On the Strength of Owicki-Gries for Resources

Alexander Malkis and Laurent Mauborgne

IMDEA Software Institute

14 October 2011

Abstract. In multithreaded programs data are often separated into lock-protected resources. Properties of those resources are typically verified by modular, Owicki-Gries-like methods. The modularity of the Owicki-Gries method has its price: proving some properties may require manual introduction of auxiliary variables. What properties can be proven without the burden of introducing auxiliary variables? We answer this question in the abstract interpretation framework. On one hand, we reveal a lattice structure of the method and supply a syntax-based abstract transformer that describes the method *exactly*. On the other hand, we bound the loss of precision from *above* and *below* by transition-relation-independent weakly relational closures. On infinitely many programs the closures coincide and describe the precision loss exactly; in general, the bounds are strict. We prove the absence of a general exact closure-based fixpoint characterization of the accuracy of the Owicki-Gries method, both in the collecting semantics and in certain trace semantics.

1 Introduction

The paper will characterize the accuracy of a popular verification method for proving safety properties of concurrent programs that operate on resources.

A program operating on resources is a multithreaded program in which threads communicate via sequentially consistent shared memory and in which all shared variables are partitioned into disjoint resources. Each resource may be *available* or *busy*. To access a variable belonging to a resource, a thread waits until the resource gets available and then starts a critical section for that resource, thus making the resource busy. While staying in the critical section, the thread can read and write the resource data, and no other thread can enter a critical section for the same resource, so no other thread can access the resource data. After the thread finishes accessing the resource, it exits the corresponding critical section, making the resource available.

Simple safety properties of such programs can be proven modularly. A modular proof of a program consists of an annotation per control flow location and an annotation per resource. Roughly, an annotation of a control flow location describes the valuations of the variables that the thread may access at that location. An annotation of a resource describes, roughly, the state of the resource when it is available. The annotations of locations have to be sequentially consistent similarly to the standard Hoare-style assertional logic, but with two changes: when a thread acquires a resource, the proof may assume that the invariant of the acquired resource holds, and on releasing a resource the resource invariant should be reestablished.

Such modularity incurs a loss of precision: for many programs, too strong but valid properties cannot be proven by the method. To prove such properties, the program can be manually augmented with so-called auxiliary variables. A set of variables is called auxiliary if, intuitively, projecting the traces of the program to the other variables gives the same result as removing all the statements involving the auxiliary variables and projecting afterwards. A modular proof is created for the augmented program, then the proven property is projected onto the original variables. The ability to use auxiliary variables makes the proof system complete. So far, auxiliary variables have been introduced purely manually, while the construction of the remaining proof can be automated.

We will estimate the loss of precision inherent to the *core* of the Owicki-Gries method, which consists of all the proof rules of the Owicki-Gries logic except the ability to use auxiliary variables, in abstract interpretation.

We will show a rich lattice structure of the Owicki-Gries-core proofs and a syntax-based abstract transformer describing the Owicki-Gries core *exactly*.

We will observe that there is no transition-relation-independent approximation operator that exactly characterizes the loss of precision of the Owicki-Gries core in the collecting semantics. We will also note the absence of equivalent trace-based characterizations of certain kinds.

We will find an approximation operator that induces a useful upper bound on this set, i.e., that will describe how much precision the Owicki-Gries core always loses. This approximation operator depends only on the syntactic structure of the program, but not on its exact transition relation. If a property is provable by the Owicki-Gries core, it is provable by abstract interpretation with this approximation.

We will present an approximation operator inducing a lower bound on the set, i.e., describing how much precision the Owicki-Gries core always retains. This approximation operator also depends only on the syntactic structure of the program, but not on its exact transition relation. If a property is provable by abstract interpretation with this approximation, it is provable by the Owicki-Gries core.

In passing, we will demonstrate an infinite class of simple programs for which both bounds are equal and a program on which both bounds are strict.

In short, the main results and their position are depicted in Fig. 1.

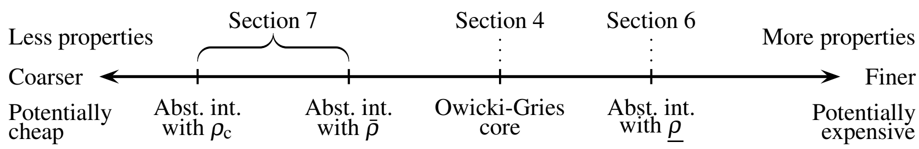

Fig. 1: Precision of analyses. The upper bound is $\underline{\rho}$, the lower bound is $\bar{\rho}$. In addition to $\bar{\rho}$, we will consider a simpler lower bound ρ_c, which is coarser than $\bar{\rho}$, but easier to understand.

The rigorous formalization, computations, proofs, recommendations for practitioners, and comparison to abstract interpretations for general programs are found in the appendix.

2 Resource-manipulating language and its semantics

2.1 Language RPL

Now we briefly recall the parallel language RPL (Restricted Parallel Language, rigorously defined in [25]) in which shared data are separated into resources and access to a resource is granted exclusively.

Let threads be indexed by the elements of the set *Tid* and, without loss of generality, start their executions together. Threads operate on data variables from the set *DataVar*. A *resource* is a set of data variables; the resources are disjoint. Let *Res* be the set of all resources.

A *statement* is either

- an atomic statement, i.e., an assignment or an "`assume` ϕ" (which skips if the argument expression evaluates to *true* and blocks otherwise),
- a sequential composition of two statements, an `if-then-else`, or a `while` loop,
- a critical section `with` r `when` ϕ `do` C `endwith` where r is a resource, ϕ a formula over data variables and C some statement. We write `with` r `do` C `endwith` when ϕ is `true`.

A thread executes a statement.

A data variable is called *local* to thread t if all the assignments to the variable are syntactically in t. If a variable appears in thread t, it should belong to a resource or be local to t. If a variable belongs to a resource r, it can appear only inside a critical section for r, i.e., inside a "`with` r`...`" statement.

Each resource r is associated with a set of *proof variables* $\mathit{ProofVar}(r)$, which are all data variables that are not assigned except maybe in critical sections for r. All our examples will satisfy $\mathit{ProofVar}(r) = r$ (in general, $\mathit{ProofVar}(r) \supseteq r$).

An RPL program is described by a set of threads, a set of resources and an initial condition on the variables.

2.2 Semantics of RPL

To connect to abstract interpretation, we describe programs in RPL as transition systems.

For each thread t we introduce a fresh non-data program counter variable pc_t, which

- is local to thread t, but not to any other thread, and
- doesn't belong to any resource, and
- takes values from the set of control flow locations *PL* (Program Location).

Let *PCVar* be the set of all program counter variables and $\mathit{Var} = \mathit{DataVar} \dot\cup \mathit{PCVar}$.

We associate each control flow point $p \in PL$ with a set $rsc(p)$ of resources r such that p is inside a `with` r`...` statement.

We allow the same control flow location, say, p, to occur in different threads.

A state of a program is a map from *Var* to the set of values *Val* that includes *PL* and the ranges of all data variables. We are going to speak about *consistent states* only: those are states that

- map program counters to elements of *PL* and
- satisfy mutual exclusion: the sets of resources held by different threads are disjoint.

A thread t *holds* a resource r in a consistent state v if $r \in rsc(v(pc_t))$. A resource r is *busy* in a consistent state v if some thread holds it in v, otherwise r is *available* in v.

Each statement of thread t induces a set of transitions $v \to_t v'$ where v, v' are consistent states. The transitions induced by the non-`with` statements are straightforward. The statement `with` r `when` ϕ `do` C blocks when r is busy or when ϕ does not hold, otherwise it changes the control flow location to the initial control flow location of C; going out of the critical section is an unconditional change of the control flow location.

Let *init* be the set of initial states corresponding to the initial condition and let the successor map

$$post(S) \;=\; \{v' \mid \exists\, v \in S,\, t \in Tid\colon\, v \to_t v'\}$$

return the set of all one-step successors of a set of consistent states.

The *syntactic structure* of an RPL program is given by: thread identifiers, control flow locations, values, resources, locals and program counter variables of the threads, proof variables, map *rsc*. A syntactic structure is just a tuple of basic mathematical objects (sets, variables, maps). It also exists independently of an RPL program.

3 Owicki-Gries-core proofs

3.1 Lattice of Owicki-Gries-core proofs

Now we formalize the core of the Owicki-Gries proof system. We abstract away from the details of logic, handling sets of variable valuations instead of formulas.

Fix an arbitrary syntactic structure.

A *program annotation* (I, M) consists of

- a resource invariant I_r for each resource r, which is a set of valuations of proof variables of r,
- a control flow annotation $M_{p,t}$ for each control flow location p and each thread t, containing valuations of data variables which are local to t or which are proof variables of a resource held at p.

We say that a consistent state v satisfies I_r (resp. $M_{p,t}$), if the projection of v to $ProofVar(r)$ (resp. to local data variables of t and all proof variables of all resources in $rsc(p)$) is in I_r (resp. in $M_{p,t}$).

A program annotation (I, M) *denotes* the set of all consistent states v such that

- for each resource r which is available in v, the state v satisfies I_r, and
- for each thread t, the state v satisfies $M_{v(pc_t),t}$.

We define $\gamma_{\mathrm{OG}}(I, M)$ as the set of all consistent states denoted by (I, M).

Now fix an arbitrary RPL program obeying the given syntactic structure.

An *Owicki-Gries-core proof* of the program is a program annotation that satisfies the following conditions.

– It is *sequentially consistent*, which means the following: consider any transition $v \to_t v'$ from a control flow location p to a control flow location p' such that v satisfies $M_{p,t}$. There are three cases.
  - The transition does not cross the border of a critical section. Then v' should satisfy $M_{p',t}$.
  - The transition starts a critical section of a resource r. Additionally, assume that v satisfies I_r. Then v' should satisfy $M_{p',t}$.
  - The transition finishes a critical section of a resource r. Then v' should satisfy both $M_{p',t}$ and I_r.

– It admits the initial states, i.e., denotes a superset of *init*.

These Hoare-style rules treat critical sections specially. First, on entering a critical section, we additionally assume the resource invariant. Second, on leaving a critical section, we should prove the resource invariant.

Every Owicki-Gries-core proof denotes an inductive invariant. A set of consistent states S is *Owicki-Gries-core provable* if there is an Owicki-Gries-core proof that denotes a subset of S.

Interestingly, the set of program annotations of a fixed program forms a complete lattice with respect to componentwise inclusion order. Restricting this order to the set of Owicki-Gries-core proofs gives a complete lattice of Owicki-Gries-core proofs, even a Moore family. So each program has a unique smallest Owicki-Gries-core proof. This proof denotes the strongest Owicki-Gries-core-provable property. To investigate the power of the method, we will look at the smallest Owicki-Gries-core proofs and strongest Owicki-Gries-core-provable properties.

3.2 Examples

Now we will look at examples of programs with their smallest Owicki-Gries-core proofs. On Readers-Writers, as well as for a whole class of similarly simple programs, all proof methods will have the same precision, on Upper all proof methods will have different precision and on the class SepThreads no precision loss will be observed at all.

We'll use Owicki-style notation. In particular, "`resource` $control(ww, ar, aw)$" means that the resource named *control* contains exactly *ww*, *ar* and *aw*.

Example 1 (Readers-Writers). A number of threads share a file simultaneously: $n > 1$ need reading access and $m > 1$ writing access. Any number of readers may access the file simultaneously, but a writer must have exclusive access. In Fig. 2, all the data variables range over nonnegative integers by default, subtracting a positive value from 0 blocks.

```
initially ww = ar = aw = 0
resource  control(ww, ar, aw)

reader_1  ||  ...  ||  reader_n  ||  writer_1  ||  ...  ||  writer_m
```

```
// reader_i:
                  while true do
startreadA:         {true}
                    with control
                    when ww = 0 do
startreadB:           {ww = 0 ∧ aw ≤ 1}
                      ar := ar + 1
startreadC:           {ww = 0 ∧ aw ≤ 1 ≤ ar}
                    endwith;
read:               {true}
                    ;
finishreadA:        {true}
                    with control do
finishreadB:          {aw ≤ 1}
                      ar := ar − 1
finishreadC:          {aw ≤ 1}
                    endwith
                  endwhile
```

```
// writer_j:
                  while true do
askwriteA:          {true}
                    with control do
askwriteB:            {aw ≤ 1}
                      ww := ww + 1
askwriteC:            {aw ≤ 1 ≤ ww}
                    endwith;
startwriteA:        {true}
                    with control
                    when ar = aw = 0 do
startwriteB:          {ar = aw = 0}
                      aw := aw + 1
startwriteC:          {ar = 0 ∧ aw = 1}
                    endwith;
write:              {true}
                    ;
finishwriteA:       {true}
                    with control do
finishwriteB:         {aw ≤ 1}
                      (aw, ww) := (aw − 1, ww − 1)
finishwriteC:         {aw = 0}
                    endwith
                  endwhile
```

$$I_{control} \;=\; \{aw \leq 1\}$$

Fig. 2: Program Readers-Writers and its smallest Owicki-Gries-core proof.

The strongest Owicki-Gries-core-provable property is

$$
\begin{aligned}
\{v \in \mathit{ConsState} \mid\ & v(ww), v(ar) \in \mathbb{N}_0 \ \wedge\ v(aw) \in \{0,1\} \\
& \wedge \forall\, i \in \{1,\ldots,n\}: \\
& \qquad (v(pc_{\texttt{reader}_i}) \in \{x\texttt{read}y, \texttt{read} \mid x \in \{\texttt{start},\texttt{finish}\} \\
& \qquad\quad \wedge\ y \in \{\texttt{A},\texttt{B},\texttt{C}\}\}) \\
& \qquad \wedge (v(pc_{\texttt{reader}_i}) = \texttt{startreadB} \Rightarrow v(ww) = 0) \\
& \qquad \wedge (v(pc_{\texttt{reader}_i}) = \texttt{startreadC} \Rightarrow v(ww) = 0 < v(ar)) \\
& \wedge \forall\, j \in \{1,\ldots,m\}: \\
& \qquad (v(pc_{\texttt{writer}_j}) \in \{x\texttt{write}y, \texttt{write} \mid x \in \{\texttt{ask},\texttt{start},\texttt{finish}\} \\
& \qquad\quad \wedge\ y \in \{\texttt{A},\texttt{B},\texttt{C}\}\}) \\
& \qquad \wedge (v(pc_{\texttt{writer}_j}) = \texttt{askwriteC} \Rightarrow v(ww) > 0) \\
& \qquad \wedge (v(pc_{\texttt{writer}_j}) = \texttt{startwriteB} \Rightarrow v(ar) = v(aw) = 0) \\
& \qquad \wedge (v(pc_{\texttt{writer}_j}) = \texttt{startwriteC} \Rightarrow v(ar) = 0 < v(aw)) \\
& \qquad \wedge (v(pc_{\texttt{writer}_j}) = \texttt{finishwriteC} \Rightarrow v(aw) = 0)\,\}\,.
\end{aligned}
$$

There are states in this set in which more than one writer is at `write`. Thus no Owicki-Gries-core proof can show that a writer has exclusive access. The smallest Owicki-Gries-core proof can prove a slightly weaker property, namely that the value of *aw*, which tracks the number of writers, never exceeds 1.

Example 2 (progUpper). The program Upper in Fig. 3 will be used for showing differences between the Owicki-Gries-core proofs and the upper bound later. The range of the data variables is $\{0,1\}$. The computation is trivial: no thread can make a step. The majority of the program text serves to create a particular distribution of variables into locals and resources. There are exactly two reachable states, namely the

initially $u \neq z = x = y = l$ resource $r(u,z)$, $r'(x,y)$	
// Thread 1	// Thread 2
A: $\{l = x\}$	O: $\{y = z\}$
with r when $l = u$ do	assume *false*;
B: $\{l = x = u \neq z\}$	P: with r' do
with r' do	Q: $y := 0$
C: $\{y \geq l = x = u \neq z\}$	R: endwith;
$x := 0$	S: with r do
D: $\{y \geq l = u \neq z \wedge x = 0\}$	T: $u := 0$;
endwith;	U: $z := 0$
E: $\{l = u \neq z \wedge x = 0\}$	V: endwith;
assume *false*	W:
F: endwith;	
G: with r' do	
H: $x := 0$	
I: endwith;	
J: with r do	
K: $u := 0$	
L: endwith;	
M: $l := 0$	
N:	
$I_r = \{u \neq z\}$, $I_{r'} = \{x \leq y\}$	

Fig. 3: Program Upper and its smallest Owicki-Gries-core proof. Control flow locations following "`assume` *false*" are annotated by $\{false\}$ by default.

initial ones. The smallest Owicki-Gries-core proof denotes many more states, e.g., $[u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 1, l \mapsto 0, pc_1 \mapsto \mathtt{A}, pc_2 \mapsto \mathtt{O}]$.

Example 3 (Simple). A program belongs to the class *Simple* if it has at most one resource, and if the resource exists, then it contains no local variables and all its proof variables belong to the resource. Readers-Writers is a family of programs from Simple.

Example 4 (SepThreads). The class *SepThreads* (Separate Threads) consists of all programs that have no resources and whose initial states are exactly those that satisfy the initial conditions of all the threads (a proper subset of such states is disallowed). For such programs the smallest Owicki-Gries-core proof denotes the set of reachable states.

4 Owicki-Gries core as abstract interpretation

4.1 Owicki-Gries-core proofs are the post-fixpoints of a sound abstract successor map

Our first step to try to give a measure of the precision of the Owicki-Gries core will be to cast it in the abstract interpretation framework. An Owicki-Gries-core proof is clearly an abstraction of the set of reachable states of a program. So we will give a characterization of Owicki-Gries core as a post-fixpoint of a sound abstraction of the successor map *post* of an RPL program. The map mimics the application of the Owicki-Gries-core proof conditions.

For a program annotation (I,M), we define the *sound Owicki-Gries-core abstract successor map* $post^{\#}_{\mathrm{OG}}(I,M) = (I',M')$, which applies the sequential consistency once:

- I'_r is the smallest superset of I_r that contains all valuations w of proof variables of r such that there is a transition $v \to_t v'$ (for some thread t) that exits a critical section for r, v satisfies $M_{v(pc_t),t}$ and w equals the projection of v' on the proof variables of r;
- $M'_{p',t}$ is the smallest superset of $M_{p',t}$ that contains all valuations w of local data variables of thread t and proof variables of resources held at location p' such that there is a transition $v \to_t v'$ such that v satisfies $M_{v(pc_t),t}$, v satisfies I_r if the transition starts a critical section for r, $v'(pc_t)=p'$, and w equals the projection of v' to local data variables of t and to the proof variables of resources held at p'.

This operator is, as the name says, sound with respect to the successor map.
Let (I^{init},M^{init}) be the annotation describing the initial states only:

- I^{init}_r is the set of all valuations of proof variables of r in the initial states;
- $M^{init}_{p,t}$ is the set of valuations of local data variables of thread t in the initial states.

The *Owicki-Gries-core abstract transformer* $F^{\#}_{\mathrm{OG}}(I,M)$ is constructed as the pointwise union of (I^{init},M^{init}) and $post^{\#}_{\mathrm{OG}}(I,M)$. The set of post-fixpoints of $F^{\#}_{\mathrm{OG}}$ coincides with the set of Owicki-Gries-core proofs; thus the following theorem holds.

Theorem 5 (Equivalence). *The least fixpoint of $F^{\#}_{\mathrm{OG}}$ is the smallest Owicki-Gries-core proof.*

4.2 Characteristic closures for Owicki-Gries core?

An elegant way of describing the precision of an approximation of a semantics given in fixpoint form is through the use of *closures*. A closure ρ is a monotone operator that is idempotent ($\rho(\rho(x)) = \rho(x)$) and extensive ($\rho(x) \geq x$). Given a concrete domain $(D,\leq)$ and a monotone function $F : D \to D$, the closure ρ on D defines an approximation of the concrete semantics $lfp(F)$ in the sense that $lfp(F) \leq lfp(\rho \circ F)$. Such a description shows the actual loss of information, as the fixpoints of the closure are exactly the abstract elements that describe the approximation, and applying the closure to one concrete element exactly shows what information is lost for that element.

In our case, we have a concrete domain of sets of consistent states ordered by inclusion, and the semantics is given as the least fixpoint of the successor map *post* over the initial states. Then we exhibited an approximation $post^{\#}_{\mathrm{OG}}$ of that successor map. In order to describe its precision, it would be nice to find a closure ρ such that $post^{\#}_{\mathrm{OG}}$ is exactly the best transformer for $\rho \circ post$. Finding one closure for each program is not difficult (just take the closure with two fixpoints, one being the strongest Owicki-Gries-core-provable property and the other the set of all consistent states), but this would not be very informative. Instead, because the concrete domain is entirely fixed by the syntactic structure of the program, we would like to find a ρ that would be fixed for a given syntactic structure. Alas, the next section shows that it is not possible in general.

5 Absence of equivalent characterizations by closures

The main result of this section is unfortunately negative: assuming we start from a reachable state semantics, there is no way to describe the strongest Owicki-Gries-core proof for a given syntactic structure using closures.

Theorem 6 (No Equivalence). *There is a syntactic structure such that for consistent states defined through the syntactic structure and the concrete domain being the powerset of consistent states, there is no closure ρ on the powerset of consistent states such that for any multithreaded program having the given syntactic structure and for any property S of consistent states we have*

$$S \text{ is Owicki-Gries-core provable} \quad \Leftrightarrow \quad lfp(\lambda x.\, \rho(init \cup post(x))) \subseteq S.$$

One such syntactic structure is given by program Upper from Example 2.

In fact, we can prove an even stronger property, as the proof requires only that ρ is monotone. Even more, the same proof holds even if we restrict the validity of the equivalence to a given transition relation:

Theorem 7. *Under the same syntactic structure as in Thm. 6, there is no family of monotone maps Ξ indexed by transition relations, such that Ξ preserves least fixpoints ($\forall$ transition relations τ_1, τ_2: $lfp(\tau_1) = lfp(\tau_2) \Rightarrow lfp(\Xi_{\tau_1}) = lfp(\Xi_{\tau_2})$), and for any property S of consistent states we have*

$$S \text{ is Owicki-Gries-core provable} \quad \Leftrightarrow \quad lfp(\Xi_{\lambda x.init \cup post(x)}) \subseteq S.$$

Theorem 6 shows that if we start by approximating traces by states and wish to describe the Owicki-Gries-core proof system using closures, we can only hope for bounds framing the proof system. We will provide such bounds in the next two sections. If we are willing to work with closures on sets of traces instead of sets of states, it might be possible to find some equivalence. But such an equivalence cannot be obtained directly by collecting the states of traces obtained from abstract interpretation with a closure. Let $\widetilde{\alpha}$ be the abstraction which associates to a set of traces the set of consistent states appearing in the traces.

Theorem 8. *Under the same syntactic structure as in Thm. 6, there is no monotone operator $\widetilde{\rho}$ on the powerset of traces of consistent states such that for any multithreaded program having the given syntactic structure, set of initial traces $\widetilde{init}$ and the trace extension operator $\widetilde{post}$ and for any property S of consistent states we have*

$$S \text{ is Owicki-Gries-core provable} \quad \Leftrightarrow \quad \widetilde{\alpha}(\mathit{lfp}(\lambda x.\,\widetilde{\rho}(\widetilde{init} \cup \widetilde{post}(x)))) \subseteq S.$$

6 Upper bound on precision

Now we will show a closure operator such that the best abstract interpretation of the program with this approximation allows proving a larger set of properties than those provable by the Owicki-Gries core. The approximation will depend only on the syntactic structure, but not on the exact transition relation of a program.

Definition 9 (Owicki-Gries-core annotation closure). *For a given syntactic structure, let $\underline{\rho}(Q)$ be the approximation defined as the set of consistent states v such that:*
- *if a resource r is available in v, then there is some other state in Q*
  - *that coincides with v on the proof variables of r and*
  - *in which r is available;*
- *and for any thread t there is a state in Q that coincides with v on local variables (including the program counter) of t and on the proof variables of the resources held by t in v.*

Then $\underline{\rho}$ is a closure on the powerset of consistent states. We call it the Owicki-Gries-core annotation closure.

The reason why we call this closure an Owicki-Gries-core annotation closure is because it is the closure corresponding to the Galois connection defined by γ_{OG} (which gives the set of consistent states denoted by a program annotation).

Now fix an arbitrary program and let $\underline{\rho}$ be defined by its syntactic structure.

Theorem 10 (Upper bound). *Abstract interpretation with $\underline{\rho}$ is at least as strong as the Owicki-Gries core. Formally:*

$$\mathit{lfp}(\lambda x.\,\underline{\rho}(init \cup post(x))) \subseteq \text{the strongest Owicki-Gries-core-provable property}.$$

From the high-level view, the best transformer using this closure is capable of taking into account *global* computation instead of *local* successor computation in the Owicki-Gries core. It is as if before checking sequential consistency we take into account annotations not only of one thread, but of all the threads, gaining precision.

Furthermore, the Owicki-Gries-core annotation closure shows where the information is always lost. For instance, if a syntactic structure says that locals are disjoint among themselves and from all the proof variables of the resources, and if two states outside of critical sections are given, then any combination of the locals and resource variables of those states is in the approximation of those two states.

Example 11 (Readers-Writers). For the program Readers-Writers from Example 1 the least fixpoint of $lfp(\lambda x.\, \underline{\rho}(init \cup post(x)))$ coincides with the strongest Owicki-Gries-core-provable property. This property is coarser than the set of reachable states. For example, in one execution $\mathtt{writer}_1$ can reach `write`, in another execution $\mathtt{writer}_2$ can reach `write`, and the initial state satisfies $ww = ar = aw = 0$, so $\underline{\rho}$ produces a combined state where both writers are at `write`, other threads are at their initial locations and all data variables have value 0. Thus, no Owicki-Gries-core proof can restore the dependency between the threads and prove mutual exclusion between the writers.

Example 12 (Upper). For the program Upper from Example 2 abstract interpretation with $\underline{\rho}$ produces the set of reachable states, which is properly included into every Owicki-Gries-core-provable property. The reason for this discrepancy is that locals of one thread and resources held by a different thread overlap. Such an overlap constrains the output of $\underline{\rho}$, but not an Owicki-Gries-core proof. Considering such overlaps actually improves precision!

Example 13 (Simple). For the programs of the class Simple abstract interpretation with $\underline{\rho}$ produces the same result as the smallest Owicki-Gries-core proof. The reason, as we will see, is that abstract interpretation with the lower bound will produce the same result as abstract interpretation with the upper bound.

Example 14 (SepThreads). For the programs of the class SepThreads of Example 4 abstract interpretation with $\underline{\rho}$ produces the same result as the smallest Owicki-Gries-core proof. The reason is that the smallest Owicki-Gries-core proof already denotes the set of reachable states.

7 Lower bound on precision

Now we will show a nontrivial Cartesian-like closure such that abstract interpretation with this closure can prove only properties weaker than or equal to the strongest Owicki-Gries-core-provable property. In fact, we will even show two such closures. One closure (namely $\bar{\rho}$) will describe a better lower bound, while the other one (namely ρ_c) is easier to comprehend.

The definitions of both bounds require some preparation.

For a family of sets $\mathscr{X} = \{X_1, \ldots, X_n\}$, let $Part(\mathscr{X})$, called *partition* of $\mathscr{X}$, be the set of all nonempty $Y_1 \cap \ldots \cap Y_n$ where each Y_i is either X_i or its complement $(\bigcup \mathscr{X}) \setminus X_i$ ($1 \leq i \leq n$). Elements of a partition are called *blocks*.

Let us fix an arbitrary syntactic structure. Our lower bounds will depend on the syntactic structure but be the same for all the programs that obey this syntactic structure. Let RL be the family of sets containing all resources r and all sets of locals $Local_t$ for all threads t as elements. Let $\widetilde{RL}$ be the partition of RL.

For example, for the syntactic structure of Upper the set RL has exactly four elements: the locals of the first thread $\{l, x, pc_1\}$, the locals of the second thread $\{y, z, pc_2\}$, the resource $r = \{u, z\}$, and the resource $r' = \{x, y\}$. The corresponding partition has exactly six blocks: $\{u\}$, $\{x\}$, $\{y\}$, $\{z\}$, $\{l, pc_1\}$, $\{pc_2\}$.

7.1 Simple cartesian closure

The simpler approximation is defined as follows.

Definition 15 (Cartesian closure). *Given a set of consistent states Q, the* Cartesian *approximation ρ_c returns all the consistent states from the product of projections of Q onto the blocks in $\widetilde{RL}$.*
In other words, $\rho_c(Q)$ contains exactly those consistent states v such that for each block there is a state $\tilde{v}$ in Q that agrees with v on the variables of the block.

We call this approximation *Cartesian*, since it is similar to the closure that can be derived from the Cartesian pair of adjoint maps from [10], p. 172. As the name says, ρ_c is a closure.

This approximation breaks all the dependencies between the blocks and retains all dependencies inside a block.

For example, for the syntactic structure of the program Upper the variables l and pc_1 belong to the same block. Thus, if in a set of states every state at some fixed control flow location satisfies $l = 0$, each state from the Cartesian approximation of this set will also satisfy $l = 0$ for that control flow location.

Abstract interpretation with ρ_c generates a property which is always weaker than or equal to the strongest Owicki-Gries-core-provable property.

7.2 More precise lower bound closure

The following definition strengthens ρ_c in two ways. Firstly, we impose restrictions on $\tilde{v}$ from the definition of ρ_c. Such restrictions will depend on the block. Secondly, we look at the prophecy variables: those are variables which are never written and which don't belong to a resource. Prophecy variables form a separate block; now we restore all the dependencies between this block and those locals of any thread that are not resource variables.

Definition 16 (Lower Bound closure). *Given a set of consistent states Q, its approximation $\bar{\rho}(Q)$ contains exactly those consistent states v such that both of the following conditions are satisfied.*

- *For every block in $\widetilde{RL}$ that is contained in a resource,*
  - *if the resource is available in v, then there is some $\tilde{v} \in Q$*
    - *in which the resource is also available*
    - *and which coincides with v on the block;*
  - *if some thread holds the resource in v, then there is some $\tilde{v} \in Q$ which coincides with v on all the variables*
    - *that are local to this thread but do not belong to any resource, or*
    - *that belong to the block.*
- *For every thread there is a state in Q that coincides with v on each variable*
  - *that is a local variable of the thread but*
  - *does not belong to any resource.*

As the name says, $\bar{\rho}$ is a closure. It is at least as precise as ρ_c. Both closures do not depend on the exact transition relation of a program. The closure $\bar{\rho}$ induces the tightest lower bound we could prove. It shows the dependencies that are always retained, creating a basis for the construction of future refinement algorithms (possibly following [20]).

Now fix an arbitrary RPL program that has the assumed syntactic structure.

The proof of the lower bound relies on several claims about the strongest property provable by abstract interpretation with $\bar{\rho}$. The following claim is the most important one.

Lemma 17. *Let Q be the strongest property provable by abstract interpretation with $\bar{\rho}$. Consider a transition of a thread t from a consistent state v to a consistent state v', let the transition start a critical section. Let $\tilde{v} \in Q$ such that v agrees with $\tilde{v}$ on the locals of t and on the variables of the resources held by t before the transition. Let $\hat{v} \in Q$ such that v agrees with $\hat{v}$ on the variables of the resource being acquired. Then there is a state in Q that agrees with v' on the locals of thread t and on the variables of the resources held by the thread after the transition.*

The lower bound theorem follows from the lemma.

Theorem 18 (Lower bound). *The core of Owicki-Gries can prove at least as many properties as abstract interpretation with $\bar{\rho}$ or ρ_c. Formally:*
the strongest Owicki-Gries-core–provable property $\subseteq$ $\mathit{lfp}(\lambda x.\, \bar{\rho}(\mathit{init} \cup \mathit{post}(x)))$
$\subseteq$ $\mathit{lfp}(\lambda x.\, \rho_c(\mathit{init} \cup \mathit{post}(x)))$.

Example 19 (Readers-Writers). For the program Readers-Writers from Example 1 abstract interpretation with ρ_c produces the set of all consistent states where readers are at ...`read`..., writers are at ...`write`..., and *ww*, *ar*, *aw* are nonnegative. The Owicki-Gries core and abstract interpretation with $\bar{\rho}$ can prove stronger properties, e.g., that at location `askwriteC` the value of *ww* is positive. Intuitively, when a resource is busy, Cartesian abstraction always breaks the dependency between the resource variables and the control flow, while $\bar{\rho}$ and the Owicki-Gries core sometimes retain the dependency.

Example 20 (Upper). For the program Upper from Example 2 abstract interpretation using either ρ_c or $\bar{\rho}$ produces the same result: the set of all states such that the first thread is at any location between `A` and `E`, the second thread is at `O` and all data variables take arbitrary values from $\{0,1\}$. The Owicki-Gries core can prove stronger properties; for instance, it can show that at location `E` the value of x is zero. Intuitively, the dependency between resource variables and control flow is always broken in Cartesian abstraction but is sometimes retained in the Owicki-Gries core.

Example 21 (Simple). For the programs from the class Simple from Example 3 the approximations $\bar{\rho}$ and $\underline{\rho}$ are so close to each other that abstract interpretations with both produce the same property. Since they define the lower and upper bounds on the precision of the Owicki-Gries core, the Owicki-Gries core can prove exactly the same properties as those provable by abstract interpretation with any of the two approximations. If a program from Simple does not have the empty resource, then $\bar{\rho}$ and $\underline{\rho}$ coincide exactly, approximating a set of states Q by the set of all consistent states v such that both of the following conditions hold.

– If there is a resource and it is available in v, there is some state in Q that coincides with v on the resource and in which the resource is available.
– For each thread there is a state in Q which coincides with v on the variables of the thread and, if the resource is present and is held, on the variables of the resource.

Example 22 (SepThreads). For the programs from the class SepThreads from Example 4 the approximations $\bar{\rho}$ and $\underline{\rho}$ coincide. Since they define the lower and upper bounds on the precision of the Owicki-Gries core, the Owicki-Gries core can prove exactly the same properties as those provable by abstract interpretation with any of the two approximations. Due to the absence of any thread interactions and independence of initial states of the threads, the mentioned methods can prove the strongest inductive property, namely the set of states reachable from the initial ones. Informally spoken, all dependencies between the locals of each thread are retained.

8 Related work

Historically, conditional critical regions were introduced in [16]. The thesis [25] of Owicki and her paper with Gries [26] describe the original proof method for RPL. Modular reasoning about RPL has not been characterized in terms of abstract interpretation via closures [8] so far.

For general multithreaded programs (i.e. without separation of data into resources), Owicki-Gries-style reasoning without auxiliary variables is equivalent to multithreaded Cartesian abstraction. The result was first mentioned without proof in [11], and the proof appears in [20].

Clarke [6] has considered a subset of integer RPL programs with only one resource, where, roughly, only additions of constants inside the critical sections are allowed, and the property to be checked is either mutual exclusion of PV-semaphores or deadlock freedom. With a predefined choice of integer auxiliary variables, the least fixpoint of a particular functional is a resource invariant that precisely tells whether the property holds or not. Two overapproximations are given: a fixed-formula resource invariant and an invariant computed by a polyhedral analysis with widening. Our work, on the contrary, does not impose any restrictions on the program form. Our results hold for even more programs than the RPL ones, e.g., where the critical sections are not well-nested.

The work of Owicki on RPL is the basis of a variety of modular programming languages equipped with proof methods of different degrees of completeness and automation.

Concurrent separation logic (CSL) [24] equips RPL with separation logic as a formula language. CSL is also incomplete without the rule of auxiliary variables, so the question of precision arises. Removing secondly important features of CSL for the sake of clarity (as in [5]) and considering variables in the heap makes our lower bound also apply to such CSL versions.

Chalice [19] is a language for verification of object-oriented concurrent programs with heap, equipped with an RPL-like proof system. Due to the powerful permission system, the proof system is in general stronger than that of Owicki, so our lower bound on precision carries over to Chalice for programs that can be directly represented both in RPL and in Chalice.

VCC [23] is a verifier for multithreaded C. When accessing structures in a lock-based manner, VCC requires the user to provide invariants of C `struct`s. On obtaining ownership of a `struct`ure, the resource invariant is assumed; on relinquishing ownership, the resource invariant has to be reestablished. Ghost contracts of lock-manipulating functions control the ownership transfer. Our lower bound applies to VCC as well.

9 Discussion

9.1 Challenges

Discussing the precision loss reveals several open problems.

Example 20 shows a gap between the accuracy of the Owicki-Gries core and the lower bound. Can the lower bound closure be strengthened?

The main inequivalence result assumes that the concrete domain is the powerset of consistent states. For the powerset of traces, we only know inequivalence for a subclass of abstract interpretations. Is there an exact characterization of the Owicki-Gries core by abstract interpretation on the powerset of traces?

We have shown what variable dependencies does the Owicki-Gries core break. Can these dependencies be restored on demand? Is there an automatic counterexample-guided abstraction refinement of the Owicki-Gries core, perhaps based on auxiliary variables [7], unions of Cartesian products [20], or abstract threads [18]?

Can one characterize the precision loss of the Owicki-Gries core by completeness notions of [15]?

Can one formalize CSL in abstract interpretation in a way that would reveal the involved approximation?

9.2 Conclusion

We have examined a modular method (Owicki-Gries core) for proving safety properties of a widely-used class of multithreaded programs.

The considered class contains structured programs in which shared data are partitioned into resources and are accessed only in critical sections that ensure mutually exclusive access to resources. The method provides a clean basis for other more sophisticated proof methods like Concurrent Separation Logic, Chalice, or VCC. The Owicki-Gries core is polynomial in the number of threads, but without manually adding auxiliary variables it cannot prove many properties of concurrent programs.

