_r(a)]$ is visible at $\langle a', a\rangle$ in $G \setminus \{b\}$.

(b) Let D be a $\langle G, \langle a', a\rangle\rangle$-reader of σ. Then $H = \mathcal{S}(G, \langle a', a\rangle, D[\mathsf{Loc}]) \cup \{\langle D(x), a, x\rangle \mid x \in \mathsf{Loc}\}$ is coherent. Let $x_b = loc(b)$. Since σ is not visible at $\langle a', a\rangle$ in $G \setminus \{b\}$, D is not a $\langle G \setminus \{b\}, \langle a', a\rangle\rangle$-reader of σ, and so $D(x_b) = b$. Hence, $b \in G.\mathtt{W} \cup G.\mathtt{U}$, and $\langle b, a\rangle \in H.E_{x_b}$. Let $H_s = (H[b \mapsto \langle \mathtt{S} \rangle] \setminus \{\langle b, a, x_b\rangle\}) \cup \{\langle b, a\rangle\}$. By Theorem 5(a), H_s is coherent. Now, choose a_{x_b} to be some node in $H_s.\mathtt{W}_{x_b} \cup H_s.\mathtt{U}_{x_b}$ such that $H'_s = H_s \cup \{\langle a_{x_b}, a, x_b\rangle\}$ is coherent and $val_w(a_{x_b}) = val_r(b)$ if $b \in \mathtt{U}$. To see that such a node exists, consider the following two cases:

- $b \notin \mathtt{U}$. By Theorem 5(c), there is some $a_{x_b} \in H_s.\mathtt{W}_{x_b} \cup H_s.\mathtt{U}_{x_b}$, such that $H_s \cup \{\langle a_{x_b}, a, x_b\rangle\}$ is coherent.
- $b \in \mathtt{U}$. Since $G \setminus \{a\}$ is complete, $\langle a_{x_b}, b\rangle \in G.E_{x_b}$ for some $a_{x_b} \in A$ (and, by definition, $val_w(a_{x_b}) = val_r(b)$). Since $\langle a_{x_b}, b\rangle \in G.E_{x_b}$, we also have $\langle a_{x_b}, b\rangle \in H.E_{x_b}$. By Theorem 5(e), $H_s \cup \{\langle a_{x_b}, a, x_b\rangle\}$ is coherent.

Let $D' = D[x_b \mapsto a_{x_b}]$ and $\sigma' = \sigma[x_b \mapsto val_w(a_{x_b})]$. H'_s is an extension of $\mathcal{S}(G \setminus \{b\}, \langle a', a\rangle, D'[\mathsf{Loc}])$, and $\langle D'(x), a\rangle \in H'_s.E_x$ for every $x \in \mathsf{Loc}$. By Lemma 5, σ' is visible at $\langle a', a\rangle$ in $G \setminus \{b\}$. Now, let $G' = H \setminus \{a\}$. G' is a prefix of G that does not include a. Let $b' \in A$ such that $\langle b', b\rangle \in G.po$. Let $H_0 = H'_s \setminus \{\langle c, b, x\rangle \in E \mid c \in A, x \in \mathsf{Loc}\}$. By repeatedly applying Theorem 5(f) on a, b and $D'(x)$ for each $x \in \mathsf{Loc}$, starting from H_0, one derives the coherence of

$H_1 = H_0 \cup \{\langle D'(x), b, x\rangle \mid x \in \mathsf{Loc}\}$. H_1 is an extension of $\mathcal{S}(G', \langle b', b\rangle, D'[\mathsf{Loc}])$ and $\langle D'(x), b\rangle \in H_1.E_x$ for every $x \in \mathsf{Loc}$. By Lemma 5, σ' is visible at $\langle b', b\rangle$ in G'.

