On the Strength of Owicki-Gries for Resources

Alexander Malkis and Laurent Mauborgne
IMDEA Software Institute
accepted 2011-08-28, published 2011-10-14, last updated 2023-07-28

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 $\rho$, the lower bound is $\bar{\rho}$. In addition to $\rho$, we will consider a simpler lower bound $\rho_c$, which is coarser than $\rho$, 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 $Data\Var$. 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 $\phi$” (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 $\text{with } r \text{ when } \phi \text{ do } C \text{ endwith }$ where $r$ is a resource, $\phi$ a formula over data variables and $C$ some statement. We write $\text{with } r \text{ do } C \text{ endwith }$ when $\phi$ 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 $Proof\Var(r)$, which are all data variables that are not assigned except maybe in critical sections for $r$. All our examples will satisfy $Proof\Var(r) = r$ (in general, $Proof\Var(r) \supseteq r$).

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 $PC\Var$ be the set of all program counter variables and $\Var = Data\Var \cup PC\Var$.

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 \( \text{Var} \) to the set of values \( \text{Val} \) that includes \( \text{PL} \) and the ranges of all data variables. We are going to consider consistent states only: those are states that

- map program counters to elements of \( \text{PL} \) 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 \( r \in \text{rsc}(v(pc_t)) \). 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 \( v \rightarrow_t v' \) where \( v, v' \) are consistent states. The transitions induced by the non-\( \text{with} \) statements are straightforward. The statement \( \text{with } r \text{ when } \phi \text{ do } C \) blocks when \( r \) is busy or when \( \phi \) 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 \( \text{init} \) be the set of initial states corresponding to the initial condition and let the successor map

\[
\text{post}(S) = \{ v' \mid \exists v \in S, t \in \text{Tid}: v \rightarrow_t v' \}
\]

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 \( \text{rsc} \). 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 \( \text{ProofVar}(r) \) (resp. to local data variables of \( t \) and all proof variables of all resources in \( \text{rsc}(p) \)) 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 \( \gamma_{OG}(I, M) \) 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 $v \rightarrow v'$ 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 will 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 $\text{control}(ww, ar, aw)$

reader$_1$ || ... || reader$_n$ || writer$_1$ || ... || writer$_m$

// reader$_i$:
while true do
    startreadA: \{true\}
    with control do
        when $ww = 0$ do
            askwriteA: \{true\}
            with control do
                when $ww = 0 \wedge aw \leq 1$ do
                    startreadB: \{true\}
                    $ww := ww + 1$
                askwriteB: \{true\}
                when $ww = 0 \wedge aw \leq 1$ do
                    $ar := ar + 1$
                    finishreadB: \{true\}
                $ww := ww + 1$
            finishwriteB: \{true\}
            with control do
                when $ar = aw = 0$ do
                    $ar := aw + 1$
                $aw := aw - 1$
                when $ar = aw = 0$ do
                    write: \{true\}
                    $ww := ww - 1$
                finishwriteB: \{true\}
                with control do
                    when $ar = aw = 0$ do
                        $ar := aw + 1$
                    $aw := aw - 1$
                    when $ar = aw = 0$ do
                        $ar := aw + 1$
                    $aw := aw - 1$
                finishwriteC: \{true\}
                with control do
                    when $ar = aw = 0$ do
                        $ar := aw + 1$
                    $aw := aw - 1$
                    when $ar = aw = 0$ do
                        $ar := aw + 1$
                    $aw := aw - 1$
                finishwriteC: \{true\}
            endwith
        finishreadA: \{true\}
        with control do
            when $ww = 0 \wedge aw \leq 1$ do
                finishreadA: \{true\}
            $ww := ww + 1$
        $ar := ar - 1$
        $ar := ar - 1$
    finishwriteA: \{true\}
    with control do
        when $ar = aw = 0$ do
            $ar := aw + 1$
        $aw := aw - 1$
        when $ar = aw = 0$ do
            write: \{true\}
            $ww := ww - 1$
        finishwriteB: \{true\}
        with control do
            when $ar = aw = 0$ do
                $ar := aw + 1$
            $aw := aw - 1$
            when $ar = aw = 0$ do
                $ar := aw + 1$
            $aw := aw - 1$
        finishwriteC: \{true\}
        with control do
            when $ar = aw = 0$ do
                $ar := aw + 1$
            $aw := aw - 1$
            when $ar = aw = 0$ do
                $ar := aw + 1$
            $aw := aw - 1$
        finishwriteC: \{true\}
    endwith
endwhile

$\text{L}_{\text{control}} = \{aw \leq 1\}$

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

The strongest Owicki-Gries-core-provable property is

$$\{v \in \text{ConsState} \mid v(ww), v(ar) \in \mathbb{N}_0 \wedge v(aw) \in \{0, 1\} \wedge \forall i \in \{1, \ldots, n\} :$$

$$\wedge (v(pc_{\text{reader}})_i) \in \{\text{read, read} \mid x \in \{\text{start, finish}\} \wedge y \in \{A, B, C\}\}$$

$$\wedge (v(pc_{\text{reader}})_i) = \text{startreadB} \Rightarrow v(ww) = 0$$

$$\wedge (v(pc_{\text{reader}})_i) = \text{startreadC} \Rightarrow v(ww) = 0 < v(ar)$$

$$\wedge \forall j \in \{1, \ldots, m\} :$$

$$\wedge (v(pc_{\text{reader}})_j) \in \{\text{write, write} \mid x \in \{\text{ask, start, finish}\} \wedge y \in \{A, B, C\}\}$$

$$\wedge (v(pc_{\text{reader}})_j) = \text{askwriteC} \Rightarrow v(ww) > 0$$

$$\wedge (v(pc_{\text{reader}})_j) = \text{startwriteB} \Rightarrow v(ar) = v(aw) = 0$$

$$\wedge (v(pc_{\text{reader}})_j) = \text{startwriteC} \Rightarrow v(ar) = 0 < v(aw)$$

$$\wedge (v(pc_{\text{reader}})_j) = \text{finishwriteC} \Rightarrow v(aw) = 0\}.$$
There are states in this set in which more than one writer is at \texttt{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 \texttt{aw},
which tracks the number of writers, never exceeds 1.

**Example 2 (Upper).** The program Upper in Fig. 3 will be used for showing differenc es
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 initial ones. The smallest
initially $u \neq z = x = y = l$

```
// Thread 1
A: \{l = x\}
with r when l = u do
B: \{l = x \neq u\}
with r' do
C: \{y \geq l = x \neq z\}
D: \{y \geq l = u \neq z \land x = 0\}
endwith;
E: \{l = u \neq z \land x = 0\}
assume false
F: endwith;
G: with r' do
H: x := 0
I: endwith;
J: with r do
K: u := 0
L: endwith;
M: l := 0
N:
```

```
// Thread 2
O: \{y = z\}
assume false;
P: with r' do
Q: y := 0
R: endwith;
S: with r do
T: u := 0;
U: z := 0
V: endwith;
W:
```

\begin{align*}
I_r &= \{u \neq z\}, & I'_r &= \{x \leq y\}
\end{align*}

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

Owicki-Gries-core proof denotes many more states, e.g., $[u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 1, 
\begin{align*}
l \mapsto 0, pc_1 \mapsto k, pc_2 \mapsto 0].
\end{align*}$

**Example 3 (Simple).** A program belongs to the class \textit{Simple} if it has at most one re-
source, 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 \textit{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-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 \(\text{post}^\#_{\text{OG}}(I, M) = (I', M')\), 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 \(v \rightarrow r'\) (for some thread \(t\)) that exits a critical section for \(r\), \(v\) satisfies \(M(v(pc), t)\) 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 \(v \rightarrow r'\) such that \(v\) satisfies \(M(v(pc), t)\), \(v\) satisfies \(I_r\) if the transition starts a critical section for \(r\), \(v'(pc) = p'\), 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 \((I^{\text{init}}, M^{\text{init}})\) be the annotation describing the initial states only:

- \(I^{\text{init}}_r\) is the set of all valuations of proof variables of \(r\) in the initial states;
- \(M^{\text{init}}_p, t\) is the set of valuations of local data variables of thread \(t\) in the initial states.

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

**Theorem 5 (Equivalence).** The least fixpoint of \(F^\#_{\text{OG}}\) 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 \(\rho\) 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 \(F : D \rightarrow D\), the closure \(\rho\) on \(D\) defines an approximation of the concrete semantics \(\text{lfp}(F)\) in the sense that \(\text{lfp}(F) \leq \text{lfp}(\rho \circ F)\). 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 $post^\#_{OG}$ of that successor map. In order to describe its precision, it would be nice to find a closure $\rho$ such that $post^\#_{OG}$ 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 $\rho$ 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 $\rho$ 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

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

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 $\rho$ 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 $\Xi$ indexed by transition relations, such that $\Xi$ preserves least fixpoints ($\forall$ transition relations $\tau_1, \tau_2$: $lfp(\tau_1) = lfp(\tau_2) \Rightarrow lfp(\Xi_{\tau_1}) = lfp(\Xi_{\tau_2})$), and for any property $S$ of consistent states we have

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

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 $\tilde{\alpha}$ 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 \( \tilde{\rho} \) on the powerset of traces of consistent states such that for any multithreaded program having the given syntactic structure, set of initial traces \( \tilde{\text{init}} \) and the trace extension operator post and for any property \( S \) of consistent states we have

\[
S \text{ is Owicki-Gries-core provable } \iff \tilde{\alpha}(\text{lfp}(\lambda x. \tilde{\rho}(\tilde{\text{init}} \cup \tilde{\text{post}}(x)))) \subseteq S.
\]

### 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 \( \rho(Q) \) 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 \( \rho \) 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 \( \gamma_{OG} \) (which gives the set of consistent states denoted by a program annotation).

Now fix an arbitrary program and let \( \rho \) be defined by its syntactic structure.

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

\[
\text{lfp}(\lambda x. \rho(\text{init} \cup \text{post}(x))) \subseteq \text{the strongest Owicki-Gries-core-provable property}.
\]

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 $lfp(\lambda x.\rho(init \cup post(x)))$ coincides with the strongest Owicki-Gries-core-provable property. This property is coarser than the set of reachable states. For example, in one execution writer$_1$ can reach write, in another execution writer$_2$ can reach write, and the initial state satisfies \(ww = ar = aw = 0\), so \(\rho\) 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 \(\rho\) 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 \(\rho\), 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 \(\rho\) 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 \(\rho\) 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 (namely \(\rho_c\)) is easier to comprehend.