The Owicki-Gries core is, intuitively, expected to succeed for properties whose dependence on thread coupling is low, and is expected to fail if complicated thread interactions have to be analyzed. We have made this notion precise, providing a characterization of the set of Owicki-Gries-core-provable properties. We have demonstrated an abstract transformer corresponding to the Owicki-Gries core: the least fixpoint of the abstract transformer denotes exactly the strongest Owicki-Gries-core-provable property.

To quantify the loss of precision inherent to modularity, we have provided a superset and a subset of Owicki-Gries-core-provable properties, described by abstract interpretations with closure operators that depend on the syntactic structure of the program only. These bounds coincide for a class of simple programs. We have also shown a principal inability to provide an exact characterization of the set of properties in terms of closures that depend only on the syntactic structure.

10 Acknowledgements

We thank Byron Cook and Microsoft Research Cambridge for support in the initial stages of the work. The work was also partially supported by the Verisoft project of the German science foundation DFG. We thank Josh Berdine and Viktor Vafeiadis for helpful comments and discussions.

References

1. G. Barthe and M. V. Hermenegildo, editors. *Verification, Model Checking, and Abstract Interpretation, 11th International Conference, VMCAI 2010, Madrid, Spain, January 17-19, 2010. Proceedings*, volume 5944 of *LNCS*. Springer, 2010.
2. G. Birkhoff. *Lattice Theory*. American Mathematical Society, second edition, 1948.
3. G. Birkhoff. *Lattice Theory*, volume 25. American Mathematical Society, third edition, 1995. Eighth printing, ISBN 0-8218-1025-1.
4. S. N. Burris and H. P. Sankappanavar. *A Course in Universal Algebra*. Springer-Verlag, Millenium edition, 1981. `http://www.math.uwaterloo.ca/~snburris/htdocs/UALG/univ-algebra.pdf`.
5. C. Calcagno, P. W. O'Hearn, and H. Yang. Local action and abstract separation logic. In *LICS*, pages 366–378. IEEE Computer Society, 2007.
6. E. M. Clarke. Synthesis of resource invariants for concurrent programs. *ACM Trans. Program. Lang. Syst.*, 2(3):338–358, 1980.
7. A. Cohen and K. S. Namjoshi. Local proofs for global safety properties. In W. Damm and H. Hermanns, editors, *CAV*, volume 4590 of *LNCS 4590*, pages 55–67. Springer, 2007.
8. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, pages 238–252, 1977.
9. P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *POPL*, pages 269–282, 1979.
10. P. Cousot and R. Cousot. Formal language, grammar and set-constraint-based program analysis by abstract interpretation. In *FPCA*, pages 170–181, 1995.
11. R. Cousot. *Fondements des méthodes de preuve d'invariance et de fatalité de programmes parallèles*. PhD thesis, Institut national polytechnique de Lorraine, 1985.
12. M. Dodds, X. Feng, M. J. Parkinson, and V. Vafeiadis. Deny-guarantee reasoning. In G. Castagna, editor, *ESOP*, volume 5502 of *Lecture Notes in Computer Science*, pages 363–377. Springer, 2009.
13. X. Feng, R. Ferreira, and Z. Shao. On the relationship between concurrent separation logic and assume-guarantee reasoning. In R. D. Nicola, editor, *ESOP*, volume 4421 of *LNCS*, pages 173–188. Springer, 2007.
14. C. Flanagan and S. Qadeer. Thread-modular model checking. In T. Ball and S. K. Rajamani, editors, *SPIN*, volume 2648 of *Lecture Notes in Computer Science*, pages 213–224. Springer, 2003.

15. R. Giacobazzi, F. Ranzato, and F. Scozzari. Making abstract interpretations complete. *Journal of the ACM*, 47(2):361–416, 2000.
16. C. A. R. Hoare. Towards a theory of parallel programming. In C. A. R. Hoare and R. H. Perrott, editors, *Operating System Techniques*, pages 61–71. Academic Press, 1972.
17. C. B. Jones. Tentative steps toward a development method for interfering programs. *ACM Trans. Program. Lang. Syst.*, 5(4):596–619, 1983.
18. S. K. Lahiri, A. Malkis, and S. Qadeer. Abstract threads. In Barthe and Hermenegildo [1], pages 231–246.
19. K. R. M. Leino. Verifying concurrent programs with Chalice. In Barthe and Hermenegildo [1], page 2.
20. A. Malkis. *Cartesian Abstraction and Verification of Multithreaded Programs*. PhD thesis, Albert-Ludwigs-Universität Freiburg, Feb. 2010.
21. A. Malkis, A. Podelski, and A. Rybalchenko. Thread-modular verification and Cartesian abstraction. TV'06, Aug. 2006.
22. A. Malkis, A. Podelski, and A. Rybalchenko. Thread-modular verification is Cartesian abstract interpretation. In K. Barkaoui, A. Cavalcanti, and A. Cerone, editors, *ICTAC*, volume 4281 of *Lecture Notes in Computer Science*, pages 183–197. Springer, 2006.
23. M. Moskal, W. Schulte, E. Cohen, M. A. Hillebrand, and S. Tobies. Verifying C programs: A VCC tutorial, 2011. MSR Redmond, EMIC Aachen.
24. P. W. O'Hearn. Resources, concurrency, and local reasoning. *Theor. Comput. Sci.*, 375(1-3):271–307, 2007.
25. S. S. Owicki. *Axiomatic Proof Techniques For Parallel Programs*. PhD thesis, Cornell University, Department of Computer Science, TR 75-251, July 1975.
26. S. S. Owicki and D. Gries. Verifying properties of parallel programs: an axiomatic approach. *Commun. ACM*, 19(5):279–285, 1976.
27. M. J. Parkinson and A. J. Summers. The relationship between separation logic and implicit dynamic frames. In G. Barthe, editor, *ESOP*, volume 6602 of *Lecture Notes in Computer Science*, pages 439–458. Springer, 2011.
28. A. Tarski. A lattice theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.
29. V. Vafeiadis. Modular fine-grained concurrency verification. Technical Report UCAM-CL-TR-726, University of Cambridge, Computer Laboratory, July 2008.
30. V. Vafeiadis. RGSep action inference. In Barthe and Hermenegildo [1], pages 345–361.
31. M. Ward. The closure operators of a lattice. *The Annals of Mathematics*, 43(2):191–196, Apr. 1942.

A RPL

We formalize RPL (restricted programming language) of [25] as a transition system as follows.

A *syntactic structure* is a tuple

$$(Tid, PL, Val, Res, (Local_t, pc_t)_{t\in Tid}, ProofVar, rsc)$$

where all the following conditions hold.

– Tid, PL, Val are any sets with $PL \subseteq Val$.

– *Res* is a set[1] of disjoint[2] sets.
– For each $t \in Tid$ we have $pc_t \in Local_t \setminus \bigcup Res$.
– For all different $t, \tilde{t}$ in *Tid* we have $pc_t \notin Local_{\tilde{t}}$.
– $ProofVar : Res \to \mathfrak{P}(DataVar)$ is extensive, where $\mathfrak{P}$ is the powerset symbol, $PCVar = \{pc_t \mid t \in Tid\}$ and $DataVar = \bigcup Res \cup \bigcup_{t \in Tid} Local_t \setminus PCVar$.
– $rsc : PL \to \mathfrak{P}(Res)$. Notice that *rsc* is independent[3] of *Tid*.

The elements of $ProofVar(r)$ are called *proof variables*[4] of a resource r ($r \in Res$). Given a syntactic structure as above, let $Var = \bigcup Res \cup \bigcup_{t \in Tid} Local_t$. A *consistent state* is an element of

$$ConsState = \{v \in Var \to Val \mid \forall\, t \in Tid : v(pc_t) \in PL \text{ and } \forall\, \tilde{t} \in Tid \setminus \{t\} : rsc(v(pc_t)) \cap rsc(v(pc_{\tilde{t}})) = \emptyset\}\,.$$

A resource r is *held* by $t \in Tid$ in $v \in ConsState$ if $r \in rsc(v(pc_t))$. A resource is *busy* in $v \in ConsState$ if $r \in \bigcup_{t \in Tid} rsc(v(pc_t))$, otherwise r is *available* in v.

The *restriction* of a map w to a set V is $w|_V = \{(a,b) \in w \mid a \in V\}$. Two maps v, w *coincide* on a set V, written $v \sim_V w$, iff $v|_V = w|_V$.

An *RPL transition system obeying a syntactic structure* as above is a tuple

$$(init, (\to_t)_{t \in Tid})$$

such that $init \subseteq ConsState$, for all $t \in Tid$ we have $\to_t \subseteq ConsState^2 \wedge \forall v \in init\colon\ rsc(v(pc_t)) = \emptyset$, and all the conditions below hold:

– for all $x \in DataVar$ and $t \in Tid$ we have

$$(\forall\, \tilde{t} \in Tid \setminus \{t\}, (v,v') \in \to_{\tilde{t}}\colon\ v(x) = v'(x)) \Rightarrow x \in Local_t\,;$$

– for all $r \in Res$ we have

$$ProofVar(r) = \left\{ x \in DataVar \,\middle|\, \forall t \in Tid, (v,v') \in \to_t\colon \begin{pmatrix} v(x) = v'(x) \quad \text{or} \\ r \in rsc(v(pc_t)) \cap rsc(v'(pc_t)) \end{pmatrix} \right\};$$

[1] In her thesis, Owicki did not explicitly forbid resources that do not protect any variables, though she mentions no examples with such empty resources. We explicitly allow them. However, we introduce a restriction: in our formulation, the empty resource, if it exists, should be unique. We don't consider this restriction important: we have never come across meaningful academic or real-life programs with empty resources.

[2] Following Owicki, we treat the general case, allowing nonempty intersection between any element of *Res* and any $Local_t$ ($t \in Tid$). If we would enforce mutual disjointness between all $r \in Res$ and all $Local_t$ ($t \in Tid$), we would get a tighter lower bound, but just because the proof system would get weaker.

[3] This way of modeling turned out to be simpler than, say, making control flow locations of different threads disjoint, or making *rsc* dependent on the thread.

[4] Owicki described `Proof−var` as a proof-related notion. Since proof variables of a resource capture a syntactic property of the transition relation of a program, we view proof variables as a syntactic notion, despite their name. A lightweight presentation [26] omits the notion of proof variables, obtaining a simpler but significantly weaker proof system.

– for all $t \in Tid$, $(v,v') \in \rightarrow_t$, $R = rsc(v(pc_t))$, $R' = rsc(v'(pc_t))$, $\bar{R} = Res \setminus (R \cup R')$ all the following conditions hold:
  (1) $v \sim_{\bigcup_{\tilde{t} \in Tid \setminus \{t\}} Local_{\tilde{t}}} v'$;
  (2) $v \sim_{\bigcup \bar{R}} v'$
  (3) $R \neq R' \Rightarrow v \sim_{Var \setminus \{pc_t\}} v'$;
  (4) for all $w \in ConsState$, if $v \sim_{(Local_t \setminus \bigcup \bar{R}) \cup \bigcup R'} w$ and $\forall r \in R' \setminus R : r \notin \bigcup rsc(w(PCVar)))$, then there is some w' such that $w \rightarrow_t w'$ and $v' \sim_{(Local_t \setminus \bigcup \bar{R}) \cup \bigcup R'} w'$;
  (5) $|(R \setminus R') \cup (R' \setminus R)| \leq 1$.

Remark 23. Notice that in (4), w' additionally satisfies $w' \in ConsState$ and $(v \sim_{Local_t \cap \bigcup \bar{R}} w \Rightarrow v' \sim_{Local_t \cup \bigcup R'} w')$.

Proof. Follows from $\rightarrow_t \subseteq ConsState^2$ and v' [by (2)] $\sim_{Local_t \cap \bigcup \bar{R}} v$ [by assumption] $\sim_{Local_t \cap \bigcup \bar{R}} w$ [by (2)] $\sim_{Local_t \cap \bigcup \bar{R}} w'$. ■

Fix an RPL transition system as above. A set of states is mapped to a set of their successors by the function

$$post : \mathfrak{P}(ConsState) \rightarrow \mathfrak{P}(ConsState), \quad S \mapsto \{v' \mid \exists\, v \in S, t \in Tid : v \rightarrow_t v'\}.$$

The goal of verification is, given a set of states $safe \subseteq ConsState$, to prove that

$$lfp(\lambda S.\, init \cup post(S)) \subseteq safe.$$

B Owicki-Gries for RPL

Fix a syntactic structure $(Tid, PL, Val, Res, (Local_t, pc_t)_{t \in Tid}, ProofVar, rsc)$ for the whole section B.

B.1 Definition of Owicki-Gries core

For each $t \in Tid$ let $LocalDataVar_t = Local_t \setminus \{pc_t\}$ be the set of *local data variables*.

A *program annotation* is an element of

$$PA = (\prod_{r \in Res} \mathfrak{P}(ProofVar(r) \rightarrow Val)) \times \prod_{\substack{p \in PL, \\ t \in Tid}} \mathfrak{P}((LocalDataVar_t \cup \bigcup ProofVar(rsc(p))) \rightarrow Val).$$

A program annotation P *denotes* a set of states given by

$$\begin{aligned} \gamma_{\mathrm{OG}}(P) = \{v \in ConsState \mid \forall\, r \in Res \setminus \bigcup rsc(v(PCVar)) : v|_{ProofVar(r)} \in I_r \\ \text{and } \forall\, t \in Tid : v|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(v(pc_t)))} \in M_{v(pc_t),t}\}. \end{aligned}$$

For an RPL transition system $T = (init, (\rightarrow_t)_{t \in Tid})$ that obeys the given syntactic structure, an *Owicki-Gries-core proof* of T is a program annotation $P := ((I_r)_{r \in Res}, (M_{p,t})_{p \in PL, t \in Tid})$ such that the following holds:

– sequential consistency: for all $t \in \mathit{Tid}$, $(v,v') \in \to_t$, $p = v(pc_t)$, $p' = v'(pc_t)$, $R = rsc(p)$, $R' = rsc(p')$, if

$$v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(R)} \in M_{p,t} \ \wedge\ \forall\, r \in R' \setminus R : v|_{\mathit{ProofVar}(r)} \in I_r\,,$$

then

$$v'|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(R')} \in M_{p',t} \ \wedge\ \forall\, r \in R \setminus R' : v'|_{\mathit{ProofVar}(r)} \in I_r\,;$$

– initial condition: $\mathit{init} \subseteq \gamma_{\mathrm{OG}}(P)$.

An Owicki-Gries-core proof P *proves* a property $\mathit{safe} \subseteq \mathit{ConsState}$ if $\gamma_{\mathrm{OG}}(P) \subseteq \mathit{safe}$.

B.2 Lattice of Owicki-Gries-core proofs

The set of program annotations is ordered by pointwise inclusion:

$$((I_r)_{r\in \mathit{Res}}, (M_{p,t})_{p\in PL, t\in \mathit{Tid}}) \sqsubseteq ((\hat{I}_r)_{r\in \mathit{Res}}, (\hat{M}_{p,t})_{p\in PL, t\in \mathit{Tid}})$$

iff $\forall\, r \in \mathit{Res} : I_r \subseteq \hat{I}_r$ and $\forall\, p \in PL, t \in \mathit{Tid} : M_{p,t} \subseteq \hat{M}_{p,t}$.

Then the least upper bound is the pointwise union $\bigsqcup$, the greatest lower bound is a pointwise intersection $\bigsqcap$, and the lattice is complete.

Consider the lattice

$$D = \mathfrak{P}(\mathit{ConsState})\,,$$

ordered by inclusion. This lattice is certainly complete.

Proposition 24. *$\gamma_{\mathrm{OG}} : PA \to D$ is a complete meet-morphism.*

Proof. Let A be a set of program annotations and $((J_r)_{r\in \mathit{Res}}, (N_{p,t})_{p\in PL, t\in \mathit{Tid}}) = \bigsqcap A$. We have to show that $\gamma_{\mathrm{OG}}(\bigsqcap A)$ and $\bigcap \gamma_{\mathrm{OG}}(A)$ coincide.

"$\subseteq$": Let $v \in \gamma_{\mathrm{OG}}(\bigsqcap A)$. Then $\forall\, r \in \mathit{Res} \setminus \bigcup rsc(v(\mathit{PCVar})) : v|_{\mathit{ProofVar}(r)} \in J_r$ and $\forall\, t \in \mathit{Tid} : v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(v(pc_t)))} \in N_{v(pc_t),t}$. The definition of $\bigsqcap$ implies that for all $B = ((I_r)_{r\in \mathit{Res}}, (M_{p,t})_{p\in PL, t\in \mathit{Tid}}) \in A$ we have $\forall\, r \in \mathit{Res} \setminus \bigcup rsc(v(\mathit{PCVar})) : v|_{\mathit{ProofVar}(r)} \in I_r$ and $\forall\, t \in \mathit{Tid} \colon v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(v(pc_t)))} \in M_{v(pc_t),t}$, i.e., $v \in \gamma_{\mathrm{OG}}(B)$. Since $B \in A$ was arbitrary, we have $v \in \bigcap \gamma_{\mathrm{OG}}(A)$.

"$\supseteq$": Let $v \in \gamma_{\mathrm{OG}}(B)$ for all $B \in A$. Let $r \in \mathit{Res} \setminus \bigcup rsc(v(\mathit{PCVar}))$. Then for each $B = (I,M) \in A$ the definition of γ_{OG} gives $v|_{\mathit{ProofVar}(r)} \in I_r$. Thus $v|_{\mathit{ProofVar}(r)} \in J_r$. So $\forall r \in \mathit{Res} \setminus \bigcup rsc(v(\mathit{PCVar}))\colon v|_{\mathit{ProofVar}(r)} \in J_r$. Now let $t \in \mathit{Tid}$. For each $B = (I,M) \in A$ the definition of γ_{OG} implies that $v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(v(pc_t)))} \in M_{v(pc_t),t}$; so $N_{v(pc_t),t} \ni v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(v(pc_t)))}$. Thus $\forall t {\in} \mathit{Tid}\colon v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(v(pc_t)))} \in N_{v(pc_t),t}$. We have shown $v \in \gamma_{\mathrm{OG}}(\bigsqcap A)$. ∎

For the remainder of the subsection B.2, fix an RPL transition system $T = (\mathit{init}, (\to_t)_{t\in \mathit{Tid}})$ that obeys the given syntactic structure. Let $\mathfrak{O}$ be the set of Owicki-Gries-core proofs of T. From PA it inherits the componentwise inclusion as the partial order, which we also denote by $\sqsubseteq$.

Theorem 25. $(\mathfrak{O}, \sqsubseteq)$ *is a complete lattice with infimum being componentwise intersection* $\bigsqcap$.

Proof. Let A be a set of Owicki-Gries-core proofs. First we show that $\bigsqcap A$ in PA is the infimum of A in $\mathfrak{O}$.

– Case $A = \emptyset$. Let $B =$ $((\mathit{ProofVar}(r) \to \mathit{Val})_{r \in \mathit{Res}}, ((\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(p))) \to \mathit{Val})_{\substack{p \in PL, \\ t \in \mathit{Tid}}})$. Certainly B is sequentially consistent. Since $\gamma_{\mathrm{OG}}(B) = \mathit{ConsState}$, B satisfies the initial condition. Thus $B \in \mathfrak{O}$. Notice that B is a lower bound of $\emptyset$ and is greater than or equal to any other element of $\mathfrak{O}$. Thus $B = \inf_{\mathfrak{O}} \emptyset$.
– Case $A \neq \emptyset$. Let $B = ((I_r)_{r \in \mathit{Res}}, (M_{p,t})_{\substack{p \in PL, \\ t \in \mathit{Tid}}}) = \bigsqcap A$. First we will show that B is an Owicki-Gries-core proof.
  Sequential consistency. Let $t \in \mathit{Tid}$, $(v, v') \in \to_t$, $p = v(pc_t)$, $p' = v'(pc_t)$, $R = rsc(p)$, $R' = rsc(p')$, $v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(R)} \in M_{p,t}$ and $\forall\, r \in R' \setminus R : v|_{\mathit{ProofVar}(r)} \in I_r$. Then for all $(J, N) \in A$ we have $v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(R)} \in N_{p,t}$ and $\forall\, r \in R' \setminus R$ we have $v|_{\mathit{ProofVar}(r)} \in J_r$. Since A contains Owicki-Gries-core proofs only, we get $v'|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(R')} \in N_{p',t}$ and $\forall\, r \in R \setminus R' : v'|_{\mathit{ProofVar}(r)} \in J_r$ ($(J, N) \in A$). Thus $v'|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(R')} \in M_{p',t}$ and $\forall\, r \in R \setminus R' : v'|_{\mathit{ProofVar}(r)} \in I_r$.
  Initial condition. For each $P = ((J_r)_{r \in \mathit{Res}}, (N_{p,t})_{\substack{p \in PL, \\ t \in \mathit{Tid}}}) \in A$ we have $\mathit{init} \subseteq \gamma_{\mathrm{OG}}(P)$, so for any $v \in \mathit{init}$ we have $\forall\, r \in \mathit{Res} \setminus \bigcup rsc(v(\mathit{PCVar})) : v|_{\mathit{ProofVar}(r)} \in J_r$ and $\forall\, t \in \mathit{Tid} : v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(v(pc_t)))} \in N_{v(pc_t),t}$. Taking componentwise intersection over all P, for any $v \in \mathit{init}$ we have $\forall\, r \in \mathit{Res} \setminus \bigcup rsc(v(\mathit{PCVar}))$: $v|_{\mathit{ProofVar}(r)} \in I_r$ and $\forall\, t \in \mathit{Tid} : v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(v(pc_t)))} \in M_{v(pc_t),t}$. Thus for all $v \in \mathit{init}$ we have $v \in \gamma_{\mathrm{OG}}(B)$, so $\mathit{init} \subseteq \gamma_{\mathrm{OG}}(B)$.
  We have shown that $B \in \mathfrak{O}$. By definition B is less than or equal to any element of A. Assume some other lower bound $C \in \mathfrak{O}$ for A. Then $C \sqsubseteq P$ for each $P \in A$, so C is less than or equal to the pointwise intersection over A, i.e., $C \sqsubseteq B$. Thus B is the greatest lower bound of A in $\mathfrak{O}$.

We have shown that the greatest lower bound of any set of elements of the poset $(\mathfrak{O}, \sqsubseteq)$ exist. By Thm. 2 in Chap. 4 of [2] (alternatively Thm. 4.2 in [4]), $\mathfrak{O}$ is a complete lattice. ∎

Notice that the infimum of $\mathfrak{O}$ is the tuple of empty sets iff *init* is empty; thus $\mathfrak{O}$ is a complete sublattice of *PA* only if *init* is empty. Usually verified programs have initial states, thus usually $\mathfrak{O}$ is not a complete sublattice of *PA*.

In any case the least Owicki-Gries-core proof exists.

Corollary 26. *The strongest Owicki-Gries-core provable property exists and is denoted by the smallest Owicki-Gries-core proof.*

Proof. The set of Owicki-Gries-core-provable properties is $\gamma_{\mathrm{OG}}(\mathfrak{O})$. The infimum of this set is $S \stackrel{def}{=} \bigcap \gamma_{\mathrm{OG}}(\mathfrak{O}) = \gamma_{\mathrm{OG}}(\bigsqcap \mathfrak{O})$. Since $\mathfrak{O}$ is a complete lattice with pointwise intersection as infimum, $\bigsqcap \mathfrak{O} \in \mathfrak{O}$. So S is itself Owicki-Gries-core provable. Notice that S is stronger than or equal to any other Owicki-Gries-core-provable property. ∎

Proposition 27. *Any Owicki-Gries-core proof denotes an inductive invariant:*

$$\forall\, P \in \mathfrak{O}: \quad init \cup post(\gamma_{\mathrm{OG}}(P)) \subseteq \gamma_{\mathrm{OG}}(P)\,.$$

Proof. Let $(I,M) \in \mathfrak{O}$. We have to prove the following two claims.

"$post \circ \gamma_{\mathrm{OG}}(I,M) \subseteq \gamma_{\mathrm{OG}}(I,M)$": Let $v' \in post \circ \gamma_{\mathrm{OG}}(I,M)$. Then there is $v \in \gamma_{\mathrm{OG}}(I,M)$ and $t \in Tid$ such that $v \to_t v'$. Notice that by (1) we have $v(pc_{\hat{t}}) = v'(pc_{\hat{t}})$ for $\hat{t} \in Tid \setminus \{t\}$. Let $R = rsc(v(pc_t))$ and $R' = rsc(v'(pc_t))$. From $v \in \gamma_{\mathrm{OG}}(I,M)$ we get $v|_{LocalDataVar_t \cup \bigcup ProofVar(R)} \in M_{v(pc_t),t}$. Now let $r \in R' \setminus R$. From $v' \in ConsState$ and $r \in rsc(v'(pc_t))$ we get $r \notin \bigcup_{\hat{t} \in Tid \setminus \{t\}} rsc(v'(pc_{\hat{t}}))$ = [since $v(pc_{\hat{t}}) = v'(pc_{\hat{t}})$ for $\hat{t} \in Tid \setminus \{t\}$] $\bigcup_{\hat{t} \in Tid \setminus \{t\}} rsc(v(pc_{\hat{t}}))$. Since $r \notin R = rsc(v(pc_t))$, we conclude that $r \notin \bigcup rsc(v(PCVar))$. Since $v \in \gamma_{\mathrm{OG}}(I,M)$, we obtain $v|_{ProofVar(r)} \in I_r$. Thus $\forall\, r \in R' \setminus R$: $v|_{ProofVar(r)} \in I_r$. By sequential consistency, $v'|_{LocalDataVar_t \cup \bigcup ProofVar(R')} \in M_{v'(pc_t),t}$ and $\forall\, r \in R \setminus R' : v'|_{ProofVar(r)} \in I_r$.

Now we will show that v' is in $\gamma_{\mathrm{OG}}(I,M)$.

- Let $r \in Res \setminus \bigcup rsc(v'(PCVar))$. From $t \in Tid$, $v \to_t v'$ and the definition of *ProofVar* we obtain that for all $x \in ProofVar(r)$ we have $v(x) = v'(x)$. So $v|_{ProofVar(r)} = v'|_{ProofVar(r)}$. From (1) we know for $\tilde{t} \in Tid \setminus \{t\}$ that $v(pc_{\tilde{t}}) = v'(pc_{\tilde{t}})$, thus $rsc(v(pc_{\tilde{t}})) = rsc(v'(pc_{\tilde{t}})) \not\ni r$. Consider two cases.
  Case $r \in rsc(v(pc_t))$. Then $r \in R \setminus R'$. By the above, $v'|_{ProofVar(r)} \in I_r$.
  Case $r \notin rsc(v(pc_t))$. Then $r \notin \bigcup rsc(v(PCVar))$. Thus $v|_{ProofVar(r)} \in I_r$. Therefore $v'|_{ProofVar(r)} \in I_r$.
  So $\forall\, r \in Res \setminus \bigcup rsc(v'(PCVar)) : v'|_{ProofVar(r)} \in I_r$.
- Let $\tilde{t} \in Tid$. Consider two cases.
  Case $\tilde{t} = t$. By the above, $v'|_{LocalDataVar_t \cup \bigcup ProofVar(R')} \in M_{v'(pc_t),t}$.
  Case $\tilde{t} \neq t$. From $v(pc_{\tilde{t}}) = v'(pc_{\tilde{t}})$ we get $LocalDataVar_{\tilde{t}} \cup \bigcup ProofVar(rsc(v(pc_{\tilde{t}})))$ $= LocalDataVar_{\tilde{t}} \cup \bigcup ProofVar(rsc(v'(pc_{\tilde{t}})))$. Let x be an element of $LocalDataVar_{\tilde{t}}$ $\cup \bigcup ProofVar(rsc(v(pc_{\tilde{t}})))$. If $x \in LocalDataVar_{\tilde{t}}$, then (1) implies $v(x) = v'(x)$. Otherwise $x \in ProofVar(r)$ for some $r \in rsc(v'(pc_{\tilde{t}}))$. From $v' \in ConsState$ we get $r \notin rsc(v'(pc_t))$. From $x \in ProofVar(r)$ we obtain $v(x) = v'(x)$. Combining together, $v'|_{LocalDataVar_{\tilde{t}} \cup \bigcup ProofVar(rsc(v'(pc_{\tilde{t}})))} = v|_{LocalDataVar_{\tilde{t}} \cup \bigcup ProofVar(rsc(v(pc_{\tilde{t}})))}$ $\in$ [since $v \in \gamma_{\mathrm{OG}}(I,M)$] $M_{v(pc_{\tilde{t}}),\tilde{t}} = M_{v'(pc_{\tilde{t}}),\tilde{t}}$.
  So $\forall\, \tilde{t} \in Tid : v'|_{LocalDataVar_{\tilde{t}} \cup \bigcup ProofVar(rsc(v'(pc_{\tilde{t}})))} \in M_{v'(pc_{\tilde{t}}),\tilde{t}}$.

We have shown $v' \in \gamma_{\mathrm{OG}}(I,M)$.

"$init \subseteq \gamma_{\mathrm{OG}}(I,M)$": By definition of an Owicki-Gries-core proof. ∎

B.3 Examples of Owicki-Gries-core proofs

For $k \in \mathbb{N}^+$ let $\mathbb{N}_k = \mathbb{N} \cap [1,k]$.

Extension of Example 1. The syntactic structure of Readers-Writers $(n,m > 1)$ is given by

– $Tid = \{\texttt{reader}_i, \texttt{writer}_j \mid 1 \le i \le n, 1 \le j \le m\}$;
– $PL = \{$`startreadA`, `startreadB`, `startreadC`, `read`, `finishreadA`, `finishreadB`, `finishreadC`, `askwriteA`, `askwriteB`, `askwriteC`,

$\texttt{startwriteA}, \texttt{startwriteB}, \texttt{startwriteC}, \texttt{write}, \texttt{finishwriteA}, \texttt{finishwriteB}, \texttt{finishwriteC}\}$;

- $Val = PL \dot{\cup} \mathbb{N}_0$, where $\mathbb{N}_0$ is the set of nonnegative integers;
- $Res = \{control\} = \{\{ww, ar, aw\}\}$;
- $Local_{\texttt{reader}_i} = \{pc_{\texttt{reader}_i}\}$ $(1 \le i \le n)$, $Local_{\texttt{writer}_j} = \{pc_{\texttt{writer}_j}\}$ $(1 \le j \le m)$;
- $ProofVar(control) = control$;
- $rsc(\ldots\texttt{A}) = rsc(\texttt{read}) = rsc(\texttt{write}) = \emptyset$, $rsc(\ldots\texttt{B}) = rsc(\ldots\texttt{C}) = \{control\}$.

Since we have assumed at least two readers and two writers, there are no local data variables. Let
$I_{control} = \{v : control \to \mathbb{N}_0 \mid v(aw) \le 1\}$,
$M \in \prod_{p \in PL, t \in Tid} \mathfrak{P}((LocalDataVar_t \cup ProofVar(rsc(p))) \to Val)$ given by
$M_{\ldots\texttt{readA},\texttt{reader}_i} = M_{\texttt{read},\texttt{reader}_i} = \{\emptyset\}$,
$M_{\texttt{startreadB},\texttt{reader}_i} = \{v \in (\{ww, ar, aw\} \to \mathbb{N}_0) \mid v(ww) = 0 \wedge v(aw) \le 1\}$,
$M_{\texttt{startreadC},\texttt{reader}_i} = \{v \in (\{ww, ar, aw\} \to \mathbb{N}_0) \mid v(ww) = 0 \wedge v(aw) \le 1 \le v(ar)\}$,
$M_{\texttt{finishreadB},\texttt{reader}_i} = M_{\texttt{finishreadC},\texttt{reader}_i} = \{v \in (\{ww, ar, aw\} \to \mathbb{N}_0) \mid v(aw) \le 1\}$ for $1 \le i \le n$,
$M_{\ldots\texttt{writeA},\texttt{writer}_j} = M_{\texttt{write},\texttt{writer}_j} = \{\emptyset\}$,
$M_{\texttt{askwriteB},\texttt{writer}_j} = M_{\texttt{finishwriteB},\texttt{writer}_j} = \{v \in (\{ww, ar, aw\} \to \mathbb{N}_0) \mid v(aw) \le 1\}$,
$M_{\texttt{askwriteC},\texttt{writer}_j} = \{v \in (\{ww, ar, aw\} \to \mathbb{N}_0) \mid v(aw) \le 1 \le v(ww)\}$,
$M_{\texttt{startwriteB},\texttt{writer}_j} = \{v \in (\{ww, ar, aw\} \to \mathbb{N}_0) \mid v(ar) = 0 = v(aw)\}$,
$M_{\texttt{startwriteC},\texttt{writer}_j} = \{v \in (\{ww, ar, aw\} \to \mathbb{N}_0) \mid v(ar) = 0 \wedge v(aw) = 1\}$,
$M_{\texttt{finishwriteC},\texttt{writer}_j} = \{v \in (\{ww, ar, aw\} \to \mathbb{N}_0) \mid 0 = v(aw)\}$ for $1 \le j \le n$,
the remaining $M_{\ldots}$ are empty sets.
Notice that $P = ([control \mapsto I_{control}], M)$ is an annotation of Readers-Writers. It is sequentially consistent and the single initial state, say, v^{init}, satisfies $v^{init}(ww) = v^{init}(ar) = v^{init}(aw) = 0$, thus $v^{init}|_{ProofVar(control)} \in I_{control}$, so $v^{init} \in \gamma_{\mathrm{OG}}(P)$. Thus P is an Owicki-Gries-core proof.
Now we will show that this proof is the smallest one. So let $([control \mapsto \hat{I}_{control}, \hat{M}])$ be an arbitrary Owicki-Gries-core proof for Readers-Writers.
First notice that by starting with the initial state and successively applying sequential consistency we obtain nonemptiness of certain annotation parts:
Fact 1. For every $i \in \mathbb{N}_n$ and every $l \in \{\texttt{startreadA}, \texttt{read}, \texttt{finishreadA}\}$ we have $\emptyset \in \hat{M}_{l,\texttt{reader}_i}$; for every $j \in \mathbb{N}_m$ and every $l \in \{\texttt{askwriteA}, \texttt{startwriteA}, \texttt{write}, \texttt{finishwriteA}\}$ we have $\emptyset \in \hat{M}_{l,\texttt{writer}_j}$.
Now we will show in three steps that for any $(a, b, c) \in \mathbb{N}_0^2 \times \{0, 1\}$ the map $[ww \mapsto a, ar \mapsto b, aw \mapsto c]$ is in $\hat{I}_{control}$.

Step 1. If $c = 0$, we get $[ww \mapsto 0, ar \mapsto 0, aw \mapsto c] \in \hat{I}_{control}$ from the initial condition. If $c = 1$, apply sequential consistency to states v, v' with $v(pc_{\texttt{writer}_1}) = \texttt{startwriteA}$ and $v \to_{\texttt{writer}_1} v'$ and to $[ww \mapsto 0, ar \mapsto 0, aw \mapsto 0] \in \hat{I}_{control}$ to get $[ww \mapsto 0, ar \mapsto 0, aw \mapsto 0] \in M_{\texttt{startwriteB},\texttt{writer}_1}$. Applying sequential consistency to the next step of the same thread to get $[ww \mapsto 0, ar \mapsto 0, aw \mapsto 1] \in M_{\texttt{startwriteC},\texttt{writer}_1}$. Applying sequential consistency to the termination of the critical section, we obtain $[ww \mapsto 0, ar \mapsto 0, aw \mapsto c] \in \hat{I}_{control}$.

Step 2. Now we will prove by induction on b that for all $b \ge 0$, $c \in \{0, 1\}$ the map $[ww \mapsto 0, ar \mapsto b, aw \mapsto c]$ is in $\hat{I}_{control}$. Notice that for $b = 0$ we already have

$[ww \mapsto 0, ar \mapsto 0, aw \mapsto c] \in \hat{I}_{control}$. If $b > 0$, the induction assumption gives $[ww \mapsto 0, ar \mapsto b-1, aw \mapsto c] \in \hat{I}_{control}$. Applying sequential consistency to some states v, v' with $v(pc_{\mathtt{reader}_1}) = \mathtt{startreadA}$ and $v \rightarrow_{\mathtt{reader}_1} v'$, we get $[ww \mapsto 0, ar \mapsto b-1, aw \mapsto c] \in M_{\mathtt{startreadB},\mathtt{reader}_1}$. Applying sequential consistency once again to the same thread, we obtain $[ww \mapsto 0, ar \mapsto b, aw \mapsto c] \in M_{\mathtt{startreadC},\mathtt{reader}_1}$. Applying sequential consistency to the transition terminating the critical section, we obtain $[ww \mapsto 0, ar \mapsto b, aw \mapsto c] \in \hat{I}_{control}$.