(c) Let $B = \{b \in A \mid b = a_2 \vee \langle b, a_1\rangle \in po^*, \langle b, c_1\rangle \notin E^*_{all}\}$. Then, $a_2 \in B$ and B is partially ordered by E^*_{all} (since G is coherent). Choose b_2 to be an E^*_{all}-minimal element in B. Since $\langle b_2, c_1\rangle \notin E^*_{all}$ and c_1 is not initial, b_2 is not initial. Choose b_1 to be a_1 if $b_2 = a_2$, or any node with $\langle b_1, b_2\rangle \in po$ otherwise. Then $\langle b_1, b_2\rangle$ is G-before $\langle a_1, a_2\rangle$ and $\langle b_2, c_1\rangle \notin E^*_{all}$. The minimality of b_2 entails that $\langle b_1, c_1\rangle \in E^*_{all}$. Let $G_c = \mathcal{S}(G, \langle c_1, c_2\rangle, \emptyset)$ and $G_b = \mathcal{S}(G, \langle b_1, b_2\rangle, \{c_1\})$. Let D be a $\langle G_c, \langle c_1, c_2\rangle\rangle$-reader of some state σ. Note that $\mathcal{S}(G_c, \langle c_1, c_2\rangle, D[\mathsf{Loc}]) = G_c$ (since $\langle D(x), c_1\rangle \in G_c.E^*_{all}$ for every $x \in \mathsf{Loc}$). Then, $G'_c = G_c \cup \{\langle D(x), c_2, x\rangle \mid x \in \mathsf{Loc}\}$ is coherent. Let H be the execution obtained adding to G'_c the node b_2 labeled with $\langle \mathtt{S}\rangle$, and edges $\langle b_1, b_2\rangle, \langle b_2, c_2\rangle$. Since $\langle b_1, c_2\rangle \in G'_c.E^+_{all}$, by Theorem 5(b), H is coherent. Now, by repeatedly applying Theorem 5(f), on c_2, b_2 and $D(x)$ for each $x \in \mathsf{Loc}$, starting from H, one derives the coherence of $H' = H \cup \{\langle D(x), b_2, x\rangle \mid x \in \mathsf{Loc}\}$. H' extends $\mathcal{S}(G_b, \langle b_1, b_2\rangle, D[\mathsf{Loc}])$ and $\langle D(x), b_2\rangle \in H'.E_x$ for every $x \in \mathsf{Loc}$. By Lemma 5, σ is visible at $\langle b_1, b_2\rangle$ in G_b. □

Proof (of Theorem 3). Let $G = \langle A, L, E\rangle$. Let $G_0 = \langle A_0, L_0, E_0\rangle$ be the initialization part of G and the following skip node, i.e. the execution consisting of initial nodes $o_1, \ldots, o_M$ with $L(o_i) = \langle \mathtt{W}, \nu_i, v_i\rangle$, a node o_0 with $L(o_0) = \langle \mathtt{S}\rangle$, and edges $\langle o_i, o_0\rangle$ for $1 \le i \le M$ (where $\langle o_0, b\rangle \in G.po^*$ for any other node $b \in A$). The local validity of Θ entails that $\Theta(\langle o_i, o_0\rangle)[v_i/\nu_i]$ is logically true for every $1 \le i \le M$. It is then easy to verify that Θ is valid for G_0. Consider an enumeration $a_1, \ldots, a_n$ of the elements of $A \setminus A_0$ according to E^*_{all} (partial) order (i.e., $i \le j$ whenever $\langle a_i, a_j\rangle \in E^*_{all}$). For every $1 \le i \le n$, let $A_i = A_0 \cup \{a_1, \ldots, a_i\}$ and $G_i = G \cap A_i$. Each G_i is complete, coherent, and initialized. We show by induction on i that Θ is valid for each G_i. In particular, it would follow that Θ is valid for $G_n = G$. Suppose that Θ is valid for G_{i-1}. We prove that it is valid for G_i.

We first show that $\Theta(\langle a, a_i\rangle)$ holds at $\langle a, a_i\rangle$ in G_i for every $a \in A$ such that $\langle a, a_i\rangle \in G_i.po$. Let $a \in A$ such that $\langle a, a_i\rangle \in G_i.po$, and let $R = \Theta(\langle a, a_i\rangle)$. Let $B_0 = \{b \in A_i \mid \langle b, a\rangle \in G.E^*_{all}\} \cup \{a_i\}$. Consider an enumeration $b_1, \ldots, b_m$ of the elements of $A_i \setminus B_0$ according to $G.E^*_{all}$ order. For every $0 \le j \le m$, let $B_j = B_0 \cup \{b_1, \ldots, b_j\}$, and $H_j = G_i \cap B_j$. Each H_j is coherent and initialized, and $H_j \setminus \{a_i\}$ is complete. Additionally, $H_j \setminus \{a_i\}$ is a prefix of G_{i-1} for every $0 \le j \le m$. Therefore, by Lemma 3, if a state σ is visible at some edge $\langle c', c\rangle \in H_j.po$ in $H_j \setminus \{a_i\}$, then it is also visible at $\langle c', c\rangle$ in G_{i-1}, and by the induction hypothesis we have $\sigma \models \Theta(\langle c', c\rangle)$. We use induction on j to show that R holds as $\langle a, a_i\rangle$ in each H_j. In particular, it would follow that R holds at $\langle a, a_i\rangle$ in $H_m = G_i$.