The definitions of both bounds require some preparation.

For a family of sets \(\mathcal{X} = \{X_1, \ldots, X_n\}\), let \(Part(\mathcal{X})\), called partition of \(\mathcal{X}\), be the set of all nonempty \(Y_1 \cap \cdots \cap Y_n\) where each \(Y_i\) is either \(X_i\) or its complement \((\bigcup \mathcal{X}) \setminus X_i\) (\(1 \leq i \leq n\)). 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 \(RL\) 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 $\rho_c$ returns all the consistent states from the product of projections of $Q$ onto the blocks in $RL$.

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, $\rho_c$ 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 $\rho_c$ 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 $\rho_c$ in two ways. Firstly, we impose restrictions on $\tilde{v}$ from the definition of $\rho_c$. 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 $RL$ 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 $\rho_c$. 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 $\rho_c$. Formally:

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

**Example 19 (Readers-Writers).** For the program Readers-Writers from Example 1 abstract interpretation with $\rho_c$ 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 $\rho_c$ 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 0 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 $\rho$ 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 $\rho$ 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 $\rho$ 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 structs. On obtaining ownership of a struct, 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


30. V. Vafeiadis. RGSep action inference. In Barthe and Hermenegildo [1], pages 345–361.


### 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 $PL \subseteq Val$. 

17
- \textit{Res} is a set\(^1\) of disjoint\(^2\) sets.
- For each \(t \in \text{Tid}\) we have \(pc_t \in \text{Local}_t \setminus \bigcup \text{Res}\).
- For all different \(t, \bar{t}\) in \(\text{Tid}\) we have \(pc_t \not\in \text{Local}_t\).
- \textit{ProofVar}: \(\text{Res} \rightarrow \mathcal{P}(\text{DataVar})\) is extensive, where \(\mathcal{P}\) is the powerset symbol, \(\text{PCVar} = \{pc_t \mid t \in \text{Tid}\}\) and \(\text{DataVar} = \bigcup \text{Res} \cup \bigcup_{t \in \text{Tid}} \text{Local}_t \setminus \text{PCVar}\).
- \(\text{rsc}: \text{PL} \rightarrow \mathcal{P}(\text{Res})\). Notice that \(\text{rsc}\) is independent\(^3\) of \(\text{Tid}\).

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

\[
\text{ConsState} = \{v \in \text{Var} \mid \forall t \in \text{Tid}: v(pc_t) \in \text{PL} \text{ and } \forall \bar{t} \in \text{Tid} \setminus \{t\} : \text{rsc}(v(pc_{\bar{t}})) \cap \text{rsc}(v(pc_t)) = \emptyset\}.
\]

A resource \(r\) is \textit{held} by \(t \in \text{Tid}\) in \(v \in \text{ConsState}\) if \(r \in \text{rsc}(v(pc_t))\). A resource is \textit{busy} in \(v \in \text{ConsState}\) if \(r \in \bigcup_{t \in \text{Tid}} \text{rsc}(v(pc_t))\), otherwise \(r\) is \textit{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 \textit{obeying a syntactic structure} as above is a tuple

\[(\text{init}, (\Rightarrow_t)_{t \in \text{Tid}})\]

such that \(\text{init} \subseteq \text{ConsState}\), for all \(t \in \text{Tid}\) we have \(\Rightarrow_t \subseteq \text{ConsState}\)\(^2\) \(\land \forall v \in \text{init}: \text{rsc}(v(pc_t)) = \emptyset\), and all the conditions below hold:

- for all \(x \in \text{DataVar}\) and \(t \in \text{Tid}\) we have

\[
(\forall \bar{t} \in \text{Tid} \setminus \{t\}, (v, v') \in \Rightarrow: v(x) = v'(x)) \Rightarrow x \in \text{Local}_t;
\]

- for all \(r \in \text{Res}\) we have

\[
\text{ProofVar}(r) = \left\{x \in \text{DataVar} \mid \forall t \in \text{Tid}, (v, v') \in \Rightarrow : \left(v(x) = v'(x) \text{ or } \begin{cases} r \in \text{rsc}(v(pc_t)) \cap \text{rsc}(v'(pc_t)) \end{cases}\right)\right\};
\]

- for all \(t \in \text{Tid}\), \((v, v') \in \Rightarrow, R = \text{rsc}(v(pc_t)), R' = \text{rsc}(v'(pc_t))\), \(\bar{R} = \text{Res} \setminus (R \cup R')\) all the following conditions hold:

\(^1\) 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 consider the general case, allowing nonempty intersection between any element of \(\text{Res}\) and any \(\text{Local}_t (t \in \text{Tid})\). If we would enforce mutual disjointness between all \(r \in \text{Res}\) and all \(\text{Local}_t (t \in \text{Tid})\), we would get a tighter lower bound, but only 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 \(\text{rsc}\) dependent on the thread.

\(^4\) Owicki described \text{ProofVar} 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 an RPL transition system

\( \Rightarrow \)

The goal of verification is, given a set of states

\[ w \]

**Proof.** Follows from

\[ \text{Fix an RPL transition system as above. A set of states is mapped to a set of their successors by the function} \]

\[ \text{Notice that in (4), } \]

\[ \text{Remark 23. Notice that in (4), } w' \text{ additionally satisfies } w' \in \text{ConsState and } (v \sim_{\text{Local} \cup \mathbf{R}} v') \text{.} \]

**Proof.** Follows from \( \rightarrow_t \subseteq \text{ConsState}^2 \) and \( v' \text{ [by (2)] } \sim_{\text{Local} \cup \mathbf{R}} v \text{ [by assumption]} \)

\[ \sim_{\text{Local} \cup \mathbf{R}} w \text{ [by (2)] } \sim_{\text{Local} \cup \mathbf{R}} w'. \]

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

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

The goal of verification is, given a set of states \( \text{safe} \subseteq \text{ConsState} \), to prove that

\[ \text{lfp}(\lambda S. \text{init} \cup \text{post}(S)) \subseteq \text{safe} . \]

## B Wocki-Gries for RPL

Fix a syntactic structure \( \text{(Tid, PL, Val, Res, (Local, pc)} \in \text{Tid, ProofVar, rsc) for the whole section B.} \)

### B.1 Definition of Wocki-Gries core

For each \( t \in \text{Tid} \) let \( \text{LocalDataVar}_t = \text{Local}_t \setminus \{ \text{pc}_t \} \) be the set of local data variables.

A program annotation is an element of

\[ PA = (\prod_{r \in \text{Res}} \mathfrak{P}(\text{ProofVar}(r) \rightarrow \text{Val})) \times (\prod_{p \in \text{PL}} (\text{LocalDataVar}_t \cup \text{ProofVar}(\text{rsc}(p))) \rightarrow \text{Val}). \]

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

\[ \gamma_{\text{GI}}(P) = \{ v \in \text{ConsState} \mid \forall r \in \text{Res} \setminus \bigcup \text{rsc}(v(\text{PCVar})) : v|_{\text{ProofVar}(r)} \in I_r, \]

\[ \text{and } \forall t \in \text{Tid} : v|_{\text{LocalDataVar}_t \cup \text{ProofVar}(\text{rsc}(v(\text{pc}_t)))} \in M_{v(\text{pc}_t)} \}. \]

For an RPL transition system \( T = (\text{init}, (\rightarrow_t)_{t \in \text{Tid}}) \) that obeys the given syntactic structure, an **Owicki-Gries-core proof** of \( T \) is a program annotation \( P : = (I_r)_{r \in \text{Res}, (M_p)_{p \in \text{PL}}} \) such that the following holds:
– sequential consistency: for all $t \in \text{Tid}$, $(v,v') \in \rightarrow_r$, $p = v(pc)$, $p' = v'(pc)$, $R = rsc(p)$, $R' = rsc(p')$, if

$$v|_{\text{LocalData} \land \lnot \text{ProofVar}(R)} \in M_{p,t} \land \forall r \in R' \setminus R : v|_{\text{ProofVar}(r)} \in I_r,$$

then

$$v|_{\text{LocalData} \land \lnot \text{ProofVar}(R')} \in M_{p',t} \land \forall r \in R' \setminus R' : v|_{\text{ProofVar}(r)} \in I_r;$$

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

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

### B.2 Lattice of Owicki-Gries-core proofs

The set of program annotations is ordered by pointwise inclusion:

$$((I_r)_{r \in \text{Res}}, (M_{p,t})_{p \in \text{PL}, t \in \text{Tid}}) \subseteq ((I_r')_{r \in \text{Res}}, (M_{p,t}')_{p \in \text{PL}, t \in \text{Tid}})$$

iff $\forall r \in \text{Res} : I_r \subseteq I'_r$ and $\forall p \in \text{PL}, t \in \text{Tid} : M_{p,t} \subseteq M'_{p,t}$.

Then the least upper bound is the pointwise union $\bigcup$, and the lattice is complete.

Consider the lattice

$$D = \mathcal{P}(\text{ConsState}),$$

ordered by inclusion. This lattice is certainly complete.

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

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

“$\subseteq$” Let $v \in \gamma_{OG}(\bigcap A)$. Then $\forall r \in \text{Res} \setminus \bigcup rsc(v(\text{PCVar})) : v|_{\text{ProofVar}(r)} \in I_r$ and $\forall t \in \text{Tid} : v|_{\text{LocalData} \land \lnot \text{ProofVar}(rsc(v(pc)) \in N_{v(pc), t}}$. The definition of $\bigcap$ implies that for all $B = ((I_r)_{r \in \text{Res}}, (M_{p,t})_{p \in \text{PL}, t \in \text{Tid}}) \in A$ we have $\forall r \in \text{Res} \setminus \bigcup rsc(v(\text{PCVar})) : v|_{\text{ProofVar}(r)} \in I_r$ and $\forall t \in \text{Tid} : v|_{\text{LocalData} \land \lnot \text{ProofVar}(rsc(v(pc)) \in M_{v(pc), t}$, i.e., $v \in \gamma_{OG}(B)$. Since $B \in A$ was arbitrary, we have $v \in \bigcap \gamma_{OG}(A)$.