Step 3. Now we will prove by induction on a that for all $a, b \geq 0$, $c \in \{0,1\}$ the map $[ww \mapsto a, ar \mapsto b, aw \mapsto c]$ is in $\hat{I}_{control}$. Notice that for $a = 0$ we have just proven $[ww \mapsto 0, ar \mapsto b, aw \mapsto c] \in \hat{I}_{control}$. For $a > 0$, the induction assumption states that $[ww \mapsto a-1, ar \mapsto b, aw \mapsto c] \in \hat{I}_{control}$. Sequential consistency, applied to any pair of states v, v' with $v(pc_{\mathtt{writer}_1}) = \mathtt{askwriteA}$ and $v \rightarrow_{\mathtt{writer}_1} v'$, gives $[ww \mapsto a-1, ar \mapsto b, aw \mapsto c] \in M_{\mathtt{askwriteB},\mathtt{writer}_1}$. Applying sequential consistency once again gives $[ww \mapsto a, ar \mapsto b, aw \mapsto c] \in M_{\mathtt{askwriteC},\mathtt{writer}_1}$, and applying it again gives $[ww \mapsto a, ar \mapsto b, aw \mapsto c] \in \hat{I}_{control}$.

We have shown that $I_{control} \subseteq \hat{I}_{control}$.
From fact 1 we get $M_{\mathtt{startreadA},\mathtt{reader}_i} \subseteq \hat{M}_{\mathtt{startreadA},\mathtt{reader}_i}$, $M_{\mathtt{finishreadA},\mathtt{reader}_i} \subseteq \hat{M}_{\mathtt{finishreadA},\mathtt{reader}_i}$, $M_{\mathtt{read},\mathtt{reader}_i} \subseteq \hat{M}_{\mathtt{read},\mathtt{reader}_i}$ $(1 \leq i \leq n)$ and $M_{\mathtt{askwriteA},\mathtt{writer}_j} \subseteq \hat{M}_{\mathtt{askwriteA},\mathtt{writer}_j}$, $M_{\mathtt{startwriteA},\mathtt{writer}_j} \subseteq \hat{M}_{\mathtt{startwriteA},\mathtt{writer}_j}$, $M_{\mathtt{write},\mathtt{writer}_j} \subseteq \hat{M}_{\mathtt{write},\mathtt{writer}_j}$, $M_{\mathtt{finishwriteA},\mathtt{writer}_j} \subseteq \hat{M}_{\mathtt{finishwriteA},\mathtt{writer}_j}$ $(1 \leq j \leq m)$.
Fix some $i \in \mathbb{N}_n$ and states $v \rightarrow_{\mathtt{reader}_i} v'$ such that $v(pc_{\mathtt{reader}_i}) = \mathtt{startreadA}$, $v(ww) = 0$, $v(ar) \geq 0$ and $v(aw) \leq 1$. Then $v|_{LocalDataVar_{\mathtt{reader}_i} \cup \bigcup ProofVar(rsc(\mathtt{startreadA}))} = \emptyset \in \hat{M}_{\mathtt{startreadA},\mathtt{reader}_i}$ and $v|_{ProofVar(control)} \in \hat{I}_{control}$. By sequential consistency, $v'|_{LocalDataVar_{\mathtt{reader}_i} \cup \bigcup ProofVar(rsc(\mathtt{startreadB}))} = v'|_{control} \in \hat{M}_{\mathtt{startreadB},\mathtt{reader}_i}$. Thus for any $b \geq 0$ and $c \in \{0,1\}$ we have $[ww \mapsto 0, ar \mapsto b, aw \mapsto c] \in \hat{M}_{\mathtt{startreadB},\mathtt{reader}_i}$. In particular, $M_{\mathtt{startreadB},\mathtt{reader}_i} \subseteq \hat{M}_{\mathtt{startreadB},\mathtt{reader}_i}$.
Applying sequential consistency to v' and v'' with $v' \rightarrow_{\mathtt{reader}_i} v''$ such that $v'(pc_{\mathtt{reader}_i}) = \mathtt{startreadB}$, $v'(ww) = 0$ and $v'(ar) \geq 0$ and $v'(aw) \in \{0,1\}$, we obtain $M_{\mathtt{startreadC},\mathtt{reader}_i} \subseteq \hat{M}_{\mathtt{startreadC},\mathtt{reader}_i}$.
Analogously we get $M_{\mathtt{finishreadB},\mathtt{reader}_i} \subseteq \hat{M}_{\mathtt{finishreadB},\mathtt{reader}_i}$, $M_{\mathtt{finishreadC},\mathtt{reader}_i} \subseteq \hat{M}_{\mathtt{finishreadC},\mathtt{reader}_i}$ $(1 \leq i \leq n)$ and $M_{\mathtt{askwriteB},\mathtt{writer}_j} \subseteq \hat{M}_{\mathtt{askwriteB},\mathtt{writer}_j}$, $M_{\mathtt{askwriteC},\mathtt{writer}_j} \subseteq \hat{M}_{\mathtt{askwriteC},\mathtt{writer}_j}$, $M_{\mathtt{startwriteB},\mathtt{writer}_j} \subseteq \hat{M}_{\mathtt{startwriteB},\mathtt{writer}_j}$, $M_{\mathtt{startwriteC},\mathtt{writer}_j} \subseteq \hat{M}_{\mathtt{startwriteC},\mathtt{writer}_j}$, $M_{\mathtt{finishwriteB},\mathtt{writer}_j} \subseteq \hat{M}_{\mathtt{finishwriteB},\mathtt{writer}_j}$, $M_{\mathtt{finishwriteC},\mathtt{writer}_j} \subseteq \hat{M}_{\mathtt{finishwriteC},\mathtt{writer}_j}$ $(1 \leq j \leq m)$. So P is really the smallest Owicki-Gries-core proof.

Now we will show that $\gamma_{\mathrm{OG}}(P)$ equals

$$\begin{aligned}
\{v \in \mathit{ConsState} \mid\ & v(ww), v(ar) \in \mathbb{N}_0 \ \wedge\ v(aw) \in \{0,1\} \\
& \wedge\ \forall\, i \in \{1,\dots,n\} : \\
& \qquad (v(pc_{\mathtt{reader}_i}) \in \{x\mathtt{ready}y, \mathtt{read} \mid x \in \{\mathtt{start},\mathtt{finish}\} \\
& \qquad\qquad\qquad \wedge\ y \in \{\mathtt{A},\mathtt{B},\mathtt{C}\}\}) \\
& \qquad \wedge (v(pc_{\mathtt{reader}_i}) = \mathtt{startreadB} \Rightarrow v(ww) = 0) \\
& \qquad \wedge (v(pc_{\mathtt{reader}_i}) = \mathtt{startreadC} \Rightarrow v(ww) = 0 < v(ar)) \\
& \wedge\ \forall\, j \in \{1,\dots,m\} : \\
& \qquad (v(pc_{\mathtt{writer}_j}) \in \{x\mathtt{write}y, \mathtt{write} \mid x \in \{\mathtt{ask},\mathtt{start},\mathtt{finish}\} \\
& \qquad\qquad\qquad \wedge\ y \in \{\mathtt{A},\mathtt{B},\mathtt{C}\}\}) \\
& \qquad \wedge (v(pc_{\mathtt{writer}_j}) = \mathtt{askwriteC} \Rightarrow v(ww) > 0) \\
& \qquad \wedge (v(pc_{\mathtt{writer}_j}) = \mathtt{startwriteB} \Rightarrow v(ar) = v(aw) = 0) \\
& \qquad \wedge (v(pc_{\mathtt{writer}_j}) = \mathtt{startwriteC} \Rightarrow v(ar) = 0 < v(aw)) \\
& \qquad \wedge (v(pc_{\mathtt{writer}_j}) = \mathtt{finishwriteC} \Rightarrow v(aw) = 0)\,\}\,.
\end{aligned}$$

"$\subseteq$": Since there are no local data variables, the defining condition of $\gamma_{\mathrm{OG}}(P)$ restricts the variables of *control* at least once: either by the resource invariant, or by the annotation of a control flow location in a critical region. Since both the resource invariant as well as the annotations of all control flow locations inside the critical regions allow only values 0 and 1 of *aw*, we have $\pi_{aw}(\gamma_{\mathrm{OG}}(P)) \subseteq \{0,1\}$. Some $M_{\dots}$ are empty, restricting the program counters of the readers and writers. If the variables *ww*, *ar* are restricted beyond being nonnegative, then the restriction comes from the nonempty annotations of the control flow points.

"$\supseteq$": Take a state v from the right hand side. It satisfies the resource invariant in any case. If *control* is available in v, then no control flow annotation imposes additional restrictions on v, so v is in the left hand side. If a state's location is in a critical section, then the location is unique. The annotation of that location is satisfied either because the annotation is exactly the resource invariant. Or the annotation is satisfied by a condition of the right hand side.

□

Extension of Example 2. The syntactic structure of Upper is given by

- $\mathit{Tid} = \{1,2\}$;
- $\mathit{PL} = \{\mathtt{A},\mathtt{B},\mathtt{C},\dots,\mathtt{V},\mathtt{W}\}$;
- $\mathit{Val} = \mathit{PL} \dot\cup \{0,1\}$;
- $\mathit{Res} = \{r, r'\}$ where $r = \{u,z\}$, $r' = \{x,y\}$;
- $\mathit{Local}_1 = \{l, x, pc_1\}$, $\mathit{Local}_2 = \{y, z, pc_2\}$;
- $\mathit{ProofVar}(r) = \{u,z\} = r$, $\mathit{ProofVar}(r') = \{x,y\} = r'$;
- $\mathit{rsc} : \mathit{PL} \to \mathfrak{P}(\mathit{Res})$ given by

p	$\mathtt{A},\mathtt{G},\mathtt{J},\mathtt{M},\mathtt{N},\mathtt{O},\mathtt{P},\mathtt{S},\mathtt{W}$	$\mathtt{B},\mathtt{E},\mathtt{F},\mathtt{K},\mathtt{L},\mathtt{T},\mathtt{U},\mathtt{V}$	$\mathtt{H},\mathtt{I},\mathtt{Q},\mathtt{R}$	$\mathtt{C},\mathtt{D}$
$\mathit{rsc}(p)$	$\emptyset$	$\{r\}$	$\{r'\}$	$\{r,r'\}$

We have
$init = \{\ [u \mapsto 1, z \mapsto 0, x \mapsto 0, y \mapsto 0, l \mapsto 0, pc_1 \mapsto \mathtt{A}, pc_2 \mapsto \mathtt{O}],$
$[u \mapsto 0, z \mapsto 1, x \mapsto 1, y \mapsto 1, l \mapsto 1, pc_1 \mapsto \mathtt{A}, pc_2 \mapsto \mathtt{O}]\ \}.$
Let $P = ([r \mapsto I_r, r' \mapsto I_{r'}], (M_{p,t})_{p \in PL, t \in Tid})$, where
$I_r = \{[u \mapsto 0, z \mapsto 1], [u \mapsto 1, z \mapsto 0]\}$,
$I_{r'} = \{[x \mapsto 0, y \mapsto 0], [x \mapsto 0, y \mapsto 1], [x \mapsto 1, y \mapsto 1]\}$,
$M_{\mathtt{A},1} = \{[x \mapsto 0, l \mapsto 0], [x \mapsto 1, l \mapsto 1]\}$,
$M_{\mathtt{B},1} = \{[u \mapsto 0, z \mapsto 1, x \mapsto 0, l \mapsto 0], [u \mapsto 1, z \mapsto 0, x \mapsto 1, l \mapsto 1]\}$,
$M_{\mathtt{C},1} = \{\ [u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 0, l \mapsto 0], [u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 1, l \mapsto 0],$
$[u \mapsto 1, z \mapsto 0, x \mapsto 1, y \mapsto 1, l \mapsto 1]\}$,
$M_{\mathtt{D},1} = \{\ [u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 0, l \mapsto 0], [u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 1, l \mapsto 0],$
$[u \mapsto 1, z \mapsto 0, x \mapsto 0, y \mapsto 1, l \mapsto 1]\}$,
$M_{\mathtt{E},1} = \{[u \mapsto 0, z \mapsto 1, x \mapsto 0, l \mapsto 0], [u \mapsto 1, z \mapsto 0, x \mapsto 0, l \mapsto 1]\}$,
$M_{\mathtt{O},2} = \{[y \mapsto 0, z \mapsto 0], [y \mapsto 1, z \mapsto 1]\}$,
$M_{x,t} = \emptyset$ for $(x,t) \in (PL \times Tid) \setminus \{(\mathtt{A},1), (\mathtt{B},1), (\mathtt{C},1), (\mathtt{D},1), (\mathtt{E},1), (\mathtt{O},2)\}$.
Notice that the two initial states satisfy annotations of locations A and O as well as the resource invariants. Thus the program annotation P admits the initial states. Furthermore, P is sequentially consistent. Thus P is an Owicki-Gries-core proof.
Now we'll show that P is smaller than or equal to any Owicki-Gries-core proof $([r \mapsto \hat{I}_r, r' \mapsto \hat{I}_{r'}], (\hat{M}_{p,t})_{p \in PL, t \in Tid})$. From the initial condition we get $[u \mapsto 0, z \mapsto 1]$, $[u \mapsto 1, z \mapsto 0] \in \hat{I}_r$, so $I_r \subseteq \hat{I}_r$. The initial condition also implies $[x \mapsto 0, l \mapsto 0]$, $[x \mapsto 1, l \mapsto 1] \in \hat{M}_{\mathtt{A},1}$, so $M_{\mathtt{A},1} \subseteq \hat{M}_{\mathtt{A},1}$. Again from the initial condition we get $[y \mapsto 0, z \mapsto 0]$, $[y \mapsto 1, z \mapsto 1] \in \hat{M}_{\mathtt{O},2}$, so $M_{\mathtt{O},2} \subseteq \hat{M}_{\mathtt{O},2}$. At last the initial condition gives $[x \mapsto 0, y \mapsto 0]$, $[x \mapsto 1, y \mapsto 1] \in \hat{I}_{r'}$. Now let
$v^{\mathtt{A}} = [u \mapsto 1, z \mapsto 0, x \mapsto 1, y \mapsto 1, l \mapsto 1, pc_1 \mapsto \mathtt{A}, pc_2 \mapsto \mathtt{O}]$ and
$v^{\mathtt{B}} = [u \mapsto 1, z \mapsto 0, x \mapsto 1, y \mapsto 1, l \mapsto 1, pc_1 \mapsto \mathtt{B}, pc_2 \mapsto \mathtt{O}]$.
Notice that $rsc(\mathtt{A}) = \emptyset$, $rsc(\mathtt{B}) = \{r\}$, so
$v^{\mathtt{A}}|_{LocalDataVar_1 \cup \bigcup ProofVar(rsc(\mathtt{A}))} = [x \mapsto 1, l \mapsto 1] \in \hat{M}_{\mathtt{A},1}$ and
$v^{\mathtt{A}}|_{ProofVar(r)} = [u \mapsto 1, z \mapsto 0] \in \hat{I}_r$.
Since $v^{\mathtt{A}}(u) = 1 = v^{\mathtt{A}}(l)$, we have $v^{\mathtt{A}} \to_1 v^{\mathtt{B}}$. By sequential consistency,
$v^{\mathtt{B}}|_{LocalDataVar_1 \cup \bigcup ProofVar(rsc(\mathtt{B}))} \in \hat{M}_{\mathtt{B},1}$, i.e., $[u \mapsto 1, z \mapsto 0, x \mapsto 1, l \mapsto 1] \in \hat{M}_{\mathtt{B},1}$. Let
$v^{\mathtt{C}} = [u \mapsto 1, z \mapsto 0, x \mapsto 1, y \mapsto 1, l \mapsto 1, pc_1 \mapsto \mathtt{C}, pc_2 \mapsto \mathtt{O}]$.
Since $v^{\mathtt{B}}|_{ProofVar(r')} = [x \mapsto 1, y \mapsto 1] \in \hat{I}_{r'}$ and $v^{\mathtt{B}} \to_1 v^{\mathtt{C}}$, we obtain from the sequential consistency condition that $v^{\mathtt{C}}|_{LocalDataVar_1 \cup \bigcup ProofVar(rsc(\mathtt{C}))} \in \hat{M}_{\mathtt{C},1}$.
Let
$v^{\mathtt{D}} = [u \mapsto 1, z \mapsto 0, x \mapsto 0, y \mapsto 1, l \mapsto 1, pc_1 \mapsto \mathtt{D}, pc_2 \mapsto \mathtt{O}]$.
Since $v^{\mathtt{C}} \to_1 v^{\mathtt{D}}$, sequential consistency gives $v^{\mathtt{D}}|_{LocalDataVar_1 \cup \bigcup ProofVar(rsc(\mathtt{D}))} \in \hat{M}_{\mathtt{D},1}$.
Let
$v^{\mathtt{E}} = [u \mapsto 1, z \mapsto 0, x \mapsto 0, y \mapsto 1, l \mapsto 1, pc_1 \mapsto \mathtt{E}, pc_2 \mapsto \mathtt{O}]$.
From $v^{\mathtt{D}} \to_1 v^{\mathtt{E}}$, sequential consistency and $r' \in rsc(\mathtt{D}) \setminus rsc(\mathtt{E})$ we get
$v^{\mathtt{E}}|_{LocalDataVar_1 \cup \bigcup ProofVar(rsc(\mathtt{E}))} \in \hat{M}_{\mathtt{E},1}$ and $\hat{I}_{r'} \ni v^{\mathtt{E}}|_{ProofVar(r')} = [x \mapsto 0, y \mapsto 1]$.
Thus $I_{r'} \subseteq \hat{I}_{r'}$.
Now let
$\bar{v}^{\mathtt{A}} = [u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 1, l \mapsto 0, pc_1 \mapsto \mathtt{A}, pc_2 \mapsto \mathtt{O}]$ and
$\bar{v}^{\mathtt{B}} = [u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 1, l \mapsto 0, pc_1 \mapsto \mathtt{B}, pc_2 \mapsto \mathtt{O}]$.

Since $\bar{v}^{\mathtt{A}}(u) = 0 = \bar{v}^{\mathtt{A}}(l)$, we have $\bar{v}^{\mathtt{A}} \to_1 \bar{v}^{\mathtt{B}}$. Since additionally $\bar{v}^{\mathtt{A}}|_{LocalDataVar_1 \cup \bigcup ProofVar(rsc(\mathtt{A}))} = [x \mapsto 0, l \mapsto 0] \in \hat{M}_{\mathtt{A},1}$, $rsc(\bar{v}^{\mathtt{B}}) \setminus rsc(\bar{v}^{\mathtt{A}}) = \{r\}$ and $\bar{v}^{\mathtt{A}}|_{ProofVar(r)} = [u \mapsto 0, z \mapsto 1] \in \hat{I}_r$, we have $\hat{M}_{\mathtt{B},1} \ni \bar{v}^{\mathtt{B}}|_{LocalDataVar_1 \cup \bigcup ProofVar(rsc(\mathtt{B}))} = [u \mapsto 0, z \mapsto 1, x \mapsto 0, l \mapsto 0]$.
Since $[u \mapsto 1, z \mapsto 0, x \mapsto 1, l \mapsto 1]$, $[u \mapsto 0, z \mapsto 1, x \mapsto 0, l \mapsto 0] \in \hat{M}_{\mathtt{B},1}$, we have $M_{\mathtt{B},1} \subseteq \hat{M}_{\mathtt{B},1}$.
From $[u \mapsto 0, z \mapsto 1, x \mapsto 0, l \mapsto 0]$, $[u \mapsto 1, z \mapsto 0, x \mapsto 1, l \mapsto 1] \in \hat{M}_{\mathtt{B},1}$, $[x \mapsto 0, y \mapsto 0]$, $[x \mapsto 0, y \mapsto 1]$, $[x \mapsto 1, y \mapsto 1] \in \hat{I}_r$, we get $[u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 0, l \mapsto 0]$, $[u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 1, l \mapsto 0]$, $[u \mapsto 1, z \mapsto 0, x \mapsto 1, y \mapsto 1, l \mapsto 1] \in \hat{M}_{\mathtt{C},1}$. Thus, $M_{\mathtt{C},1} \subseteq \hat{M}_{\mathtt{C},1}$.
Applying sequential consistency with respect to $x := 0$ we get $[u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 0, l \mapsto 0]$, $[u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 1, l \mapsto 0]$, $[u \mapsto 1, z \mapsto 0, x \mapsto 0, y \mapsto 1, l \mapsto 1] \in \hat{M}_{\mathtt{D},1}$, i.e., $M_{\mathtt{D},1} \subseteq \hat{M}_{\mathtt{D},1}$.
Applying sequential consistency to $[u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 0, l \mapsto 0]$, $[u \mapsto 1, z \mapsto 0, x \mapsto 0, y \mapsto 1, l \mapsto 1] \in \hat{M}_{\mathtt{D},1}$ we get $[u \mapsto 0, z \mapsto 1, x \mapsto 0, l \mapsto 0]$, $[u \mapsto 1, z \mapsto 0, x \mapsto 0, l \mapsto 1] \in \hat{M}_{\mathtt{E},1}$, i.e., $M_{\mathtt{E},1} \subseteq \hat{M}_{\mathtt{E},1}$.
We have shown that P is the smallest Owicki-Gries-core proof. It denotes the following states (column = state):

$$\begin{array}{l|llllllll}
u & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
z & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
x & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
y & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
l & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
pc_1 & \mathtt{A} & \mathtt{A} & \mathtt{A} & \mathtt{B} & \mathtt{C} & \mathtt{D} & \mathtt{E} & \mathtt{E} \\
pc_2 & \mathtt{0} & \mathtt{0} & \mathtt{0} & \mathtt{0} & \mathtt{0} & \mathtt{0} & \mathtt{0} & \mathtt{0}
\end{array}$$

□

Extension of Example 3. A syntactic structure = $(Tid, PL, Val, Res, (Local_t, pc_t)_{t \in Tid}, ProofVar, rsc)$ belongs to the class *StrSimple* iff $|Res| \leq 1$, $(\forall\, t \in Tid\colon Local_t \cap \bigcup Res = \emptyset)$ and $(\forall\, r \in Res\colon ProofVar(r) = r \neq \emptyset)$. An RPL transition system $(init, (\to_t)_{t \in Tid})$ belongs to the class *Simple* iff it obeys a syntactic structure from StrSimple.
All programs from the family Readers-Writers are in Simple. □

Extension of Example 4. A syntactic structure $(Tid, PL, Val, Res, (Local_t, pc_t)_{t \in Tid}, ProofVar, rsc)$ belongs to the class *StrSepThreads* iff $Res = \emptyset$ and Tid is finite. An RPL transition system $(init, (\to_t)_{t \in Tid})$ belongs to the class *SepThreads* iff it obeys a syntactic structure from StrSepThreads and $\forall\, v \in ConsState\colon\ (\forall\, t \in Tid\ \exists\, \tilde{v} \in init\colon\ v \sim_{Local_t} \tilde{v}) \Rightarrow v \in init$.
Notice that StrSepThreads $\subseteq$ StrSimple and SepThreads $\subseteq$ Simple.
We will show later, using the lower bound, that the smallest Owicki-Gries-core proof of an RPL transition system from SepThreads denotes the strongest invariant[5]. □

[5] For infinite *Tid* only those states are reachable that have finitely many non-initial components, but the Owicki-Gries-core proof denotes states that may have arbitrary many non-initial components. Thus we study here the finite-threaded case only.

C Owicki-Gries core as abstract interpretation

Within the section C, fix a syntactic structure and an RPL transition system obeying it as in section A. Let

$$\begin{aligned}
&\mathit{next}^{\#}_{\mathrm{OG}} : PA \to PA\,,\\
&\quad (I,M) \mapsto ((\{v'|_{\mathit{ProofVar}(r)} \mid \exists\, t \in \mathit{Tid}, v:\ v \to_t v' \wedge\\
&\qquad\qquad v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(v(pc_t)))} \in M_{v(pc_t),t} \wedge\\
&\qquad\qquad r \in rsc(v(pc_t)) \setminus rsc(v'(pc_t))\})_{r \in Res},\\
&\qquad (\{v'|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(p))} \mid \exists\, v:\ v \to_t v' \wedge v'(pc_t) = p\\
&\qquad\qquad \wedge\, v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(v(pc_t)))} \in M_{v(pc_t),t} \wedge\\
&\qquad\qquad \forall\, r \in rsc(p) \setminus rsc(v(pc_t)) : v|_{\mathit{ProofVar}(r)} \in I_r\})_{\substack{p \in PL,\\ t \in \mathit{Tid}}})\,,\\
&I^{init} = (\{v|_{\mathit{ProofVar}(r)} \mid v \in \mathit{init}\})_{r \in Res}\,,\\
&M^{init} = (\{v|_{\mathit{LocalDataVar}_t} \mid v \in \mathit{init} \wedge v(pc_t) = p\})_{p \in PL, t \in \mathit{Tid}}\,.
\end{aligned}$$

Let the *sound Owicki-Gries-core abstract successor map* be

$$\begin{aligned}
&\mathit{post}^{\#}_{\mathrm{OG}} : PA \to PA\,,\\
&\quad (I,M) \mapsto (I,M) \sqcup \mathit{next}^{\#}_{\mathrm{OG}}(I,M)\,,
\end{aligned}$$

and the *Owicki-Gries-core abstract transformer* be

$$\begin{aligned}
&F^{\#}_{\mathrm{OG}} : PA \to PA\,,\\
&\quad (I,M) \mapsto (I^{init}, M^{init}) \sqcup \mathit{post}^{\#}_{\mathrm{OG}}(I,M)\,.
\end{aligned}$$

Lemma 28. *The map $\mathit{next}^{\#}_{\mathrm{OG}}$ is monotone.*

Proof. Let $(I,M) \sqsubseteq (\hat{I},\hat{M})$ be program annotations, $\mathit{next}^{\#}_{\mathrm{OG}}(I,M) = (I',M')$, $\mathit{next}^{\#}_{\mathrm{OG}}(\hat{I},\hat{M}) = (\hat{I}',\hat{M}')$.

"$\forall\, r \in Res : I'_r \subseteq \hat{I}'_r$": Let $r \in Res$. Let $w \in I'_r$. Then there is a thread $t \in \mathit{Tid}$ and some transition $v \to_t v'$ such that $v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(v(pc_t)))} \in M_{v(pc_t),t}$, $r \in rsc(v(pc_t)) \setminus rsc(v'(pc_t))$ and $v'|_{\mathit{ProofVar}(r)} = w$. Then $\hat{M}_{v(pc_t),t} \ni v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(v(pc_t)))}$. So $w \in \hat{I}'_r$.

"$\forall\, t \in \mathit{Tid}, p \in PL : M'_{p,t} \subseteq \hat{M}'_{p,t}$": Let $t \in \mathit{Tid}$, $p \in PL$. Let $w \in M'_{p,t}$. Then there is some transition $v \to_t v'$ such that $v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(v(pc_t)))} \in M_{v(pc_t),t}$, $v'(pc_t) = p$, $\forall\, r \in rsc(p) \setminus rsc(v(pc_t)) : v|_{\mathit{ProofVar}(r)} \in I_r$ and $w = v'|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(p))}$. Then $\hat{M}_{v(pc_t),t} \ni v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(v(pc_t)))}$ and $\forall\, r \in rsc(p) \setminus rsc(v(pc_t)) : v|_{\mathit{ProofVar}(r)} \in \hat{I}_r$. So $w \in \hat{M}'_{p,t}$. ∎

Corollary 29. *The Owicki-Gries-core abstract transformer $F^{\#}_{\mathrm{OG}}$ is monotone.*

Corollary 29 implies that the Owicki-Gries-core abstract transformer has a least fixpoint, which follows from a well-known fixpoint theorem[6].

A *postfix point* of a map $f : X \to X$ on a poset $(X, \leq)$ is any $x \in X$ such that $f(x) \leq x$. Let $\mathit{postfp}(f)$ be the set of postfix points of f.

[6] Known today as Tarski's fixpoint theorem (Thm. 1 in [28]). This existence of the least fixpoint, however, was also mentioned independently, e.g., Exercise 5 in §4 of Chapter IV of [2] in year 1948. See also [3], Ch. V, §3, p. 115.

Theorem 30. *Every Owicki-Gries-core proof is a postfix point of the Owicki-Gries-core abstract transformer and vice versa. Formally:*

$$\mathfrak{O} = postfp(F^{\#}_{\mathrm{OG}})\,.$$

Proof. "$\subseteq$": Let $(I,M) \in \mathfrak{O}$. We will show that $(I',M') := F^{\#}_{\mathrm{OG}}(I,M)$ is less than or equal to (I,M). Let $(\hat{I},\hat{M}) = next^{\#}_{\mathrm{OG}}(I,M)$.

"$\forall\, r \in Res : I'_r \subseteq I_r$": Let $r \in Res$. Let $w \in I'_r$. Then $w \in I^{init}_r \cup I_r \cup \hat{I}_r$.

Case $w \in I^{init}_r$. Then there is some $v \in init$ such that $v|_{ProofVar(r)} = w$. By the initial condition, $v \in \gamma_{\mathrm{OG}}(I,M)$. Since every resource is available in an initial state, $r \in Res \setminus \bigcup rsc(v(PCVar))$. Thus $v|_{ProofVar(r)} \in I_r$.

Case $w \in I_r$. Then we don't have to prove anything more.

Case $w \in \hat{I}_r$. Then there is $t \in Tid$, $(v,v') \in \rightarrow_t$ such that $v|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(v(pc_t)))} \in M_{v(pc_t),t}$, $r \in rsc(v(pc_t)) \setminus rsc(v'(pc_t))$ and $w = v'|_{ProofVar(r)}$. By (5) we have $rsc(v(pc_t)) \supseteq rsc(v'(pc_t))$. Sequential consistency implies $v'|_{ProofVar(r)} \in I_r$.

In all cases $w \in I_r$. So $I'_r \subseteq I_r$.

"$\forall\, t \in Tid, p \in PL : M'_{p,t} \subseteq M_{p,t}$": Let $t \in Tid$, $p \in PL$. Let $w \in M'_{p,t}$. Then $w \in M^{init}_{p,t} \cup M_{p,t} \cup \hat{M}_{p,t}$.

Case $w \in M^{init}_{p,t}$. Then there is some $v \in init$ such that $v(pc_t) = p$ and $w = v|_{LocalDataVar_t}$. By the initial condition, $v \in \gamma_{\mathrm{OG}}(I,M)$. From definition of γ_{OG} and $rsc(v(pc_t)) = [v \in init]\ \emptyset$ we get $v|_{LocalDataVar_t} \in M_{p,t}$.

Case $w \in M_{p,t}$. Then nothing more has to be proven.

Case $w \in \hat{M}_{p,t}$. Then there is $(v,v') \in \rightarrow_t$ such that $v|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(v(pc_t)))} \in M_{v(pc_t),t}$, $v'(pc_t) = p$, $\forall\, r \in rsc(p) \setminus rsc(v(pc_t))$: $v|_{ProofVar(r)} \in I_r$ and $w = v'|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(p))}$. Then $v'|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(p))} \in M_{p,t}$ by sequential consistency.

In any case $w \in M_{p,t}$. So $M'_{p,t} \subseteq M_{p,t}$.

"$\supseteq$": Let $(I,M) \in postfp(F^{\#}_{\mathrm{OG}})$, i.e., for $(I',M') = F^{\#}_{\mathrm{OG}}(I,M)$ we have $(I',M') \sqsubseteq (I,M)$. We'll show that (I,M) is an Owicki-Gries-core proof.

Sequential consistency: Let $t \in Tid$, $(v,v') \in \rightarrow_t$, $p = v(pc_t)$, $p' = v'(pc_t)$, $R = rsc(p)$, $R' = rsc(p')$, $v|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(p))} \in M_{p,t}$ and $\forall\, r \in R' \setminus R$: $v|_{ProofVar(r)} \in I_r$. By definition of $next^{\#}_{\mathrm{OG}}$ we obtain $v'|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(p'))} \in M'_{p',t} \subseteq M_{p',t}$. If any $r \in R \setminus R'$ is given, the definition of $next^{\#}_{\mathrm{OG}}$ implies $v'|_{ProofVar(r)} \in I'_r \subseteq I_r$.

Initial condition: Let $v \in init$.

Let $r \in Res \setminus \bigcup rsc(v(PCVar))$. Then $v|_{ProofVar(r)} \in I^{init}_r \subseteq I'_r \subseteq I_r$.

Now let $t \in Tid$. Then $v|_{LocalDataVar_t} \in M^{init}_{v(pc_t),t}$. Since initial states don't have any busy resources, $v|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(v(pc_t)))} \in M^{init}_{v(pc_t),t} \subseteq M'_{v(pc_t),t} \subseteq M_{v(pc_t),t}$.

We have shown $v \in \gamma_{\mathrm{OG}}(I,M)$. Thus $init \subseteq \gamma_{\mathrm{OG}}(I,M)$. ■

Restatement of Theorem 5. The smallest Owicki-Gries-core proof is the least fixpoint of the Owicki-Gries-core abstract transformer, formally

$$\inf(\mathfrak{O}) = lfp(F^{\#}_{\mathrm{OG}})\,.$$

Proof. $\inf(\mathfrak{O}) =$ [Thm. 30] $\inf(\mathit{postfp}(F^{\#}_{\mathrm{OG}})) =$ [Tarski's fixpoint theorem] $\mathit{lfp}(F^{\#}_{\mathrm{OG}})$. ■

Proposition 31. *The map $\mathit{post}^{\#}_{\mathrm{OG}}$ is sound with respect to the concrete successor map post. Formally:*

$$\forall\, P \in PA\colon\ \mathit{post} \circ \gamma_{\mathrm{OG}}(P) \subseteq \gamma_{\mathrm{OG}} \circ \mathit{post}^{\#}_{\mathrm{OG}}(P)\,.$$

Proof. Let $(I,M) \in PA$. Let $v' \in \mathit{post}(\gamma_{\mathrm{OG}}(I,M))$. Let $(I',M') = \mathit{next}^{\#}_{\mathrm{OG}}(I,M)$ and $(\hat{I},\hat{M}) = \mathit{post}^{\#}_{\mathrm{OG}}(I,M)$. Then $(\hat{I},\hat{M}) = (I,M) \sqcup (I',M')$. By definition of *post* there is some $t \in \mathit{Tid}$ and some $v \in \gamma_{\mathrm{OG}}(I,M)$ such that $v \to_t v'$. By definition of γ_{OG} we have $v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(\mathit{rsc}(v(pc_t)))} \in M_{v(pc_t),t}$. To show that v' is in $\gamma_{\mathrm{OG}}(\hat{I},\hat{M})$ we have to prove the following two claims.

"$\forall\, r \in \mathit{Res} \setminus \bigcup \mathit{rsc}(v'(\mathit{PCVar}))\colon\ v'|_{\mathit{ProofVar}(r)} \in \hat{I}_r$": Let $r \in \mathit{Res} \setminus \bigcup \mathit{rsc}(v'(\mathit{PCVar}))$.

Case $r \in \mathit{rsc}(v(pc_{\tilde{t}}))$ for some $\tilde{t} \in \mathit{Tid}$. Then $r \in \mathit{rsc}(v(pc_{\tilde{t}})) \setminus \mathit{rsc}(v'(pc_{\tilde{t}}))$, so $v(pc_{\tilde{t}}) \neq v'(pc_{\tilde{t}})$. Then (1) implies that $\tilde{t} = t$. So $r \in \mathit{rsc}(v(pc_t)) \setminus \mathit{rsc}(v'(pc_t))$. Then $v'|_{\mathit{ProofVar}(r)} \in I'_r$ by definition of $\mathit{next}^{\#}_{\mathrm{OG}}$.

Case $r \notin \bigcup \mathit{rsc}(v(\mathit{PCVar}))$. The definition of γ_{OG} implies that $v|_{\mathit{ProofVar}(r)} \in I_r$. Notice that $r \notin \mathit{rsc}(v(pc_t)) \cap \mathit{rsc}(v'(pc_t))$. Thus for all $x \in \mathit{ProofVar}(r)$ we have $v(x) = v'(x)$. So $v \sim_{\mathit{ProofVar}(r)} v'$. So $v'|_{\mathit{ProofVar}(r)} \in I_r$.

In both cases $v'|_{\mathit{ProofVar}(r)} \in \hat{I}_r$.

"$\forall\, \tilde{t} \in \mathit{Tid}\colon\ v'|_{\mathit{LocalDataVar}_{\tilde{t}} \cup \bigcup \mathit{ProofVar}(\mathit{rsc}(v'(pc_{\tilde{t}})))} \in \hat{M}_{v'(pc_{\tilde{t}},\tilde{t})}$": Let $\tilde{t} \in \mathit{Tid}$.

Case $\tilde{t} = t$. Consider an arbitrary $r \in \mathit{rsc}(v'(pc_t)) \setminus \mathit{rsc}(v(pc_t))$ (there may be none). From $v' \in \mathit{ConsState}$ we get $r \notin \bigcup_{\hat{t} \in \mathit{Tid} \setminus \{t\}} \mathit{rsc}(v'(pc_{\hat{t}}))$, so (1) gives $r \notin \bigcup_{\hat{t} \in \mathit{Tid} \setminus \{t\}} \mathit{rsc}(v(pc_{\hat{t}}))$. Thus r is available in v. From $v \in \gamma_{\mathrm{OG}}(I,M)$ we get $v|_{\mathit{ProofVar}(r)} \in I_r$. Thus $\forall\, r \in \mathit{rsc}(v'(pc_t)) \setminus \mathit{rsc}(v(pc_t))\colon\ v|_{\mathit{ProofVar}(r)} \in I_r$. From the definition of $\mathit{next}^{\#}_{\mathrm{OG}}$ we get $M'_{v'(pc_t),t} \ni v'|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(\mathit{rsc}(v'(pc_t)))}$.