1. For $j = 0$, the claim follows by the local validity of Θ for G. Indeed, consider a state σ that is visible at $\langle a, a_i\rangle$ in H_0. Let $x_a = loc(a)$ (undefined if $L(a) = \langle \mathtt{S}\rangle$), $P = \bigwedge_{\langle a', a\rangle \in G.po} \Theta(\langle a', a\rangle)$, and $H'_0 = H_0 \setminus \{a_i\}$. Consider four cases:
   - If $a \in \mathtt{S}$, then by Lemma 7(a), σ is visible at $\langle a', a\rangle$ in H'_0 for every $a' \in H_0$ with $\langle a', a\rangle \in H_0.po$. It follows that $\sigma \models P$ (note that H_0 includes all edges of G going to a). Since Θ is locally valid for G, we obtain that $\sigma \models R$.

- If $a \in \mathtt{R}$, then, as in the previous case, $\sigma \models P$. In addition, in this case Lemma 7(a) also implies $\sigma(x_a) = val_r(a)$. Hence, since Θ is locally valid for G, we obtain that $\sigma \models R$.
- If $a \in \mathtt{W}$, then by Lemma 7(a), $\sigma(x_a) = val_w(a)$, and there exists an x_a-variant σ' of σ that is visible at $\langle a', a\rangle$ in H'_0 for every $a' \in H_0$ such that $\langle a', a\rangle \in H_0.po$. It follows that $\sigma' \models P$. Since Θ is locally valid for G, one of the following holds:
  - $P \vdash R[val_w(a)/x_a]$. In this case, we obtain that $\sigma' \models R[val_w(a)/x_a]$. Since $\sigma(x_a) = val_w(a)$, it follows that $\sigma \models R$.
  - There is a unique node a' such that $\langle a', a\rangle \in G.po$, and we have $a' \in \mathtt{R}$ and $P \wedge (loc(a') = val_r(a')) \vdash R[val_w(a)/x_a]$. Then, $\langle c, a'\rangle \in H'_0.E^*_{all}$ for every $c \in H'_0$. By Lemma 7(a), we have $\sigma'(loc(a')) = val_r(a')$. Thus we obtain that $\sigma' \models R[val_w(a)/x_a]$. Again, it follows that $\sigma \models R$.
- If $a \in \mathtt{U}$, then by Lemma 7(a), $\sigma(x_a) = val_w(a)$, and $\sigma' = \sigma[x_a \mapsto val_r(a)]$ is visible at $\langle a', a\rangle$ in H'_0 for every $a' \in H_0$ such that $\langle a', a\rangle \in H_0.po$. It follows that $\sigma' \models P$. Since Θ is locally valid for G, we obtain that $\sigma' \models R[val_w(a)/x_a]$. Since $\sigma(x_a) = val_w(a)$, it follows that $\sigma \models R$.

2. Assume that the claim holds for H_{j-1}, we prove it for H_j. Consider a state σ that is visible at $\langle a, a_i\rangle$ in H_j. If σ is visible at $\langle a, a_i\rangle$ in H_{j-1}, then we are done by the (inner) induction hypothesis. Otherwise, by Lemma 2, $b_j \in \mathtt{W} \cup \mathtt{U}$ and $val_w(b_j) = \sigma(loc(b_j))$. Let $x_{b_j} = loc(b_j)$, and $P = \bigwedge_{\langle b', b_j\rangle \in G.po} \Theta(\langle b', b_j\rangle)$.

   Claim. $\sigma' \models R \wedge P$ for some x_{b_j}-variant σ' of σ. Furthermore, if $b \in \mathtt{U}$, then this holds for $\sigma' = \sigma[x_{b_j} \mapsto val_r(b_j)]$.