“$\supseteq$” Let $v \in \gamma_{OG}(B)$ for all $B \in A$. Let $r \in \text{Res} \setminus \bigcup rsc(v(\text{PCVar}))$. Then for each $B = (I, M) \in A$ the definition of $\gamma_{OG}$ gives $v|_{\text{ProofVar}(r)} \in I_r$. Thus $v|_{\text{ProofVar}(r)} \in J_r$. So $\forall r \in \text{Res} \setminus \bigcup rsc(v(\text{PCVar})) : v|_{\text{ProofVar}(r)} \in J_r$. Now let $t \in \text{Tid}$. For each $B = (I, M) \in A$ the definition of $\gamma_{OG}$ implies that $v|_{\text{LocalData} \land \lnot \text{ProofVar}(rsc(v(pc)) \in M_{v(pc), t}$; so $N_{v(pc), t} \supseteq v|_{\text{LocalData} \land \lnot \text{ProofVar}(rsc(v(pc)))$. Thus $\forall t \in \text{Tid} : v|_{\text{LocalData} \land \lnot \text{ProofVar}(rsc(v(pc))) \in N_{v(pc), t}$. We have shown $v \in \gamma_{OG}(\bigcap A)$. \[\blacksquare\]

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

20
Theorem 25. \((\mathcal{D}, \sqsubseteq)\) is a complete lattice with infimum being componentwise intersection \(\sqcap\).

Proof. Let \(A\) be a set of Owicki-Gries-core proofs. First we show that \(\bigcap A\) in \(PA\) is the infimum of \(A\) in \(\mathcal{D}\).

- Case \(A = \emptyset\). Let \(B = ((|Proof\!Var(r)| \rightarrow Val)_{r \in Res}, \bigcup_{r \in Tid} (|Proof\!Var(rsc(r)|) \rightarrow Val)_{r \in Rsc(r)}\). Certainly \(B\) is sequentially consistent. Since \(\gammaOG(B) = ConsState\), \(B\) satisfies the initial condition. Thus \(B \in \mathcal{D}\). Notice that \(B\) is a lower bound of \(\emptyset\) and is greater than or equal to any other element of \(\mathcal{D}\). Thus \(B = \inf\mathcal{D} \emptyset\).

- Case \(A \neq \emptyset\). Let \(B = ((|I_r| \in Res, (M_{r,s})_{r \in Tid} = \bigcap A\). First we will show that \(B\) is an Owicki-Gries-core proof.

Sequential consistency. Let \(t \in Tid, (v, v') \in \gamma_r, p = v(pc_r), p' = v'(pc_r), R = rsc(p), R' = rsc(p'), v|_{LocalDataVar} \cup \cup_{t \in Tid} (|Proof\!Var(rsc(r)|) \rightarrow Val)_{r \in Rsc(r)}\). Then for all \((J, N) \in A\) we have \(v|_{LocalDataVar} \cup \cup_{t \in Tid} (|Proof\!Var(rsc(r)|) \rightarrow Val)_{r \in Rsc(r)}\). Thus \(v|_{LocalDataVar} \cup \cup_{t \in Tid} (|Proof\!Var(rsc(r)|) \rightarrow Val)_{r \in Rsc(r)}\). Taking componentwise intersection over all \(P\), for any \(v \in init\) we have \(\forall \gamma \in Res, \cup_{t \in Tid} (|Proof\!Var(rsc(r)|) \rightarrow Val)_{r \in Rsc(r)}\). So for all \(v \in init\) we have \(v \in \gammaOG(B), \text{ so } init \subseteq \gammaOG(B)\).

We have shown that \(B \in \mathcal{D}\). By definition \(B\) is less than or equal to any element of \(A\). Assume some other lower bound \(C \in \mathcal{D}\) for \(A\). Then \(C \subseteq P\) for each \(P \in A\), so \(C\) is less than or equal to the pointwise intersection over \(A\), i.e., \(C \subseteq B\). Thus \(B\) is the greatest lower bound of \(A\) in \(\mathcal{D}\).

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

Notice that the infimum of \(\mathcal{D}\) is the tuple of empty sets iff \(init\) is empty; thus \(\mathcal{D}\) is a complete sublattice of \(PA\) only if \(init\) is empty. Usually verified programs have initial states, thus usually \(\mathcal{D}\) 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 \(\gammaOG(\mathcal{D})\). The infimum of this set is \(S = \bigcap \gammaOG(\mathcal{D}) = \gammaOG(\bigcap \mathcal{D})\). Since \(\mathcal{D}\) is a complete lattice with pointwise intersection as infimum, \(\bigcap \mathcal{D} \in \mathcal{D}\). 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:

\[\forall P \in \mathcal{D}: \text{ init } \cup \text{ post}(\gammaOG(P)) \subseteq \gammaOG(P)\].

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

- **post \circ \gamma_{OG}(I,M) \subseteq \gamma_{OG}(I,M)**: Let \(v' \in \text{post} \circ \gamma_{OG}(I,M)\). Then there is \(v \in \gamma_{OG}(I,M)\) and \(i \in Tid\) such that \(v \Rightarrow_{i} v'\). Notice that by \((1)\) we have \(v(pc_{i}) = v'(pc_{i})\) for \(i \in Tid \setminus \{t\}\). Let \(R = \text{rsc}(v(pc_{i}))\) and \(R' = \text{rsc}(v'(pc_{i}))\). From \(v \in \gamma_{OG}(I,M)\) we get \(v|_{\text{LocalDataVar}}(\bigcup_{\text{ProofVar}}(R)) \in M_{v(pc_{i})}^R\). Now let \(r \in R' \setminus R\). From \(v \in \text{ConsState}\) and \(r \in \text{rsc}(v'(pc_{i}))\) we get \(r \in \bigcup_{i \in Tid \setminus \{t\}} \text{rsc}(v'(pc_{i})) = \{\text{since } v(pc_{i}) = v'(pc_{i})\} \text{ for } i \in Tid \setminus \{t\}\}. Since \(r \not\in R = \text{rsc}(v(pc_{i}))\), we conclude that \(r \not\in \bigcup \text{rsc}(v(PCVar))\). Since \(v \in \gamma_{OG}(I,M)\), we obtain \(v|_{\text{ProofVar}}(r) \in I_r\). Thus \(\forall r \in R' \setminus R\): \(v|_{\text{ProofVar}}(r) \in I_r\) By sequential consistency, \(v|_{\text{LocalDataVar}}(\bigcup_{\text{ProofVar}}(R)) \in M_{v(pc_{i})}^R\) and \(\forall r \in R \setminus R' : v|_{\text{ProofVar}}(r) \in I_r\).

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

- Let \(r \in \text{Res} \setminus \text{rsc}(v'(PCVar))\). From \(i \in Tid\), \(v \Rightarrow_{i} v'\) and the definition of \(\text{ProofVar}\) we obtain that for all \(x \in \text{ProofVar}(r)\) we have \(v(x) = v'(x)\). So \(v|_{\text{ProofVar}(r)} = v'|_{\text{ProofVar}(r)}\). From \((1)\) we know for \(i \in Tid \setminus \{t\}\) that \(v(pc_{i}) = v'(pc_{i})\), thus \(\text{rsc}(v(pc_{i})) = \text{rsc}(v'(pc_{i})) \not\in r\). Consider two cases.

  - Case \(\hat{r} = t\). By the above, \(v'|_{\text{LocalDataVar}}(\bigcup_{\text{ProofVar}}(R)) \in M_{v(pc_{i})}^R\).

  - Case \(\hat{r} \neq t\). From \(v(pc_{j}) = v'(pc_{j})\) we get \(\text{LocalDataVar}(\bigcup_{\text{ProofVar}}(rsc(v(pc_{j})))) = \text{LocalDataVar}(\bigcup_{\text{ProofVar}}(rsc(v'(pc_{j}))))\). Let \(x\) be an element of \(\text{LocalDataVar}(\bigcup_{\text{ProofVar}}(rsc(v(pc_{j}))))\). If \(x \in \text{LocalDataVar}(\bigcup_{\text{ProofVar}}(rsc(v'(pc_{j}))))\), then \((1)\) implies \(v(x) = v'(x)\). Otherwise \(x \in \text{ProofVar}(r)\) for some \(r \in \text{rsc}(v'(pc_{j}))\). From \(v \in \text{ConsState}\) we get \(r \not\in \text{rsc}(v'(pc_{j}))\). From \(x \in \text{ProofVar}(r)\) we obtain \(v(x) = v'(x)\). Combining together, \(v|_{\text{LocalDataVar}}(\bigcup_{\text{ProofVar}}(rsc(v(pc_{j})))) = v|_{\text{LocalDataVar}}(\bigcup_{\text{ProofVar}}(rsc(v'(pc_{j}))))\) in \(\text{since } v \in \gamma_{OG}(I,M)\). \(M_{v(pc_{i})}^R = M_{v'(pc_{i})}^R\).

So \(\forall \hat{r} \in Tid\): \(v'|_{\text{LocalDataVar}}(\bigcup_{\text{ProofVar}}(rsc(v'(pc_{j})))) \in M_{v(pc_{i})}^R\) We have shown \(v' \in \gamma_{OG}(I,M)\).

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

\[\square\]

### B.3 Examples of Owicki-Gries-core proofs

For \(k = N^{+}\) 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

- \(\text{Tid} = \{\text{reader}_{i}, \text{writer}_{j} \mid 1 \leq i \leq n, 1 \leq j \leq m\}\);
- \(\text{PL} = \{\text{startreadA}, \text{startreadB}, \text{startreadC}, \text{read}, \text{finishreadA}, \text{finishreadB}, \text{finishreadC}, \text{askwriteA}, \text{askwriteB}, \text{askwriteC}, \text{startwriteA}, \text{startwriteB}, \text{write}, \text{finishwriteA}, \text{finishwriteB}, \text{finishwriteC}\};
- \(\text{Val} = \text{PL} \cup \mathbb{N}_0\), where \(\mathbb{N}_0\) is the set of nonnegative integers;
- \(\text{Res} = \{\text{control}\} = \{\{\text{ww}, \text{ar}, \text{aw}\}\};

22
- Local\(_{reader}\), \(\{pc\_{reader}\}\) (1 \(\leq i \leq n\)), Local\(_{writer}\), \(\{pc\_{writer}\}\) (1 \(\leq j \leq m\));
- ProofVar(control) = control;
- \(rsc(\ldots A) = rsc(read) = rsc(write) = \emptyset, rsc(\ldots B) = rsc(\ldots C) = \{control\}\).