Case $\tilde{t} \neq t$. By (1) we have $v \sim_{\mathit{Local}_{\tilde{t}}} v'$, so $v(pc_{\tilde{t}}) = v'(pc_{\tilde{t}})$. Take any $r \in \mathit{rsc}(v'(pc_{\tilde{t}}))$ (there may be none). From $v' \in \mathit{ConsState}$ we get $r \notin \mathit{rsc}(v'(pc_t))$. In particular, $r \notin \mathit{rsc}(v(pc_t)) \cap \mathit{rsc}(v'(pc_t))$. By definition of *ProofVar*, for all $x \in \mathit{ProofVar}(r)$ we have $v(x) = v'(x)$. Thus $v \sim_{\mathit{ProofVar}(r)} v'$. So $v \sim_{\mathit{LocalDataVar}_{\tilde{t}} \cup \bigcup \mathit{ProofVar}(\mathit{rsc}(v'(pc_{\tilde{t}})))} v'$. Notice that $v|_{\mathit{LocalDataVar}_{\tilde{t}} \cup \bigcup \mathit{ProofVar}(\mathit{rsc}(v(pc_{\tilde{t}})))} \in M_{v(pc_{\tilde{t}}),\tilde{t}}$. Then we have $M_{v'(pc_{\tilde{t}}),\tilde{t}} \ni v'|_{\mathit{LocalDataVar}_{\tilde{t}} \cup \bigcup \mathit{ProofVar}(\mathit{rsc}(v'(pc_{\tilde{t}})))}$.

In both cases $v'|_{\mathit{LocalDataVar}_{\tilde{t}} \cup \bigcup \mathit{ProofVar}(\mathit{rsc}(v'(pc_{\tilde{t}})))} \in \hat{M}_{v'(pc_{\tilde{t}}),\tilde{t}}$.

Thus $v' \in \gamma_{\mathrm{OG}}(\hat{I},\hat{M})$. Since $v' \in \mathit{post} \circ \gamma_{\mathrm{OG}}(I,M)$ was arbitrary, we have shown that $\mathit{post} \circ \gamma_{\mathrm{OG}}(I,M) \subseteq \gamma_{\mathrm{OG}} \circ \mathit{post}^{\#}_{\mathrm{OG}}(I,M)$. ■

Remark 32. A reader might wonder why we have not taken the map $PA \to PA$, $(I,M) \mapsto (I^{init},M^{init}) \sqcup \mathit{next}^{\#}_{\mathrm{OG}}(I,M)$ as the abstract transformer. It is also monotone and the set of its postfix points is equal to the set of Owicki-Gries-core proofs. However, $\mathit{next}^{\#}_{\mathrm{OG}}$ is in general not a sound approximation of *post* under $(\alpha_{\mathrm{OG}},\gamma_{\mathrm{OG}})$. □

D Upper bound on precision

Within the current section D, fix a syntactic structure $(\mathit{Tid},\mathit{PL},\mathit{Val},\mathit{Res},(\mathit{Local}_t,pc_t)_{t \in \mathit{Tid}},\mathit{ProofVar},\mathit{rsc})$, let *ConsState* be the set of consistent states, $D = \mathfrak{P}(\mathit{ConsState})$ and $(PA,\sqsubseteq)$ the lattice of program annotations.

Let

$$\alpha_{\mathrm{OG}} : D \to PA\,,$$
$$S \mapsto ((\{v|_{\mathit{ProofVar}(r)} \;\mid\; v \in S \wedge r \notin \bigcup rsc(v(\mathit{PCVar}))\})_{r \in \mathit{Res}}\,,$$
$$(\{v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(p))} \;\mid\; v \in S \wedge p = v(pc_t)\})_{\substack{p \in PL,\\ t \in \mathit{Tid}}})\,.$$

A *pair of adjoint maps*[7] between two posets $(X,\leq)$, $(Y,\preceq)$ is a pair of maps (α,γ) such that $\alpha : X \to Y$, $\gamma : Y \to X$ and

$$\forall\, x \in X, y \in Y :\; \alpha(x) \preceq y \Leftrightarrow x \leq \gamma(Y)\,.$$

Proposition 33. $(\alpha_{\mathrm{OG}},\gamma_{\mathrm{OG}})$ *is a pair of adjoint maps between* $(D,\subseteq)$ *and* $(PA,\sqsubseteq)$.

Proof. Let $S \in D$, $\alpha_{\mathrm{OG}}(S) = (I,M)$, $T = (J,N) \in PA$. We will show that $\alpha_{\mathrm{OG}}(S) \sqsubseteq T$ if and only if $S \subseteq \gamma_{\mathrm{OG}}(T)$.

"only if": Let $v \in S$. By definition of D we get $v \in \mathit{ConsState}$. Let $r \in \mathit{Res} \setminus \bigcup rsc(v(\mathit{PCVar}))$. By definition of α_{OG} we get $v|_{\mathit{ProofVar}(r)} \in I_r$. By assumption, $I_r \subseteq J_r$, so $v|_{\mathit{ProofVar}(r)} \in J_r$.
Now let $t \in \mathit{Tid}$ and $p = v(pc_t)$. By definition of α_{OG} we get $v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(p))} \in M_{p,t}$. By assumption, $N_{p,t} \supseteq M_{p,t}$. Thus $N_{v(pc_t),t} \ni v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(v(pc_t)))}$.
Combining, we obtain $v \in \gamma_{\mathrm{OG}}(T)$.

"if": Let $r \in \mathit{Res}$. Let $u \in I_r$. By definition of α_{OG} there is $v \in S$ such that $r \notin \bigcup rsc(v(\mathit{PCVar}))$ and $u = v|_{\mathit{ProofVar}(r)}$. By assumption, $v \in \gamma_{\mathrm{OG}}(T)$. By definition of γ_{OG} we have $u \in J_r$. Thus $I_r \subseteq J_r$.
Let $t \in \mathit{Tid}$ and $p \in PL$. Let $u \in M_{p,t}$. By definition of α_{OG} there is some $v \in S$ such that $p = v(pc_t)$ and $v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(p))} = u$. By assumption, $v \in \gamma_{\mathrm{OG}}(T)$. By definition of γ_{OG} we get $u \in N_{p,t}$. Thus $M_{p,t} \subseteq N_{p,t}$.
Combining, we obtain $\alpha_{\mathrm{OG}}(S) \sqsubseteq T$. ∎

Let $\underline{\rho} = \gamma_{\mathrm{OG}} \circ \alpha_{\mathrm{OG}}$.
Then for $Q \in D$ we have $\underline{\rho}(Q) =$

$$\left\{ v \in \mathit{ConsState} \;\middle|\; \begin{array}{l} \forall\, r \in \mathit{Res} \setminus \bigcup rsc(v(\mathit{PCVar}))\, \exists\, \hat{v} \in Q : \begin{pmatrix} r \notin \bigcup rsc(\hat{v}(\mathit{PCVar})) \\ \wedge\; v \sim_{\mathit{ProofVar}(r)} \hat{v} \end{pmatrix} \\ \wedge \\ \forall\, t \in \mathit{Tid}\; \exists\, \tilde{v} \in Q :\; v \sim_{\mathit{Local}_t \cup \bigcup \mathit{ProofVar}(rsc(v(pc_t)))} \tilde{v} \end{array} \right\}.$$

A *closure* (sometimes called an *upper closure operator*) on a poset $(X,\leq)$ is a map $\rho : X \to X$ which is monotone, extensive and idempotent. (For an introduction to closures see [31].)

Proposition 34. $\underline{\rho} : D \to D$ *is a closure operator.*

[7] As of year 2011, it is also called a *(monotone) Galois connection*. The definition of a Galois connection is changing over time, see, e.g., § 6 in Chapter IV of [2], also § 8 in Chapter V of [3], for a more antique definition where the maps α and γ are antitone.

Proof. By a standard result (exercise for the reader, see also [9], Corollary 5.3.0.5), applying the right adjoint after the left adjoint gives a closure operator. ■

Restatement of Theorem 10. Abstract interpretation with $\underline{\rho}$ proves the strongest Owicki-Gries-provable property.
Formally, let $T = (init, \ldots)$ be an RPL transition system that obeys the given syntactic structure and has successor operator *post* and let $\mathfrak{O}$ be the lattice of Owicki-Gries-core proofs of T; then

$$lfp(\lambda x.\underline{\rho}(init \cup post(x))) \subseteq \gamma_{\mathrm{OG}}(\inf \mathfrak{O}).$$

Proof. Let $P = \inf \mathfrak{O}$. By Prop. 27 we have

$$init \cup post(\gamma_{\mathrm{OG}}(P)) \subseteq \gamma_{\mathrm{OG}}(P).$$

Since $(\alpha_{\mathrm{OG}}, \gamma_{\mathrm{OG}})$ is a pair of adjoint maps, we have

$$\alpha_{\mathrm{OG}}(init \cup post(\gamma_{\mathrm{OG}}(P))) \sqsubseteq P.$$

Right adjoints are monotone, so

$$\gamma_{\mathrm{OG}}(\alpha_{\mathrm{OG}}(init \cup post(\gamma_{\mathrm{OG}}(P)))) \subseteq \gamma_{\mathrm{OG}}(P),$$

otherwise stated

$$\underline{\rho}(init \cup post(\gamma_{\mathrm{OG}}(P))) \subseteq \gamma_{\mathrm{OG}}(P).$$

Let $F = \lambda x.\underline{\rho}(init \cup post(x))$, then $\gamma_{\mathrm{OG}}(P) \in postfp(F)$. Since the least fixpoint of a monotone map is the infimum of its postfix points, $lfp(F) \subseteq \gamma_{\mathrm{OG}}(P)$. Apply Corollary 26. ■

In particular, abstract interpretation with $\underline{\rho}$ allows proving at least as many properties as the core of the Owicki-Gries method.

Extension of Example 11. We will see later in Example 19 that $lfp(\lambda x.\underline{\rho}(init \cup post(x))) = \gamma_{\mathrm{OG}}(I,M)$. So, for Readers-Writers, the Owicki-Gries-core proof method and abstract interpretation with $\underline{\rho}$ are equally strong. □

Extension of Example 12. Notice that $\underline{\rho}(init \cup post(init)) = \underline{\rho}(init \cup \emptyset) = \underline{\rho}(init) \subseteq init$. Since *init* is the set of reachable states, $lfp(\lambda x.\underline{\rho}(init \cup post(x))) = init$. Thus for the RPL transition system Upper abstract interpretation under $\underline{\rho}$ can prove strictly stronger properties than the core of the Owicki-Gries method. □

Extension of Example 13. For the programs of the class Simple we will show later in Example 22 that $lfp(\lambda x.\underline{\rho}(init \cup post(x))) = \gamma_{\mathrm{OG}}(\inf \mathfrak{O})$.

Extension of Example 14. For the RPL transition systems in the class SepThreads we will show later that the abstract interpretation using the lower bound produces exactly the set of reachable states. Thus abstract interpretation using the upper bound $\underline{\rho}$ will also produce the set of reachable states. □

E Lower bound on precision

For a set X of sets, let

$$Part(X) = \{Y \mid \exists\, T \subseteq X:\ Y = \big(\bigcap T\big) \cap \big(\bigcap_{Z \in X \setminus T} ((\bigcup X) \setminus Z)\big) \neq \emptyset\}$$

be the *partition* of X, where $\bigcap \emptyset = \bigcup X$ by convention. The elements of a partition are called *blocks*.

Lemma 35. *Let X be a set of sets. Then:*
1. *$\bigcup Part(X) = \bigcup X$;*
2. *for all different $s_1, s_2 \in Part(X)$ we have $s_1 \cap s_2 = \emptyset$;*
3. *if $s \in Part(X)$ and $S \in X$, then either $s \cap S = \emptyset$ or $s \subseteq S$;*
4. *if X is finite, so is $Part(X)$.*

Proof. 1. "$\subseteq$": Intersection of subsets of $\bigcup X$ produces a subset of $\bigcup X$.
"$\supseteq$": Let $x \in \bigcup X$. Let $T = \{S \in X \mid x \in S\}$. Then for all $Z \in X \setminus T$ we have $x \notin Z$. So $x \in \big(\bigcap T\big) \cap \big(\bigcap_{Z \in X\setminus T} ((\bigcup X) \setminus Z)\big)$.

2. Let $T_1, T_2 \subseteq X$ be such that for $s_1 = \big(\bigcap T_1\big) \cap \big(\bigcap_{Z \in X \setminus T_1} ((\bigcup X) \setminus Z)\big) \neq \emptyset$ and $s_2 = \big(\bigcap T_2\big) \cap \big(\bigcap_{Z \in X \setminus T_2} ((\bigcup X) \setminus Z)\big) \neq \emptyset$ we have $s_1 \neq s_2$. Assume for the purpose of contradiction that there is some y in $s_1 \cap s_2$. Since $s_1 \neq s_2$, there is some x in $(s_1 \setminus s_2) \cup (s_2 \setminus s_1)$. Without loss of generality let $x \in s_1 \setminus s_2$, the other case is symmetrical. Then $x \in (\bigcup X) \setminus s_2$, so there are two cases.
   - Case $\exists\, S \in T_2 : x \notin S$. Take such S. Then $S \notin T_1$, thus $S \in X \setminus T_1$, so $y \in (\bigcup X) \setminus S$. Since $S \in T_2$, we have $y \in S$. Contradiction!
   - Case $\exists\, Z \in X \setminus T_2 : x \notin (\bigcup X) \setminus Z$. Then $Z \notin X \setminus T_1$ (otherwise x were not in s_1). So $Z \in T_1$. So $y \in Z$. But $y \in s_2$, so $y \in (\bigcup X) \setminus Z$. Contradiction!
3. Let $s \in Part(X)$ and $S \in X$. Assume for the purpose of contradiction that $s \cap S$ contains some element, say, x, but $s \setminus S$ also contains some element, say, y. There is some $T \subseteq X$ such that $s = \big(\bigcap T\big) \cap \big(\bigcap_{Z \in X \setminus T} ((\bigcup X) \setminus Z)\big)$. So $y \in \bigcap T$. If $S \in T$, we get $y \notin \bigcap T$, a contradiction. If $S \notin T$, then $S \in X \setminus T$, so $x \in s \subseteq (\bigcup X) \setminus S$ implies $x \notin S$, also a contradiction.
4. If X is finite, $|Part(X)| \leq 2^{|X|}$. ∎

For the rest of section E, fix a syntactic structure $(Tid, PL, Val, Res, (Local_t, pc_t)_{t \in Tid}, ProofVar, rsc)$, let $ConsState$ be the set of consistent states and $D = \mathfrak{P}(ConsState)$. Let

$$RL = Res \cup \{Local_t \mid t \in Tid\} \quad \text{and} \quad \widetilde{RL} = Part(RL)\,.$$

Let

$$\bar{\rho} : D \to D, \quad Q \mapsto$$
$$\left\{ v \in ConsState \;\middle|\; \begin{array}{l} \left(\begin{array}{l} \forall\, s \in \widetilde{RL}, r \in Res:\ s \subseteq r \Rightarrow \\ \left(\begin{array}{l} \big(r \notin \bigcup rsc(v(PCVar)) \Rightarrow \exists \hat{v} \in Q:\ v \sim_s \hat{v} \wedge r \notin \bigcup rsc(\hat{v}(PCVar))\big) \\ \wedge \\ \big(\forall t \in Tid : r \in rsc(v(pc_t)) \Rightarrow \exists \check{v} \in Q:\ v \sim_{(Local_t \setminus \bigcup Res) \cup s} \check{v}\big) \end{array}\right) \end{array}\right) \\ \wedge \\ \big(\forall\, t \in Tid\ \exists\, \tilde{v} \in Q:\ v \sim_{Local_t \setminus \bigcup Res} \tilde{v}\big) \end{array} \right\},$$

and
$\rho_c : D \to D, \quad Q \mapsto \{v \in \mathit{ConsState} \mid \forall\, s \in \widetilde{RL}\ \exists\, \bar{v} \in Q : v \sim_s \bar{v}\}.$

Note that neither $\bar{\rho}$ nor ρ_c depend on the map *ProofVar*. Thus for syntactic structures that differ only in the map *ProofVar* we have the same $\bar{\rho}$ and the same ρ_c.

Proposition 36. *$\bar{\rho}$ is a closure operator on $(D, \subseteq)$.*

Proof. Monotonicity. Let $Q \subseteq Q' \subseteq \mathit{ConsState}$. Let $v \in \bar{\rho}(Q)$.

- Let $s \in \widetilde{RL}$, $r \in \mathit{Res}$, $s \subseteq r$.
  - ∗ Assume for a moment that r is available in v. Then there is $\hat{v} \in Q$ such that $v \sim_s \hat{v}$ and r is available in $\hat{v}$. Notice that $\hat{v} \in Q'$.
  - ∗ Let $t \in \mathit{Tid}$ and $r \in \mathit{rsc}(v(pc_t))$. Then there is $\check{v} \in Q$ such that $v \sim_{(\mathit{Local}_t \setminus \bigcup \mathit{Res}) \cup s}$ $\check{v}$. Notice that $\check{v} \in Q'$.
- Let $t \in \mathit{Tid}$. Then there is $\tilde{v} \in Q$ such that $v \sim_{\mathit{Local}_t \setminus \bigcup \mathit{Res}} \tilde{v}$. Notice that $\tilde{v} \in Q'$.

We have shown that $v \in \bar{\rho}(Q')$. Thus $\bar{\rho}(Q) \subseteq \bar{\rho}(Q')$.

Extensivity. Let $Q \in D$. Let $v \in Q$.

- Let $s \in \widetilde{RL}$, $r \in \mathit{Res}$, $s \subseteq r$.
  - ∗ Assume for the moment that $r \notin \bigcup \mathit{rsc}(v(\mathit{PCVar}))$. Take $\hat{v} := v$.
  - ∗ Let $t \in \mathit{Tid}$ and $r \in \mathit{rsc}(v(pc_t))$. Take $\check{v} := v$.
- Let $t \in \mathit{Tid}$. Take $\tilde{v} := v$.

Thus $v \in \bar{\rho}(Q)$. We have shown that $Q \subseteq \bar{\rho}(Q)$.

Idempotence. Let $Q \in D$. Let $v \in \bar{\rho}(\bar{\rho}(Q))$. We'll show that v is in $\bar{\rho}(Q)$.

- Let $s \in \widetilde{RL}$, $r \in \mathit{Res}$, $s \subseteq r$.
  - ∗ Assume for a moment that r is available in v. Then there is $\hat{v} \in \bar{\rho}(Q)$ such that $v \sim_s \hat{v}$ and r is available in $\hat{v}$. Then there is $\hat{\hat{v}} \in Q$ such that $\hat{v} \sim_s \hat{\hat{v}}$ and r is available in $\hat{\hat{v}}$. Notice that $v \sim_s \hat{\hat{v}}$.
  - ∗ Let $t \in \mathit{Tid}$ such that $r \in \mathit{rsc}(v(pc_t))$. Then some $\check{v} \in \bar{\rho}(Q)$ exists such that $v \sim_{(\mathit{Local}_t \setminus \bigcup \mathit{Res}) \cup s} \check{v}$. Then $v(pc_t) = \check{v}(pc_t)$, so $r \in \mathit{rsc}(\check{v}(pc_t))$. Then there is $\check{\check{v}} \in Q$ such that $\check{v} \sim_{(\mathit{Local}_t \setminus \bigcup \mathit{Res}) \cup s} \check{\check{v}}$. Notice that $v \sim_{(\mathit{Local}_t \setminus \bigcup \mathit{Res}) \cup s} \check{\check{v}}$.
- Let $t \in \mathit{Tid}$. Then there is $\tilde{v} \in \bar{\rho}(Q)$ such that $v \sim_{\mathit{Local}_t \setminus \bigcup \mathit{Res}} \tilde{v}$. Then there is $\tilde{\tilde{v}} \in Q$ such that $\tilde{v} \sim_{\mathit{Local}_t \setminus \bigcup \mathit{Res}} \tilde{\tilde{v}}$. Notice that $v \sim_{\mathit{Local}_t \setminus \bigcup \mathit{Res}} \tilde{\tilde{v}}$.

Thus $v \in \bar{\rho}(Q)$. We have shown that $\bar{\rho}(\bar{\rho}(Q)) \subseteq \bar{\rho}(Q)$. The opposite inclusion follows from extensivity. ■

Proposition 37. *ρ_c is a closure operator on $(D, \subseteq)$*

Proof. We will show the three properties of a closure operator.

– Monotonicity. Let $Q \subseteq Q'$ be sets of consistent states and $v \in \rho_c(Q)$. Then $v \in \mathit{ConsState}$ and for each block $s \in \widetilde{RL}$ there is a state $\bar{v} \in Q$ such that $v \sim_s \bar{v}$. Notice that such $\bar{v}$ are also in Q'. We have shown $\rho_c(Q) \subseteq \rho_c(Q')$.

– Extensivity. Let $Q \in D$. Let $v \in Q$. Let $s \in \widetilde{RL}$. Certainly $v \sim_s v$. Thus $v \in \rho_c(Q)$. We have shown $Q \subseteq \rho_c(Q)$.

– Idempotence. Let $Q \in D$. Let $v \in \rho_c(\rho_c(Q))$. Let $s \in \widetilde{RL}$. Then there is some $\bar{v} \in \rho_c(Q)$ such that $v \sim_s \bar{v}$. Then there is some $\bar{\bar{v}} \in Q$ such that $\bar{v} \sim_s \bar{\bar{v}}$. Then $v \sim_s \bar{\bar{v}}$. Thus $v \in \rho_c(Q)$. We have shown $\rho_c(\rho_c(Q)) \subseteq \rho_c(Q)$. The opposite inclusion follows from extensivity. ■

Proposition 38. *For all $Q \in D$ we have $\bar{\rho}(Q) \subseteq \rho_c(Q)$.*

Proof. Let $Q \in D$. Let $v \in \bar{\rho}(Q)$. Let $s \in \widetilde{RL}$. By part 3 of Lemma 35 we have two cases.
Case there is $r \in Res$ such that $s \subseteq r$. There are two subcases.

Case r is available in v. Then there is some $\hat{v} \in Q$ such that $v \sim_s \hat{v}$. Let $\bar{v} := \hat{v}$.

Case $r \in rsc(v(pc_t))$ for some $t \in Tid$. Then there is $\check{v} \in Q$ such that $v \sim_{(Local_t \setminus \bigcup Res) \cup s} \check{v}$. Let $\bar{v} := \check{v}$.

Case $s \cap \bigcup Res = \emptyset$. Since $s \neq \emptyset$, there is some $t \in Tid$ such that $s \subseteq Local_t \setminus \bigcup Res$. There is $\tilde{v} \in Q$ such that $v \sim_{Local_t \setminus \bigcup Res} \tilde{v}$. Let $\bar{v} := \tilde{v}$.

In any case there is $\bar{v} \in Q$ such that $v \sim_s \bar{v}$. ■

For the purpose of the proof let $T = (init, (\rightarrow_t)_{t \in Tid})$ be an RPL transition system that obeys the given syntactic structure and has successor operator *post*. Let $F = \lambda x \in D.\, \bar{\rho}(init \cup post(x))$, let $\mu \in \mathrm{Ord}$ have cardinality greater than the cardinality of D and let $(Y^{(i)})_{i \in \mu}$ be the μ-termed upper iteration sequence for F starting with $\emptyset \in D$:

$$Y^{(0)} = \emptyset, \quad Y^{(\sigma+1)} = F\left(Y^{(\sigma)}\right) \text{ for } \sigma + 1 \in \mu, \quad Y^{(\omega\sigma)} = \bigcup_{\bar{\sigma} \in \omega\sigma} Y^{(\bar{\sigma})} \text{ for } \omega\sigma \in \mu.$$

Let $Q = lfp(F)$. The sequence $(Y^{(i)})_{i \in \mu}$ is monotonously increasing, and its limit is Q.

Lemma 39. *Let $t \neq \tilde{t}$ in Tid. Then*

1. $\forall\, \tau \in Tid, (v,v') \in \rightarrow_\tau\colon\ v \sim_{Local_t \cap Local_{\tilde{t}}} v' \ \wedge\ v \sim_{\bigcap_{\hat{t} \in Tid} Local_{\hat{t}}} v'$;
2. $Local_t \cap Local_{\tilde{t}} = \bigcap_{\hat{t} \in Tid} Local_{\hat{t}}$.

Proof. 1. Let $\tau \in Tid$, $v \rightarrow_\tau v'$. Then $\tau \neq t$ or $\tau \neq \tilde{t}$. From (1) we get $v \sim_{Local_t} v'$ or $v \sim_{Local_{\tilde{t}}} v'$. In any case $v \sim_{Local_t \cap Local_{\tilde{t}}} v'$. In particular, $v \sim_{\bigcap_{\hat{t} \in Tid} Local_{\hat{t}}} v'$.

2. "$\subseteq$": Let $x \in Local_t \cap Local_{\tilde{t}}$ and $\hat{t} \in Tid$. To show that x is in $Local_{\hat{t}}$, let $\tau \in Tid \setminus \{\hat{t}\}$ and $(v,v') \in \rightarrow_\tau$. As shown above, $v \sim_{Local_t \cap Local_{\tilde{t}}} v'$, so $v(x) = v'(x)$. By definition of RPL, $x \in Local_{\hat{t}}$.
"$\supseteq$": Follows from $Local_t \supseteq \bigcap_{\hat{t} \in Tid} Local_{\hat{t}}$ and $Local_{\tilde{t}} \supseteq \bigcap_{\hat{t} \in Tid} Local_{\hat{t}}$. ■

Lemma 40. *Let $s \in \widetilde{RL}$, $t \neq \tilde{t}$ in Tid such that $s \subseteq Local_t \cap Local_{\tilde{t}} \cap \bigcup Res$. Then $\forall\, \sigma \in \mu,\ v \in Y^{(\sigma)}\ \exists\, w \in init\colon\ v \sim_s w$.*

Proof. By transfinite induction. Let $\sigma \in \mu$ and $\forall\, \bar{\sigma} \in \sigma\ \forall\, v \in Y^{(\bar{\sigma})}\ \exists\, w \in init\colon v \sim_s w$. Let $v \in Y^{(\sigma)}$. Then $\sigma > 0$. There are two cases.
Case $\sigma = \bar{\sigma} + 1$ for some $\bar{\sigma} \in \mu$. There is some $r \in Res$ such that $s \subseteq r$. There are two cases.

Case $r \in rsc(v(pc_t))$ for some $t \in Tid$. By definition of $\bar{\rho}$ there is $\check{v} \in init \cup post(Y^{(\bar{\sigma})})$ such that $v \sim_{(Local_t \setminus \bigcup Res) \cup s} \check{v}$. In particular, $v \sim_s \check{v}$ and $v(pc_t) = \check{v}(pc_t)$. Thus $r \in rsc(\check{v}(pc_t))$. Since in initial states all resources are available, $\check{v} \notin init$. So there is some $z \in Y^{(\bar{\sigma})}$ such that $\check{v} \in post(\{z\})$. Part 1 of Lemma 39 implies $\check{v} \sim_s z$, so $v \sim_s z$.

Case $r \notin \bigcup rsc(v(PCVar))$. Then there is $\hat{v} \in init \cup post(Y^{(\bar{\sigma})})$ such that $v \sim_s \hat{v}$. If $\hat{v} \in init$, take $w := \hat{v}$. Otherwise there is $z \in Y^{(\bar{\sigma})}$ such that $\hat{v} \in post(\{z\})$. Part 1 of Lemma 39 implies $\hat{v} \sim_s z$, so $v \sim_s z$.

If we have not found a suitable w so far, there is $z \in Y^{(\bar{\sigma})}$ such that $v \sim_s z$. Applying the induction hypothesis to $\bar{\sigma} < \sigma$ and $z \in Y^{(\bar{\sigma})}$ we get some $w \in init$ such that $z \sim_s w$, which implies $v \sim_s w$.

Case σ is a limit ordinal. Then there is $\bar{\sigma} \in \sigma$ such that $v \in Y^{(\bar{\sigma})}$. Apply the induction hypothesis. ■

Lemma 41. *Let $t \neq \tilde{t}$ in Tid. Then $\forall\, \sigma \in \mu,\, v \in Y^{(\sigma)}\ \exists\, w \in init\colon\ v \sim_{(Local_t \cap Local_{\tilde{t}}) \setminus \bigcup Res} w$.*

Proof. Let $C = (Local_t \cap Local_{\tilde{t}}) \setminus \bigcup Res$. We'll use transfinite induction. Let $\sigma \in \mu$ and $\forall\, \bar{\sigma} \in \sigma\ \forall\, v \in Y^{(\bar{\sigma})}\ \exists\, w \in init\colon\ v \sim_C w$. Let $v \in Y^{(\sigma)}$. Then $\sigma > 0$.

Case $\sigma = \bar{\sigma} + 1$ for some $\bar{\sigma} \in \mu$. By definition of $Y^{(\bar{\sigma}+1)}$ there is $\tilde{v} \in init \cup post(Y^{(\bar{\sigma})})$ such that $v \sim_{Local_t \setminus \bigcup Res} \tilde{v}$. Then $v \sim_C \tilde{v}$. If $\tilde{v} \in init$, let $w := \tilde{v}$. Otherwise there is $z \in Y^{(\bar{\sigma})}$ such that $\tilde{v} \in post(\{z\})$. Part 1 of Lemma 39 implies that $\tilde{v} \sim_C z$, so $v \sim_C z$. Applying the induction hypothesis to $\bar{\sigma}$ and z we obtain some $w \in init$ such that $z \sim_C w$, so $v \sim_C w$.

Case σ is a limit ordinal. Then there is $\bar{\sigma} \in \sigma$ such that $v \in Y^{(\bar{\sigma})}$. Apply the induction hypothesis. ■

Lemma 42. *Let $s \in \widetilde{RL}$, $t \in Tid$, $r \in Res$, $s \subseteq r \cap Local_t$, and $\forall\, \tilde{t} \in Tid \setminus \{t\}\colon\ s \cap Local_{\tilde{t}} = \emptyset$. Then $\forall\, \sigma \in \mu,\, v \in Y^{(\sigma)}\colon\ r \notin rsc(v(pc_t))\ \Rightarrow\ \exists\, w \in Y^{(\sigma)}\colon v \sim_s w \wedge r$ is available in w.*

Proof. We'll use transfinite induction. Let $\sigma \in \mu$ and $\forall\, \bar{\sigma} \in \sigma\ \forall\, v \in Y^{(\bar{\sigma})}\colon\ r \notin rsc(v(pc_t)) \Rightarrow \exists\, w \in Y^{(\bar{\sigma})}\colon v \sim_s w \wedge r$ is available in w. Let $v \in Y^{(\sigma)}$ such that $r \notin rsc(v(pc_t))$. Then $\sigma > 0$.

Case $\sigma = \bar{\sigma} + 1$ for some $\bar{\sigma} \in \mu$. If r is available in v, take $w := v$. Otherwise there is some $\bar{t} \in Tid \setminus \{t\}$ such that $r \in rsc(v(pc_{\bar{t}}))$. By definition of $\bar{\rho}$ there is $y \in init \cup post(Y^{(\bar{\sigma})})$ such that $v \sim_{(Local_{\bar{t}} \setminus \bigcup Res) \cup s} y$. Then $v(pc_{\bar{t}}) = y(pc_{\bar{t}})$ and $v \sim_s y$. In particular, $r \in rsc(y(pc_{\bar{t}}))$. Then $y \notin init$. So there is $z \in Y^{(\bar{\sigma})}$, $\tau \in Tid$ such that $z \to_\tau y$. Remark that $r \notin rsc(y(pc_t))$.

Case $\tau = t$. From $\tau \neq \bar{t}$ and (1) we obtain $rsc(z(pc_{\bar{t}})) = rsc(y(pc_{\bar{t}}))$, thus $r \in rsc(z(pc_{\bar{t}}))$. From $z \in ConsState$ we get $r \notin rsc(z(pc_t))$. Thus $r \in Res \setminus (rsc(y(pc_t)) \cup rsc(z(pc_t)))$. From (2) we get $z \sim_r y$, so $z \sim_s y$, so $z \sim_s v$. Apply the induction hypothesis to $\bar{\sigma}$ and z.

Case $\tau \neq t$. From (1) we obtain $z \sim_{Local_t} y$, so $z \sim_s y$, so $z \sim_s v$. Also $z(pc_t) = y(pc_t)$, thus $r \notin rsc(z(pc_t))$. Apply the induction hypothesis to $\bar{\sigma}$ and z.

Case σ is a limit ordinal. Then there is $\bar{\sigma} \in \sigma$ such that $v \in Y^{\bar{\sigma}}$. Apply the induction hypothesis to $\bar{\sigma}$ and v. ■

Lemma 43. *Let $t \in Tid$, $v \in Q$, $r \in Res \setminus rsc(v(pc_t))$, $s \in \widetilde{RL}$, $s \subseteq r \cap Local_t$. Then $\exists\, w \in Q\colon\ v \sim_s w \wedge r$ is available in w.*

Proof. There is some $\sigma \in \mu$ such that $v \in Y^{(\sigma)}$.

Case there is $\tilde{t} \in Tid \setminus \{t\}$ such that $s \subseteq Local_{\tilde{t}}$. By Lemma 40 there is some $w \in init \subseteq Q$ such that $v \sim_s w$. Notice that all resources are available in w.

Case $\forall \tilde{t} \in Tid \setminus \{t\}\colon s \cap Local_{\tilde{t}} = \emptyset$. Lemma 42 provides a suitable $w \in Y^{(\sigma)} \subseteq Q$.

■

Lemma 44. *Let $t \in Tid$, $v \in Q$, $r \in Res \setminus rsc(v(pc_t))$. Then there is $w \in Q$ such that r is available in w and $w \sim_{Local_t \cup \bigcup rsc(w(pc_t))} v$.*

Proof. If r is available in w, take $w := v$. Otherwise there is $\bar{t} \in Tid \setminus \{t\}$ such that $r \in rsc(v(pc_{\bar{t}}))$. Lemma 41 implies the existence of some $u \in init$ such that $u \sim_{(Local_t \cap Local_{\bar{t}}) \setminus \bigcup Res}$ v. Let

$$w : Var \to Val\,, \quad x \mapsto \begin{cases} u(x)\,, & \text{if } x \in (Local_{\bar{t}} \setminus \bigcup Res) \cup ((\bigcup rsc(v(pc_{\bar{t}}))) \setminus Local_t)\,, \\ v(x)\,, & \text{otherwise.} \end{cases}$$

We'll show:

"$rsc(v(pc_{\bar{t}})) \cap \bigcup rsc(w(PCVar)) = \emptyset$": $rsc(v(pc_{\bar{t}})) \cap \bigcup rsc(w(PCVar)) = rsc(v(pc_{\bar{t}})) \cap (rsc(w(pc_{\bar{t}})) \cup \bigcup_{\hat{t} \in Tid \setminus \{\bar{t}\}} rsc(w(pc_{\hat{t}}))) = rsc(v(pc_{\bar{t}})) \cap (\emptyset \cup \bigcup_{\hat{t} \in Tid \setminus \{\bar{t}\}} rsc(v(pc_{\hat{t}}))) =$ [since $v \in ConsState$] $\emptyset$.

"r is available in w": Follows from $r \in rsc(v(pc_{\bar{t}}))$.

"$w \sim_{Local_t \cup \bigcup rsc(w(pc_t))} v$": Let $x \in Local_t \cup \bigcup rsc(w(pc_t))$. There are two cases.

Case $x \in (Local_{\bar{t}} \setminus \bigcup Res) \cup ((\bigcup rsc(v(pc_{\bar{t}}))) \setminus Local_t)$. Then $w(x) = u(x)$.

Case $x \in Local_{\bar{t}} \setminus \bigcup Res$. Then $x \in (Local_t \cup \bigcup rsc(w(pc_t))) \cap (Local_{\bar{t}} \setminus \bigcup Res) = (Local_t \cap Local_{\bar{t}}) \setminus \bigcup Res$. Then $u(x) = v(x)$, so $w(x) = v(x)$.

Case $x \in (\bigcup rsc(v(pc_{\bar{t}}))) \setminus Local_t$. Then there is $\tilde{r} \in rsc(v(pc_{\bar{t}}))$ such that $x \in \tilde{r}$. By the already proven, $\tilde{r} \notin \bigcup rsc(w(PCVar))$, so $x \notin \bigcup rsc(w(pc_t))$. Also $x \notin Local_t$. Thus $x \notin Local_t \cup \bigcup rsc(w(pc_t))$, a contradiction.

Case $x \notin (Local_{\bar{t}} \setminus \bigcup Res) \cup ((\bigcup rsc(v(pc_{\bar{t}}))) \setminus Local_t)$. Then $w(x) = v(x)$.

"$w \in Q$": We'll show first that w is in $\bar{\rho}(Q)$ as follows.

Initially, we'll show that w is a consistent state.

For all $\hat{t} \in Tid$ we have $w(pc_{\hat{t}}) \in \{u(pc_{\hat{t}}), v(pc_{\hat{t}})\} \subseteq PL$ and $rsc(u(pc_{\hat{t}})) = \emptyset \subseteq rsc(v(pc_{\hat{t}}))$. Thus for $t1 \neq t2$ in Tid we have $rsc(w(pc_{t1})) \cap rsc(w(pc_{t2})) \subseteq rsc(v(pc_{t1})) \cap rsc(v(pc_{t2})) = \emptyset$. Thus $w \in ConsState$.

Next, let $s \in \widetilde{RL}$ and $\tilde{r} \in Res$ such that $s \subseteq \tilde{r}$. We will show that
" $(\tilde{r} \notin \bigcup rsc(w(PCVar)) \Rightarrow \exists \hat{w} \in Q\colon w \sim_s \hat{w} \wedge \tilde{r} \notin \bigcup rsc(\hat{w}(PCVar)))$
$\wedge \left(\forall \tilde{t} \in Tid\colon \tilde{r} \in rsc(w(pc_{\tilde{t}})) \Rightarrow \exists \check{w} \in Q\colon w \sim_{(Local_{\tilde{t}} \setminus \bigcup Res) \cup s} \check{w}\right)$".