   Proof. By Lemma 7(b), there exist an x_{b_j}-variant σ' of σ and a prefix H' of H_j that does not include a_i, such that σ' is visible at $\langle a, a_i\rangle$ in H_{j-1} and at $\langle b', b_j\rangle$ in H' for every $b' \in H_j$ such that $\langle b', b_j\rangle \in H_j.po$. By the inner induction hypothesis, $\sigma' \models R$. In addition, H' is a prefix of $H_j \setminus \{a_i\}$, and by Lemma 3, σ' is visible at $\langle b', b_j\rangle$ in $H_j \setminus \{a_i\}$ for every $b' \in H_j$ such that $\langle b', b_j\rangle \in H_j.po$. By the outer induction hypothesis, it follows that $\sigma' \models P$. Additionally, if $b_j \in \mathtt{U}$, then Lemma 7(b) also entails that the above holds for $\sigma' = \sigma[x_{b_j} \mapsto val_r(b_j)]$.

   Claim. b_j interferes with $\langle a, a_i\rangle$ in G for Θ.

   Proof. Our construction ensures that $\langle b_j, a\rangle \notin G.po^*$ and $\langle a_i, b_j\rangle \notin G.po^*$. Let $o \in G$ with $L(o) = \langle \mathtt{R}, x_o, v_o\rangle$. Suppose that $\langle o, b_j\rangle \in G.po$, and $\langle o, a\rangle \notin G.po^*$. By Lemma 6, some state σ_o is visible at $\langle o, b_j\rangle$ in $\mathcal{S}(H_j, \langle o, b_j\rangle, \emptyset)$. By Lemma 7(a), since $o \in \mathtt{R}$ and $\mathcal{S}(H_j, \langle o, b_j\rangle, \emptyset) \setminus \{b_j\}$ is complete, $\sigma_o(x_o) = v_o$. By Lemma 7(c), σ_o is also visible at $\langle d', d\rangle$ in $\mathcal{S}(H_j, \langle d', d\rangle, \{o\})$ for some $\langle d', d\rangle \in H_j.po$ such that $\langle d', d\rangle$ is H_j-before $\langle a, a_i\rangle$ and $\langle d, o\rangle \notin H_j.po^*$. Clearly, $\langle d', d\rangle$ is also G-before $\langle a, a_i\rangle$, and $\langle d, o\rangle \notin G.po^*$. By Lemma 4, σ_o is visible at $\langle o, b_j\rangle$ in $H_j \setminus \{a_i\}$. Hence, $\sigma_o \models \Theta(\langle o, b_j\rangle)$. We show that $\sigma_o \models \Theta(\langle d', d\rangle)$. First, if $d = a_i$, then, since σ_o is visible at $\langle d', d\rangle$ in H_{j-1} (by Lemma 4), the induction hypothesis entails that $\sigma_o \models \Theta(\langle d', d\rangle)$. Otherwise, by Lemma 4, σ_o is visible at $\langle d', d\rangle$ in $H_j \setminus \{a_i\}$. Hence, $\sigma_o \models \Theta(\langle d', d\rangle)$ in this case as well. It follows that $\Theta(\langle d', d\rangle) \wedge \Theta(\langle o, b_j\rangle) \not\vdash x_o \neq v_o$.

Now, since Θ is stable for G, it follows that $\sigma' \models R[\sigma(x_{b_j})/x_{b_j}]$, and this implies that $\sigma \models R$.

Next, let $\langle a, b\rangle \in G_i.po$ such that $b \neq a_i$, and let $R = \Theta(\langle a, b\rangle)$. Consider a state σ that is visible at $\langle a, b\rangle$ in G_i. If it is visible at $\langle a, b\rangle$ in G_{i-1}, then we are done by the induction hypothesis. Otherwise, Lemma 2 implies that $a_i \in \mathtt{W} \cup \mathtt{U}$, $val_w(a_i) = \sigma(loc(a_i))$, and $\langle b, a_i\rangle \notin G_i.E^*_{all}$. Let $x_{a_i} = loc(a_i)$, and $P = \bigwedge_{\langle a', a_i\rangle \in G.po} \Theta(\langle a', a_i\rangle)$.

Claim. $\sigma' \models R \wedge P$ for some x_{a_i}-variant σ' of σ. Furthermore, if $a_i \in \mathtt{U}$, then this holds for $\sigma' = \sigma[x_{a_i} \mapsto val_r(a_i)]$.