Since we have assumed at least two readers and two writers, there are no local data variables. Let
\[
I_{control} = \{v : control \rightarrow N_0 | v(aw) \leq 1\},
\]
\[
M \in \prod_{\text{pc} \in \mathcal{PC}, \text{loc} \in \mathcal{Loc}} \mathcal{P}((\text{LocalDataVar} \cup \text{ProofVar}(rsc(p))) \rightarrow \text{Val})
\]
given by
\[
M_{\text{read}\_reader} = M_{\text{read}\_writer} = \{\emptyset\},
\]
\[
M_{\text{startread}\_reader} = \{v \in \{(ww, ar, aw) \rightarrow N_0 | v(ww) = 0 \wedge v(aw) \leq 1\},
\]
\[
M_{\text{startread}\_writer} = \{v \in \{(ww, ar, aw) \rightarrow N_0 | v(ww) = 0 \wedge v(aw) \leq 1 \leq v(ar)\},
\]
\[
M_{\text{finishread}\_reader} = M_{\text{finishread}\_writer} = \{v \in \{(ww, ar, aw) \rightarrow N_0 | v(aw) \leq 1\} \text{ for } 1 \leq i \leq n,
\]
the remaining \(M_{\cdot}\) are empty sets.

Notice that \(P = ([control \leftrightarrow I_{control}], M)\) is an annotation of Readers-Writers. It is sequentially consistent and the single initial state, say, \(v^{\text{init}}\), satisfies \(v^{\text{init}}(ww) = v^{\text{init}}(ar) = 0\), thus \(v^{\text{init}}|_{\text{ProofVar}(control)} \in I_{control}\), so \(v^{\text{init}} \in \gamma_0(P)\). Thus \(P\) is an Owicki-Gries-core proof.

Now we will show that this proof is the smallest one. So let \([control \leftrightarrow \hat{I}_{control}], \hat{M}\) 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 \(i \in N_n\) and every \(l \in \{\text{startreadA, read, finishreadA}\}\) we have \(\emptyset \in \hat{M}_{\text{reader}}\); for every \(j \in N_m\) and every \(l \in \{\text{ankwriteA, startwriteA, write, finishwriteA}\}\) we have \(\emptyset \in \hat{M}_{\text{writer}}\).

Now we will show in three steps that for any \((a, b, c) \in N_\mathcal{A}^2 \times \{0, 1\}\) 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 \(v(pc_{\text{writer}}) = \text{startwriteA}\) and \(v( reads) = v'\) and to \([ww \mapsto 0, ar \mapsto 0, aw \mapsto 0]\) \(\in \hat{I}_{control}\) to get \([ww \mapsto 0, ar \mapsto 0, aw \mapsto 0]\) \(\in \hat{M}_{\text{startwrite}\_\text{writer}}\). Applying sequential consistency to the next step of the same thread to get \([ww \mapsto 0, ar \mapsto 0, aw \mapsto 1]\) \(\in \hat{M}_{\text{startwrite}\_\text{writer}}\). Applying sequential consistency to the termination of the critical section, we obtain \([ww \mapsto 0, ar \mapsto 0, aw \mapsto c]\) \(\in \hat{I}_{control}\).
the same thread, we obtain \([ww → 0, ar → b, aw → c] ∈ M_{\text{startreadC.reader}}\). Applying sequential consistency to the transition terminating the critical section, we obtain \([ww → 0, ar → b, aw → c] ∈ I_{\text{control}}\).

Step 3. Now we will prove by induction on \(a\) that for all \(a, b ≥ 0, c ∈ \{0, 1\}\) the map \([ww → a, ar → b, aw → c] \) is in \(I_{\text{control}}\). Notice that for \(a = 0\) we have just proven \([ww → 0, ar → b, aw → c] ∈ I_{\text{control}}\). For \(a > 0\), the induction assumption states that \([ww → a−1, ar → b, aw → c] ∈ I_{\text{control}}\). Sequential consistency, applied to any pair of states \(v, v′\) with \(v(pc_{\text{reader}}) = \text{askwriteA}\) and \(v → \text{writer}, v′\), gives \([ww → a−1, ar → b, aw → c] ∈ M_{\text{askwriteA.writer}}\). Applying sequential consistency once again gives \([ww → a, ar → b, aw → c] ∈ M_{\text{askwriteC.writer}}\), and applying it again gives \([ww → a, ar → b, aw → c] ∈ I_{\text{control}}\).

We have shown that \(I_{\text{control}} ⊆ I_{\text{control}}\).

From fact 1 we get \(M_{\text{startreadA.reader}} ⊆ M_{\text{startreadA.reader}}, M_{\text{finishreadA.reader}} ⊆ M_{\text{finishreadA.reader}}, M_{\text{read.reader}} ⊆ M_{\text{read.reader}}, (1 ≤ i ≤ n) and M_{\text{askwriteA.writer}} ⊆ M_{\text{askwriteA.writer}}, M_{\text{startwriteA.writer}} ⊆ M_{\text{startwriteA.writer}}, M_{\text{write.writer}} ⊆ M_{\text{write.writer}}, M_{\text{finishwriteA.writer}} \subseteq M_{\text{finishwriteA.writer}}, \).

Fix some \(i ∈ \mathbb{N}_0\) and states \(v → \text{reader}, v′\) such that \(v(pc_{\text{reader}}) = \text{startreadA}, v(ww) = 0, v(ar) ≥ 0\) and \(v(aw) ≤ 1\). Then \(v|_{\text{LocalDataV ar}, \cup \\\text{ProofVar}(\{\text{startreadA}\})} = \emptyset \in M_{\text{startreadA.reader}}\) and \(v|_{\text{ProofVar(control)}} \in I_{\text{control}}\). By sequential consistency, \(v′|_{\text{LocalDataV ar}, \cup \\\text{ProofVar}(\{\text{startreadA}\})} = v′|_{\text{control}} \in M_{\text{startreadA.reader}}\). Thus for any \(b ≥ 0\) and \(c ∈ \{0, 1\}\) we have \([ww → 0, ar → b, aw → c] ∈ M_{\text{startreadB.reader}}\). In particular, \(M_{\text{startreadB.reader}} \subseteq M_{\text{startreadB.reader}}\). Applying sequential consistency to \(v′\) and \(v″\) with \(v′ → \text{reader}, v″\) such that \(v′(pc_{\text{reader}}) = \text{startreadB}, v′(ww) = 0\) and \(v″(ar) ≥ 0\) and \(v″(aw) ∈ \{0, 1\}\), we obtain \(M_{\text{startreadC.reader}} \subseteq M_{\text{startreadC.reader}}\).

Analogously we get \(M_{\text{finishreadB.reader}} \subseteq M_{\text{finishreadB.reader}}, M_{\text{finishreadC.reader}} \subseteq M_{\text{finishreadC.reader}}, (1 ≤ i ≤ n) and M_{\text{askwriteB.writer}} \subseteq M_{\text{askwriteB.writer}}, M_{\text{askwriteC.writer}} \subseteq M_{\text{askwriteC.writer}}, M_{\text{askwriteD.writer}} \subseteq M_{\text{askwriteD.writer}}, M_{\text{askwriteE.writer}} \subseteq M_{\text{askwriteE.writer}}, M_{\text{finishwriteC.writer}} \subseteq M_{\text{finishwriteC.writer}}, M_{\text{finishwriteD.writer}} \subseteq M_{\text{finishwriteD.writer}}, M_{\text{finishwriteE.writer}} \subseteq M_{\text{finishwriteE.writer}}\).

So \(P\) is really the smallest Owicki-Gries-core proof.
Now we will show that $\gamma_{OG}(P)$ equals

$$\{ v \in ConsState \mid v(ww), v(ar) \in \mathbb{N}_0 \land v(aw) \in \{0,1\}$$
$$\land \forall i \in \{1,\ldots,n\} :$$
$$\land (v(pc_{\text{reader}}) \in \{\text{ready, read} \mid x \in \{\text{start, finish}\}$$
$$\land y \in \{A,B,C\} ) )$$
$$\land (v(pc_{\text{reader}}) = \text{startreadB} \Rightarrow v(ww) = 0)$$
$$\land (v(pc_{\text{reader}}) = \text{startreadC} \Rightarrow v(ww) = 0 < v(ar) )$$
$$\land \forall j \in \{1,\ldots,m\} :$$
$$\land (v(pc_{\text{writer}}) \in \{\text{write, write} \mid x \in \{\text{ask, start, finish}\}$$
$$\land y \in \{A,B,C\} ) )$$
$$\land (v(pc_{\text{writer}}) = \text{askwriteC} \Rightarrow v(ww) > 0)$$
$$\land (v(pc_{\text{writer}}) = \text{startwriteB} \Rightarrow v(ar) = v(aw) = 0)$$
$$\land (v(pc_{\text{writer}}) = \text{startwriteC} \Rightarrow v(ar) = 0 < v(aw) )$$
$$\land (v(pc_{\text{writer}}) = \text{finishwriteC} \Rightarrow v(aw) = 0 ) \}. $$

"⊆": Since there are no local data variables, the defining condition of $\gamma_{OG}(P)$ 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 $\pi_{\text{var}}(\gamma_{OG}(P)) \subseteq \{0,1\}$. Some $M_{aw}$ 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.