Notice that $s \cap (Local_t \setminus \bigcup Res) = \emptyset$.

- Assume for a moment that $\tilde{r} \notin \bigcup rsc(w(PCVar))$. We distinguish two cases.

  Case $\tilde{r} \in rsc(v(pc_{\bar{t}}))$. By part 3 of Lemma 35 one of the following cases holds.

  Case $s \subseteq Local_t$. Then $s \cap ((\bigcup rsc(v(pc_{\bar{t}}))) \setminus Local_t) = \emptyset$, so $w \sim_s v$. From $v \in ConsState$ we get $\tilde{r} \notin rsc(v(pc_t))$. By Lemma 43 there is $\hat{w} \in Q$ such that $v \sim_s \hat{w}$ and $\tilde{r}$ is available in $\hat{w}$. Then $w \sim_s \hat{w}$.

  Case $s \cap Local_t = \emptyset$. Then $w \sim_s u$. Take $\hat{w} := u$.

  Case $\tilde{r} \notin rsc(v(pc_{\bar{t}}))$. Then $w \sim_s v$. From $\tilde{r} \notin \bigcup_{\hat{t} \in Tid \setminus \{\bar{t}\}} rsc(w(pc_{\hat{t}})) = \bigcup_{\hat{t} \in Tid \setminus \{\bar{t}\}} rsc(v(pc_{\hat{t}}))$ we get that $\tilde{r}$ is available in v. Take $\hat{w} := v$.

- Let $\tilde{t} \in Tid$ such that $\tilde{r} \in rsc(w(pc_{\tilde{t}}))$. From $\tilde{r} \notin \emptyset = rsc(w(pc_{\bar{t}}))$ we get $\tilde{t} \neq \bar{t}$, so $\tilde{r} \in rsc(v(pc_{\tilde{t}}))$. From consistency of v we get $\tilde{r} \notin rsc(v(pc_{\bar{t}}))$. Thus $s \cap \bigcup rsc(v(pc_{\bar{t}})) = \emptyset$. So $s \cap ((Local_{\bar{t}} \setminus \bigcup Res) \cup ((\bigcup rsc(v(pc_{\bar{t}}))) \setminus Local_t)) = \emptyset$, thus $w \sim_s v$. Now let $x \in Local_{\tilde{t}} \setminus \bigcup Res$. Then $x \notin (\bigcup rsc(v(pc_{\bar{t}}))) \setminus Local_t$.

  Case $x \in Local_{\bar{t}} \setminus \bigcup Res$. Then $x \in (Local_{\tilde{t}} \cap Local_{\bar{t}}) \setminus \bigcup Res =$ [part 2 of Lemma 39] $(\bigcap_{\hat{t} \in Tid} Local_{\hat{t}}) \setminus \bigcup Res =$ [part 2 of Lemma 39] $(Local_t \cap Local_{\bar{t}}) \setminus \bigcup Res$. So $w(x) =$ [definition of w] $u(x) =$ [defining property of u] $v(x)$.

  Case $x \notin Local_{\bar{t}} \setminus \bigcup Res$. Then $w(x) = v(x)$ by definition of w.

  In total, $w \sim_{(Local_{\tilde{t}} \setminus \bigcup Res) \cup s} v$. Take $\check{w} := v$.

At last, we'll prove that "$\forall\,\tilde{t}\in Tid\;\exists\,\tilde{w}\in Q:\; w\sim_{Local_{\tilde{t}}\setminus\bigcup Res}\tilde{w}$".

Let $\tilde{t}\in Tid$. We distinguish two cases.

Case $\tilde{t}=\bar{t}$. Then $w\sim_{Local_{\bar{t}}\setminus\bigcup Res} u$. Take $\tilde{w}:=u$.

Case $\tilde{t}\neq\bar{t}$. Notice that $Local_{\tilde{t}}\setminus\bigcup Res=((Local_{\tilde{t}}\cap Local_{\bar{t}})\setminus\bigcup Res)\cup(Local_{\tilde{t}}\setminus(Local_{\bar{t}}\cup\bigcup Res))$. We have $w\sim_{(Local_{\tilde{t}}\cap Local_{\bar{t}})\setminus\bigcup Res} u\sim_{(Local_{t}\cap Local_{\bar{t}})\setminus\bigcup Res} v$. Part 2 of Lemma 39 implies that $Local_{\tilde{t}}\cap Local_{\bar{t}}=$ [since $\tilde{t}\neq\bar{t}$] $\bigcap_{\hat{t}\in Tid} Local_{\hat{t}}=$ [since $t\neq\bar{t}$] $Local_{t}\cap Local_{\bar{t}}$. Therefore, $w\sim_{(Local_{\tilde{t}}\cap Local_{\bar{t}})\setminus\bigcup Res} v$. Also $w\sim_{Local_{\tilde{t}}\setminus(Local_{\bar{t}}\cup\bigcup Res)} v$. Take $\tilde{w}:=v$.

We have shown $w\in\bar{\rho}(Q)$. From $\bar{\rho}(Q)=Q$ we obtain $w\in Q$. ■

Restatement of Lemma 17. Let $t\in Tid$, $(v,v')\in\rightarrow_t$, $R=rsc(v(pc_t))$, $R'=rsc(v'(pc_t))$, $r\in R'\setminus R$, $\hat{v},\tilde{v}\in Q$, r is available in $\hat{v}$, $v\sim_{Local_t\cup\bigcup R}\tilde{v}$, $v\sim_r\hat{v}$. Then there is $w'\in Q$ such that $w'\sim_{Local_t\cup\bigcup R'} v'$.

Proof. Notice that $r\notin rsc(v(pc_t))=rsc(\tilde{v}(pc_t))$. Applying Lemma 44 to $\tilde{v}$ we obtain some $\check{v}\in Q$ such that r is available in $\check{v}$ and $\check{v}\sim_{Local_t\cup\bigcup rsc(\check{v}(pc_t))}\tilde{v}$. Let

$$w: Var\rightarrow Val,\quad x\mapsto\begin{cases}\hat{v}(x), & \text{if } x\in r,\\ \check{v}(x), & \text{if } x\notin r.\end{cases}$$

We claim:

"r is available in w": Follows from $rsc(w(PCVar))=rsc(\check{v}(PCVar))$.

"$w\sim_{Local_t\cup\bigcup R'} v$": By (5) we have $Local_t\cup\bigcup R'=(Local_t\setminus r)\cup r\cup\bigcup R$. Then $w\sim_{(Local_t\setminus r)\cup\bigcup R}\check{v}\sim_{(Local_t\setminus r)\cup\bigcup R}\tilde{v}\sim_{(Local_t\setminus r)\cup\bigcup R} v$ and $w\sim_r\hat{v}\sim_r v$.

"$w\in Q$": First we'll show that w is in $\bar{\rho}(Q)$.

Initially, we'll show that w is consistent. For all $\hat{t}\in Tid$ we have $w(pc_{\hat{t}})=\check{v}(pc_{\hat{t}})\in PL$. So for $t_1\neq t_2$ in Tid we have $rsc(w(pc_{t_1}))\cap rsc(w(pc_{t_2}))=rsc(\check{v}(pc_{t_1}))\cap rsc(\check{v}(pc_{t_2}))$ $=$ [since $\check{v}\in ConsState$] $\emptyset$.

Next, let $s\in\widetilde{RL}$, $\tilde{r}\in Res$, $s\subseteq\tilde{r}$. We have to show two statements.

"$\tilde{r}\notin\bigcup rsc(w(PCVar))\Rightarrow\exists\hat{w}\in Q:\; w\sim_s\hat{w}\wedge\tilde{r}\notin\bigcup rsc(\hat{w}(PCVar))$": Assume for a moment that $\tilde{r}$ is available in w. We distinguish two cases.

Case $\tilde{r}=r$. Notice that $w\sim_{\tilde{r}}\hat{v}$, $\hat{v}\in Q$ and $\tilde{r}$ is available in $\hat{v}$. Take $\hat{w}:=\hat{v}$.

Case $\tilde{r}\neq r$. Then $\tilde{r}\cap r=\emptyset$, so we have $w\sim_{\tilde{r}}\check{v}$. Since $rsc(w(PCVar))=rsc(\check{v}(PCVar))$, we obtain that $\tilde{r}$ is available in $\check{v}$. Take $\hat{w}:=\check{v}$.

"$\forall\tilde{t}\in Tid:\;\tilde{r}\in rsc(w(pc_{\tilde{t}}))\Rightarrow\exists\check{w}\in Q:\; w\sim_{(Local_{\tilde{t}}\setminus\bigcup Res)\cup s}\check{w}$": Let $\tilde{t}\in Tid$ such that $\tilde{r}\in rsc(w(pc_{\tilde{t}}))$. Since r is available in w, we have $r\neq\tilde{r}$, so $w\sim_{(Local_{\tilde{t}}\setminus\bigcup Res)\cup s}\check{v}$. Take $\check{w}:=\check{v}$.

At last, we'll show

"$\forall\,\tilde{t}\in Tid\;\exists\,\tilde{w}\in Q:\; w\sim_{Local_{\tilde{t}}\setminus\bigcup Res}\tilde{w}$": Let $\tilde{t}\in Tid$ be given. Then $w\sim_{Local_{\tilde{t}}\setminus\bigcup Res}\check{v}$. Take $\tilde{w}:=\check{v}$.

We have shown $w\in\bar{\rho}(Q)$. From $\bar{\rho}(Q)=Q$ we obtain $w\in Q$.

By (4) and Remark 23 there is w' such that $w\rightarrow_t w'$ and $w'\sim_{Local_t\cup\bigcup R'} v'$. Notice that $w'\in post(Q)\subseteq Q$. ■

Let $\mathfrak{O}$ be the lattice of Owicki-Gries-core proofs of T.

Restatement of Theorem 18. The strongest property provable by abstract interpretation with $\bar{\rho}$ is provable by the Owicki-Gries core. Formally:

$$\gamma_{\mathrm{OG}}(\inf \mathfrak{O}) \;\subseteq\; \mathit{lfp}(\lambda x.\, \bar{\rho}(\mathit{init} \cup \mathit{post}(x)))\,.$$

Proof. For each $r \in \mathit{Res}$ let

$$I_r = \{u \in (\mathit{ProofVar}(r) \to \mathit{Val}) \mid \exists\, \hat{v} \in Q : u \sim_r \hat{v} \,\wedge\, r \text{ is available in } \hat{v}\}\,,$$

and for all $t \in \mathit{Tid}$, $p \in \mathit{PL}$ let

$$M_{p,t} = \{u \in ((\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(\mathit{rsc}(p))) \to \mathit{Val}) \mid \;\exists\, \tilde{v} \in Q : \\ u \sim_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{rsc}(p)} \tilde{v} \,\wedge\, \tilde{v}(pc_t) = p\}\,.$$

Let $P = ((I_r)_{r \in \mathit{Res}}, (M_{p,t})_{p \in \mathit{PL}, t \in \mathit{Tid}})$. We'll show that P is an Owicki-Gries-core proof that denotes a subset of Q by proving the following three properties.

Sequential consistency. Let $t \in \mathit{Tid}$, $(v, v') \in \to_t$, $p = v(pc_t)$, $p' = v'(pc_t)$, $R = \mathit{rsc}(p)$, $R' = \mathit{rsc}(p')$, $v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(R)} \in M_{p,t}$ and $\forall\, r \in R' \setminus R : v|_{\mathit{ProofVar}(r)} \in I_r$. Then there is $\tilde{v} \in Q$ such that $v \sim_{\mathit{LocalDataVar}_t \cup \bigcup R} \tilde{v}$ and $\tilde{v}(pc_t) = p$. Then $\tilde{v}(pc_t) = v(pc_t)$. Thus $\tilde{v} \sim_{\mathit{Local}_t \cup \bigcup R} v$. By (5) there are two cases.

Case $R' \subseteq R$. Then $\tilde{v} \sim_{\mathit{Local}_t \cup \bigcup R'} v$. By (4) and Remark 23 there is some $\tilde{v}' \in \mathit{ConsState}$ such that $(\tilde{v}, \tilde{v}') \in \to_t$ and $\tilde{v}' \sim_{\mathit{Local}_t \cup \bigcup R'} v'$. Then $\tilde{v}' \in \mathit{post}(Q) \subseteq Q$. Also $\tilde{v}'(pc_t) = v'(pc_t) = p'$ and $\tilde{v}' \sim_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{rsc}(p')} v'$. Thus $v'|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(\mathit{rsc}(p'))} \in M_{p',t}$.

Now take any $r \in R \setminus R'$ (by (5) there is one or none). Notice that $r \in \mathit{rsc}(\tilde{v}(pc_t))$. Since $\tilde{v} \in \mathit{ConsState}$, we get $r \notin \bigcup_{\tilde{t} \in \mathit{Tid} \setminus \{t\}} \mathit{rsc}(\tilde{v}(pc_{\tilde{t}})) =$ [by (1)] $\bigcup_{\tilde{t} \in \mathit{Tid} \setminus \{t\}} \mathit{rsc}(\tilde{v}'(pc_{\tilde{t}}))$. Moreover, $r \notin \mathit{rsc}(\tilde{v}'(pc_t))$. Thus r is available in $\tilde{v}'$. We have $v' \sim_r$ [by (3)] $v \sim_r$ [since $r \in R$] $\tilde{v} \sim_r$ [by (3)] $\tilde{v}'$. Thus $v'|_{\mathit{ProofVar}(r)} \in I_r$.

Case $R' = R \dot{\cup} \{r\}$ for some $r \in \mathit{Res}$. From assumption $v|_{\mathit{ProofVar}(r)} \in I_r$ we obtain some $\hat{v} \in Q$ such that $v \sim_r \hat{v}$ and r is available in $\hat{v}$. By Lemma 17 there is $w' \in Q$ such that $w' \sim_{\mathit{Local}_t \cup \bigcup R'} v'$. From $w' \in Q$, $w'(pc_t) = p'$ and $w' \sim_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{rsc}(p')} v'$ we obtain $v'|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(\mathit{rsc}(p'))} \in M_{p',t}$.

Initial condition. Let $v \in \mathit{init}$. Then $v \in \mathit{ConsState}$, all resources are available in v and $v \in Q$. Thus for any $r \in \mathit{Res}$ we have $v|_{\mathit{ProofVar}(r)} \in I_r$. Now let $t \in \mathit{Tid}$. By definition of $M_{v(pc_t),t}$ we obtain $v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(\mathit{rsc}(v(pc_t)))} \in M_{v(pc_t),t}$. So $v \in \gamma_{\mathrm{OG}}(P)$. We've shown $\mathit{init} \subseteq \gamma_{\mathrm{OG}}(P)$.

Safe condition. Let $v \in \gamma_{\mathrm{OG}}(P)$. Then $v \in \mathit{ConsState}$.

Let $s \in \widetilde{RL}$, $r \in \mathit{Res}$, $s \subseteq r$. We'll show two statements.

"$r \notin \bigcup \mathit{rsc}(v(\mathit{PCVar})) \Rightarrow \exists \hat{v} \in Q\colon\, v \sim_s \hat{v} \wedge r \notin \bigcup \mathit{rsc}(\hat{v}(\mathit{PCVar}))$": Assume for a moment that $r \notin \bigcup \mathit{rsc}(v(\mathit{PCVar}))$. The definition of γ_{OG} implies that $v|_{\mathit{ProofVar}(r)} \in I_r$. So there is $\hat{v} \in Q$ such that $v \sim_r \hat{v}$ and r is available in $\hat{v}$. Then $v \sim_s \hat{v}$.

"$\forall t \in \mathit{Tid} : r \in \mathit{rsc}(v(pc_t)) \Rightarrow \exists \check{v} \in Q\colon\, v \sim_{(\mathit{Local}_t \setminus \bigcup \mathit{Res}) \cup s} \check{v}$": Let $t \in \mathit{Tid}$ and $r \in \mathit{rsc}(v(pc_t))$. Then $v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(\mathit{rsc}(v(pc_t)))} \in$ [definition of γ_{OG}] $M_{v(pc_t),t}$. Then there is $\tilde{v} \in Q$ such that $\tilde{v} \sim_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{rsc}(v(pc_t))} v$ and $\tilde{v}(pc_t) = v(pc_t)$. Thus $\tilde{v} \sim_{\mathit{Local}_t \setminus \bigcup \mathit{Res}} v$ and $\tilde{v} \sim_s v$. Take $\check{v} := \tilde{v}$.

Now we'll show that

"$\forall\, t \in \mathit{Tid}\; \exists\, \tilde{v} \in Q\colon\, v \sim_{\mathit{Local}_t \setminus \bigcup \mathit{Res}} \tilde{v}$": Let $t \in \mathit{Tid}$. The definition of $\gamma_{\mathrm{OG}}(P)$ implies $v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(\mathit{rsc}(v(pc_t)))} \in M_{v(pc_t),t}$. So there is $\tilde{v} \in Q$ such that $\tilde{v}(pc_t) =$

$v(pc_t)$ and $\tilde{v} \sim_{LocalDataVar_t \cup \bigcup rsc(v(pc_t))} v$. Then $\tilde{v} \sim_{Local_t \setminus \bigcup Res} v$.
We've shown that $v \in \bar{\rho}(Q) \subseteq$ [since Q is in the image of $\bar{\rho}$ by definition, it is a fixpoint of $\bar{\rho}$] Q.
Thus $\gamma_{\mathrm{OG}}(P) \subseteq Q$. ∎

Thus any property provable by abstract interpretation with $\bar{\rho}$ has an Owicki-Gries-core proof.

Corollary 45. *Let $lfp(\lambda x.\, \rho_{\mathrm{c}}(init \cup post(x))) \subseteq safe \in D$. Then safe has an Owicki-Gries-core proof.*

Proof. Let $S = lfp(\lambda x.\, \rho_{\mathrm{c}}(init \cup post(x)))$, $S \subseteq safe \in D$. From Prop. 38 we get $\bar{\rho}(init \cup post(S)) \subseteq \rho_{\mathrm{c}}(init \cup post(S)) = S$, so S is a postfix point of $\lambda x.\, \bar{\rho}(init \cup post(x))$ which is by Tarski's fixpoint theorem greater than or equal to its least fixpoint. Thus $lfp(\lambda x.\, \bar{\rho}(init \cup post(x))) \subseteq S \subseteq safe$. Apply Thm. 18. ∎

Proposition 46. *Fix a syntactic structure $(Tid, PL, Val, Res, (Local_t, pc_t)_{t \in Tid}, ProofVar, rsc)$ from StrSimple and let ConsState be the set of consistent states of this syntactic structure and D its powerset. Let $\bar{\rho}$ and $\underline{\rho}$ be defined for StrSimple as in Sections D and E, respectively. Then the following statements hold.*

1. *$\underline{\rho} = \bar{\rho}$.*
2. *For all RPL transitions systems $(init, (\to_t)_{t \in Tid})$ from Simple that have successor map post and the set of Owicki-Gries-core proofs $\mathfrak{O}$ we have*

$$\gamma_{\mathrm{OG}}(\inf(\mathfrak{O})) = lfp(\lambda x.\, \bar{\rho}(init \cup post(x))) = lfp(\lambda x.\, \underline{\rho}(init \cup post(x)))\,.$$

Proof. 1. Let $Q \in D$. Since $|Res| \leq 1$, we distinguish two cases.
Case $Res = \emptyset$. Then

$$\underline{\rho}(Q) \;=\; \{v \in ConsState \mid \forall\, t \in Tid\; \exists\, \tilde{v} \in Q\colon\; v \sim_{Local_t} \tilde{v}\} \;=\; \bar{\rho}(Q)\,.$$

Case $Res = \{r\}$. Then $r \neq \emptyset$. Then $\underline{\rho}(Q) =$

$$\left\{ v \in ConsState \;\middle|\; \begin{array}{l} \left(r \notin \bigcup rsc(v(PCVar)) \Rightarrow \exists\, \hat{v} \in Q\colon \begin{pmatrix} r \notin \bigcup rsc(\hat{v}(PCVar)) \\ \wedge\; v \sim_r \hat{v} \end{pmatrix} \right) \\ \wedge \\ \forall\, t \in Tid\; \exists\, \tilde{v} \in Q\colon\; v \sim_{Local_t \cup \bigcup rsc(v(pc_t))} \tilde{v} \end{array} \right\}$$

[since $r \neq \emptyset$ and is disjoint from all $Local_t$ ($t \in Tid$), we have $r \in \widetilde{RL}$]
$= \bar{\rho}(Q)$.

2. Let $(init, (\to_t)_{t \in Tid})$ be an RPL transition system from Simple that obeys the fixed syntactic structure and has a set of Owicki-Gries-core proofs $\mathfrak{O}$ and successor map *post*. Then $lfp(\lambda x.\, \underline{\rho}(init \cup post(x))) \subseteq$ [Thm. 10] $\gamma_{\mathrm{OG}}(\inf \mathfrak{O}) \subseteq$ [Thm. 18] $lfp(\lambda x.\, \bar{\rho}(init \cup post(x))) \subseteq$ [part 1 of this Lemma] $lfp(\lambda x.\, \underline{\rho}(init \cup post(x)))$.

Extension of Example 19. Consider the RPL transition system Readers-Writers from Example 1.
Notice that $|Res| = 1$. Moreover, $\forall\, t \in Tid\colon\; Local_t \cap control = \{pc_t\} \cap \{ww, ar, aw\} = \emptyset$

and $\mathit{ProofVar}(\mathit{control}) = \mathit{control}$. Thus Readers-Writers belongs to Simple and its syntactic structure belongs to StrSimple. By Prop. 46, part 2, the strongest Owicki-Gries-core-provable property is equal to $\mathit{lfp}(\lambda x.\,\bar{\rho}(\mathit{init}\cup \mathit{post}(x))) = \mathit{lfp}(\lambda x.\,\underline{\rho}(\mathit{init}\cup \mathit{post}(x)))$. $\square$

Extension of Example 20. For the RPL transition system Upper from Example 2 we have $RL = \{r, r', \mathit{Local}_1, \mathit{Local}_2\} = \{\{u,z\},\{x,y\},\{x,l,pc_1\},\{y,z,pc_2\}\}$, $\widetilde{RL} = \{\{u\},\{z\},\{x\},\{y\},\{l,pc_1\},\{pc_2\}\}$. Let

$$C \;=\; \{v \in \mathit{ConsState} \mid v(u),v(z),v(x),v(y),v(l) \in \{0,1\} \,\wedge\, v(pc_2) = \mathtt{0}\}\,.$$

The upper iteration sequence $(X^{(i)})_{i\in\omega}$ for $\lambda x.\,\bar{\rho}(\mathit{init}\cup \mathit{post}(x))$ starting from $\emptyset \in D$ is below.

$$\begin{aligned}
X^{(0)} &= \emptyset\,,\\
X^{(1)} &= \bar{\rho}(\mathit{init}) \;=\; \{v \in C \mid v(pc_1) = \mathtt{A}\}\,,\\
\mathit{post}(X^{(1)}) &= \{v \in C \mid v(pc_1) = \mathtt{B} \wedge v(l) = v(u)\}\,,\\
X^{(2)} &= \bar{\rho}(\mathit{init}\cup \mathit{post}(X^{(1)})) \;=\; \left\{ v \in C \;\middle|\; \begin{array}{c} v(pc_1) = \mathtt{A} \\ \vee\ (v(pc_1) = \mathtt{B} \wedge v(u) = v(l)) \end{array} \right\},\\
\mathit{post}(X^{(2)}) &= \{v \in C \mid v(pc_1) \in \{\mathtt{B},\mathtt{C}\} \wedge v(u) = v(l)\}\,,\\
X^{(3)} &= \bar{\rho}(\mathit{init}\cup \mathit{post}(X^{(2)})) \;=\; \left\{ v \in C \;\middle|\; \begin{array}{c} v(pc_1) = \mathtt{A} \\ \vee\ (v(pc_1) \in \{\mathtt{B},\mathtt{C}\} \wedge v(u) = v(l)) \end{array} \right\},\\
\mathit{post}(X^{(3)}) &= \left\{ v \in C \;\middle|\; \begin{array}{c} (v(pc_1) \in \{\mathtt{B},\mathtt{C}\} \wedge v(u) = v(l)) \\ \vee\ (v(pc_1) = \mathtt{D} \wedge v(u) = v(l) \wedge v(x) = 0) \end{array} \right\},\\
X^{(4)} &= \bar{\rho}(\mathit{init}\cup \mathit{post}(X^{(3)})) \;=\; \left\{ v \in C \;\middle|\; \begin{array}{c} v(pc_1) = \mathtt{A} \\ \vee\ (v(pc_1) \in \{\mathtt{B},\mathtt{C}\} \wedge v(u) = v(l)) \\ \vee\ (v(pc_1) = \mathtt{D} \wedge v(u) = v(l) \wedge v(x) = 0) \end{array} \right\},\\
\mathit{post}(X^{(4)}) &= \left\{ v \in C \;\middle|\; \begin{array}{c} (v(pc_1) \in \{\mathtt{B},\mathtt{C}\} \wedge v(u) = v(l)) \\ \vee\ (v(pc_1) \in \{\mathtt{D},\mathtt{E}\} \wedge v(u) = v(l) \wedge v(x) = 0) \end{array} \right\},\\
X^{(5)} &= \bar{\rho}(\mathit{init}\cup \mathit{post}(X^{(4)})) \;=\; \left\{ v \in C \;\middle|\; \begin{array}{c} v(pc_1) = \mathtt{A} \\ \vee\ (v(pc_1) \in \{\mathtt{B},\mathtt{C},\mathtt{E}\} \wedge v(u) = v(l)) \\ \vee\ (v(pc_1) = \mathtt{D} \wedge v(u) = v(l) \wedge v(x) = 0) \end{array} \right\},\\
\mathit{post}(X^{(5)}) &= \left\{ v \in C \;\middle|\; \begin{array}{c} (v(pc_1) \in \{\mathtt{B},\mathtt{C}\} \wedge v(u) = v(l)) \\ \vee\ (v(pc_1) \in \{\mathtt{D},\mathtt{E}\} \wedge v(u) = v(l) \wedge v(x) = 0) \end{array} \right\} \subseteq X^{(5)},\ \text{so}\\
X^{(6)} &= \bar{\rho}(\mathit{init}\cup \mathit{post}(X^{(5)})) \;\subseteq\; \bar{\rho}(\mathit{init}\cup X^{(5)}) \;=\; \bar{\rho}(X^5) \;=\; X^{(5)}\,.
\end{aligned}$$

Thus

$$\mathit{lfp}(\lambda x.\,\bar{\rho}(\mathit{init}\cup \mathit{post}(x))) = \left\{ v \in C \;\middle|\; \begin{array}{c} v(pc_1) = \mathtt{A} \\ \vee\ (v(pc_1) \in \{\mathtt{B},\mathtt{C},\mathtt{E}\} \wedge v(u) = v(l)) \\ \vee\ (v(pc_1) = \mathtt{D} \wedge v(u) = v(l) \wedge v(x) = 0) \end{array} \right\}$$

is strictly weaker than the strongest Owicki-Gries-core-provable property, e.g., containing the state $[u\mapsto 0, z\mapsto 0, x\mapsto 0, y\mapsto 0, l\mapsto 0, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{0}] \notin \gamma_{\mathrm{OG}}(\inf \mathfrak{D})$. $\square$

Extension of Example 22. Fix an RPL transition system from SepThreads. For all $S \subseteq \mathit{ConsState}$ we have

$$\bar{\rho}(S) \;=\; \{v \in \mathit{ConsState} \mid \forall\, t \in \mathit{Tid}\ \exists\, \tilde{v} \in S\colon v \sim_{\mathit{Local}_t} \tilde{v}\} \;=\; \underline{\rho}(S)\,.$$

Let $F = \lambda x.\, \mathit{init} \cup \mathit{post}(x)$ and $Q = \mathit{lfp}(F)$. Notice that post is a join-morphism, so is F. By Kleene's fixpoint theorem $Q = \bigcup_{i \in \mathbb{N}_0} F^i(\emptyset)$.

Let $v \in \bar{\rho}(Q)$. Then there is a map $m\colon \mathit{Tid} \to \mathbb{N}_0$ such that $\forall\, t \in \mathit{Tid}\ \exists\, \tilde{v} \in F^{m(t)}(\emptyset)\colon v \sim_{\mathit{Local}_t} \tilde{v}$. Since Tid is finite and the sequence $(F^i(\emptyset))_{i \in \mathbb{N}_0}$ is monotonically increasing, $\forall\, t \in \mathit{Tid}\ \exists\, \tilde{v} \in F^{\max(m(\mathit{Tid}))}(\emptyset)\colon v \sim_{\mathit{Local}_t} \tilde{v}$. Thus $v \in \bar{\rho}(F^{\max(m(\mathit{Tid}))}(\emptyset))$.

We have shown that $\bar{\rho}(Q) \subseteq \bigcup_{i \in \mathbb{N}_0} \bar{\rho}(F^i(\emptyset))$. Now we are going to show by induction that for each $i \in \mathbb{N}_0$ the set $\bar{\rho}(F^i(\emptyset))$ is a subset of Q.

Case $i = 0$.

Case $\mathit{Tid} = \emptyset$. Then $\mathit{ConsState} = \{v \in \mathit{Var} \to \mathit{Val} \mid \forall\, t \in \mathit{Tid} : \ldots\} = (\mathit{Var} \to \mathit{Val}) = \{\emptyset\} = \bar{\rho}(\emptyset)$. By definition of SepThreads, $\forall\, v \in \mathit{ConsState}\colon\ ((\forall t \in \mathit{Tid}\colon \ldots) \Rightarrow v \in \mathit{init})$. Thus $\mathit{init} = \{\emptyset\}$. Then $\bar{\rho}(F^0(\emptyset)) = \mathit{init} \subseteq Q$.

Case $\mathit{Tid} \neq \emptyset$. Then $\bar{\rho}(F^0(\emptyset)) = \emptyset \subseteq Q$.

Case $i = 1$. We have $\bar{\rho}(F^1(\emptyset)) = \bar{\rho}(\mathit{init}) \subseteq$ [def. of SepThreads] $\mathit{init} \subseteq Q$.

Case $i > 1$. Let $v \in \bar{\rho}(F^i(\emptyset))$, i.e., $v \in \bar{\rho}(\mathit{init} \cup \mathit{post}(F^{i-1}(\emptyset)))$. There is a subset of Tid (e.g., the empty subset) such that for all t from this subset there is $\tilde{v} \in \mathit{init}$ such that $v \sim_{\mathit{Local}_t} \tilde{v}$. Let I be a maximal (with respect to inclusion) subset of such form. Then for all $t \in \mathit{Tid} \setminus I$ there is $\tilde{v} \in \mathit{post}(F^{i-1}(\emptyset))$ such that $v \sim_{\mathit{Local}_t} \tilde{v}$. There is a subset of $\mathit{Tid} \setminus I$ (e.g., the empty subset) such that for all t from this subset there is $\tilde{v} \in F^{i-1}(\emptyset)$ such that $v \sim_{\mathit{Local}_t} \tilde{v}$. Let J be a maximal (with respect to inclusion) such subset. Let $K = \mathit{Tid} \setminus (I \cup J)$. For each $t \in K$ there is some $\hat{v} \in F^{i-1}(\emptyset)$ such that there is some $\tilde{v} \in \mathit{post}(\{\hat{v}\})$ such that $v \sim_{\mathit{Local}_t} \tilde{v}$; maximality of J implies $\hat{v} \not\sim_{\mathit{Local}_t} v$, so $\hat{v} \not\sim_{\mathit{Local}_t} \tilde{v}$, so (1) implies $\hat{v} \to_t \tilde{v}$. Thus there are maps $f, f'\colon K \to \mathit{ConsState}$ such that $\forall\, t \in K\colon f(t) \in F^{i-1}(\emptyset) \wedge f(t) \to_t f'(t) \wedge f'(t) \sim_{\mathit{Local}_t} v$. Let $C = \bigcup_{\substack{t_1, t_2 \in \mathit{Tid} \\ t_1 \neq t_2}} (\mathit{Local}_{t_1} \cap \mathit{Local}_{t_2})$. There is a map $h\colon ((\mathit{Var} \setminus C) \to \mathit{Tid})$ such that $\forall\, x \in \mathit{Var} \setminus C\colon x \in \mathit{Local}_{h(x)}$. Such a map h is unique. There is some enumeration of K, fix it: $K = \{\tau_k \mid 1 \leq k \leq n\}$ for $n = |K|$. For each $k \in \mathbb{N}_0 \cap [0, n]$ let

$$w^{(k)}\colon \mathit{Var} \to \mathit{Val},\ x \mapsto \begin{cases} v(x), & \text{if } x \in C \cup \left(\bigcup\limits_{t \in I \cup J} \mathit{Local}_t\right) \cup \left(\bigcup\limits_{j=1}^{k} \mathit{Local}_{\tau_j}\right), \\ f(h(x))(x), & \text{if } x \in \bigcup\limits_{j=k+1}^{n} \mathit{Local}_{\tau_j} \setminus C. \end{cases}$$

To show that each $w^{(k)}$ is well-defined ($0 \leq k \leq n$), notice that

- $\mathit{Var} \subseteq C \cup \left(\bigcup\limits_{t \in I \cup J} \mathit{Local}_t\right) \cup \left(\bigcup\limits_{j=1}^{k} \mathit{Local}_{\tau_j}\right) \cup \left(\bigcup\limits_{j=k+1}^{n} \mathit{Local}_{\tau_j} \setminus C\right)$;
- $(I \cup J) \cap K = \emptyset$ and the definition of C imply that for $a \in I \cup J \cup \{\tau_j \mid 1 \leq j \leq k\}$ and $b \in \{\tau_j \mid k < j \leq n\}$ we have $\mathit{Local}_a \cap (\mathit{Local}_b \setminus C) = \emptyset$.

Now we'll show by induction that for all $k \in \mathbb{N}_0 \cap [0, n]$ we have $w^{(k)} \in Q$.

Case $k = 0$. First we'll show that $w^{(0)}$ is in $\bar{\rho}(F^{i-1}(\emptyset))$. Let $t \in \mathit{Tid}$.

Case $t \in I$. Then $w^{(0)} \sim_{Local_t} v$. By definition of I, there is some $\tilde{v} \in init$ such that $v \sim_{Local_t} \tilde{v}$. Then $w^{(0)} \sim_{Local_t} \tilde{v}$. From $i-1 \geq 1$ we get $F^{i-1}(\emptyset) \supseteq init \ni \tilde{v}$.

Case $t \in J$. Then $w^{(0)} \sim_{Local_t} v$. By definition of J, there is some $\tilde{v} \in F^{i-1}(\emptyset)$ such that $v \sim_{Local_t} \tilde{v}$. Then $w^{(0)} \sim_{Local_t} \tilde{v}$.

Case $t \in K$. Let $x \in Local_t$. There is some $j \in \mathbb{N}^+ \cap [1,n]$ such that $t = \tau_j$; then $x \in Local_{\tau_j}$.

Case $x \in C$. Then $w^{(0)}(x) = v(x) = f'(\tau_j)(x) =$ [part 1 of Lemma 39) and $x \in C$] $f(\tau_j)(x)$.

Case $x \notin C$. Then $w^{(0)}(x) = f(h(x))(x) =$ [from $x \in Local_{\tau_j}$ and uniqueness of h] $f(\tau_j)(x)$.

We have shown that for all $x \in Local_{\tau_j}$ we have $w^{(0)}(x) = f(\tau_j)(x)$. Thus $w^{(0)} \sim_{Local_t} f(t)$. Notice that $f(t) \in F^{i-1}(\emptyset)$.

Thus $w^{(0)} \in \bar{\rho}(F^{i-1}(\emptyset))$. By induction hypothesis, $\bar{\rho}(F^{i-1}(\emptyset)) \subseteq Q$. Thus $w^{(0)} \in Q$.

Case $k \geq 1$. We are going to show two auxiliary statements.

"$f(\tau_k) \sim_{Local_{\tau_k}} w^{(k-1)}$": Let $x \in Local_{\tau_k}$.

Case $x \in C$. Then $f(\tau_k)(x) =$ [part 1 of Lemma 39] $f'(\tau_k)(x) = v(x) = w^{(k-1)}(x)$.

Case $x \notin C$. Notice that $h(x) = \tau_k$. Then $f(\tau_k)(x) = w^{(k-1)}(x)$.

"$w^{(k-1)} \to_{\tau_k} w^{(k)}$: From (4) and $f(\tau_k) \to_{\tau_k} f'(\tau_k)$ we obtain some w' such that $w^{(k-1)} \to_{\tau_k} w'$ and $w' \sim_{Local_{\tau_k}} f'(\tau_k)$. We'll show now that w' is equal to $w^{(k)}$. So let $x \in Var$.

Case $x \in C \cup \bigcup_{t \in I \cup J} Local_t$. If $x \in C$, part 1 of Lemma 39 implies $w'(x) = w^{(k-1)}(x)$. If $x \in Local_t$ for some $t \in I \cup J$, then $t \neq \tau_k$, so (1) implies $w'(x) = w^{(k-1)}(x)$. In both cases $w'(x) = w^{(k-1)} = v(x) = w^{(k)}(x)$.

Case $x \in Local_{\tau_k}$. Then $w'(x) = f'(\tau_k)(x) = v(x) = w^{(k)}(x)$.