Proof. By Lemma 7(b), there exist an x_{a_i}-variant σ' of σ and a prefix G' of G_i that does not include b, such that σ' is visible both at $\langle a, b\rangle$ in G_{i-1} and at $\langle a', a_i\rangle$ in G' for every $a' \in G_i$ such that $\langle a', a_i\rangle \in G_i.po$. By the induction hypothesis, $\sigma' \models R$. In addition, by Lemma 3, σ' is visible at $\langle a', a_i\rangle$ in G for every $a' \in G_i$ such that $\langle a', a_i\rangle \in G_i.po$. Since $\Theta(\langle a', a_i\rangle)$ holds at $\langle a', a_i\rangle$ in G_i for every such a', it follows that $\sigma' \models P$ (note that G_i includes all edges going to a_i). Additionally, if $a_i \in \mathtt{U}$, then Lemma 7(b) also entails that the above holds for $\sigma' = \sigma[x_{a_i} \mapsto val_r(a_i)]$.

Claim. a_i interferes with $\langle a, b\rangle$ in G for Θ.

Proof. Our construction ensures that $\langle a_i, a\rangle \notin G.po^*$ and $\langle b, a_i\rangle \notin G.po^*$. Let $o \in G$, with $L(o) = \langle \mathtt{R}, x_o, v_o\rangle$. Suppose that $\langle o, a_i\rangle \in G.po$ and $\langle o, a\rangle \notin G.po^*$. By Lemma 6, some state σ_o is visible at $\langle o, a_i\rangle$ in $\mathcal{S}(G_i, \langle o, a_i\rangle, \emptyset)$. By Lemma 7(a), since $o \in \mathtt{R}$ and $\mathcal{S}(G_i, \langle o, a_i\rangle, \emptyset) \setminus \{a_i\}$ is complete, $\sigma_o(x_o) = v_o$. By Lemma 7(c), σ_o is also visible at $\langle d', d\rangle$ in $\mathcal{S}(G_i, \langle d', d\rangle, \{o\})$ for some $\langle d', d\rangle \in G_i.po$ such that $\langle d', d\rangle$ is G_i-before $\langle a, b\rangle$ and $\langle d, o\rangle \notin G_i.po^*$. We show that this implies that $\sigma_o \models ann(\langle d', d\rangle) \wedge \Theta(\langle o, a_i\rangle)$. First, by Lemma 4, σ_o is also visible at $\langle o, a_i\rangle$ in G_i. Thus, $\sigma_o \models \Theta(\langle o, a_i\rangle)$ (since $\Theta(\langle o, a_i\rangle)$ holds at $\langle o, a_i\rangle$ in G_i). Second, since σ_o is visible at $\langle d', d\rangle$ in $\mathcal{S}(G_i, \langle o, a_i\rangle, \emptyset)$, Lemma 4 implies that it is also visible at $\langle d', d\rangle$ in G_{i-1}. By the induction hypothesis, $\sigma_o \models \Theta(\langle d', d\rangle)$. It follows that $\Theta(\langle d', d\rangle) \wedge \Theta(\langle o, a_i\rangle) \not\vdash x_o \neq v_o$. Clearly, since $\langle d', d\rangle$ is G_i-before $\langle a, b\rangle$, it is also G-before $\langle a, a_i\rangle$; and since $\langle d, o\rangle \notin G_i.po^*$, we also have $\langle d, o\rangle \notin G.po^*$.

Since Θ is stable for G, it follows that $\sigma' \models R[\sigma(x_{a_i})/x_{a_i}]$, and this implies that $\sigma \models R$. □