"⊇": 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.

\[ \square \]

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

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$rsc(p)$</td>
<td>$\emptyset$</td>
<td>${r}$</td>
<td>${r'}$</td>
<td>${r,r'}$</td>
</tr>
</tbody>
</table>

25
We have

\[ \text{init} = \{ u \mapsto 1, z \mapsto 0, x \mapsto 0, y \mapsto 0, l \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto 0 \}. \]

Let \( P = (\langle r \mapsto I_r, r' \mapsto I_{r'} \rangle, (M_{p,r})_{p \in PL \times Td}) \), where

\[ I_r = \{ [u \mapsto 0, z \mapsto 1], [u \mapsto 1, z \mapsto 0] \}, \]

\[ I_{r'} = \{ [x \mapsto 0, y \mapsto 0], [x \mapsto 0, y \mapsto 1], [x \mapsto 1, y \mapsto 1] \}, \]

\[ M_{A} = \{ [x \mapsto 0, l \mapsto 0], [x \mapsto 1, l \mapsto 1] \}, \]

\[ M_{B} = \{ [u \mapsto 0, z \mapsto 1, x \mapsto 0, l \mapsto 0], [u \mapsto 1, z \mapsto 0, x \mapsto 1, l \mapsto 1] \}, \]

\[ M_{C} = \{ [u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 0, l \mapsto 0], [u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 1, l \mapsto 0], [u \mapsto 1, z \mapsto 0, x \mapsto 0, y \mapsto 1, l \mapsto 1], [u \mapsto 1, z \mapsto 0, x \mapsto 0, y \mapsto 1, l \mapsto 1] \}, \]

\[ M_{D} = \{ [y \mapsto 0, z \mapsto 0], [y \mapsto 1, z \mapsto 1] \}, \]

\[ M_{E} = \emptyset \text{ for } (x, t) \in (PL \times Td) \setminus \{(A,1), (B,1), (C,1), (D,1), (E,1), (0,2)\}. \]

Notice that the two initial states satisfy annotations of locations A and 0 as well as the resource invariants. Thus the program annotation proves the initial states are consistently sequenced. 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 \( (\langle r \mapsto I_r, r' \mapsto I_{r'} \rangle, (M_{p,r})_{p \in PL \times Td}) \). From the initial condition we get \([u \mapsto 0, z \mapsto 1], [u \mapsto 1, z \mapsto 0] \in I_r \), so \( I_r \subseteq I_r \). The initial condition also implies \([x \mapsto 0, l \mapsto 0], [x \mapsto 1, l \mapsto 1] \in M_{A,1} \), so \( M_{A,1} \subseteq M_{A,1} \). Again from the initial condition we get \([y \mapsto 0, z \mapsto 0], [y \mapsto 1, z \mapsto 1] \in M_{D,2} \), so \( M_{D,2} \subseteq M_{D,2} \). At last the initial condition gives \([x \mapsto 0, y \mapsto 0], [x \mapsto 1, y \mapsto 1] \in I_r \). Now let

\[ \nu^A = \{ u \mapsto 1, z \mapsto 0, x \mapsto 1, y \mapsto 1, l \mapsto 1, pc_1 \mapsto A, pc_2 \mapsto 0 \} \]

and

\[ \nu^B = \{ u \mapsto 1, z \mapsto 0, x \mapsto 1, y \mapsto 1, l \mapsto 1, pc_1 \mapsto B, pc_2 \mapsto 0 \} \]

Notice that \( rsc(A) = \emptyset, rsc(B) = \{ r \} \), so

\[ \nu^A|_{LocalData \cap \langle L \rangle_{ProverL \cap (rsc(A))}} = [x \mapsto 1, l \mapsto 1] \in \tilde{M}_{A,1} \]

and

\[ \nu^B|_{LocalData \cap \langle L \rangle_{ProverL \cap (rsc(B))}} = [u \mapsto 1, z \mapsto 0] \in \tilde{I}^v. \]

Since \( \nu^A(u) = 1 = \nu^A(l) \), we have \( \nu^A \rightarrow^1 \nu^B \). By sequential consistency,

\[ \nu^B|_{LocalData \cap \langle L \rangle_{ProverL \cap (rsc(B))}} \in \tilde{M}_{B,1} \], i.e., \([u \mapsto 1, z \mapsto 0, x \mapsto 1, l \mapsto 1] \in \tilde{M}_{B,1} \).

Let

\[ \nu^C = \{ u \mapsto 1, z \mapsto 0, x \mapsto 1, y \mapsto 1, l \mapsto 1, pc_1 \mapsto C, pc_2 \mapsto 0 \} \]

and

\[ \nu^B|_{LocalData \cap \langle L \rangle_{ProverL \cap (rsc(c))}} = [x \mapsto 1, y \mapsto 1] \in \tilde{I}^v \] and \( \nu^B \rightarrow^1 \nu^C \), we obtain from the sequential consistency condition that \( \nu^C|_{LocalData \cap \langle L \rangle_{ProverL \cap (rsc(c))}} \in \tilde{M}_{C,1} \).

Let

\[ \nu^D = \{ u \mapsto 1, z \mapsto 0, x \mapsto 0, y \mapsto 1, l \mapsto 1, pc_1 \mapsto D, pc_2 \mapsto 0 \} \]

and

\[ \nu^B \rightarrow^1 \nu^C \), sequential consistency gives \( \nu^B|_{LocalData \cap \langle L \rangle_{ProverL \cap (rsc(B))}} \in \tilde{M}_{D,1} \).

Let

\[ \nu^E = \{ u \mapsto 1, z \mapsto 0, x \mapsto 0, y \mapsto 1, l \mapsto 1, pc_1 \mapsto E, pc_2 \mapsto 0 \} \]

From \( \nu^B \rightarrow^1 \nu^C \), sequential consistency and \( r' \in rsc(D) \setminus rsc(E) \) we get

\[ \nu^E|_{LocalData \cap \langle L \rangle_{ProverL \cap (rsc(E))}} \in \tilde{M}_{E,1} \] and \( L_{r'} \supseteq \nu^B|_{ProverL \cap (r') = [x \mapsto 0, y \mapsto 1] \}

Thus \( I_{r'} \subseteq \tilde{I}^v \).

Now let

\[ \nu^A = \{ u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 1, l \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto 0 \} \]

and

\[ \nu^B = \{ u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 1, l \mapsto 0, pc_1 \mapsto B, pc_2 \mapsto 0 \}. \]
Since \( v^A(u) = 0 = v^A(l) \), we have \( v^A \rightarrow_1 v^B \). Since additionally 
\[ v^A|_{LocalData,rsc(\{k\})} = [x \mapsto 0, l \mapsto 0] \in M_{k,1}, \ rsc(v^B) \setminus rsc(v^A) = \{ r \} \] and 
\[ v^A|_{ProofVar(r)} = [u \mapsto 0, z \mapsto 1] \in \bar{I}, \] we have 
\[ M_{k,1} \supseteq v^B|_{LocalData,rsc(\{k\})} = [u \mapsto 0, z \mapsto 1, x \mapsto 0, l \mapsto 0]. \]
Since \([u \mapsto 1, z \mapsto 0, x \mapsto 1, l \mapsto 1],[u \mapsto 0, z \mapsto 1, x \mapsto 0, l \mapsto 0] \in M_{B,1}\), we have 
\[ M_{B,1} \subseteq M_{B,1}. \]
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 M_{B,1}, [x \mapsto 0, y \mapsto 0],\]
\([x \mapsto 0, y \mapsto 1],[x \mapsto 1, y \mapsto 1] \in \bar{I}'\) we get \([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] \subseteq M_{E,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 M_{B,1}, \]
i.e., \( M_{B,1} \subseteq M_{B,1}. \)
Applying sequential consistency to \([u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 0, l \mapsto 0],[u \mapsto 1, z \mapsto 0, x \mapsto 0, y \mapsto 1, l \mapsto 1] \in M_{B,1}\) 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 M_{E,1}, \]
i.e., \( M_{E,1} \subseteq M_{E,1}. \)
We have shown that \( P \) is the smallest Owicki-Gries-core proof. It denotes the following states (column = state):

<table>
<thead>
<tr>
<th></th>
<th>u</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>z</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>l</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pc_1</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>pc_2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ \square \]

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

\[ \square \]

Extension of Example 4. A syntactic structure \((Tid, PL, Val, Res, \{Local_i, pc_x\})_{i \in Tid}; ProofVar, rsc\) belongs to the class \(StrSepThreads\) iff \( Res = \emptyset \) and \( Tid \) is finite. An RPL transition system \((init, \rightarrow_{ProofVar})_{i \in Tid}\) belongs to the class \(SepThreads\) iff it obeys a syntactic structure from \(StrSepThreads\) and \( \forall v \in ConsState:\ (\forall t \in Tid \exists v \in init: v \sim_{Local_i} \bar{v}) \Rightarrow v \in init. \)
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\(^5\). \[ \square \]

\(^5\) 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.
Lemma 28.

Let the sound 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

\[ \text{next}^\#_{\text{OG}} : PA \to PA, \]
\[ (I, M) \mapsto ((\{ v^t \}_{\text{LocalData}(v)} \mid \exists t \in \text{Tid}, v \to t v^t \land \]
\[ v_{\text{LocalData}(v)} \cup \{ v_{\text{LocalData}(v)}(\text{pc}_i) \} \in M_{(\text{pc}_i), t} \land \]
\[ r \in \text{rsc}(v_{\text{pc}_i}) \} \}_{r \in \text{Res}}, \]
\[ \forall r \in \text{rsc}(p) \} \}_{r \in \text{Tid}, p \in \text{Tid}}, \]
\[ I_{\text{init}} = \{ (\{ v_{\text{LocalData}(v)}(v) \in \text{init}) \}_{r \in \text{Res}}, \]
\[ M_{\text{init}} = \{ (\{ v_{\text{LocalData}(v)}(v \in \text{init} \land \text{pc}_i = p) \}_{r \in \text{PL}, p \in \text{Tid}}. \]

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

\[ \text{post}^\#_{\text{OG}} : PA \to PA, \]
\[ (I, M) \mapsto (I, M) \sqcup \text{next}^\#_{\text{OG}}(I, M), \]

and the Owicki-Gries-core abstract transformer be

\[ F^\#_{\text{OG}} : PA \to PA, \]
\[ (I, M) \mapsto (I_{\text{init}}, M_{\text{init}}) \sqcup \text{post}^\#_{\text{OG}}(I, M). \]

Lemma 28. The map \( \text{next}^\#_{\text{OG}} \) is monotone.

Proof. Let \((I, M) \subseteq (I', M')\) be program annotations, \(\text{next}^\#_{\text{OG}}(I, M) = (I', M')\), \(\text{next}^\#_{\text{OG}}(I', M') = (I'', M'')\).