Case $x \in Local_{\tau_j}$ for some $j \neq k$. Then $w'(x) =$ [by (1)] $w^{(k-1)}(x)$. If $j < k$, then $w^{(k-1)}(x) =$ [since $j \leq k-1$] $v(x) =$ [since $j \leq k$] $w^{(k)}(x)$. If $j > k$, then $w^{(k-1)}(x) =$ [since $j \geq k$] $f(h(x))(x) =$ [since $j \geq k+1$] $w^{(k)}(x)$. In both cases $w'(x) = w^{(k)}(x)$.

We have shown that $w' = w^{(k)}$. Thus $w^{(k-1)} \to_{\tau_k} w^{(k)}$

By induction hypothesis, $w^{(k-1)} \in Q$. Then $w^{(k)} \in post(\{w^{(k-1)}\}) \subseteq post(Q) \subseteq Q$.

In particular, $v = w^{(n)} \in Q$.

We have proven by induction that $\forall\, i \in \mathbb{N}_0 \colon \bar{\rho}(F^i(\emptyset)) \subseteq Q$. Thus $\bar{\rho}(Q) \subseteq Q$. So $\bar{\rho}(F(Q)) = \bar{\rho}(Q) = Q$, i.e., Q is a fixpoint of $\bar{\rho} \circ F$. Since $Q = lfp(F) \subseteq lfp(\bar{\rho} \circ F) \subseteq Q$,

$$lfp(\lambda x.\, \bar{\rho}(init \cup post(x))) \;=\; lfp(\lambda x.\, init \cup post(x)).$$

By theorems 10 and 18 we also get

$$lfp(\lambda x.\, \underline{\rho}(init \cup post(x))) \;=\; \gamma_{\mathrm{OG}}(\inf \mathfrak{O}) \;=$$
$$lfp(\lambda x.\, \bar{\rho}(init \cup post(x))) \;=\; lfp(\lambda x.\, init \cup post(x)).$$

$\square$

F Absence of equivalent semantic characterization in abstract interpretation

Given a complete lattice D, a *semantic transformation* over D is a map $F^\# \in (D \to D) \to D \to D$ such that
- for all join morphisms $\tau \in D \to D$, the map $F^\#_\tau : D \to D$ is monotone and
- for all join morphisms $\tau_1, \tau_2 \in D \to D$ we have $\mathit{lfp}(\tau_1) = \mathit{lfp}(\tau_2) \Rightarrow \mathit{lfp}(F^\#_{\tau_1}) = \mathit{lfp}(F^\#_{\tau_2})$.

Lemma 47. *Let $(D, \subseteq)$ be a complete lattice and $f : (D \times D) \to D$ be any map that is monotone in the second component. Then*

$$F^\# \colon (D \to D) \to D \to D\,, \quad \tau \mapsto \lambda x.\, f\left(\left(\begin{cases} \mathit{lfp}(\tau)\,, & \text{if } \tau \text{ is monotone}\,, \\ \sup D\,, & \text{otherwise}\,. \end{cases}\right), x\right)$$

is a semantic transformation.

Proof. For all join morphisms $\tau \in D \to D$ and all elements $x \subseteq y$ of D we have $F^\#_\tau(x) = f(\mathit{lfp}(\tau), x) \subseteq f(\mathit{lfp}(\tau), y) = F^\#_\tau(y)$.

Now take two join morphisms $\tau_1, \tau_2 \in D \to D$ such that $\mathit{lfp}(\tau_1) = \mathit{lfp}(\tau_2)$. Then $\mathit{lfp}(F^\#_{\tau_1}) = \mathit{lfp}(\lambda x.\, f(\mathit{lfp}(\tau_1), x)) = \mathit{lfp}(\lambda x.\, f(\mathit{lfp}(\tau_2), x)) = \mathit{lfp}(F^\#_{\tau_2})$. ∎

Example 48. Given a complete lattice D, the following maps are semantic transformations (where infimum over the empty set is the top of D):
- $\lambda \tau \in D \to D.\, \lambda x \in D.\, \sup(D)$;
- $\lambda \tau \in D \to D.\, \lambda x \in D.\, \inf\{y \in D \mid \tau(y) = y\}$;
- $\lambda \tau \in D \to D.\, \lambda x \in D.\, \sup\{\inf\{y \in D \mid \tau(y) = y\}, x\}$. □

Given a syntactic structure $(\mathit{Tid}, \mathit{PL}, \mathit{Val}, \mathit{Res}, (\mathit{Local}_t, \mathit{pc}_t)_{t \in \mathit{Tid}}, \mathit{ProofVar}, \mathit{rsc})$, let

$$\mathit{Trace} = \mathit{ConsState}^*\,, \quad \widetilde{D} = \mathfrak{P}(\mathit{Trace}) \quad \text{and} \quad \widetilde{\alpha} \colon \widetilde{D} \to D,\ T \mapsto \bigcup_{\delta \in T} \pi_2(\delta)\,,$$

where a trace is viewed as a finite word, i.e., a map from $\mathbb{N}_n$ for some natural $n \in \mathbb{N}^+$ to consistent states, and where $\pi_2(\delta) = \{y \mid \exists\, x \colon (x, y) \in \delta\}$ is the set of states occurring in the trace δ.

Given a program $(\mathit{init}, (\to_t)_{t \in \mathit{Tid}})$ that obeys the above syntactic structure, has the set of *initial traces* $\widetilde{\mathit{init}} = \{\{(1, v)\} \mid v \in \mathit{init}\}$ and the *trace extension operator*

$$\widetilde{\mathit{post}} \colon \widetilde{D} \to \widetilde{D}\,, \quad T \mapsto \{\delta v v' \mid \delta v \in T \wedge \exists t \in \mathit{Tid} \colon (v, v') \in \to_t\}\,.$$

Restatement of Theorem 6. There is a syntactic structure
$(\mathit{Tid}, \mathit{PL}, \mathit{Val}, \mathit{Res}, (\mathit{Local}_t, \mathit{pc}_t)_{t \in \mathit{Tid}}, \mathit{ProofVar}, \mathit{rsc})$
such that for *ConsState*, D, $\widetilde{D}$, $\widetilde{\alpha}$ defined as above, all the following statements hold:
1. There is no semantic transformation $F^\#$ over D such that for any RPL transition system $(\mathit{init}, \ldots)$ that obeys the syntactic structure, has successor map *post* and the least Owicki-Gries-core proof P, we have

$$\gamma_{\mathrm{OG}}(P) = \mathit{lfp}(F^\#_{\lambda x.\mathit{init} \cup \mathit{post}(x)})\,;$$

2. There is no monotone operator $\rho\colon D \to D$ such that for any RPL transition system $(init, \ldots)$ that obeys the syntactic structure, has successor map *post* and the least Owicki-Gries-core proof P, we have

$$\gamma_{\mathrm{OG}}(P) = lfp(\lambda x.\, \rho(init \cup post(x)))\,.$$

3. There is no monotone operator $\widetilde{\rho}\colon \widetilde{D} \to \widetilde{D}$ such that for any RPL transition system that obeys the syntactic structure, has a set of initial traces $\widetilde{init}$, has a trace extension operator $\widetilde{post}$ and has the least Owicki-Gries-core proof P, we have

$$\gamma_{\mathrm{OG}}(P) = \widetilde{\alpha}(lfp(\lambda x.\, \widetilde{\rho}(\widetilde{init} \cup \widetilde{post}(x))))\,.$$

Proof. Consider the syntactic structure S given by

- $Tid = \{1,2\}$;
- $PL = \{\mathtt{A},\mathtt{B},\mathtt{C},\ldots,\mathtt{V},\mathtt{W}\}$;
- $Val = PL \dot\cup \{0,1\}$;
- $Res = \{r,r'\}$ where $r = \{u,z\}$, $r' = \{x,y\}$;
- $Local_1 = \{l,x,pc_1\}$, $Local_2 = \{y,z,pc_2\}$;
- $ProofVar(r) = r$ and $ProofVar(r') = r'$.
- $rsc : PL \to \mathfrak{P}(Res)$ given by

p	A, G, J, M, N, O, P, S, W	B, E, F, K, L, T, U, V	H, I, Q, R	C, D
$rsc(p)$	$\emptyset$	$\{r\}$	$\{r'\}$	$\{r,r'\}$

The distribution of variables into different sets is given in Figure 4.

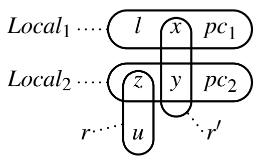

Fig. 4: Distribution of variables of program UpperA.

Let D be the powerset of the set of consistent states defined by S as stated in the theorem.

The RPL transition system UpperA defined by Fig. 5 obeys S. It differs from Upper only at location A.

We are going to show an Owicki-Gries-core proof of property *init* for UpperA. Let
$I_r = \{\, [u \mapsto 0, z \mapsto 1], [u \mapsto 1, z \mapsto 0] \,\}$, $I_{r'} = \{\, [x \mapsto 0, y \mapsto 0], [x \mapsto 1, y \mapsto 1] \,\}$,
$M_{\mathtt{A},1} = \{\, [x \mapsto 0, l \mapsto 0], [x \mapsto 1, l \mapsto 1] \,\}$,
$M_{\mathtt{O},2} = \{\, [y \mapsto 0, z \mapsto 0], [y \mapsto 1, z \mapsto 1] \,\}$ and
$M_{x,t} = \emptyset$ for $(x,t) \in (PL \times Tid) \setminus \{(\mathtt{A},1),(\mathtt{O},2)\}$.

Let $P_{\mathrm{UpperA}} = ([r \mapsto I_r, r' \mapsto I_{r'}], (M_{x,t})_{x \in PL, t \in Tid})$. Now we will show that P_{UpperA} is an Owicki-Gries-core proof denoting *init*.

By definition, $P_{\mathrm{UpperA}} \in$
$(\mathfrak{P}(ProofVar(r) \to Val) \times \mathfrak{P}(ProofVar(r') \to Val)) \times$

```
// Thread 1                        // Thread 2
A: with r when false do            O: assume false;
B:    with r' do                   P: with r' do
C:       x:=0                      Q:    y:=0
D:    endwith;                     R: endwith;
E:    assume false                 S: with r do
F: endwith;                        T:    u:=0;
G: with r' do                      U:    z:=0
H:    x:=0                         V: endwith
I: endwith;                        W:
J: with r do
K:    u:=0
L: endwith;
M: l:=0
N:
```

$$
\begin{aligned}
\mathit{init} = \{ & [u \mapsto 1, z \mapsto 0, x \mapsto 0, y \mapsto 0, l \mapsto 0, pc_1 \mapsto \mathtt{A}, pc_2 \mapsto \mathtt{O}], \\
& [u \mapsto 0, z \mapsto 1, x \mapsto 1, y \mapsto 1, l \mapsto 1, pc_1 \mapsto \mathtt{A}, pc_2 \mapsto \mathtt{O}] \}.
\end{aligned}
$$

Fig. 5: Program UpperA.

$(\mathfrak{P}(\mathit{LocalDataVar}_1 \to \mathit{Val}) \times \mathfrak{P}(\mathit{LocalDataVar}_2 \to \mathit{Val}) \times$
$\mathfrak{P}((\mathit{LocalDataVar}_1 \cup \mathit{ProofVar}(r)) \to \mathit{Val}) \times$
$\ldots)$,
where the components are indexed by r, r' in the first line and $(\mathtt{A}, 1)$, $(\mathtt{O}, 2)$, $(\mathtt{B}, 1)$ and so on in the remainder. Thus $P_{\text{UpperA}} \in PA$. We'll show:

- sequential consistency. Let $t \in \mathit{Tid}$, $(v, v') \in \to_t$, $R = \mathit{rsc}(v(pc_t))$, $R' = \mathit{rsc}(v'(pc_t))$, as well as $v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(R)} \in M_{v(pc_t),t}$ and $\forall\, \tilde{r} \in R' \setminus R : v|_{\mathit{ProofVar}(\tilde{r})} \in I_{\tilde{r}}$.
  - Case $v(pc_t) \in \{\mathtt{A}, \mathtt{O}\}$. Then thread t can't proceed from v, a contradiction.
  - Case $v(pc_t) \notin \{\mathtt{A}, \mathtt{O}\}$. Then $v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(R)} \in \emptyset$, a contradiction.
- initial condition. We have
  $$
  \begin{aligned}
  \gamma_{\text{OG}}(P_{\text{UpperA}}) = \{ & [u \mapsto 1, z \mapsto 0, x \mapsto 0, y \mapsto 0, l \mapsto 0, pc_1 \mapsto \mathtt{A}, pc_2 \mapsto \mathtt{O}], \\
  & [u \mapsto 0, z \mapsto 1, x \mapsto 1, y \mapsto 1, l \mapsto 1, pc_1 \mapsto \mathtt{A}, pc_2 \mapsto \mathtt{O}]\}.
  \end{aligned}
  $$
  Notice that $\mathit{init} \subseteq \gamma_{\text{OG}}(P_{\text{UpperA}})$.

Since $\gamma_{\text{OG}}(P_{\text{UpperA}}) \subseteq \mathit{init}$, P_{UpperA} is an Owicki-Gries-core proof of property *init* for the program UpperA.

The RPL transition system Upper from Example 2 also obeys S and has the same initial states as UpperA, but, as stated in the example, has no Owicki-Gries-core proof of *init*.

Now we'll show the claims of the theorem.

Let *postU* be the successor map of Upper and *postUA* the successor map of UpperA. Notice that $\mathit{postU}(\mathit{init}) = \mathit{postUA}(\mathit{init})$, since the values of program variables l and u are different in each state of *init*. Moreover, *init* is exactly the set of reachable states of both Upper and UpperA. In particular, *init* is the strongest Owicki-Gries-core-provable property of UpperA.

1. Assume for the purpose of contradiction that a semantic transformation $F^\#$ over D exists such that for any RPL transition system $T = (\mathit{init}, \ldots)$ that obeys S and has

successor map *post* we have

$$\gamma_{\mathrm{OG}}(\text{the smallest Owicki-Gries-core proof of } T) = \mathit{lfp}(F^{\#}_{\lambda x.\mathit{init}\cup \mathit{post}(x)})\,;$$

Notice that $\lambda x.\,\mathit{init} \cup \mathit{postU}(x)$ and $\lambda x.\,\mathit{init} \cup \mathit{postUA}(x)$ are join-morphisms on the same lattice and their least fixpoints coincide. By the assumption, for UpperA with successor map *postUA*, the strongest Owicki-Gries-core-provable property is $\mathit{lfp}(F^{\#}_{\lambda x.\mathit{init}\cup \mathit{postUA}(x)})$. Thus $\mathit{init} = \mathit{lfp}(F^{\#}_{\lambda x.\mathit{init}\cup \mathit{postUA}(x)}) =$ [property of $F^{\#}$] $\mathit{lfp}(F^{\#}_{\lambda x.\mathit{init}\cup \mathit{postU}(x)})$. By the assumption again, *init* is also the strongest Owicki-Gries-core-provable property of Upper. Contradiction!

2. Assume for the purpose of contradiction that a monotone operator $\rho : D \to D$ exists such that for any RPL transition system $T = (\mathit{init}, \ldots)$ that obeys S and has successor map *post* we have

$$\gamma_{\mathrm{OG}}(\text{the smallest Owicki-Gries-core proof of } T) = \mathit{lfp}(\lambda x.\,\rho(\mathit{init} \cup \mathit{post}(x)))\,.$$

Then for UpperA with initial states *init* and successor map *postUA* we have $\mathit{lfp}(\lambda x.\,\rho(\mathit{init} \cup \mathit{postUA}(x))) = \mathit{init}$. Thus $\rho(\mathit{init} \cup \mathit{postUA}(\mathit{init})) = \mathit{init}$. Since $\mathit{postUA}(\mathit{init}) = \mathit{postU}(\mathit{init})$, we get $\rho(\mathit{init} \cup \mathit{postU}(\mathit{init})) = \mathit{init}$. Then $\mathit{lfp}(\lambda x.\,\rho(\mathit{init} \cup \mathit{postU}(x))) \subseteq \mathit{init}$. Applying the assumption to Upper, we get an Owicki-Gries-core proof P of Upper such that $\gamma_{\mathrm{OG}}(P) \subseteq \mathit{init}$. But Upper has no Owicki-Gries-core proof of *init*. Contradiction!

3. Assume for the purpose of contradiction that a monotone operator $\widetilde{\rho} : \widetilde{D} \to \widetilde{D}$ exists such that for any RPL transition system $T = (\mathit{init}, \ldots)$ that obeys S and has $\widetilde{\mathit{init}}$ as a set of initial traces and the trace extension operator $\widetilde{\mathit{post}}$ we have

$$\gamma_{\mathrm{OG}}(\text{the smallest Owicki-Gries-core proof of } T) \;=\; \widetilde{\alpha}(\mathit{lfp}(\lambda x.\widetilde{\rho}(\widetilde{\mathit{init}} \cup \widetilde{\mathit{post}}(x)))).$$

Then for UpperA with the set of initial traces $\widetilde{\mathit{init}}$ and the trace extension operator $\widetilde{\mathit{postUA}}$ we have $\mathit{init} = \widetilde{\alpha}(\mathit{lfp}(\lambda x.\,\widetilde{\rho}(\widetilde{\mathit{init}} \cup \widetilde{\mathit{postUA}}(x))))$. From definition of $\widetilde{\alpha}$ we get $\mathit{lfp}(\lambda x.\,\widetilde{\rho}(\widetilde{\mathit{init}} \cup \widetilde{\mathit{postUA}}(x))) \subseteq \mathit{init}^*$. Consider the upper fixpoint iteration sequence $(X^i)_{i\in\mu}$ for $\lambda x.\,\widetilde{\rho}(\widetilde{\mathit{init}} \cup \widetilde{\mathit{postUA}}(x))$ that starts with $X^0 = \emptyset$ and where $\mu > |\widetilde{D}|$ is some ordinal. This sequence is monotonously increasing and its limit is a subset of init^*. Since $\widetilde{\mathit{postU}}(\mathit{init}^*) = \emptyset = \widetilde{\mathit{postUA}}(\emptyset)$, we have $\widetilde{\rho}(\widetilde{\mathit{init}} \cup \widetilde{\mathit{postU}}(\mathit{init}^*)) = \widetilde{\rho}(\widetilde{\mathit{init}} \cup \widetilde{\mathit{postUA}}(X^0)) = X^1 \subseteq \mathit{init}^*$. So $\mathit{init}^* \in \mathit{postfp}(\lambda x.\,\widetilde{\rho}(\widetilde{\mathit{init}} \cup \widetilde{\mathit{postU}}(x)))$. Thus $\mathit{lfp}(\lambda x.\,\widetilde{\rho}(\widetilde{\mathit{init}} \cup \widetilde{\mathit{postU}}(x))) \subseteq \mathit{init}^*$, so $\widetilde{\alpha}(\mathit{lfp}(\lambda x.\,\widetilde{\rho}(\widetilde{\mathit{init}} \cup \widetilde{\mathit{postU}}(x)))) \subseteq \mathit{init}$. By assumption, the smallest Owicki-Gries-core proof of Upper denotes a subset of *init*. But Upper has no Owicki-Gries-core proof of *init*. Contradiction! ■

Remark 49. The proof relied on the statements `assume` *false* to denote blocking. Strictly speaking, Owicki's RPL does not have an `assume` statement. A purist may emulate it by `while` *true* `do begin end` or by `with` r'' `when` *false* `do endwith` for an empty fresh resource r''.

It is also possible to rewrite Upper and UpperA into nonblocking terminating programs using `if-then-else` statements.

Moreover, the programs of the proof depend on realizability of the overlap of resources and locals, but require a relatively small number (five) of data variables.

As a consequence, Thm. 6 holds even for the restricted class of programs that are written strictly in the original Owicki's syntax, in which every execution terminates with threads at their final locations, and that use at most five data variables. $\square$

G Improving precision

Now we will show a method, called Relaxed Frontier Search [21], that allows improving precision of many abstract interpretations, in particular, of the Owicki-Gries core. First we will show the general case and then specialize it to the Owicki-Gries core.

G.1 Relaxed Frontier Search - in general

Now we will show relaxed frontier search for a large class of abstract transfer functions.

Within Section G.1, let $(D, \subseteq)$ be a complete lattice of sets, $(D^\#, \sqsubseteq)$ a complete lattice, (α, γ) a pair of adjoint maps, $init \in D$, $post : D \to D$ a complete join morphism and $post^\# : D^\# \to D^\#$ a monotonic operator such that $\forall\, Y \in D^\# : \alpha post \gamma Y \sqsubseteq post^\# Y$.

Intuitively, D is the powerset of program states, *init* is the set of initial states, *post* maps a set of states to the set of their direct successors, $D^\#$ is an abstract lattice used for analysis and $post^\#$ is an abstract transformer.

The *relaxed frontier search sequence* is the sequence $(T^{(i)})_{i \in \mathbb{N}_0}$ of elements of $D^\#$ defined recursively by

$T^{(0)} = \alpha init$,

$$T^{(i+1)} = \begin{cases} post^\# T^{(i)}, & \text{if } post^\# T^{(i)} \not\sqsubseteq \bigsqcup_{j=0}^{i} T^{(j)}, \\ T^{(i)} \sqcup post^\# T^{(i)}, & \text{if } post^\# T^{(i)} \sqsubseteq \bigsqcup_{j=0}^{i} T^{(j)}. \end{cases}$$

Let $k = \min\{i \mid T^{(i+1)} \sqsubseteq T^{(i)}\} \in \mathbb{N}_0 \dot\cup \{\infty\}$ (where $\infty = \min \emptyset$).

The *height* of a partial order is the supremum over cardinalities of its chains.

The above (in general non-monotone) sequence is the basis of our algorithm, which needs only elements of the sequence up to position k. First we prove that k is well-defined, and is asymptotically at most quadratic in the height of $\sqsubseteq$.

Proposition 50. *If the height h of $(D^\#, \sqsubseteq)$ is finite, then $k \le h(h-1)$.*

Proof. Let h be finite. The elements of the monotone sequence $(\bigsqcup_{i=0}^{k'} T^{(i)})_{k' \ge 0}$ build a chain, so this sequence stabilizes for some $k' \ge 0$, i.e., $T^{(j)} \sqsubseteq \bigsqcup_{i=0}^{k'} T^{(i)}$ for all $j \ge k'$. Take the smallest such k'. For all $j \ge k'$ we have $T^{(j+1)} \sqsupseteq post^\# T^{(j)}$ and thus $post^\# T^{(j)} \sqsubseteq \bigsqcup_{i=0}^{k'} T^{(i)}$. By definition of the algorithm, the sequence $(T^{(j)})_{j \ge k'}$ is monotonously increasing. The domain has finite height, so there is a $k \ge k'$ with $T^{(k+1)} = T^{(k)}$. Take the smallest such k.

Now we derive the number of iteration steps. There are two cases

Case $post^\# T^{(0)} \sqsubseteq T^{(0)}$. Then $post^\# T^{(0)} \sqsubseteq \bigsqcup_{j=0}^{0} T^{(j)}$, so $T^{(1)} = T^{(0)}$. Then $k = 0 \le h(h-1)$.

Case $post^{\#}T^{(0)} \not\sqsubseteq T^{(0)}$. Then $post^{\#}T^{(0)} \not\sqsubseteq \bigsqcup_{j=0}^{0} T^{(j)}$, so $T^{(1)} = post^{\#}T^{(0)} \not\sqsubseteq T^{(0)}$. Therefore $\bigsqcup_{i=0}^{0} T^{(i)} \neq \bigsqcup_{i=0}^{1} T^{(i)}$ (otherwise $T^{(1)}$ were less than or equal to $T^{(0)}$).

Since $\{\bigsqcup_{i=0}^{j} T^{(i)} \mid j \in \mathbb{N}_0\}$ is a chain, there are at most $h-1$ indices $j > 0$ such that $\bigsqcup_{i=0}^{j-1} T^{(i)} \neq \bigsqcup_{i=0}^{j} T^{(i)}$, the smallest such index is 1. Consider any two such "neighbor" indices $a < b$ with

$$\bigsqcup_{i=0}^{a-1} T^{(i)} \neq \bigsqcup_{i=0}^{a} T^{(i)} \text{ and } \bigsqcup_{i=0}^{b-1} T^{(i)} \neq \bigsqcup_{i=0}^{b} T^{(i)} \text{ and } 0 < a$$

so that for all c with $a < c < b$ we have

$$\bigsqcup_{i=0}^{c-1} T^{(i)} = \bigsqcup_{i=0}^{c} T^{(i)} .$$

The sequence $(T^{(c)})_{a \le c < b}$ is increasing by the definition of the relaxed frontier sequence. The increase is strict, i.e., $T^{(c-1)} \neq T^{(c)}$ for $a < c < b$ (since otherwise the sequence $(T^{(i)})_{i \ge c}$ were constant for some $c < b$, contradicting the choice of b). The maximal chain length gives $b - a \le h$ and finally $k - k' < h$. The number of such neighbor pairs (a,b) as above is at most $h-2$, the last such b is k', the first such a is 1. Thus $k' - 1 \le h(h-2)$. Thus $k' \le h^2 - 2h + 1$ and $k \le k' + h - 1 \le h^2 - h$. ∎

Proposition 51. *The set $\bigcup_{i \in \mathbb{N}_0} \gamma T^{(i)}$ is an inductive invariant. Formally:*

$$init \cup post\left(\bigcup_{i \in \mathbb{N}_0} \gamma T^{(i)}\right) \subseteq \bigcup_{i \in \mathbb{N}_0} \gamma T^{(i)} .$$

Proof. By extensivity of $\gamma\alpha$ we obtain $init \subseteq \gamma\alpha init = \gamma T^{(0)}$. Moreover

$$post\left(\bigcup_{i \in \mathbb{N}_0} \gamma T^{(i)}\right) = \bigcup_{i \in \mathbb{N}_0} post\gamma T^{(i)} \subseteq \text{[by definition of a pair of adjoint maps]}$$
$$\subseteq \bigcup_{i \in \mathbb{N}_0} \gamma post^{\#} T^{(i)} \subseteq \bigcup_{i \in \mathbb{N}_0} \gamma T^{(i+1)} \subseteq \bigcup_{i \in \mathbb{N}_0} \gamma T^{(i)} .$$

∎

Proposition 52. *If $k < \infty$, then $\bigcup_{i=0}^{k} \gamma T^{(i)}$ is an inductive invariant. Formally:*

$$k < \infty \quad \Rightarrow \quad init \cup post\left(\bigcup_{i=0}^{k} \gamma T^{(i)}\right) \subseteq \bigcup_{i=0}^{k} \gamma T^{(i)} .$$

Proof. By definition of k we have $T^{(k+1)} \sqsubseteq T^{(k)}$. If for some $j > k$ we have $T^{(j)} \sqsubseteq T^{(k)}$, then $T^{(j+1)} \sqsubseteq T^{(j)} \sqcup post^{\#}T^{(j)} \sqsubseteq$ [by assumption and monotonicity of $post^{\#}$] $T^{(k)} \sqcup post^{\#}T^{(k)} \sqsubseteq T^{(k)} \sqcup T^{(k+1)} \sqsubseteq$ [by definition of k] $T^{(k)}$. By induction $T^{(j)} \sqsubseteq T^{(k)}$ for all $j \ge k$. Thus $\gamma T^{(j)} \subseteq \gamma T^{(k)}$ for all $j \ge k$, implying $\bigcup_{i=0}^{k} \gamma T^{(i)} = \bigcup_{i \in \mathbb{N}_0} \gamma T^{(i)}$. ∎

As a corollary, if $(D^{\#}, \sqsubseteq)$ has a finite height, $\bigcup_{i=0}^{k} \gamma T^{(i)}$ is an inductive invariant.

G.2 Relaxed Frontier Search - specialized for the Owicki-Gries core

Now we are going to specialize relaxed frontier search from Section G.1 to the Owicki-Gries core[8].

Let $(Tid, PL, Val, Res, (Local_t, pc_t)_{t\in Tid}, ProofVar, rsc)$ be an arbitrary but fixed syntactic structure, D the powerset of consistent states, $D^\#$ the set of program annotations equipped with componentwise order, $(init, (\rightarrow_t)_{t\in Tid})$ an RPL program that obeys the syntactic structure, $post$ its successor map, $\mathfrak{O}$ the lattice of Owicki-Gries-core proofs, and

$$
\begin{aligned}
&p^\#_{\mathrm{OG}} : PA \to PA\,, \quad (I,M) \mapsto \\
&\left(\left(\left\{ \begin{array}{l} u \in \\ ProofVar(r) \to Val \end{array} \,\middle|\, \begin{array}{l} \exists\, t \in Tid, (v,v') \in \rightarrow_t \colon\; r \notin rsc(v'(pc_t)) \wedge \\ v|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(v(pc_t)))} \in M_{v(pc_t),t} \wedge \\ \begin{pmatrix} (r \notin rsc(v(pc_t)) \;\wedge\; u \in I_r) \;\vee \\ (r \in rsc(v(pc_t)) \;\wedge\; u = v|_{ProofVar(r)}) \end{pmatrix} \end{array} \right\}\right)_{r\in Res}, \right. \\
&\left.\left(\left\{ \begin{array}{l} u \in \\ \begin{pmatrix} LocalDataVar_t \cup \\ \bigcup ProofVar(rsc(p)) \end{pmatrix} \\ \to Val \end{array} \,\middle|\, \begin{array}{l} \begin{pmatrix} \exists\, \tilde{t} \in Tid \backslash \{t\}, (v,v') \in \rightarrow_{\tilde{t}} \colon\; u \in M_{p,t} \wedge \\ v|_{LocalDataVar_{\tilde{t}} \cup \bigcup ProofVar(rsc(v(pc_{\tilde{t}})))} \in M_{v(pc_{\tilde{t}}),\tilde{t}} \wedge \\ (rsc(v(pc_{\tilde{t}})) \cup rsc(v'(pc_{\tilde{t}}))) \cap rsc(p) = \emptyset \end{pmatrix} \\ \vee \\ \begin{pmatrix} \exists\, (v,v') \in \rightarrow_t \colon v'(pc_t) = p \wedge \\ u = v'|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(p))} \wedge \\ v|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(v(pc_t)))} \in M_{v(pc_t),t} \wedge \\ \forall r \in rsc(p) \backslash rsc(v(pc_t)) : v|_{ProofVar(r)} \in I_r \end{pmatrix} \end{array} \right\}\right)_{\substack{p\in PL,\\ t\in Tid}} \right).
\end{aligned}
$$

We call $p^\#_{\mathrm{OG}}$ the *improved Owicki-Gries-core abstract successor map*. As before in Section C, let

$$
\begin{aligned}
I^{init} &= (\{ v|_{ProofVar(r)} \mid v \in init \})_{r\in Res}\,, \\
M^{init} &= (\{ v|_{LocalDataVar_t} \mid v \in init \wedge v(pc_t) = p \})_{p\in PL, t\in Tid}\,.
\end{aligned}
$$

and the *improved Owicki-Gries-core abstract transformer* be

$$
\begin{aligned}
&G^\#_{\mathrm{OG}} : PA \to PA\,, \\
&\quad (I,M) \mapsto (I^{init}, M^{init}) \sqcup p^\#_{\mathrm{OG}}(I,M)\,.
\end{aligned}
$$

Proposition 53. *$p^\#_{\mathrm{OG}}$ is monotone.*

Proof. Let $(I,M) \sqsubseteq (\hat{I},\hat{M})$ be program annotations, $p^\#_{\mathrm{OG}}(I,M) = (I',M')$, $p^\#_{\mathrm{OG}}(\hat{I},\hat{M}) = (\hat{I}',\hat{M}')$.

"$\forall\, r \in Res : I'_r \subseteq \hat{I}'_r$": Let $r \in Res$. Let $w \in I'_r$. Then there is a thread $t \in Tid$ and some transition $v \rightarrow_t v'$ such that $v|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(v(pc_t)))} \in M_{v(pc_t),t}$, which is a subset of $\hat{M}_{v(pc_t),t}$, $r \notin rsc(v'(pc_t))$, and either $r \notin rsc(v(pc_t)) \wedge w \in I_r$, which is a subset of $\hat{I}_r$, or $r \in rsc(v(pc_t)) \wedge w = v|_{ProofVar(r)}$. Then $v|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(v(pc_t)))} \in$

[8] Using $post^\#_{\mathrm{OG}}$ would not give us more precision: the generated inductive invariant would be equal to $\gamma_{\mathrm{OG}}(lfp(F^\#_{\mathrm{OG}}))$. We cannot use $next^\#_{\mathrm{OG}}$, as it does not overapproximate $\alpha_{\mathrm{OG}} \circ post \circ \gamma_{\mathrm{OG}}$ in general. We are going to suggest an operator in between: it will overapproximate $\alpha_{\mathrm{OG}} \circ post \circ \gamma_{\mathrm{OG}}$ and underapproximate $post^\#_{\mathrm{OG}}$, allowing more precision in the relaxed frontier search.

$\hat{M}_{v(pc_t),t}$; also either $r \notin rsc(v(pc_t)) \wedge w \in \hat{I}_r$, or $r \in rsc(v(pc_t)) \wedge w = v|_{ProofVar(r)}$. Thus $w \in \hat{I}'_r$.

"$\forall\, t \in Tid, p \in PL : M'_{p,t} \subseteq \hat{M}'_{p,t}$": Let $t \in Tid$, $p \in PL$. Let $w \in M'_{p,t}$. There are two cases.

- Either there is $\tilde{t} \in Tid \backslash \{t\}$ and $(v,v') \in \rightarrow_{\tilde{t}}$ such that $v|_{LocalDataVar_{\tilde{t}} \cup \bigcup ProofVar(rsc(v(pc_{\tilde{t}})))} \in M_{v(pc_{\tilde{t}}),\tilde{t}}$, $w \in M_{p,t}$ and $\emptyset = rsc(p) \cap (rsc(v(pc_{\tilde{t}})) \cup rsc(v'(pc_{\tilde{t}})))$. Notice that $M_{p,t} \subseteq \hat{M}_{p,t}$ and $M_{v(pc_{\tilde{t}}),\tilde{t}} \subseteq \hat{M}_{v(pc_{\tilde{t}}),\tilde{t}}$. Thus $v|_{LocalDataVar_{\tilde{t}} \cup \bigcup ProofVar(rsc(v(pc_{\tilde{t}})))} \in \hat{M}_{v(pc_{\tilde{t}}),\tilde{t}}$ and $w \in \hat{M}_{p,t}$. So $w \in \hat{M}'_{p,t}$.
- Or there is a transition $v \rightarrow_t v'$ of the same thread t such that $v'(pc_t) = p$, $w = v'|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(p))}$, $v|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(v(pc_t)))} \in M_{v(pc_t),t}$ and $\forall\, r \in rsc(p) \backslash rsc(v(pc_t))$: $v|_{ProofVar(r)} \in I_r$. Notice that $M_{v(pc_t),t} \subseteq \hat{M}_{v(pc_t),t}$ and for all $r \in Res$ we have $I_r \subseteq \hat{I}_r$. Then $\hat{M}_{v(pc_t),t} \ni v|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(v(pc_t)))}$ and $\forall r \in rsc(p) \backslash rsc(v(pc_t)) : v|_{ProofVar(r)} \in \hat{I}_r$. Then $w \in \hat{M}'_{p,t}$.

∎

Thus the improved Owicki-Gries-core abstract transformer is also monotone.

Proposition 54. *The set of Owicki-Gries-core proofs coincides with the set of postfix points of the improved Owicki-Gries-core abstract transformer. Formally:*

$$\mathfrak{O} = postfp(G^{\#}_{\mathrm{OG}}).$$

Proof. "$\subseteq$": Let $(I,M) \in \mathfrak{O}$ and $(\hat{I},\hat{M}) = p^{\#}_{\mathrm{OG}}(I,M)$. We will show that $(I',M') := G^{\#}_{\mathrm{OG}}(I,M)$ is less than or equal to (I,M).

"$\forall\, r \in Res : I'_r \subseteq I_r$": Let $r \in Res$. Let $w \in I'_r$.

If $w \in I_r^{init}$, there is some $v \in init$ such that $v|_{ProofVar(r)} = w$. By the initial condition, $v \in \gamma_{\mathrm{OG}}(I,M)$. Since every resource is available in an initial state, $r \in Res \setminus \bigcup rsc(v(PCVar))$. Thus $v|_{ProofVar(r)} \in I_r$.

Otherwise $w \in \hat{I}_r$. So there is some $t \in Tid$ and some transition $v \rightarrow_t v'$ such that $v|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(v(pc_t)))} \in M_{v(pc_t),t}$, $r \notin rsc(v'(pc_t))$ and either $r \notin rsc(v(pc_t)) \wedge w \in I_r$ or $r \in rsc(v(pc_t)) \wedge w = v|_{ProofVar(r)}$. Assume for the purpose of contradiction that $w \notin I_r$. Then $r \in rsc(v(pc_t)) \setminus rsc(v'(pc_t)) \wedge w = v|_{ProofVar(r)}$. By (5) we have $rsc(v(pc_t)) \supseteq rsc(v'(pc_t))$. Sequential consistency implies $v'|_{ProofVar(r)} \in I_r$. Contradiction!

We have shown that in both cases $w \in I_r$. So $I'_r \subseteq I_r$.

"$\forall\, t \in Tid, p \in PL : M'_{p,t} \subseteq M_{p,t}$": Let $t \in Tid$, $p \in PL$. Let $w \in M'_{p,t}$.