Proof (of Theorem 1). Suppose that $\mathcal{H} = \mathcal{R}; \mathcal{G} \Vdash \{P\}\, c\, \{Q\}$ is derivable for some $\mathcal{R}$ and $\mathcal{G}$. By Theorem 2, $\{P\}\, c\, \{Q\}$ is safe. Let $G = \langle A, L, E\rangle$ in $\mathcal{WG}(P); [\![c]\!]; \mathcal{SG}$, and let $E' \subseteq A \times A \times \mathsf{Loc}$, such that $G \cup E'$ is complete and coherent. Then, there exist a state σ that satisfies P, distinct $a_1, \ldots, a_M, a, b \in \mathbb{N}$, and execution $G_c = \langle A_c, L_c, E_c\rangle$ in $[\![c]\!]$, such that $A = \{a_1, \ldots, a_M, b\} \uplus A_c$, $L = \{a_i \mapsto \langle \mathtt{W}, \nu_i, \sigma(\nu_i)\rangle \mid 1 \leq i \leq M\} \cup \{b \mapsto \langle \mathtt{S}\rangle\} \cup L_c$, and $E = \{\langle a_i, G_c.i\rangle \mid 1 \leq i \leq M\} \cup \{\langle G_c.o, b\rangle\} \cup E_c$. Let $G_0 \in \mathcal{SG}; \{G \setminus \{a_1, \ldots, a_M\}\}$. Then, $G_0 \in \mathcal{SG}; [\![c]\!]; \mathcal{SG}$. Since $\{P\}\, c\, \{Q\}$ is safe, there exists an annotation Θ_0, that is locally valid and stable for G_0, and assigns some assertion P', such that $P \vdash P'$, to the initial edge of G_0, Θ_0 assigns some assertion Q', such that $Q' \vdash Q$, to the terminal

edge of G_0. Let $\Theta = \Theta_0 \cup \{\langle a_i, G_c.i\rangle \mapsto \nu_i = \sigma(\nu_i) \mid 1 \leq i \leq M\}$. It is easy to verify that Θ is locally valid and stable for $G \cup E'$ (using the fact that Θ_0 is locally valid and stable for G_0, and that it assigns P' to the initial edge of G_0). Since $G \cup E'$ is a complete, coherent and initialized execution, Theorem 3 implies that Θ is valid for $G \cup E'$. Since Θ_0 assigns Q' to the terminal edge of G_0, Θ assigns Q' to this edge as well (which is also the terminal edge of G). Therefore, Q' holds at this edge in $G \cup E'$, and since $Q' \vdash Q$, Q holds as well. □

D Appendix: Basic Owicki-Gries

In this appendix we present the basic OG proof system for SC in a rely/guarantee style. We assume a programming language with the following commands:

$$c ::= \texttt{skip} \mid \texttt{if}\ e\ \texttt{then}\ c\ \texttt{else}\ c \mid \texttt{while}\ e\ \texttt{do}\ c \mid c\,;c \mid c \parallel c \mid x := e$$

Definition 22. A *basic OG judgment* $\mathcal{R};\mathcal{G} \Vdash \{P\}\,c\,\{Q\}$ extends a Hoare triple with two extra components:

- A finite set $\mathcal{R}$ of pairs of assertions; and
- A finite set $\mathcal{G}$ of *guarded assignments*, i.e., pairs of the form $\{R\}c$, where R is an assertion and c is an assignment command. We write $\mathcal{G} \leq \mathcal{G}'$ for such sets $\mathcal{G}, \mathcal{G}'$, if for every pair $\{R\}c \in \mathcal{G}$ there exists a pair $\{R'\}c \in \mathcal{G}'$ such that $R \vdash R'$.

Definition 23. An assertion R is *stable* under $\{P\}x := e$ if $R \wedge P \vdash R[e/x]$.

Definition 24. The pairs $\mathcal{R}_1;\mathcal{G}_1$ and $\mathcal{R}_2;\mathcal{G}_2$ are *non-interfering* if every $R \in \mathcal{R}_i$ is stable under every $\{P\}c \in \mathcal{G}_j$ for $i \neq j$.

Figure 12 presents the proof system. Soundness of this system under SC holds only if each assignment command and each expression is executed (or evaluated) as an indivisible action (exactly as in usual OG, see [10]). Alternatively, one may consider only programs in which each assignment and each expression refers to at most one global variable (see condition (3.1) in [10]). Soundness of this system under SC can be derived from Theorem 1 as follows. Take Loc to consist of just one location X, and Val to be the set of functions from program variables to values. Every (indivisible) assignment $x := e$ in the current language can be straightforwardly encoded as an atomic assignment (RMW) $X \stackrel{\mathsf{at}}{:=} e'(X)$ of the language studied in the paper. Moreover, the current rule for assignment and the current stability condition correspond exactly to the rule for atomic assignments and the stability condition for them that are included in this paper.