\(\forall r \in \text{Res} : I'_r \subseteq I''_r\): Let \(r \in \text{Res}\). Let \(w \in I'_r\). Then there is a thread \(t \in \text{Tid}\) and some transition \(v \to t v^t\) such that \(v_{\text{LocalData}(v)} \cup \{ v_{\text{LocalData}(v)}(\text{pc}_i) \} \in M_{(\text{pc}_i), t} \land \]
\[ r \in \text{rsc}(v_{\text{pc}_i}) \} \}_{r \in \text{Res}}, \]
\[ \forall r \in \text{rsc}(p) \} \}_{r \in \text{Tid}}, \]
\[ I_{\text{init}} = \{ (\{ v_{\text{LocalData}(v)}(v \in \text{init}) \}_{r \in \text{Res}}, \]
\[ M_{\text{init}} = \{ (\{ v_{\text{LocalData}(v)}(v \in \text{init} \land \text{pc}_i = p) \}_{r \in \text{PL}, p \in \text{Tid}}. \]

So \(w \in I''_r\).

\(\forall t \in \text{Tid}, p \in \text{PL} : M'_{p,t} \subseteq M''_{p,t}\): Let \(t \in \text{Tid} \land p \in \text{PL}\). Let \(w \in M'_{p,t}\). Then there is some transition \(v \to t v^t\) such that \(v_{\text{LocalData}(v)} \cup \{ v_{\text{LocalData}(v)}(\text{pc}_i) \} \in M_{(\text{pc}_i), t} \land \]
\[ r \in \text{rsc}(v_{\text{pc}_i}) \} \}_{r \in \text{Res}}, \]
\[ \forall r \in \text{rsc}(p) \} \}_{r \in \text{Tid}} \land w = \{ v_{\text{LocalData}(v)}(v_{\text{pc}_i}, t) \}_{t \in \text{Tid}}, \]
\[ \forall r \in \text{rsc}(p) \} \}_{r \in \text{Tid}}, \]
\[ v_{\text{LocalData}(v)}(v_{\text{pc}_i}, t) \}_{t \in \text{Tid}}. \]

Corollary 29. The Owicki-Gries-core abstract transformer \(F^\#_{\text{OG}}\), 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 \(\text{postfix}(f)\) 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:

$$\Delta = \text{postfp}(F^{\Delta}_{OG})$$.

Proof. “$$\subseteq$$”: Let $$(I, M) \in \Delta$$. We will show that $$(I', M') := F^\Delta_{OG}(I, M)$$ is less than or equal to $$(I, M)$$. Let $$(I, M) = \text{next}^0_{OG}(I, M)$$. 

“$$\forall r \in \text{Res} : I_r' \subseteq I_r$$: Let $$r \in \text{Res}$$. Let $$w \in I_r'$$. Then $$w \in I_r^{\text{init}} \cup I_r \cup I_r$$. Case $$w \in I_r^{\text{init}}$$. Then there is some $$v \in \text{init}$$ such that $$v_{\text{ProofVar}(r)} = w$$. By the initial condition, $$v \in \gamma_{OG}(I, M)$$. Since every resource is available in an initial state, $$r \in \text{Res} \setminus \text{res}(v(PC\text{Var}))$$. Thus $$v_{\text{ProofVar}(r)} \in I_r$$.

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

Case $$w \in I_r$$. Then there is $$t \in \text{Tid}, (v, v') \in \gamma_r$$ such that $$v_{\text{LocalDataVar} \cup \text{ProofVar}(\text{res}(v(pc_i)))} \in M(p_{c}) \cup r \in \text{res}(v(p_{c})) \setminus \text{res}(v(p_{c}))$$ and $$w = v_{\text{ProofVar}(r)}$$. By (5) we have $$\text{res}(v(pc_i)) \supseteq \text{res}(v(pc_i))$$. Sequential consistency implies $$v_{\text{ProofVar}(r)} \in I_r$$.

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

“$$\forall t \in \text{Tid}, p \in \text{PL} : M'_{p,t} \subseteq M_{p,t}$$”: Let $$t \in \text{Tid}, p \in \text{PL}$$. Let $$w \in M'_{p,t}$$, Then $$w \in M^\text{init}_{p,t} \cup M_{p,t} \cup M_{p,t}$$. Case $$w \in M^\text{init}_{p,t}$$. Then there is some $$v \in \text{init}$$ such that $$v(p_{c_i}) = p$$ and $$w = v_{\text{LocalDataVar}}$$.

By the initial condition, $$v \in \gamma_{OG}(I, M)$$. From definition of $$\gamma_{OG}$$ and $$\text{res}(v(p_{c_i})) = \{v \in \text{init} \mid \emptyset\}$$ we get $$v_{\text{LocalDataVar}} \in M_{p,t}$$. Case $$w \in M_{p,t}$$. Then nothing more has to be proven.

Case $$w \in M_{p,t}$$. Then there is $$t \in \text{Tid}, (v, v') \in \gamma_t$$ such that $$v_{\text{LocalDataVar} \cup \text{ProofVar}(\text{res}(v(pc_i)))} \in M(p_{c}) \cup r \in \text{res}(p) \setminus \text{res}(v(p_{c}))$$, $$v_{\text{ProofVar}(r)} \in I_r$$, and $$w = v_{\text{LocalDataVar} \cup \text{ProofVar}(\text{res}(p))}$$. Then $$v_{\text{LocalDataVar} \cup \text{ProofVar}(\text{res}(p))} \in M_{p,t}$$ by sequential consistency.

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

“$$\supseteq$$”: Let $$(I, M) \in \text{postfp}(F^\Delta_{OG})$$, i.e., for $$(I', M') = F^\Delta_{OG}(I, M)$$ we have $$(I', M') \subseteq (I, M)$$. We’ll show that $$(I, M)$$ is an Owicki-Gries-core proof.

Sequential consistency: Let $$t \in \text{Tid}, (v, v') \in \gamma_t$$, $$p = v(p_{c_i})$$, $$p' = v'(p_{c_i})$$, $$R = \text{res}(p)$$, $$R' = \text{res}(p')$$, $$v_{\text{LocalDataVar} \cup \text{ProofVar}(\text{res}(p))} \in M_{p,t}$$, and $$\forall r \in R \setminus R' : v_{\text{ProofVar}(r)} \in I_r$$. By definition of $$\text{next}_{OG}^\Delta$$, we obtain $$v_{\text{LocalDataVar} \cup \text{ProofVar}(\text{res}(p'))} \in M'_{p',t} \subseteq M'_{p',t}$$. If any $$r \in R \setminus R'$$ is given, the definition of $$\text{next}_{OG}$$ implies $$v_{\text{ProofVar}(r)} \in I_r \subseteq I_r$$.

Initial condition: Let $$v \in \text{init}$$. Let $$r \in \text{Res} \setminus \text{res}(v(PC\text{Var}))$$. Then $$v_{\text{ProofVar}(r)} \in I_r^{\text{init}} \subseteq I_r \subseteq I_r$$.

Now let $$t \in \text{Tid}$$. Then $$v_{\text{LocalDataVar}} \in M^{\text{init}}_{p_{c_i},t}$$.

Since initial states don’t have any busy resources, $$v_{\text{LocalDataVar} \cup \text{ProofVar}(\text{res}(v(pc_i)))} \in M^{\text{init}}_{p_{c_i},t} \subseteq M'_{p_{c_i},t} \subseteq M_{p_{c_i},t}$$.

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

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

$$\text{inf}(\Delta) = \text{lfp}(F^\Delta_{OG})$$.
Thus

\[ T \]

Proof. Let \((I,M) \in PA. Let v' \in post(I_M, M)\). Let \((I', M') = \text{next}_I(I, M)\) and \((I', M') = post(I, M)\). Then \((I, M) = (I, M) \cup (I', M')\). By definition of post there is some \(t \in Tid\) and some \(v \in \gamma(I, M)\) such that \(v \rightarrow_t v'\). By definition of \(\gamma\) we have \(v'\) is in \(\gamma(I, M)\). We have to prove the following two claims.

Let \(r \in Res \setminus \bigcup_{\gamma \in Tid} \text{rsc}(v'(PCVar))\): \(v'\) is in \(\gamma(I, M)\). Let \(r \in Res \setminus \bigcup_{\gamma \in Tid} \text{rsc}(v'(PCVar))\). Then \(r \in \text{rsc}(v'(pc_i))\) for some \(\gamma \in Tid\). Then \(r \in \text{rsc}(v'(pc_i))\) and \(v'(pc_i) \neq v'(pc_i)\). Then (1) implies that \(\gamma = t\). So \(r \in \text{rsc}(v'(pc_i))\) and \(v'(pc_i) \neq v'(pc_i)\). Then \(v'\) is in \(\gamma\) by definition of \(\gamma\).

Case \(r \notin \bigcup_{\gamma \in Tid} \text{rsc}(v'(PCVar))\). The definition of \(\gamma\) implies that \(v'\) is in \(\gamma\). Notice that \(r \notin \text{rsc}(v'(pc_i))\). Thus for all \(x \in \text{PCVar}\) we have \(v(x) = v'(x)\). Thus \(v'\) is in \(\gamma\).

In both cases \(v'(PCVar) \in \gamma\).

Proof. Let \(\gamma(OG) \subseteq post^\#_I(I, M)\). Let \((I, M) \in PA. Let v' \in post(I, M)\). Let \((I', M') = \text{next}_I(I, M)\) and \((I', M') = post(I, M)\). Then \((I, M) = (I, M) \cup (I', M')\). By definition of post there is some \(t \in Tid\) and some \(v \in \gamma(I, M)\) such that \(v \rightarrow_t v'\). By definition of \(\gamma\) we have \(v'\) is in \(\gamma(I, M)\). We have to prove the following two claims.

Let \(r \in Res \setminus \bigcup_{\gamma \in Tid} \text{rsc}(v'(PCVar))\): \(v'\) is in \(\gamma(I, M)\). Let \(r \in Res \setminus \bigcup_{\gamma \in Tid} \text{rsc}(v'(PCVar))\). Then \(r \in \text{rsc}(v'(pc_i))\) for some \(\gamma \in Tid\). Then \(r \in \text{rsc}(v'(pc_i))\) and \(v'(pc_i) \neq v'(pc_i)\). Then (1) implies that \(\gamma = t\). So \(r \in \text{rsc}(v'(pc_i))\) and \(v'(pc_i) \neq v'(pc_i)\). Then \(v'\) is in \(\gamma\) by definition of \(\gamma\).