If $w \in M^{init}_{p,t}$, there is some $v \in init$ such that $v(pc_t) = p$ and $w = v|_{LocalDataVar_t}$. By the initial condition, $v \in \gamma_{\mathrm{OG}}(I,M)$. From definition of γ_{OG} and $rsc(v(pc_t)) = [v \in init]\ \emptyset$ we get $v|_{LocalDataVar_t} \in M_{p,t}$.

Otherwise $w \in \hat{M}_{p,t}$. Assume for the purpose of contradiction that $w \notin M_{p,t}$. By definition of $p^{\#}_{\mathrm{OG}}$ there is a transition $v \rightarrow_t v'$ such that $w = v'|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(p))}$, $v|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(v(pc_t)))} \in M_{v(pc_t),t}$, $v'(pc_t) = p$ and $\forall\, r \in rsc(p) \backslash rsc(v(pc_t))$: $v|_{ProofVar(r)} \in I_r$. From sequential consistency we get $v'|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(p))} \in M_{p,t}$. Contradiction!

In both cases $w \in M_{p,t}$. So $M'_{p,t} \subseteq M_{p,t}$.

"$\supseteq$": Let $(I,M) \in postfp(G^{\#}_{\mathrm{OG}})$, i.e., for $(I',M') = G^{\#}_{\mathrm{OG}}(I,M)$ we have $(I',M') \sqsubseteq (I,M)$. We'll show that (I,M) is an Owicki-Gries-core proof.

Sequential consistency: Let $t \in \mathit{Tid}$, $(v,v') \in \rightarrow_t$, $p = v(pc_t)$, $p' = v'(pc_t)$, $R = rsc(p)$, $R' = rsc(p')$, $v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(p))} \in M_{p,t}$ and $\forall\ r \in R' \setminus R$: $v|_{\mathit{ProofVar}(r)} \in I_r$. By definition of $p^{\#}_{\mathrm{OG}}$ we obtain $v'|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(p'))} \in M'_{p',t} \subseteq M_{p',t}$. If any $r \in R \setminus R'$ is given, the definition of $p^{\#}_{\mathrm{OG}}$ implies $v'|_{\mathit{ProofVar}(r)}$ = [by (3)] $v|_{\mathit{ProofVar}(r)} \in I'_r \subseteq I_r$.

Initial condition: Let $v \in \mathit{init}$.

Let $r \in \mathit{Res} \setminus \bigcup rsc(v(\mathit{PCVar}))$. Then $v|_{\mathit{ProofVar}(r)} \in I^{\mathit{init}}_r \subseteq I'_r \subseteq I_r$.

Now let $t \in \mathit{Tid}$. Then $v|_{\mathit{LocalDataVar}_t} \in M^{\mathit{init}}_{v(pc_t),t}$. Since initial states don't have any busy resources, $v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(v(pc_t)))} \in M^{\mathit{init}}_{v(pc_t),t} \subseteq M'_{v(pc_t),t} \subseteq M_{v(pc_t),t}$.

We have shown $v \in \gamma_{\mathrm{OG}}(I,M)$. Thus $\mathit{init} \subseteq \gamma_{\mathrm{OG}}(I,M)$. ∎

Corollary 55. *The smallest Owicki-Gries-core proof is the least fixpoint of the improved Owicki-Gries-core abstract transformer, formally*

$$lfp(G^{\#}_{\mathrm{OG}}) = \inf \mathfrak{O}\,.$$

Proof. $\inf \mathfrak{O}$ = [Prop. 54] $\inf(postfp(G^{\#}_{\mathrm{OG}}))$ = [Tarski's fixpoint theorem] $lfp(G^{\#}_{\mathrm{OG}})$. ∎

Proposition 56. *The map $p^{\#}_{\mathrm{OG}}$ overapproximates the best abstract successor map with respect to the pair of adjoint maps $(\alpha_{\mathrm{OG}}, \gamma_{\mathrm{OG}})$. Formally:*

$$\forall\ P \in PA: \quad \alpha_{\mathrm{OG}} \circ post \circ \gamma_{\mathrm{OG}}(P) \sqsubseteq p^{\#}_{\mathrm{OG}}(P)\,.$$

Proof. Let $P = (I,M) \in PA$, $(I^L,M^L) = \alpha_{\mathrm{OG}} \circ post \circ \gamma_{\mathrm{OG}}(P)$, $(I^R,M^R) = p^{\#}_{\mathrm{OG}}(P)$. We have to show two facts.

"$\forall\ r \in \mathit{Res} : I^L_r \subseteq I^R_r$": Let $r \in \mathit{Res}$. Let $w \in I^L_r$. Then there is $t \in \mathit{Tid}$, $(v,v') \in \rightarrow_t$ such that $v \in \gamma_{\mathrm{OG}}(P)$, r is available in v' and $v'|_{\mathit{ProofVar}(r)} = w$. By definition of γ_{OG} we have $v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(v(pc_t)))} \in M_{v(pc_t),t}$. There are two cases.

Case $r \in rsc(v(pc_t))$. Then $rsc(v(pc_t)) \neq rsc(v'(pc_t))$. From (3) we conclude $v|_{\mathit{ProofVar}(r)} = v'|_{\mathit{ProofVar}(r)}$. Then $w = v|_{\mathit{ProofVar}(r)}$. By definition of $p^{\#}_{\mathrm{OG}}$ we get $w \in I^R_r$.

Case $r \notin rsc(v(pc_t))$. Then $r \notin rsc(v(pc_t)) \cup rsc(v'(pc_t))$. In particular, $r \notin rsc(v(pc_t)) \cap rsc(v'(pc_t))$. If $x \in \mathit{ProofVar}(r)$, we have $v(x) = v'(x)$ by definition of $\mathit{ProofVar}$. Thus $w = v|_{\mathit{ProofVar}(r)}$. Since r is available in v', it is also available in v, since resources of threads other than t don't change in the transition. By definition of γ_{OG} we get $v|_{\mathit{ProofVar}(r)} \in I_r$, thus $w \in I_r$. By definition of $p^{\#}_{\mathrm{OG}}$ we have $w \in I^R_r$.

We have shown $I^L_r \subseteq I^R_r$.

"$\forall\ p \in PL, t \in \mathit{Tid} : M^L_{p,t} \subseteq M^R_{p,t}$": Let $p \in PL$ and $t \in \mathit{Tid}$. Let $w \in M^L_{p,t}$. Then there is some $\tilde{t} \in \mathit{Tid}$, $(v,v') \in \rightarrow_{\tilde{t}}$ such that $w = v'|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(p))}$, $v \in \gamma_{\mathrm{OG}}(P)$ and $v'(pc_t) = p$. There are two cases.

Case $t \neq \tilde{t}$. By (1) we have $v \sim_{\mathit{Local}_t} v'$. Take any $r \in rsc(p)$; then $r \notin rsc(v'(pc_{\tilde{t}}))$; thus $v \sim_{\mathit{ProofVar}(r)} v'$. So $v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(p))} = w$. From definition of γ_{OG} we get $v|_{\mathit{LocalDataVar}_{\tilde{t}} \cup \bigcup \mathit{ProofVar}(rsc(v(pc_{\tilde{t}})))} \in M_{v(pc_{\tilde{t}}),\tilde{t}}$ and $v|_{\mathit{LocalDataVar}_t \cup \bigcup \mathit{ProofVar}(rsc(v(pc_t)))} \in M_{v(pc_t),t}$. Notice that $v(pc_t) = p$. Thus $w \in M_{p,t}$ and from definition of consistent states we get $rsc(p) \cap rsc(v(pc_{\tilde{t}})) = \emptyset$. Thus $rsc(p) \cap (rsc(v(pc_{\tilde{t}})) \cup rsc(v'(pc_{\tilde{t}}))) = \emptyset$. By definition of $p^{\#}_{\mathrm{OG}}$ we get $w \in M^R_{p,t}$.

Case $t = \tilde{t}$. By definition of γ_{OG} we get $v|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(v(pc_t)))} \in M_{v(pc_t),t}$.

Consider an arbitrary $r \in rsc(p) \setminus rsc(v(pc_t))$ (there may be none). From $v' \in ConsState$ and (1) we get $r \notin \bigcup_{\hat{t} \in Tid \setminus \{t\}} rsc(v(pc_{\hat{t}}))$. Thus r is available in v. By definition of γ_{OG} we get $v|_{ProofVar(r)} \in I_r$. By definition of $p^{\#}_{\mathrm{OG}}$ we get $v'|_{LocalDataVar_t \cup \bigcup ProofVar(rsc(p))} \in M^R_{p,t}$. Thus $w \in M^R_{p,t}$.

We have shown $M^L_{p,t} \subseteq M^R_{p,t}$.

■

The last proposition allows using $p^{\#}_{\mathrm{OG}}$ in the relaxed frontier search.

G.3 Examples

Example 57. Consider the program Upper from Example 2. Let $(T^{(i)})_{i \in \mathbb{N}_0}$ be the relaxed frontier search sequence for Upper. Then:

$$T^{(0)} = \alpha_{\mathrm{OG}}(init) = \left(\begin{bmatrix} r \mapsto \{[u \mapsto 0, z \mapsto 1], [u \mapsto 1, z \mapsto 0]\}, \\ r' \mapsto \{[x \mapsto 0, y \mapsto 0], [x \mapsto 1, y \mapsto 1]\} \end{bmatrix}, \begin{bmatrix} (\mathtt{A}, 1) \mapsto \{[x \mapsto 0, l \mapsto 0], [x \mapsto 1, l \mapsto 1]\}, \\ (\mathtt{O}, 2) \mapsto \{[y \mapsto 0, z \mapsto 0], [y \mapsto 1, z \mapsto 1]\}, \\ (\text{everything else}) \mapsto \emptyset \end{bmatrix} \right),$$

$$p^{\#}_{\mathrm{OG}}(T^{(0)}) = \left(\begin{array}{l} [r \mapsto \emptyset, r' \mapsto \{[x \mapsto 0, y \mapsto 0], [x \mapsto 1, y \mapsto 1]\}], \\ \begin{bmatrix} (\mathtt{B}, 1) \mapsto \{[u \mapsto 0, z \mapsto 1, x \mapsto 0, l \mapsto 0], [u \mapsto 1, z \mapsto 0, x \mapsto 1, l \mapsto 1]\}, \\ (\mathtt{O}, 2) \mapsto \{[y \mapsto 0, z \mapsto 0], [y \mapsto 1, z \mapsto 1]\}, \\ (\text{everything else}) \mapsto \emptyset \end{bmatrix} \end{array} \right)$$

$$\not\sqsubseteq \bigsqcup_{j=0}^{0} T^{(j)}, \text{ so } T^{(1)} = p^{\#}_{\mathrm{OG}}(T^{(0)}),$$

$$p^{\#}_{\mathrm{OG}}(T^{(1)}) = \left(\begin{array}{l} [r \mapsto \emptyset, r' \mapsto \emptyset], \\ \begin{bmatrix} (\mathtt{C}, 1) \mapsto \left\{ \begin{array}{l} [u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 0, l \mapsto 0], \\ [u \mapsto 1, z \mapsto 0, x \mapsto 1, y \mapsto 1, l \mapsto 1] \end{array} \right\}, \\ (\mathtt{O}, 2) \mapsto \{[y \mapsto 0, z \mapsto 0], [y \mapsto 1, z \mapsto 1]\}, \\ (\text{everything else}) \mapsto \emptyset \end{bmatrix} \end{array} \right)$$

$$\not\sqsubseteq \bigsqcup_{j=0}^{1} T^{(j)}, \text{ so } T^{(2)} = p^{\#}_{\mathrm{OG}}(T^{(1)}),$$

$$p^{\#}_{\mathrm{OG}}(T^{(2)}) = \left(\begin{array}{l} [r \mapsto \emptyset, r' \mapsto \emptyset], \\ \begin{bmatrix} (\mathtt{D}, 1) \mapsto \left\{ \begin{array}{l} [u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 0, l \mapsto 0], \\ [u \mapsto 1, z \mapsto 0, x \mapsto 0, y \mapsto 1, l \mapsto 1] \end{array} \right\}, \\ (\mathtt{O}, 2) \mapsto \{[y \mapsto 0, z \mapsto 0], [y \mapsto 1, z \mapsto 1]\}, \\ (\text{everything else}) \mapsto \emptyset \end{bmatrix} \end{array} \right)$$

$$\not\sqsubseteq \bigsqcup_{j=0}^{2} T^{(j)}, \text{ so } T^{(3)} = p^{\#}_{\mathrm{OG}}(T^{(2)}),$$

$$p^{\#}_{\mathrm{OG}}(T^{(3)}) = \left(\begin{array}{l}[r \mapsto \emptyset, r' \mapsto \{[x\mapsto 0, y\mapsto 0], [x\mapsto 0, y\mapsto 1]\}], \\ \left[\begin{array}{l}(\mathtt{E},1) \mapsto \{[u\mapsto 0, z\mapsto 1, x\mapsto 0, l\mapsto 0], [u\mapsto 1, z\mapsto 0, x\mapsto 0, l\mapsto 1]\}, \\ (\mathtt{O},2) \mapsto \{[y\mapsto 0, z\mapsto 0], [y\mapsto 1, z\mapsto 1]\}, \\ (\text{everything else}) \mapsto \emptyset\end{array}\right]\end{array}\right)$$

$$\not\sqsubseteq \bigsqcup_{j=0}^{3} T^{(j)}, \text{ so } T^{(4)} = p^{\#}_{\mathrm{OG}}(T^{(3)}),$$

$$p^{\#}_{\mathrm{OG}}(T^{(4)}) = \big([r\mapsto\emptyset, r'\mapsto\emptyset], (\emptyset)_{p\in PL, t\in Tid}\big) \sqsubseteq \bigsqcup_{j=0}^{4} T^{(j)}, \text{ so}$$

$$T^{(5)} = T^{(4)} \sqcup p^{\#}_{\mathrm{OG}}(T^{(4)}) \;=\; T^{(4)}.$$

From $T^{(5)} \sqsubseteq T^{(4)}$ we get $k = 4$. By Prop. 52, the generated inductive invariant is

$$\bigcup_{j=0}^{k} \gamma_{\mathrm{OG}}(T^{(j)}) = \left\{\begin{array}{l}[u\mapsto 0, z\mapsto 1, x\mapsto 1, y\mapsto 1, l\mapsto 1, pc_1\mapsto\mathtt{A}, pc_2\mapsto\mathtt{O}], \\ [u\mapsto 1, z\mapsto 0, x\mapsto 0, y\mapsto 0, l\mapsto 0, pc_1\mapsto\mathtt{A}, pc_2\mapsto\mathtt{O}], \\ [u\mapsto 0, z\mapsto 1, x\mapsto 0, y\mapsto 1, l\mapsto 0, pc_1\mapsto\mathtt{E}, pc_2\mapsto\mathtt{O}], \\ [u\mapsto 1, z\mapsto 0, x\mapsto 0, y\mapsto 0, l\mapsto 1, pc_1\mapsto\mathtt{E}, pc_2\mapsto\mathtt{O}]\end{array}\right\}.$$

Notice that $\bigcup_{j=0}^{k}\gamma_{\mathrm{OG}}(T^{(j)})$ is strictly stronger than $\gamma_{\mathrm{OG}}(\inf\mathfrak{O})$, e.g., $[u\mapsto 0, z\mapsto 1, x\mapsto 0, y\mapsto 1, l\mapsto 0, pc_1\mapsto\mathtt{A}, pc_2\mapsto\mathtt{O}] \in \gamma_{\mathrm{OG}}(\inf\mathfrak{O}) \setminus \Big(\bigcup_{j=0}^{k}\gamma_{\mathrm{OG}}(T^{(j)})\Big)$. □

H Comparison to modular methods for general programs

Modular methods for general programs are variants of the original Owicki method [25], rely-guarantee reasoning [17], thread-modular model-checking [14], local proofs [7] and multithreaded Cartesian abstraction [20, 22]. We are going to compare the Owicki-Gries core with multithreaded Cartesian abstraction. We will show that the methods for programs with and without resources, when treated without auxiliary variables, have similar but incomparable precision.

Fix a syntactic structure and an RPL transition system obeying this syntactic structure.

Let us consider, for instance, the variant of multithreaded Cartesian approximation on the powerset of states that describes the core of Owicki-Gries for general multithreaded programs exactly:

$$S \mapsto \{\tilde{v} \mid \forall\, t \in Tid\; \exists\, v \in S : \tilde{v} \sim_{\{pc_t\}\cup DataVar} v\}.$$

This approximation is not directly usable for RPL, since its values are not necessarily subsets of *ConsState*. But a similar mapping on the powerset of consistent states $D = \mathfrak{P}(ConsState)$, namely

$$\rho_{\mathrm{OG,mc}} : D \to D,\; S \mapsto \{v \in ConsState \mid \forall\, t \in Tid\; \exists\, \tilde{v} \in S : v \sim_{\{pc_t\}\cup DataVar} \tilde{v}\},$$

is usable for RPL.

Proposition 58. $\rho_{\mathrm{OG,mc}} : D \to D$ *is a closure operator on D.*

Proof. We will prove the defining properties of a closure.

Monotonicity. Let $Q \subseteq Q'$ be sets of consistent states and $v \in \rho_{\mathrm{OG,mc}}(Q)$. Let $t \in \mathit{Tid}$. Then there is some $\tilde{v} \in Q$ such that $v \sim_{\{pc_t\}\cup \mathit{DataVar}} \tilde{v}$. Notice that $\tilde{v} \in Q'$. Thus $v \in \rho_{\mathrm{OG,mc}}(Q')$. We have shown $\rho_{\mathrm{OG,mc}}(Q) \subseteq \rho_{\mathrm{OG,mc}}(Q')$.

Extensivity. Let $Q \in D$. Let $v \in Q$. Certainly $\forall\, t \in \mathit{Tid} : v \sim_{\{pc_t\}\cup \mathit{DataVar}} v$. Thus $v \in \rho_{\mathrm{OG,mc}}(Q)$. We have shown $Q \subseteq \rho_{\mathrm{OG,mc}}(Q')$.

Idempotence. Let $Q \in D$. Let $v \in \rho_{\mathrm{OG,mc}}(\rho_{\mathrm{OG,mc}}(Q))$. Let $t \in \mathit{Tid}$. Then there is some $\tilde{v} \in \rho_{\mathrm{OG,mc}}(Q)$ such that $v \sim_{\{pc_t\}\cup \mathit{DataVar}} \tilde{v}$. Then there is some $\tilde{\tilde{v}} \in Q$ such that $\tilde{v} \sim_{\{pc_t\}\cup \mathit{DataVar}} \tilde{\tilde{v}}$. Then $v \sim_{\{pc_t\}\cup \mathit{DataVar}} \tilde{\tilde{v}}$. Thus $v \in \rho_{\mathrm{OG,mc}}(Q)$. We have shown $\rho_{\mathrm{OG,mc}}(\rho_{\mathrm{OG,mc}}(Q)) \subseteq \rho_{\mathrm{OG,mc}}(Q)$. ∎

Theoretically, other equivalent proof methods (rely-guarantee reasoning, thread-modular model-checking or local proofs) could be adapted to RPL. However, to the best of our knowledge, such adaptations have not been cleanly formulated so far. In their original formulation, they are also not exactly equivalent, e.g., the "locals" of threads in thread-modular reasoning have to be disjoint, the "locals" of threads in RPL may overlap, the "locals" of threads in Owicki-Gries for general multithreaded programs consist only of the program counters.

The question, whether the straightforward adaptation of thread-modular model checking is equivalent to abstract interpretation with $\rho_{\mathrm{OG,mc}}$, is not so easy for RPL. However, at least one direction holds: following the proof of Thm. 1.7.2 in [20], we relatively easily obtain that abstract interpretation with $\rho_{\mathrm{OG,mc}}$ always proves properties at least as strong as those provable by thread-modular model checking.

We are going to proceed with comparison of the Owicki-Gries core with abstract interpretation under the closure $\rho_{\mathrm{OG,mc}}$.

Example 59 (Readers-Writers). For the program Readers-Writers, let $F = \lambda x.\, \rho_{\mathrm{OG,mc}}(\mathit{init} \cup \mathit{post}(x))$ and $Q = \mathit{lfp}(F)$. We will show that Q equals the strongest Owicki-Gries-core-provable property

$$
\begin{aligned}
\{v \in \mathit{ConsState} \mid\ & v(ww), v(ar) \in \mathbb{N}_0 \ \wedge\ v(aw) \in \{0,1\} \\
& \wedge\ \forall\, i \in \mathbb{N}_n : \\
& \qquad (v(pc_{\mathtt{reader}_i}) \in \{x\mathtt{ready}y, \mathtt{read} \mid x \in \{\mathtt{start}, \mathtt{finish}\} \\
& \qquad\quad \wedge\ y \in \{\mathtt{A}, \mathtt{B}, \mathtt{C}\}\}) \\
& \qquad \wedge (v(pc_{\mathtt{reader}_i}) = \mathtt{startreadB} \Rightarrow v(ww) = 0) \\
& \qquad \wedge (v(pc_{\mathtt{reader}_i}) = \mathtt{startreadC} \Rightarrow v(ww) = 0 < v(ar)) \\
& \wedge\ \forall\, j \in \mathbb{N}_m : \\
& \qquad (v(pc_{\mathtt{writer}_j}) \in \{x\mathtt{write}y, \mathtt{write} \mid x \in \{\mathtt{ask}, \mathtt{start}, \mathtt{finish}\} \\
& \qquad\quad \wedge\ y \in \{\mathtt{A}, \mathtt{B}, \mathtt{C}\}\}) \\
& \qquad \wedge (v(pc_{\mathtt{writer}_j}) = \mathtt{askwriteC} \Rightarrow v(ww) > 0) \\
& \qquad \wedge (v(pc_{\mathtt{writer}_j}) = \mathtt{startwriteB} \Rightarrow v(ar) = v(aw) = 0) \\
& \qquad \wedge (v(pc_{\mathtt{writer}_j}) = \mathtt{startwriteC} \Rightarrow v(ar) = 0 < v(aw)) \\
& \qquad \wedge (v(pc_{\mathtt{writer}_j}) = \mathtt{finishwriteC} \Rightarrow v(aw) = 0)\,\}\,.
\end{aligned}
$$

"$\subseteq$": First we show the right hand side (RHS) is closed under $\rho_{\mathrm{OG,mc}}$. Let $v \in \rho_{\mathrm{OG,mc}}(\mathrm{RHS})$. By definition of $\rho_{\mathrm{OG,mc}}$ there is some $w \in \mathrm{RHS}$ such that $v \sim_{\{pc_{\mathtt{reader}_1}, ww, ar, aw\}} w$. Then $v(ww)$, $v(ar) \in \mathbb{N}_0$ and $v(aw) \in \{0,1\}$.
Let $i \in \mathbb{N}_n$. Then there is $w \in \mathrm{RHS}$ such that $v \sim_{\{pc_{\mathtt{reader}_i}, ww, ar, aw\}} w$. Thus $v(pc_{\mathtt{reader}_i})$ $\in \{\mathtt{startreadA}, \mathtt{startreadB}, \mathtt{startreadC}, \mathtt{read}, \mathtt{finishreadA}, \mathtt{finishreadB}, \mathtt{finishreadC}\}$. If $v(pc_{\mathtt{reader}_i}) = \mathtt{startreadB}$, then $w(pc_{\mathtt{reader}_i}) = \mathtt{startreadB}$, so $w(ww) = 0$ by definition of RHS, so $v(ww) = 0$. If $v(pc_{\mathtt{reader}_i}) = \mathtt{startreadC}$, then $w(pc_{\mathtt{reader}_i}) = \mathtt{startreadC}$, so $w(ww) = 0 < w(ar)$ by definition of RHS, so $v(ww) = 0 < v(ar)$.
Let $j \in \mathbb{N}_m$. Then there is $w \in \mathrm{RHS}$ such that $v \sim_{\{pc_{\mathtt{writer}_j}, ww, ar, aw\}} w$. Thus $v(pc_{\mathtt{writer}_j})$ $\in \{\mathtt{askwriteA}, \mathtt{askwriteB}, \mathtt{askwriteC}, \mathtt{startwriteA}, \mathtt{startwriteB}, \mathtt{startwriteC}, \mathtt{write}, \mathtt{finishwriteA}, \mathtt{finishwriteB}, \mathtt{finishwriteC}\}$. If $v(pc_{\mathtt{writer}_j}) = \mathtt{askwriteC}$, then $w(pc_{\mathtt{writer}_j}) = \mathtt{askwriteC}$, so $w(ww) > 0$, so $v(ww) > 0$. If $v(pc_{\mathtt{writer}_j}) = \mathtt{startwriteB}$, then $w(pc_{\mathtt{writer}_j}) = \mathtt{startwriteB}$, so $w(ar) = w(aw) = 0$, so $v(ar) = v(aw) = 0$. If $v(pc_{\mathtt{writer}_j}) = \mathtt{startwriteC}$, then $w(pc_{\mathtt{writer}_j}) = \mathtt{startwriteC}$, so $w(ar) = 0 < w(aw)$, so $v(ar) = 0 < v(aw)$. If $v(pc_{\mathtt{writer}_j}) = \mathtt{finishwriteC}$, then $w(pc_{\mathtt{writer}_j}) = \mathtt{finishwriteC}$, so $w(aw) = 0$, so $v(aw) = 0$.
Thus $v \in \mathrm{RHS}$. Since v was arbitrary, $\rho_{\mathrm{OG,mc}}(\mathrm{RHS}) \subseteq \mathrm{RHS}$. Notice that $\mathit{init} \cup \mathit{post}(\mathrm{RHS}) \subseteq \mathrm{RHS}$, thus $\rho_{\mathrm{OG,mc}}(\mathit{init} \cup \mathit{post}(\mathrm{RHS})) \subseteq \mathrm{RHS}$, so RHS is a postfix point of F, so $Q \subseteq \mathrm{RHS}$.

"$\supseteq$": Let

$$
\begin{aligned}
\mathit{CritR} &= \{x\mathtt{ready}y & &\mid x \in \{\mathtt{start}, \mathtt{finish}\}, y \in \{\mathtt{B}, \mathtt{C}\}\}, \\
\mathit{CritW} &= \{x\mathtt{write}y & &\mid x \in \{\mathtt{ask}, \mathtt{start}, \mathtt{finish}\}, y \in \{\mathtt{B}, \mathtt{C}\}\}, \\
\mathit{NonCritR} &= \{\mathtt{read}, x\mathtt{readA} & &\mid x \in \{\mathtt{start}, \mathtt{finish}\}\}, \\
\mathit{NonCritW} &= \{\mathtt{write}, x\mathtt{writeA} & &\mid x \in \{\mathtt{ask}, \mathtt{start}, \mathtt{finish}\}\}\,.
\end{aligned}
$$

(I) First we will show that for all $a,b \in \mathbb{N}_0$, $c \in \{0,1\}$, the state v with $(\forall\, i \in \mathbb{N}_n : v(pc_{\mathtt{reader}_i}) = \mathtt{startreadA}) \wedge (\forall\, j \in \mathbb{N}_n : v(pc_{\mathrm{writer}_j}) = \mathtt{askwriteA}) \wedge v(ww) = a \wedge v(ar) = b \wedge v(aw) = c$ is in Q.
We will do that in 3 steps.

Step 1. We'll show this claim for $a = b = 0$. If $c = 0$, notice that such a state is in *init* $\subseteq Q$. Otherwise $c = 1$. Take the initial state, make 3 steps of $\mathtt{writer}_1$, resulting in a state $v1 \in Q$ with $v1(pc_{\mathtt{writer}_1}) = \mathtt{startwriteA}$, all other writers are at $\mathtt{askwriteA}$, all readers are at $\mathtt{startreadA}$, $v1(ww) = 1$, $v1(ar) = 0$, $v1(aw) = 0$. Making 3 steps of $\mathtt{writer}_2$ from the initial state results in a state $v2 \in Q$ with $v2(pc_{\mathtt{writer}_2}) = \mathtt{startwriteA}$, all other writers are at $\mathtt{askwriteA}$, all readers are at $\mathtt{startreadA}$, $v2(ww) = 1$, $v2(ar) = 0$, $v2(aw) = 0$. Consider the state $v3$ such that $v3(pc_{\mathtt{writer}_1}) = v3(pc_{\mathtt{writer}_2}) = \mathtt{startwriteA} \wedge (\forall\, j \in \mathbb{N}_m \setminus \{1,2\} : v3(pc_{\mathtt{writer}_j}) = \mathtt{askwriteA}) \wedge (\forall\, i \in \mathbb{N}_n : v3(pc_{\mathtt{reader}_i}) = \mathtt{startreadA}) \wedge v3(ww) = 1 \wedge v3(ar) = v3(aw) = 0$. By definition of $\rho_{\mathrm{OG,mc}}$, $v3 \in Q$. Making seven steps of $\mathtt{writer}_1$ from $v3$ results in a state $v4 \in Q$ such that $(\forall\, i \in \mathbb{N}_n : v4(pc_{\mathtt{reader}_i}) = \mathtt{startreadA}) \wedge v4(pc_{\mathtt{writer}_2}) = \mathtt{startwriteA} \wedge (\forall\, j \in \mathbb{N}_m \setminus \{2\} : v4(pc_{\mathtt{writer}_j}) = \mathtt{askwriteA}) \wedge v4(ww) = v4(ar) = v4(aw) = 0$. Making three steps of $\mathtt{writer}_2$ from $v4$ results in a state $v5 \in Q$ such that $(\forall\, i \in \mathbb{N}_n : v5(pc_{\mathtt{reader}_i}) = \mathtt{startreadA}) \wedge v5(pc_{\mathtt{writer}_2}) = \mathtt{write} \wedge (\forall\, j \in \mathbb{N}_m \setminus \{2\} : v5(pc_{\mathtt{writer}_j}) = \mathtt{askwriteA}) \wedge v5(ww) = v5(ar) = 0 \wedge v5(aw) = 1$. Analogously, starting from $v3$, making seven steps of $\mathtt{writer}_2$ and three steps of $\mathtt{writer}_1$ results in a state $v6 \in Q$ such that $(\forall\, i \in \mathbb{N}_n : v6(pc_{\mathtt{reader}_i}) = \mathtt{startreadA}) \wedge v6(pc_{\mathtt{writer}_1}) = \mathtt{write} \wedge (\forall\, j \in \mathbb{N}_m \setminus \{1\} : v6(pc_{\mathtt{writer}_j}) = \mathtt{askwriteA}) \wedge v6(ww) = v6(ar) = 0 \wedge v6(aw) = 1$. By definition of $\rho_{\mathrm{OG,mc}}$ there is some $v7 \in Q$ such that $(\forall\, i \in \mathbb{N}_n : v7(pc_{\mathtt{reader}_i}) = \mathtt{startreadA}) \wedge (\forall\, j \in \mathbb{N}_m : v7(pc_{\mathtt{writer}_j}) = \mathtt{askwriteA}) \wedge v7(ww) = v7(ar) = 0 \wedge v7(aw) = 1$. Then $v7$ satisfies the stated conditions for v.

Step 2. Now we will show that for all $b \in \mathbb{N}_0$, $c \in \{0,1\}$, the state v with $(\forall\, i \in \mathbb{N}_n : v(pc_{\mathtt{reader}_i}) = \mathtt{startreadA}) \wedge (\forall\, j \in \mathbb{N}_m : v(pc_{\mathtt{writer}_j}) = \mathtt{askwriteA}) \wedge v(ww) = 0 \wedge v(ar) = b \wedge v(aw) = c$ is in Q.
For $b = 0$ the claim has been shown in step 1. Now assume that $b > 0$ and the claim has been proven for $b - 1$. Let $c \in \{0,1\}$. By induction assumption there is a state $v1 \in Q$ in which each thread sits at its initial location and $v1(ww) = 0 \wedge v1(ar) = b - 1 \wedge v1(aw) = c$. Making three steps of $\mathtt{reader}_1$ from $v1$ results in a state $v2 \in Q$ such that $v2(pc_{\mathtt{reader}_1}) = \mathtt{read} \wedge (\forall\, i \in \mathbb{N}_n \setminus \{1\} : v2(pc_{\mathtt{reader}_i}) = \mathtt{startreadA}) \wedge (\forall\, j \in \mathbb{N}_m : v2(pc_{\mathtt{writer}_j}) = \mathtt{askwriteA}) \wedge v2(ww) = 0 \wedge v2(ar) = b \wedge v2(aw) = c$. Making three steps of $\mathtt{reader}_2$ from $v1$, we get a state $v3 \in Q$ such that $v3(pc_{\mathtt{reader}_2}) = \mathtt{read} \wedge (\forall\, i \in \mathbb{N}_n \setminus \{2\} : v3(pc_{\mathtt{reader}_i}) = \mathtt{startreadA}) \wedge (\forall\, j \in \mathbb{N}_m : v3(pc_{\mathtt{writer}_j}) = \mathtt{askwriteA}) \wedge v3(ww) = 0 \wedge v3(ar) = b \wedge v3(aw) = c$. Since Q is a fixpoint of $\rho_{\mathrm{OG,mc}}$, the state v

in which all threads are at their initial locations, $v(ww) = 0 \wedge v(ar) = b \wedge v(aw) = c$ is in Q.

Step 3. Now we will show that for all $a \in \mathbb{N}_0$, $b \in \mathbb{N}_0$, $c \in \{0,1\}$, the state v with $(\forall\, i \in \mathbb{N}_n : v(pc_{\texttt{reader}_i}) = \texttt{startreadA}) \wedge (\forall\, j \in \mathbb{N}_m : v(pc_{\texttt{writer}_j}) = \texttt{askwriteA}) \wedge v(ww) = a \wedge v(ar) = b \wedge v(aw) = c$ is in Q.
For $a = 0$ the claim has been shown in step 2. Now assume that $a > 0$ and the claim has been proven for $a-1$. Let $b \in \mathbb{N}_0$, $c \in \{0,1\}$. By induction assumption there is a state $v1 \in Q$ in which each threads sits at its initial location, $v(ww) = a-1 \wedge v(ar) = b \wedge v(aw) = c$. Making three steps of $\texttt{writer}_1$ from $v1$ results in a state $v2 \in Q$ such that $v2(pc_{\texttt{writer}_1}) = \texttt{startwriteA}$, all other threads are at their initial locations, $v2(ww) = a \wedge v2(ar) = b \wedge v2(aw) = c$. Making three steps of $\texttt{writer}_2$ from $v1$ results in a state $v3 \in Q$ such that $v3(pc_{\texttt{writer}_2}) = \texttt{startwriteA}$, all other threads are at their initial locations, $v3(ww) = a \wedge v3(ar) = b \wedge v3(aw) = c$. From $v2 \in Q$, $v3 \in Q$ and the definition of $\rho_{\mathrm{OG,mc}}$ we obtain that the state v such that all threads in v are at their initial locations, $v(ww) = a \wedge v(ar) = b \wedge v(aw) = c$ is in Q.

(II) We'll show now that for all $a, b \in \mathbb{N}_0$, $c \in \{0,1\}$, $i \in \mathbb{N}_n$ the state v such that $v(pc_{\texttt{reader}_i}) = \texttt{read} \wedge (\forall\, k \in \mathbb{N}_n \setminus \{i\} : v(pc_{\texttt{reader}_k}) = \texttt{startreadA}) \wedge (\forall\, j \in \mathbb{N}_m : v(pc_{\texttt{writer}_j}) = \texttt{askwriteA}) \wedge v(ww) = a \wedge v(ar) = b \wedge v(aw) = c$ is in Q. We'll use induction on a.

Case $a = 0$. Let $b \in \mathbb{N}$, $c \in \{0,1\}$, $i \in \mathbb{N}_n$. From (I) we already know that there is some $v1 \in Q$ in which all threads are at their initial locations, $v1(ww) = 0 \wedge v1(ar) = b \wedge v1(ar) = c$. Making three steps by thread $\texttt{reader}_i$ results in a state $v2 \in Q$ such that $v2(pc_{\texttt{reader}_i}) = \texttt{read}$, all other threads are at their initial location, $v2(ww) = 0 \wedge v2(ar) = b+1 \wedge v2(aw) = c$. Since $n \geq 2$, there is some $k \in \mathbb{N}_n \setminus \{i\}$. Making three steps from $v1$ by thread $\texttt{reader}_k$ results in a state $v3 \in Q$ such that $v3(pc_{\texttt{reader}_k}) = \texttt{read}$, all other threads are at their initial locations, $v3(ww) = 0 \wedge v3(ar) = b+1 \wedge v3(aw) = c$. Combining $v2 \in Q$, $v3 \in Q$ and using the definition of $\rho_{\mathrm{OG,mc}}$ we obtain $v4 \in Q$ such that $v4(pc_{\texttt{reader}_i}) = v4(pc_{\texttt{reader}_k}) = \texttt{read}$, all threads except $\texttt{reader}_i$ and $\texttt{reader}_k$ are at their corresponding initial location, $v4(ww) = 0$, $v4(ar) = b+1$ and $v4(aw) = c$. Making four steps from $v4$ by thread $\texttt{reader}_k$ results in the state $v \in Q$ such that $v(pc_{\texttt{reader}_i}) = \texttt{read} \wedge (\forall\, l \in \mathbb{N}_n \setminus \{i\} : v(pc_{\texttt{reader}_l}) = \texttt{startreadA}) \wedge v(ww) = 0 \wedge v(ar) = b \wedge v(aw) = c$.