$$\text{(CONSEQ)}\ \frac{\mathcal{R};\mathcal{G} \Vdash \{P\}\,c\,\{Q\} \quad P' \vdash P \quad Q \vdash Q' \quad \mathcal{R} \subseteq \mathcal{R}' \quad \mathcal{G} \leq \mathcal{G}'}{\mathcal{R}';\mathcal{G}' \Vdash \{P'\}\,c\,\{Q'\}} \qquad \text{(SKIP)}\ \frac{P \in \mathcal{R}}{\mathcal{R};\emptyset \Vdash \{P\}\,\texttt{skip}\,\{P\}} \qquad \text{(ASSN)}\ \frac{P \vdash Q[e/x] \quad \{P,Q\} \subseteq \mathcal{R}}{\mathcal{R};\{\{P\}x := e\} \Vdash \{P\}\,x := e\,\{Q\}}$$

$$\text{(SEQ)}\ \frac{\mathcal{R}_1;\mathcal{G}_1 \Vdash \{P\}\,c_1\,\{R\} \quad \mathcal{R}_2;\mathcal{G}_2 \Vdash \{R\}\,c_2\,\{Q\}}{\mathcal{R}_1 \cup \mathcal{R}_2;\mathcal{G}_1 \cup \mathcal{G}_2 \Vdash \{P\}\,c_1;c_2\,\{Q\}} \qquad \text{(PAR)}\ \frac{\mathcal{R}_1;\mathcal{G}_1 \Vdash \{P_1\}\,c_1\,\{Q_1\} \quad \mathcal{R}_2;\mathcal{G}_2 \Vdash \{P_2\}\,c_2\,\{Q_2\} \quad \mathcal{R}_1;\mathcal{G}_1 \text{ and } \mathcal{R}_2;\mathcal{G}_2 \text{ are non-interfering}}{\mathcal{R}_1 \cup \mathcal{R}_2;\mathcal{G}_1 \cup \mathcal{G}_2 \Vdash \{P_1 \wedge P_2\}\,c_1 \parallel c_2\,\{Q_1 \wedge Q_2\}}$$

$$\text{(ITE)}\ \frac{P \in \mathcal{R} \quad \mathcal{R};\mathcal{G} \Vdash \{P \wedge (e \neq 0)\}\,c_1\,\{Q\} \quad \mathcal{R};\mathcal{G} \Vdash \{P \wedge (e = 0)\}\,c_2\,\{Q\}}{\mathcal{R};\mathcal{G} \Vdash \{P\}\,\texttt{if}\ e\ \texttt{then}\ c_1\ \texttt{else}\ c_2\,\{Q\}} \qquad \text{(WHILE)}\ \frac{\{P,Q\} \subseteq \mathcal{R} \quad P \wedge (e = 0) \vdash Q \quad \mathcal{R};\mathcal{G} \Vdash \{P \wedge (e \neq 0)\}\,c\,\{P\}}{\mathcal{R};\mathcal{G} \Vdash \{P\}\,\texttt{while}\ e\ \texttt{do}\ c\,\{Q\}}$$

Fig. 12. Rely/guarantee presentation of Owicki-Gries proof system.

E Appendix: Invariant Method

A particular simple and useful instance of OG reasoning is the *invariant method* [11]. Here, one has a global invariant that holds at any program point and should be stable with any assignment. The assertions in each thread's proof are confined to speak only about locations that are not modified in other threads (so non-interference is trivial). However, even this method is unsound for release-acquire as shown by the following proof that the "store buffering" program cannot return $a = b = 0$.

Let $J \triangleq x \neq 2 \land y \neq 2 \land (y = 1 \to a = y \lor a = 2 \lor b = x \lor b = 2)$ and consider the following proof outline:

$$
\begin{array}{c}
\{x = y = 0 \land a = b = 2\} \\
\{J \land a = 2 \land b = 2\} \\
\begin{array}{l|l}
\{J \land a = 2\} & \{J \land b = 2\} \\
x := 1; & y := 1; \\
\{J \land x = 1\} & \{J \land y = 1\} \\
a := y & b := x \\
\{J \land x = 1 \land a \neq 2\} & \{J \land y = 1 \land b \neq 2\}
\end{array} \\
\{J \land x = y = 1 \land a \neq 2 \land b \neq 2\} \\
\{a = 1 \lor b = 1\}
\end{array}
$$