Case \(r \notin \bigcup_{\gamma \in Tid} \text{rsc}(v'(PCVar))\). The definition of \(\gamma\) implies that \(v'\) is in \(\gamma\). Notice that \(r \notin \text{rsc}(v'(pc_i))\). Thus for all \(x \in \text{PCVar}\) we have \(v(x) = v'(x)\). Thus \(v'\) is in \(\gamma\).

In both cases \(v'(PCVar) \in \gamma\).

Proof. Let \(\gamma(OG) \subseteq post^\#_I(I, M)\). Let \((I, M) \in PA. Let v' \in post(I, M)\). Let \((I', M') = \text{next}_I(I, M)\) and \((I', M') = post(I, M)\). Then \((I, M) = (I, M) \cup (I', M')\). By definition of post there is some \(t \in Tid\) and some \(v \in \gamma(I, M)\) such that \(v \rightarrow_t v'\). By definition of \(\gamma\) we have \(v'\) is in \(\gamma(I, M)\). We have to prove the following two claims.

Let \(r \in Res \setminus \bigcup_{\gamma \in Tid} \text{rsc}(v'(PCVar))\): \(v'\) is in \(\gamma(I, M)\). Let \(r \in Res \setminus \bigcup_{\gamma \in Tid} \text{rsc}(v'(PCVar))\). Then \(r \in \text{rsc}(v'(pc_i))\) for some \(\gamma \in Tid\). Then \(r \in \text{rsc}(v'(pc_i))\) and \(v'(pc_i) \neq v'(pc_i)\). Then (1) implies that \(\gamma = t\). So \(r \in \text{rsc}(v'(pc_i))\) and \(v'(pc_i) \neq v'(pc_i)\). Then \(v'\) is in \(\gamma\) by definition of \(\gamma\).

Case \(r \notin \bigcup_{\gamma \in Tid} \text{rsc}(v'(PCVar))\). The definition of \(\gamma\) implies that \(v'\) is in \(\gamma\). Notice that \(r \notin \text{rsc}(v'(pc_i))\). Thus for all \(x \in \text{PCVar}\) we have \(v(x) = v'(x)\). Thus \(v'\) is in \(\gamma\).

In both cases \(v'(PCVar) \in \gamma\).

**Remark 32.** A reader might wonder why we have not taken the map \(PA \rightarrow PA, (I, M) \mapsto (\text{next}_I, \text{next}_I) \cup \text{next}_I^\#(I, M)\) 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, \(\text{next}_I^\#\) is in general not a sound approximation of \(\text{post}\) under \((\alpha, \gamma(OG))\).
Let

\[ \alpha_{OG} : D \to PA, \]

\[ S \mapsto (\{ v |_{\text{ProofVar}(r)} \mid v \in S \land r \not\in \bigcup_{r \in \text{Res}} (v(\text{PCVar})) \})_{r \in \text{Res}}, \]

\[ (\{ v |_{\text{LocalDVar} \cup \text{ProofVar}(\text{rsc}(p))} \mid v \in S \land p = v(p_i) \})_{p \in PL}. \]

A pair of adjoint maps\(^7\) between two posets \((X, \leq), (Y, \preceq)\) is a pair of maps \((\alpha, \gamma)\) such that \(\alpha : X \to Y, \gamma : Y \to X\) and

\[ \forall x \in X, y \in Y : \alpha(x) \preceq y \iff x \leq \gamma(Y). \]

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

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

“only if”: Let \(v \in S\). By definition of \(D\) we get \(v \in \text{ConsState}\). Let \(r \in \text{Res} \setminus \bigcup_{r \in \text{Res}} (v(\text{PCVar}))\).

By definition of \(\alpha_{OG}\) we get \(v |_{\text{ProofVar}(r)} \in I_r\). By assumption, \(I_r \subseteq J_r\), so \(v |_{\text{ProofVar}(r)} \in J_r\).

Now let \(t \in \text{Tid} \) and \(p = v(p_i)\). By definition of \(\alpha_{OG}\) we get \(v |_{\text{LocalDVar} \cup \text{ProofVar}(\text{rsc}(p))} \in M_{p,t}\). By assumption, \(N_{p,t} \supseteq M_{p,t}\). Thus \(N_{p,t} \ni v |_{\text{LocalDVar} \cup \text{ProofVar}(\text{rsc}(v(p_i)))}\).

Combining, we obtain \(v \in \gamma_{OG}(T)\).

“if”: Let \(r \in \text{Res}\). Let \(u \in I_r\). By definition of \(\alpha_{OG}\) there is \(v \in S\) such that \(r \not\in \bigcup_{r \in \text{Res}} (v(\text{PCVar}))\) and \(u = v |_{\text{ProofVar}(r)}\). By assumption, \(v \in \gamma_{OG}(T)\). By definition of \(\gamma_{OG}\) we have \(u \in J_{p,t}\). Thus \(I_r \subseteq J_{p,t}\).

Let \(t \in \text{Tid}\) and \(p \in \text{PL}\). Let \(u \in M_{p,t}\). By definition of \(\alpha_{OG}\) there is some \(v \in S\) such that \(p = v(p_i)\) and \(v |_{\text{LocalDVar} \cup \text{ProofVar}(\text{rsc}(p))} = u\). By assumption, \(v \in \gamma_{OG}(T)\).

By definition of \(\gamma_{OG}\) we get \(u \in N_{p,t}\). Thus \(M_{p,t} \subseteq N_{p,t}\).

Combining, we obtain \(\alpha_{OG}(S) \sqsubseteq T\). \(\blacksquare\)

Let \(\rho = \gamma_{OG} \circ \alpha_{OG}\).

Then for \(Q \in D\) we have \(\rho(Q) = \)

\[ \begin{cases} v \in \text{ConsState} \mid & \forall r \in \text{Res} \setminus \bigcup_{r \in \text{Res}} (v(\text{PCVar})) \exists \tilde{v} \in Q : (r \not\in \bigcup_{r \in \text{Res}} (v(\text{PCVar})) \land v \sim v |_{\text{ProofVar}(r)} \tilde{v}) \\ & \forall t \in \text{Tid} \exists \tilde{v} \in Q : v \sim v |_{\text{LocalDVar} \cup \text{ProofVar}(\text{rsc}(v(p_i)))) \tilde{v} \end{cases}. \]

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.** \(\rho : D \to D\) 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 \(\alpha\) and \(\gamma\) 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 \( \rho \) proves the strongest Owicki-Gries-provable property. Formally, let \( T = (\text{init},...) \) be an RPL transition system that obeys the given syntactic structure and has successor operator \( post \) and let \( \mathcal{D} \) be the lattice of Owicki-Gries-core proofs of \( T \); then

\[
\text{\text{lfp}}(\lambda x. \rho(\text{init} \cup post(x))) \subseteq \gamma_{\text{OG}}(\text{inf} \mathcal{D}).
\]

Proof. Let \( P = \text{inf} \mathcal{D}. \) By Prop. 27 we have

\[
\text{init} \cup post(\gamma_{\text{OG}}(P)) \subseteq \gamma_{\text{OG}}(P).
\]

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

\[
\alpha_{\text{OG}}(\text{init} \cup post(\gamma_{\text{OG}}(P))) \subseteq P.
\]

Right adjoints are monotone, so

\[
\gamma_{\text{OG}}(\alpha_{\text{OG}}(\text{init} \cup post(\gamma_{\text{OG}}(P)))) \subseteq \gamma_{\text{OG}}(P),
\]

otherwise stated

\[
\rho(\text{init} \cup post(\gamma_{\text{OG}}(P))) \subseteq \gamma_{\text{OG}}(P).
\]

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

In particular, abstract interpretation with \( \rho \) 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 \( \text{lfp}(\lambda x. \rho(\text{init} \cup post(x))) = \gamma_{\text{OG}}(I, M) \). So, for Readers-Writers, the Owicki-Gries-core proof method and abstract interpretation with \( \rho \) are equally strong.

Extension of Example 12. Notice that \( \rho(\text{init} \cup post(\text{init})) = \rho(\text{init} \cup \emptyset) = \rho(\text{init}) \subseteq \text{init} \). Since \( \text{init} \) is the set of reachable states, \( \text{lfp}(\lambda x. \rho(\text{init} \cup post(x))) = \text{init} \). Thus for the RPL transition system Upper abstract interpretation under \( \rho \) 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 \( \text{lfp}(\lambda x. \rho(\text{init} \cup post(x))) = \gamma_{\text{OG}}(\text{inf} \mathcal{D}) \).

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 \( \rho \) will also produce the set of reachable states.
E Lower bound on precision

For a set \( X \) of sets, let

\[
\text{Part}(X) = \{ Y \mid \exists T \subseteq X : Y = (\bigcap T) \cap (\bigcap_{Z \in X \setminus T} ((\bigcup X) \setminus Z)) \neq \emptyset \}
\]

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. \( \bigcup \text{Part}(X) = \bigcup X \);
2. for all different \( s_1, s_2 \in \text{Part}(X) \) we have \( s_1 \cap s_2 = \emptyset \);
3. if \( s \in \text{Part}(X) \) and \( S \in X \), then either \( s \cap S = \emptyset \) or \( s \subseteq S \);
4. if \( X \) is finite, so is \( \text{Part}(X) \).

**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 \( x \in (\bigcap T) \cap (\bigcap_{Z \in X \setminus T} ((\bigcup X) \setminus Z)) \).

2. Let \( T_1, T_2 \subseteq X \) be such that for \( s_1 = (\bigcap T_1) \cap (\bigcap_{Z \in X \setminus T_1} ((\bigcup X) \setminus Z)) \neq \emptyset \) and \( s_2 = (\bigcap T_2) \cap (\bigcap_{Z \in X \setminus T_2} ((\bigcup X) \setminus Z)) \neq \emptyset \) 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 \( \exists S \in T_1 : x \notin S \). 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 \( x \notin X \setminus T_1 \) (otherwise \( x \) were not in \( s_1 \)). So \( z \in T_1 \). So \( y \in S \). But \( y \in s_2 \), so \( y \in (\bigcup X) \setminus Z \). Contradiction!

3. Let \( s \in \text{Part}(X) \) 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 \( s = (\bigcap T) \cap (\bigcap_{Z \in X \setminus T} ((\bigcup X) \setminus Z)) \). So \( y \in \bigcap T \). If \( S \in T \), we get \( y \notin T \), 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, \( |\text{Part}(X)| \leq 2^{|X|} \).