Case $a > 0$. Let $b \in \mathbb{N}$, $c \in \{0,1\}$, $i \in \mathbb{N}_n$. By assumption hypothesis there is $v1 \in Q$ such that $v1(pc_{\texttt{reader}_i}) = \texttt{read}$, all other threads are at their initial locations, $v1(ww) = a-1 \wedge v1(ar) = b \wedge v1(aw) = c$. Taking three steps of $\texttt{writer}_1$ from $v1$ results in a state $v2 \in Q$ such that $v2(pc_{\texttt{reader}_i}) = \texttt{read} \wedge v2(pc_{\texttt{writer}_1}) = \texttt{startwriteA}$, all other threads are at their initial locations, $v2(ww) = a \wedge v2(ar) = b \wedge v2(aw) = c$. In particular, $v2(pc_{\texttt{writer}_2}) = \texttt{askwriteA}$. Taking three steps of $\texttt{writer}_2$ from $v1$ results in a state $v3 \in Q$ such that $v3(pc_{\texttt{reader}_i}) = \texttt{read} \wedge v3(pc_{\texttt{writer}_2}) = \texttt{startwriteA}$, all other threads are at their initial locations, $v3(ww) = a$

$\wedge\ v3(ar) = b \wedge v3(aw) = c$. In particular, $v3(pc_{\texttt{writer}_1}) = \texttt{askwriteA}$. Combining $v2 \in Q$ and $v3 \in Q$ we get that the state v with $v(pc_{\texttt{reader}_i}) = \texttt{read}$, all other threads are at their initial locations, $v(ww) = a \wedge v(ar) = b \wedge v(aw) = c$ is in Q.

(III) We claim that for every $i \in \mathbb{N}_n$, $a,b \in \mathbb{N}_0$, $c \in \{0,1\}$, the state v such that $v(pc_{\texttt{reader}_i}) = \texttt{finishreadA} \wedge (\forall\, k \in \mathbb{N}_n \setminus \{i\} : v(pc_{\texttt{reader}_k}) = \texttt{startreadA}) \wedge (\forall\, j \in \mathbb{N}_m : v(pc_{\texttt{reader}_j}) = \texttt{askwriteA}) \wedge v(ww) = a \wedge v(ar) = b \wedge v(aw) = c$ is in Q.
To see that, make one step from a state described in (II).

(IV) We claim that for all $a,b \in \mathbb{N}_0$, $c \in \{0,1\}$, $j \in \mathbb{N}_m$, the state v such that $(\forall\, i \in \mathbb{N}_n : v(pc_{\texttt{reader}_i}) = \texttt{startreadA}) \wedge v(pc_{\texttt{writer}_j}) = \texttt{startwriteA} \wedge (\forall\, k \in \mathbb{N}_m \setminus \{j\} : v(pc_{\texttt{writer}_k}) = \texttt{askwriteA}) \wedge v(ww) = a \wedge v(ar) = b \wedge v(aw) = c$ is in Q. Let $a \in \mathbb{N}_0$. We case split on a.
Case $a = 0$. We'll induct on b.
Case $b = 0$. Let $j \in \mathbb{N}_m$, $c \in \{0,1\}$. Making from the initial state (which is in Q) three steps of $\texttt{writer}_j$ results in $v1 \in Q$ such that $v1(pc_{\texttt{writer}_j}) = \texttt{startwriteA}$, all other threads are at their initial locations, $v1(ww) = 1 \wedge v1(ar) = v1(aw) = 0$. Since $m \geq 2$, there is some $k \in \mathbb{N}_m \setminus \{j\}$. Taking three steps of $\texttt{writer}_k$ from the initial state results in $v2 \in Q$ such that $v2(pc_{\texttt{writer}_k}) = \texttt{startwriteA}$, all other threads are at their initial locations, $v2(ww) = 1 \wedge v2(ar) = v2(aw) = 0$. Combining $v1 \in Q$ and $v2 \in Q$, we obtain a state $v3 \in Q$ such that $v3(pc_{\texttt{writer}_j}) = v3(pc_{\texttt{writer}_k}) = \texttt{startwriteA}$, all other threads are at their initial locations, $v3(ww) = 1 \wedge v3(ar) = v3(aw) = 0$. Making seven steps of $\texttt{writer}_k$ from $v3$ results in a state $v4 \in Q$ such that $v4(pc_{\texttt{writer}_j}) = \texttt{startwriteA}$, all other threads are at their initial locations, $v4(ww) = v4(ar) = v4(aw) = 0$.
Case $c = 0$. Notice that $v4$ satisfies all the requirements for v.
Case $c = 1$. Making seven steps of $\texttt{writer}_j$ from $v3$ results in a state $v5 \in Q$ such that $v5(pc_{\texttt{writer}_k}) = \texttt{startwriteA}$, all other threads are at their initial locations, $v5(ww) = v5(ar) = v5(aw) = 0$. Combining $v4 \in Q$, $v5 \in Q$ and the definition of $\rho_{\mathrm{OG,mc}}$, results in a state $v6 \in Q$ such that $v6(pc_{\texttt{writer}_j}) = v6(pc_{\texttt{writer}_k}) = \texttt{startwriteA}$, all other threads are at their initial locations, $v6(ww) = v6(ar) = v6(aw) = 0$. Making three steps of $\texttt{writer}_k$ from $v6$ results in a state $v7 \in Q$ such that $v7(pc_{\texttt{writer}_j}) = \texttt{startwriteA} \wedge v7(pc_{\texttt{writer}_k}) = \texttt{write}$, all other threads are at their initial locations, $v7(ww) = v7(ar) = 0 \wedge v7(aw) = 1$. By (I) there is some state $v8 \in Q$ in which all threads are at their initial locations and $v8(ww) = v8(ar) = 0 \wedge v8(aw) = 1$. Combining $v7 \in Q$ and $v8 \in Q$, we obtain the state $v \in Q$ such that $v(pc_{\texttt{writer}_j}) = \texttt{startwriteA}$, all other threads are at their initial locations, and $v(ww) = v(ar) = 0 \wedge v(aw) = 1$.
Case $b \geq 1$. Let $j \in \mathbb{N}_m$, $c \in \{0,1\}$. By induction assumption there is some $v1 \in Q$ such that $v1(pc_{\texttt{writer}_j}) = \texttt{startwriteA}$, all other

threads are at their initial locations, $v1(ww) = 0 \wedge v1(ar) = b-1 \wedge v1(aw) = c$. Making three steps of $\texttt{reader}_1$ from $v1$ results in a state $v2 \in Q$ such that $v2(pc_{\texttt{reader}_1}) = \texttt{read} \wedge v2(pc_{\texttt{writer}_j}) = \texttt{startwriteA}$, all other threads are at their initial locations, $v2(ww) = 0 \wedge v2(ar) = b \wedge v2(aw) = c$. Making three steps of $\texttt{reader}_2$ from $v1$ results in a state $v3 \in Q$ such that $v3(pc_{\texttt{reader}_2}) = \texttt{read} \wedge v3(pc_{\texttt{writer}_j}) = \texttt{startwriteA}$, all other threads are at their initial locations, $v3(ww) = 0 \wedge v3(ar) = b \wedge v3(aw) = c$. Notice that $v2(pc_{\texttt{reader}_2}) = v3(pc_{\texttt{reader}_1}) = \texttt{startreadA}$. From $v2 \in Q$, $v3 \in Q$ we get the state $v \in Q$ such that $v(pc_{\texttt{writer}_j}) = \texttt{startwriteA}$, all other threads are at their initial locations, $v(ww) = 0 \wedge v(ar) = b \wedge v(aw) = c$.

Case $a \geq 1$. Let $j \in \mathbb{N}_m$, $b \in \mathbb{N}_0$, $c \in \{0,1\}$. From (I) we obtain a state $v1 \in Q$ in which all threads are at their initial locations, $v1(ww) = a-1 \wedge v1(ar) = b \wedge v1(aw) = c$. Making three steps by $\texttt{writer}_j$ results in a state $v \in Q$ such that $v(pc_{\texttt{writer}_j}) = \texttt{startwriteA}$, all other threads are at their initial locations, $v(ww) = a \wedge v(ar) = b \wedge v(aw) = c$.

(V) We claim that for all $a, b \in \mathbb{N}_0$, $c \in \{0,1\}$, $j \in \mathbb{N}_m$ the state v such that $(\forall\, i \in \mathbb{N}_n : v(pc_{\texttt{reader}_i}) = \texttt{startreadA}) \wedge v(pc_{\texttt{writer}_j}) = \texttt{write} \wedge (\forall\, k \in \mathbb{N}_m \setminus \{j\} : v(pc_{\texttt{writer}_k}) = \texttt{askwriteA}) \wedge v(ww) = a \wedge v(ar) = b \wedge v(aw) = c$ is in Q. We'll use induction on a.

Case $a = 0$. We'll use induction on b.

Case $b = 0$. Let $c \in \{0,1\}$, $j \in \mathbb{N}_m$.

Case $c = 0$. Making six steps of $\texttt{writer}_j$ from the initial state results in a state $v1 \in Q$ such that $v1(pc_{\texttt{writer}_j}) = \texttt{write}$, other threads are at their initial locations, $v1(ww) = v1(aw) = 1$ and $v1(ar) = 0$. Let $k \in \mathbb{N}_m \setminus \{j\}$. Making six steps of $\texttt{writer}_k$ from the initial state results in a state $v2 \in Q$ such that $v2(pc_{\texttt{writer}_j}) = \texttt{write}$, other threads are at their initial locations, $v2(ww) = v2(aw) = 1$ and $v2(ar) = 0$. Combining $v2 \in Q$, $v3 \in Q$ and $k \neq j$, we obtain a state $v3 \in Q$ such that $v3(pc_{\texttt{writer}_j}) = v3(pc_{\texttt{writer}_k}) = \texttt{write}$, other threads are at their initial locations, $v3(ww) = v3(aw) = 1$ and $v3(ar) = 0$. Making four steps of $\texttt{writer}_k$ from $v3$ results in the state $v \in Q$ such that $v(pc_{\texttt{writer}_j}) = \texttt{write}$, all other threads are at their initial locations, $v(ww) = v(aw) = v(ar) = 0$.

Case $c = 1$. By (IV) the state $v1$ such that $v1(pc_{\texttt{writer}_j}) = \texttt{startwriteA}$, all other threads are at their initial locations, $v1(ww) = v1(ar) = v1(aw) = 0$ is in Q. Make three steps by thread $\texttt{writer}_j$.

Case $b \geq 1$. Let $c \in \{0,1\}$, $j \in \mathbb{N}_m$.

By induction assumption, there is a state $v1 \in Q$ such that $v1(pc_{\texttt{writer}_j}) = \texttt{write}$, other threads are at their initial locations, $v1(ww) = 0 \wedge v1(ar) = b-1 \wedge v1(aw) = c$. Making three steps of $\texttt{reader}_1$ from $v1$ results in a state $v2 \in Q$ such that $v2(pc_{\texttt{writer}_j}) = \texttt{write}$, $v2(pc_{\texttt{reader}_1}) = \texttt{read}$, all other threads (including $\texttt{reader}_2$) are at their initial locations, $v2(ww) = 0 \wedge v2(ar) = b \wedge v2(aw) = c$. Making three steps of $\texttt{reader}_2$ from $v1$ results in a state $v3 \in Q$ such that $v3(pc_{\texttt{writer}_j}) = \texttt{write}$, $v3(pc_{\texttt{reader}_2}) = \texttt{read}$, all other threads (including $\texttt{reader}_1$) are at their initial locations, $v3(ww) = 0 \wedge v3(ar) = b \wedge v3(aw) = c$. Combining $v2 \in Q$ and $v3 \in Q$ we get the state $v \in Q$ such that $v(pc_{\texttt{writer}_j}) = \texttt{write}$, all other threads are at their initial locations and $v(ww) = 0 \wedge v(ar) = b \wedge v(aw) = c$.

Case $a \geq 1$. Let $b \in \mathbb{N}_0$, $c \in \{0,1\}$, $j \in \mathbb{N}_m$. By induction hypothesis, there is some $v1 \in Q$ such that $v1(pc_{\texttt{writer}_j}) = \texttt{write}$, all other threads are at their initial locations, $v1(ww) = a-1$, $v1(ar) = b$, $v1(aw) = c$. Since $m \geq 2$, there is $k \in \mathbb{N}_m$ such that $k \neq j$. Making three steps of $\texttt{writer}_k$ from $v1$ results in a state $v2 \in Q$ such that $v2(pc_{\texttt{writer}_j}) = \texttt{write}$, $v2(pc_{\texttt{writer}_k}) = \texttt{startwriteA}$, $v2(ww) = a$, $v2(ar) = b$, $v2(aw) = c$. By (I), there is a state $v3 \in Q$ in which all threads are at their initial locations, $v3(ww) = a$, $v3(ar) = b$, $v3(aw) = c$. In particular, $v3(pc_{\texttt{writer}_k}) = \texttt{askwriteA}$. From $v2 \in Q$ and $v3 \in Q$ we obtain that the state v such that $v(pc_{\texttt{writer}_j}) = \texttt{write}$, all other threads are at their initial locations, $v(ww) = a \wedge v(ar) = b \wedge v(aw) = c$ is in Q.

(VI) We claim that for all $a,b \in \mathbb{N}_0$, $c \in \{0,1\}$, $j \in \mathbb{N}_m$ the state $v \in Q$ such that $(\forall\, i \in \mathbb{N}_n : v(pc_{\texttt{reader}_i}) = \texttt{startreadA}) \wedge v(pc_{\texttt{writer}_j}) = \texttt{finishwriteA} \wedge (\forall\, k \in \mathbb{N}_m \setminus \{j\} : v(pc_{\texttt{writer}_k}) = \texttt{askwriteA}) \wedge v(ww) = a \wedge v(ar) = b \wedge v(aw) = c$ is in Q.

To see that, let $a,b \in \mathbb{N}_0$, $c \in \{0,1\}$, $j \in \mathbb{N}_m$, and make one step of $\texttt{writer}_j$ from the corresponding state from (V).

(VII) We claim that for all $a,b \in \mathbb{N}_0$, $c \in \{0,1\}$, any state v such that $(\forall\, i \in \mathbb{N}_n : v(pc_{\texttt{reader}_i}) \in \mathit{NonCritR}) \wedge (\forall\, j \in \mathbb{N}_m : v(pc_{\texttt{writer}_j}) \in \mathit{NonCritW}) \wedge v(ww) = a \wedge v(ar) = b \wedge v(aw) = c$ is in Q.

This follows from (I), (II), (III), (IV), (V), (VI) and the definition of $\rho_{\mathrm{OG,mc}}$.

(VIII) We claim that for all $b \in \mathbb{N}_0$, $c \in \{0,1\}$, $i \in \mathbb{N}_n$, any state v such that $v(pc_{\texttt{reader}_i}) = \texttt{startreadB} \wedge (\forall\, k \in \mathbb{N}_n \setminus \{i\} : v(pc_{\texttt{reader}_k}) \in \mathit{NonCritR}) \wedge (\forall\, j \in \mathbb{N}_m : v(pc_{\texttt{writer}_j}) \in \mathit{NonCritW}) \wedge v(ww) = 0 \wedge v(ar) = b \wedge v(aw) = c$ is in Q.

To see that, let $b \in \mathbb{N}_0$, $c \in \{0,1\}$, $i \in \mathbb{N}_n$, and v as above. By (VII), the state $v1 = v[pc_{\texttt{reader}_i} \mapsto \texttt{startreadA}]$ is in Q. Make one step of $\texttt{reader}_i$.

(IX) We claim that for all $b \in \mathbb{N}^+$, $c \in \{0,1\}$, $i \in \mathbb{N}_n$, any state v such that $v(pc_{\texttt{reader}_i}) = \texttt{startreadC} \wedge (\forall\, k \in \mathbb{N}_n \setminus \{i\} : v(pc_{\texttt{reader}_k}) \in \mathit{NonCritR}) \wedge (\forall\, j \in \mathbb{N}_m : v(pc_{\texttt{writer}_j}) \in \mathit{NonCritW}) \wedge v(ww) = 0 \wedge v(ar) = b \wedge v(aw) = c$ is in Q.

To see that, let $b \in \mathbb{N}^+$, $c \in \{0,1\}$, $i \in \mathbb{N}_n$ and v as stated. By (VIII) the state $v1 = v[pc_{\texttt{reader}_i} \mapsto \texttt{startreadB}, ar \mapsto b-1]$ is in Q. Make one step of thread $\texttt{reader}_i$.

(X) We claim that for all $a,b \in \mathbb{N}_0$, $c \in \{0,1\}$, $i \in \mathbb{N}_n$, any state v such that $v(pc_{\texttt{reader}_i}) = \texttt{finishreadB} \wedge (\forall\, k \in \mathbb{N}_n \setminus \{i\} : v(pc_{\texttt{reader}_k}) \in \mathit{NonCritR}) \wedge (\forall\, j \in \mathbb{N}_m : v(pc_{\texttt{writer}_j}) \in \mathit{NonCritW}) \wedge v(ww) = a \wedge v(ar) = b \wedge v(aw) = c$ is in Q.

To see that, let $a,b \in \mathbb{N}_0$, $c \in \{0,1\}$, $i \in \mathbb{N}_n$, v as required. By (VII), the state $v1 = v[pc_{\texttt{reader}_i} \mapsto \texttt{finishreadA}]$ is in Q. Take one step by $\texttt{reader}_i$ from $v1$.

(XI) We claim that for all $a,b \in \mathbb{N}_0$, $c \in \{0,1\}$, $i \in \mathbb{N}_n$, any state v such that $v(pc_{\texttt{reader}_i}) = \texttt{finishreadC} \wedge (\forall\, k \in \mathbb{N}_n \setminus \{i\} : v(pc_{\texttt{reader}_k}) \in \mathit{NonCritR}) \wedge (\forall\, j \in \mathbb{N}_m : v(pc_{\texttt{writer}_j}) \in \mathit{NonCritW}) \wedge v(ww) = a \wedge v(ar) = b \wedge v(aw) = c$ is in Q.

To see that, let $a,b \in \mathbb{N}_0$, $c \in \{0,1\}$, $i \in \mathbb{N}_n$ and v as above. By (X), the state $v1 = v[pc_{\texttt{reader}_i} \mapsto \texttt{finishreadB}, ar \mapsto b+1]$ is in Q. Take one step by thread $\texttt{reader}_i$.

(XII) We claim that for all $a,b \in \mathbb{N}_0$, $c \in \{0,1\}$, $j \in \mathbb{N}_m$ any state v such that $(\forall\, i \in \mathbb{N}_n : v(pc_{\texttt{reader}_i}) \in \mathit{NonCritR}) \wedge v(pc_{\texttt{writer}_j}) = \texttt{askwriteB} \wedge (\forall\, k \in \mathbb{N}_m \setminus \{j\} : v(pc_{\texttt{writer}_k}) \in \mathit{NonCritW}) \wedge v(ww) = a \wedge v(ar) = b \wedge v(aw) = c$ is in Q.

To see that, let $a,b \in \mathbb{N}_0$, $c \in \{0,1\}$, $j \in \mathbb{N}_m$, v as stated. By (VII), the state $v1 = v[pc_{\texttt{writer}_j} \mapsto \texttt{askwriteA}]$ is in Q. Apply one step of $\texttt{writer}_j$.

(XIII) We claim that for all $a \in \mathbb{N}^+$, $b \in \mathbb{N}_0$, $c \in \{0,1\}$, $j \in \mathbb{N}_m$ any state v such that $(\forall\, i \in \mathbb{N}_n : v(pc_{\texttt{reader}_i}) \in \mathit{NonCritR}) \wedge v(pc_{\texttt{writer}_j}) = \texttt{askwriteC} \wedge (\forall\, k \in \mathbb{N}_m \setminus \{j\} : v(pc_{\texttt{writer}_k}) \in \mathit{NonCritW}) \wedge v(ww) = a \wedge v(ar) = b \wedge v(aw) = c$ is in Q.

To see that, let $a \in \mathbb{N}^+$, $b \in \mathbb{N}_0$, $c \in \{0,1\}$, $j \in \mathbb{N}_m$ and v as stated. By (XII) the state $v1 = v[pc_{\texttt{writer}_j} \mapsto \texttt{askwriteB}, ww \mapsto a-1]$ is in Q. Apply one step of $\texttt{writer}_j$ to $v1$.

(XIV) We claim that for all $a \in \mathbb{N}_0$, $j \in \mathbb{N}_m$ any state v such that $(\forall\, i \in \mathbb{N}_n : v(pc_{\texttt{reader}_i}) \in \mathit{NonCritR}) \wedge v(pc_{\texttt{writer}_j}) = \texttt{startwriteB} \wedge (\forall\, k \in \mathbb{N}_m \setminus \{j\} : v(pc_{\texttt{writer}_k}) \in \mathit{NonCritW}) \wedge v(ww) = a \wedge v(ar) = v(aw) = 0$ is in Q.

To see that, let $a \in \mathbb{N}_0$, $j \in \mathbb{N}_m$, v as stated. By (VII) the state $v1 = v[pc_{\texttt{writer}_j} \mapsto \texttt{startwriteA}]$ is in Q. Apply one step of $\texttt{writer}_j$ to $v1$.

(XV) We claim that for all $a \in \mathbb{N}_0$, $j \in \mathbb{N}_m$ any state v such that $(\forall\, i \in \mathbb{N}_n : v(pc_{\texttt{reader}_i}) \in \mathit{NonCritR}) \wedge v(pc_{\texttt{writer}_j}) = \texttt{startwriteC} \wedge (\forall\, k \in \mathbb{N}_m \setminus \{j\} : v(pc_{\texttt{writer}_k}) \in \mathit{NonCritW}) \wedge v(ww) = a \wedge v(ar) = 0 \wedge v(aw) = 1$ is in Q.

To see that, let $a \in \mathbb{N}_0$, $j \in \mathbb{N}_m$ and v as stated. By (XIV) the state $v1 = v[pc_{\texttt{writer}_j} \mapsto \texttt{startwriteB}, aw \mapsto 0]$ is in Q. Make one step from $v1$ by thread $\texttt{writer}_j$.

(XVI) We claim that for all $a,b \in \mathbb{N}_0$, $c \in \{0,1\}$, $j \in \mathbb{N}_m$ any state v such that $(\forall\, i \in \mathbb{N}_n : v(pc_{\texttt{reader}_i}) \in \mathit{NonCritR}) \wedge v(pc_{\texttt{writer}_j}) = \texttt{finishwriteB} \wedge (\forall\, k \in \mathbb{N}_m \setminus \{j\} : v(pc_{\texttt{writer}_k}) \in \mathit{NonCritW}) \wedge v(ww) = a \wedge v(ar) = b \wedge v(aw) = c$ is in Q.

To see that, let $a,b \in \mathbb{N}_0$, $c \in \{0,1\}$, $j \in \mathbb{N}_m$, v as stated. By (VII) the state $w = v[pc_{\texttt{writer}_j} \mapsto \texttt{finishwriteA}]$ is in Q. Apply one step of $\texttt{writer}_j$ from w.

(XVII) We claim that for all $a,b \in \mathbb{N}_0$, $j \in \mathbb{N}_m$ any state v such that $(\forall\, i \in \mathbb{N}_n : v(pc_{\texttt{reader}_i}) \in \mathit{NonCritR}) \wedge v(pc_{\texttt{writer}_j}) = \texttt{finishwriteC} \wedge (\forall\, k \in \mathbb{N}_m \setminus \{j\} : v(pc_{\texttt{writer}_k}) \in \mathit{NonCritW}) \wedge v(ww) = a \wedge v(ar) = b \wedge v(aw) = 0$ is in Q.

To see that, let $a,b \in \mathbb{N}_0$, $j \in \mathbb{N}_m$, v as stated. By (XVI) the state $w = v[pc_{\mathtt{writer}_j} \mapsto \mathtt{finishwriteB}, aw \mapsto 1]$ is in Q. Apply one step of $\mathtt{writer}_j$ from w.

Let $v \in$ RHS. If *control* is available in v, apply (VII). Otherwise there is some thread at a critical location ...B or ...C. If the critical location is in a reader, apply one of (VIII), (IX), (X), (XI), if the critical location is in a writer, apply one of (XII), (XIII), (XIV), (XV), (XVI), (XVII).

Thus the precision of the Owicki-Gries core and abstract interpretation with $\rho_{\mathrm{OG,mc}}$ coincide on Readers-Writers. □

Example 60 (Upper). Consider the RPL transition system Upper. We'll show that $\rho_{\mathrm{OG,mc}}(init)$ is a subset of *init*.

Let $v \in \rho_{\mathrm{OG,mc}}(init)$. For the left thread we have some $v1 \in init$ such that $v \sim_{\{pc_1,u,z,x,y,l\}} v1$. Thus $v(pc_1) = \mathtt{A}$ and $v(u) \neq v(z) = v(x) = v(y) = v(l)$. For the right thread we have some $v2 \in init$ such that $v \sim_{\{pc_2,u,z,x,y,l\}} v2$. Thus $v(pc_2) = \mathtt{0}$. Then $v \in init$.

Then $\rho_{\mathrm{OG,mc}}(init \cup post(init)) \subseteq \rho_{\mathrm{OG,mc}}(init) \subseteq init$. Thus $lfp(\lambda x.\, \rho_{\mathrm{OG,mc}}(init \cup post(x))) = init$.

Thus the result of abstract interpretation with $\rho_{\mathrm{OG,mc}}$ is strictly stronger than the strongest Owicki-Gries-core-provable property. □

Example 61 (SepThreads). Consider an RPL transition system from SepThreads. For the syntactic structure StrSepThreads and $Q \in D$ we have $\rho_{\mathrm{OG,mc}}(Q) = \{v \in ConsState \mid \forall t \in Tid\ \exists \tilde{v} \in Q\colon v \sim_{\{pc_t\} \cup DataVar} \tilde{v}\} \subseteq \{v \in ConsState \mid \forall t \in Tid\ \exists \tilde{v} \in Q\colon v \sim_{Local_t} \tilde{v}\} = \bar{\rho}(Q)$. Thus $lfp(\lambda x.\, init \cup post(x)) \subseteq lfp(\lambda x.\, \rho_{\mathrm{OG,mc}}(init \cup post(x))) \subseteq lfp(\lambda x.\, \bar{\rho}(init \cup post(x))) \subseteq$ [Example 22] $lfp(\lambda x.\, init \cup post(x)) = \gamma_{\mathrm{OG}}(\inf \mathfrak{O})$. Then $\gamma_{\mathrm{OG}}(\inf \mathfrak{O}) = lfp(\lambda x.\, \rho_{\mathrm{OG,mc}}(init \cup post(x)))$. The precision of Owicki-Gries-core and abstract interpretation with $\rho_{\mathrm{OG,mc}}$ is the same on RPL transition systems from SepThreads. □

Example 62. In Fig. 6 we see two threads executing the same code, together with its smallest Owicki-Gries-core proof.

`resource` $r(x)$
`initially` $x = 0$;

`A:` $\{true\}$	`A:` $\{true\}$
`with` r `do`	`with` r `do`
`B:` $\{x = 0\}$	`B:` $\{x = 0\}$
$x := 1$;	$x := 1$;
`C:` $\{x = 1\}$	`C:` $\{x = 1\}$
`assume` *false*	`assume` *false*
`D:` $\{false\}$	`D:` $\{false\}$
`endwith`	`endwith`
`E:` $\{false\}$	`E:` $\{false\}$

$I_r = \{x = 0\}$

Fig. 6: RPL transition system progAssign1with its smallest Owicki-Gries-core proof.

The proof denotes the property

$$\gamma_{\mathrm{OG}}(\inf \mathfrak{O}) \;=\; \left\{ \begin{array}{l} [x\mapsto 0, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{A}], \\ [x\mapsto 0, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{B}],\ [x\mapsto 0, pc_1\mapsto \mathtt{B}, pc_2\mapsto \mathtt{A}], \\ [x\mapsto 1, pc_1\mapsto \mathtt{C}, pc_2\mapsto \mathtt{A}],\ [x\mapsto 1, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{C}] \end{array} \right\}.$$

Consider the upper fixpoint iteration sequence $(X^{(i)})_{i\in\omega}$ for $\lambda x.\, \rho_{\mathrm{OG,mc}}(\mathit{init} \cup \mathit{post}(x))$:

$$\begin{aligned}
X^{(0)} &= \emptyset, \\
X^{(1)} &= \{[x\mapsto 0, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{A}]\}, \\
X^{(2)} &= \left\{ \begin{array}{l} [x\mapsto 0, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{A}], \\ [x\mapsto 0, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{B}], [x\mapsto 0, pc_1\mapsto \mathtt{B}, pc_2\mapsto \mathtt{A}] \end{array} \right\}, \\
\mathit{init} \cup \mathit{post}(X^{(2)}) &= \left\{ \begin{array}{l} [x\mapsto 0, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{A}], \\ [x\mapsto 0, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{B}], [x\mapsto 0, pc_1\mapsto \mathtt{B}, pc_2\mapsto \mathtt{A}], \\ [x\mapsto 1, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{C}], [x\mapsto 1, pc_1\mapsto \mathtt{C}, pc_2\mapsto \mathtt{A}] \end{array} \right\}, \\
X^{(3)} &= \left\{ \begin{array}{l} [x\mapsto 0, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{A}], [x\mapsto 1, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{A}] \\ [x\mapsto 0, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{B}], [x\mapsto 0, pc_1\mapsto \mathtt{B}, pc_2\mapsto \mathtt{A}], \\ [x\mapsto 1, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{C}], [x\mapsto 1, pc_1\mapsto \mathtt{C}, pc_2\mapsto \mathtt{A}] \end{array} \right\}, \\
\mathit{init} \cup \mathit{post}(X^{(3)}) &= \left\{ \begin{array}{l} [x\mapsto 0, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{A}], \\ [x\mapsto 0, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{B}], [x\mapsto 0, pc_1\mapsto \mathtt{B}, pc_2\mapsto \mathtt{A}], \\ [x\mapsto 1, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{B}], [x\mapsto 1, pc_1\mapsto \mathtt{B}, pc_2\mapsto \mathtt{A}], \\ [x\mapsto 1, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{C}], [x\mapsto 1, pc_1\mapsto \mathtt{C}, pc_2\mapsto \mathtt{A}] \end{array} \right\}, \\
X^{(4)} &= \left\{ \begin{array}{l} [x\mapsto 0, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{A}], [x\mapsto 1, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{A}], \\ [x\mapsto 0, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{B}], [x\mapsto 0, pc_1\mapsto \mathtt{B}, pc_2\mapsto \mathtt{A}], \\ [x\mapsto 1, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{B}], [x\mapsto 1, pc_1\mapsto \mathtt{B}, pc_2\mapsto \mathtt{A}], \\ [x\mapsto 1, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{C}], [x\mapsto 1, pc_1\mapsto \mathtt{C}, pc_2\mapsto \mathtt{A}] \end{array} \right\}, \\
\mathit{init} \cup \mathit{post}(X^{(4)}) &= \left\{ \begin{array}{l} [x\mapsto 0, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{A}], \\ [x\mapsto 0, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{B}], [x\mapsto 0, pc_1\mapsto \mathtt{B}, pc_2\mapsto \mathtt{A}], \\ [x\mapsto 1, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{B}], [x\mapsto 1, pc_1\mapsto \mathtt{B}, pc_2\mapsto \mathtt{A}], \\ [x\mapsto 1, pc_1\mapsto \mathtt{A}, pc_2\mapsto \mathtt{C}], [x\mapsto 1, pc_1\mapsto \mathtt{C}, pc_2\mapsto \mathtt{A}] \end{array} \right\}, \\
X^{(5)} &= X^{(4)}.
\end{aligned}$$

Notice that $\gamma_{\mathrm{OG}}(\inf \mathfrak{O}) \subsetneq \mathit{lfp}(\lambda x.\, \rho_{\mathrm{OG,mc}}(\mathit{init} \cup \mathit{post}(x)))$.
So abstract interpretation with $\rho_{\mathrm{OG,mc}}$ produces a strictly weaker property than the strongest Owicki-Gries-core proof on this example. □

From Examples 60 and 62 we obtain that Owicki-Gries core is incomparable with multithreaded Cartesian abstraction, adapted to RPL in the most straightforward way.

Example 63. The RPL program given by Fig. 7 has $\mathit{Tid} = \{1,2\}$, $\mathit{PL} = \{\mathtt{A},\mathtt{B}\}$, $\mathit{Val} = \mathit{PL} \dot\cup \{0,1\}$, $\mathit{Res} = \emptyset$, $\mathit{Local}_1 = \{pc_1, l_1\}$, $\mathit{Local}_2 = \{pc_2, l_2\}$, $\mathit{ProofVar} = \emptyset$, $\mathit{rsc}(\mathtt{A}) = \mathit{rsc}(\mathtt{B}) = \emptyset$.

$$
\begin{array}{l|l}
\multicolumn{2}{c}{\texttt{integer } l_1 = l_2 \in \{0,1\}} \\
\texttt{A: } \{\mathit{true}\} & \texttt{A: } \{\mathit{true}\} \\
\quad \texttt{assume } \mathit{false}; & \quad \texttt{assume } \mathit{false}; \\
\texttt{B: } \{\mathit{false}\} & \texttt{B: } \{\mathit{false}\} \\
\quad l_1 := 0 & \quad l_2 := 0 \\
\texttt{C: } \{\mathit{false}\} & \texttt{C: } \{\mathit{false}\}
\end{array}
$$

Fig. 7: RPL transition system progAssign2 with its smallest Owicki-Gries-core proof.

Then

$$
\begin{aligned}
\gamma_{\mathrm{OG}}(\inf \mathfrak{O}) \; = \; \{ \; & [l_1 \mapsto 0, l_2 \mapsto 0, pc_1 \mapsto \texttt{A}, pc_2 \mapsto \texttt{A}], \\
& [l_1 \mapsto 0, l_2 \mapsto 1, pc_1 \mapsto \texttt{A}, pc_2 \mapsto \texttt{A}], \\
& [l_1 \mapsto 1, l_2 \mapsto 0, pc_1 \mapsto \texttt{A}, pc_2 \mapsto \texttt{A}], \\
& [l_1 \mapsto 1, l_2 \mapsto 1, pc_1 \mapsto \texttt{A}, pc_2 \mapsto \texttt{A}] \, \},
\end{aligned}
$$

and

$$
\rho_{\mathrm{OG,mc}}(\mathit{init}) = \mathit{init}, \quad \text{thus} \quad \mathit{lfp}(\lambda x.\, \rho_{\mathrm{OG,mc}}(\mathit{init} \cup \mathit{post}(x))) \; = \; \mathit{init}\,.
$$

So the strongest Owicki-Gries-core-provable property is strictly weaker than the strongest property provable by abstract interpretation with $\rho_{\mathrm{OG,mc}}$. □

Both progAssign1 and progAssign2 belong to Simple. For progAssign2 the strongest Owicki-Gries-core-provable property is strictly weaker than the strongest property provable by abstract interpretation with $\rho_{\mathrm{OG,mc}}$, for progAssign1 the strongest Owicki-Gries-core provable property is strictly stronger. Thus, for Simple, the Owicki-Gries core is in general incomparable with abstract interpretation with $\rho_{\mathrm{OG,mc}}$.

Owicki-Gries for RPL and general programs target different verification tasks. Owicki-Gries for RPL allows verifying properties assuming mutual exclusion (external verification). Owicki-Gries for general programs (= multithreaded Cartesian abstraction) allows checking synchronization details without assuming mutual exclusion (internal verification). This section presents an initial step in comparing the two proof systems. The incomparability result shows that if both systems can be unified, it is not going to be trivial.

Although it is impossible to present a verification methodology that has exactly the same precision as both proof systems, there is a verification method that produces properties which are stronger than both proof systems, and which can be viewed as a generalization of both proof systems, namely the deny-guarantee framework [12], SL+ [27], SAGL [13], or RGSep [29, 30]. These logics, however, do not generalize the "pure" Owicki-Gries system, but separation-logic-based approaches.

We also conjecture that it is possible to prove all Owicki-Gries-core-provable properties by abstract interpretation with $\rho_{\mathrm{OG,mc}}$ with the following fixed choice of auxiliary variables. Namely, for each resource, we add a resource variable that indicates which thread is in the critical section for that resource.

I Practical implications

Now we look at the importance of results for practical verification.

Many real-life specified properties of programs are simple: they involve only few variables, which are, for example, local, or all belong to the same resource. In this case the property is likely to be a fixpoint of one of the considered closures, which retain dependencies between certain variables. Checking this fact is easy: it does not depend on the program's transition relation. If in addition the property is expected to be inductive, we obtain a clear choice of an analysis to confirm our expectation: it would be abstract interpretation with any closure for which the property is a fixpoint.

Abstract interpretation with a chosen closure might also derive a property which is coarser than expected. In this case, in the chain

$$\rho_{\mathrm{c}} \; - \; \bar{\rho} \; - \; \text{Owicki-Gries core} \; - \; \underline{\rho} \; - \; \text{identity}$$

it is useless to try out analyses with a lower precision. On the contrary, an analysis with a higher precision, which tracks more variable dependencies, might help.

An analysis from this chain might also run out of time or space, failing to produce any property. Without further assumptions, predicting time or space consumption of other analyses is hard. However, for certain finite-state scenarios predictions are still possible. For example, abstract interpretation with ρ_{c} and the Owicki-Gries core can be easily executed as fixpoint iterations on the domains of Cartesian products and program annotations, respectively. The worst-case space for storing the iterates is larger for the Owicki-Gries core than for ρ_{c}, also the height of the domain of program annotations is larger than the height of the domain of products of blocks. If the worst case is expected (e.g., by an insight of an expert), machines which are slow or have low memory might be too restricted to execute the Owicki-Gries core, so abstract interpretation with ρ_{c} should be recommended to attempt first.