For the rest of section E, fix a syntactic structure \((\text{Tid}, \text{PL}, \text{Val}, \text{Res}, (\text{Local}, \text{pc}), rsc) \in \text{Tid}, \text{ProofVar}, \text{rsc})\), let \( \text{ConsState} \) be the set of consistent states and \( D = \exists x (\text{ConsState}) \). Let

\[
\text{RL} = \text{Res} \cup \{ \text{Local} \mid \exists t \in \text{Tid} \} \quad \text{and} \quad \widetilde{\text{RL}} = \text{Part}(\text{RL}) .
\]

Let \( \rho : D \to D, \phi \mapsto Q \mapsto \{ v \in \text{ConsState} \mid \begin{cases} \forall s \in \text{RL}, r \in \text{Res} : s \subseteq r = (r \notin \bigcup rsc(v(\text{PCVar}))) \Rightarrow \exists \text{ } v \in Q : v \sim \text{ } \forall s \in \bigcup rsc(v(\text{PCVar})) \land \text{ } \forall t \in \text{Tid} \exists \text{ } v \in Q : \text{ } v \sim \text{Local}(\text{Local}) \subseteq s \land v \text{ }, \text{ } \end{cases} \} . \)
and
\[\rho_c : D \to D, \quad Q \mapsto \{ v \in ConsState \mid \forall s \in RL \exists \tilde{v} \in Q : v \sim_s \tilde{v}\}\]

Note that neither \(\bar{\rho}\) nor \(\rho_c\) 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 \(\rho_c\).

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

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

- Let \(s \in RL, r \in Res, s \subseteq r\).
  * Assume for a moment that \(r\) is available in \(v\). Then there is \(\tilde{v} \in Q\) such that \(v \sim_s \tilde{v}\) and \(r\) is available in \(\tilde{v}\). Notice that \(\tilde{v} \in Q'\).
  * Let \(t \in Tid\) and \(r \in rsc(v(pc_i))\). Then there is \(\hat{v} \in Q\) such that \(v \sim_{Local} \hat{v}\). Notice that \(\hat{v} \in Q'\).

- Let \(t \in Tid\). Then there is \(\hat{v} \in Q\) such that \(v \sim_{Local} \hat{v}\). Notice that \(\hat{v} \in Q'\).

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

Extensivity. Let \(Q \subseteq D\). Let \(v \in Q\).

- Let \(s \in RL, r \in Res, s \subseteq r\).
  * Assume for a moment that \(r\) is available in \(v\). Take \(\tilde{v} := v\).

- Let \(t \in Tid\) and \(r \in rsc(v(pc_i))\). Take \(\hat{v} := v\).

- Let \(t \in Tid\). Take \(\hat{v} := v\).

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

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

- Let \(s \in RL, r \in Res, s \subseteq r\).
  * Assume for a moment that \(r\) is available in \(v\). Then there is \(\tilde{v} \in \bar{\rho}(Q)\) such that \(v \sim_s \tilde{v}\) and \(r\) is available in \(\tilde{v}\). Then there is \(\hat{v} \in Q\) such that \(\tilde{v} \sim_s \hat{v}\) and \(r\) is available in \(\hat{v}\). Notice that \(v \sim_s \hat{v}\).
  * Let \(t \in Tid\) such that \(r \in rsc(v(pc_i))\). Then some \(\tilde{v} \in \bar{\rho}(Q)\) exists such that \(v \sim_{Local} \tilde{v}\). Then \(v(pc_i) = \tilde{v}(pc_i)\), so \(r \in rsc(v(pc_i))\). Then there is \(\hat{v} \in Q\) such that \(\tilde{v} \sim_{Local} \hat{v}\). Notice that \(v \sim_{Local} \hat{v}\).

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.** \(\rho_c\) is a closure operator on \((D, \subseteq)\)

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

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

2. Extensivity. Let \(Q \subseteq D\). Let \(v \in Q\). Let \(s \in RL\). Certainly \(v \sim_s v\). Thus \(v \in \rho_c(Q)\). We have shown \(Q \subseteq \rho_c(Q)\).

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

34
Proposition 38. For all \( Q \in D \) we have \( \hat{\rho}(Q) \subseteq \rho_c(Q) \).

**Proof.** Let \( Q \in D \). Let \( v \in \hat{\rho}(Q) \). Let \( s \in RL \). 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 \( \tilde{v} \in \hat{q} \) such that \( v \sim_s \tilde{v} \). Let \( \tilde{v} := \tilde{v} \).

Case \( r \in rsc(pc_i) \) for some \( t \in Tid \). Then there is \( \tilde{v} \in Q \) such that \( v \sim_{(Local_i \setminus Res),s} \tilde{v} \). Let \( \tilde{v} := \tilde{v} \).

Case \( s \cap \cup Res = \emptyset \). Since \( s \neq \emptyset \), there is some \( t \in Tid \) such that \( s \subseteq Local_i \setminus Res \). There is \( \tilde{v} \in Q \) such that \( v \sim_{Local_i \setminus Res} \tilde{v} \). Let \( \tilde{v} := \tilde{v} \).

In any case there is \( \tilde{v} \in Q \) such that \( v \sim_s \tilde{v} \). ■

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

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

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

**Lemma 39.** Let \( t \neq \hat{i} \) in Tid. Then

1. \( \forall \tau \in Tid \backslash \{ \hat{i} \} \quad v \sim_{Local_i \setminus Local_j} v' \wedge v \sim_{\hat{i} \in Tid Local_j} v' \).

2. \( Local_i \cap Local_j \subseteq \cap_{\hat{i} \in Tid} Local_j \).

**Proof.** 1. Let \( \tau \in Tid \backslash \{ \hat{i} \} \). From (1) we get \( v \sim_{Local_i \setminus Local_j} v' \) or \( v \sim_{Local_i \setminus Local_j} v' \). In any case \( v \sim_{Local_i \setminus Local_j} v' \). In particular, \( v \sim_{\hat{i} \in Tid Local_j} v' \).

2. “\( \subseteq \)” Let \( x \in Local_i \cap Local_j \) and \( \hat{i} \in Tid \). We are going to show that \( x \in Local_j \).

First, the definition of the syntactic structure implies \( x \notin PCV\). Thus, \( x \in Data\Varepsilon \).

Next, consider arbitrary \( \tau \in Tid \backslash \{ \hat{i} \} \) and \( (v, v') \in \rightarrow_\tau \). As shown above, \( v \sim_{Local_i \setminus Local_j} v' \). We have shown \( (\forall \tau \in Tid \backslash \{ \hat{i} \}, (v, v') \in \rightarrow_\tau : v(x) = v'(x) \).

By definition of RPL, \( x \in Local_j \).

“\( \supseteq \)” Follows from \( Local_i \supseteq \cap_{\hat{i} \in Tid} Local_j \) and \( Local_i \supseteq \cap_{\hat{i} \in Tid} Local_j \). ■

**Lemma 40.** Let \( s \in RL \) \( t \neq \hat{i} \) in Tid such that \( s \subseteq Local_i \cap Local_j \cap Res \).

Then \( \forall \sigma \in \mu, \quad v \in Y^{(\sigma)} \exists w \in \text{init}: v \sim_s w \).

**Proof.** By transfinite induction. Let \( \sigma \in \mu \) and \( \forall \hat{\sigma} \in \sigma \forall v \in Y^{(\hat{\sigma})} \exists w \in \text{init}: v \sim_s w \). Let \( v \in Y^{(\sigma)} \). Then \( \sigma > 0 \). There are two cases.

Case \( \sigma = \tilde{\sigma} + 1 \) for some \( \tilde{\sigma} \in \mu \). There is some \( r \in Res \) such that \( s \subseteq r \). There are two cases.

Case \( r \in rsc(pc_i) \) for some \( t \in Tid \). By definition of \( \hat{\rho} \) there is \( \tilde{v} \in \text{init} \cup post(Y^{(\sigma)}) \) such that \( v \sim_{(Local_i \setminus Res),s} \tilde{v} \). Let \( \tilde{v} := \tilde{v} \).

Case \( r \in rsc(\hat{v}(pc_i)) \) for some \( t \in Tid \). By definition of \( \hat{\rho} \) there is \( \tilde{v} \in \text{init} \cup post(Y^{(\sigma)}) \) such that \( v \sim_{PCV, s} \tilde{v} \). Let \( \tilde{v} := \tilde{v} \).

Since in initial states all resources are available, \( \tilde{v} \notin init \).

Thus there is some \( z \in Y^{(\sigma)} \). Part 1 of Lemma 39 implies \( \tilde{v} \sim_s z \), so \( v \sim_s z \).
Case \( r \notin \text{Unsc}(v(PC\text{Var})) \). Then there is \( \tilde{v} \in \text{init} \cup \text{post}(Y(\sigma)) \) such that \( v \sim_{w} \tilde{v} \). If \( \tilde{v} \in \text{init} \), take \( w := \tilde{v} \). Otherwise there is \( z \in Y(\sigma) \) such that \( \tilde{v} \in \text{post}(\{z\}) \). Part 1 of Lemma 39 implies \( \tilde{v} \sim_{v} z \), so \( v \sim_{w} z \).

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

Case \( \sigma \) is a limit ordinal. Then there is \( \sigma < \sigma \) such that \( v \in Y(\sigma) \). Apply the induction hypothesis.

\[ \text{Lemma 41. Let } t \neq \tilde{t} \text{ in } \text{Tid}. \text{ Then } \forall \sigma \in \mu, v \in Y(\sigma) \exists w \in \text{init} : v \sim_{(\text{Local} \cup \{ \text{Local}_l \}) \cup \{ R \text{es} \}} w. \]

\[ \text{Proof. Let } C = (\text{Local} \cap \text{Local}_l) \cup \{ \text{Res} \}. \text{ We’ll use transfinite induction. Let } \sigma \in \mu \text{ and } \forall \sigma < \sigma \forall v \in Y(\sigma) \exists w \in \text{init} : v \sim_{w} w. \text{ Let } v \in Y(\sigma) \text{. Then } \sigma > 0. \\]

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

Case \( \sigma \) is a limit ordinal. Then there is \( \sigma < \sigma \) such that \( v \in Y(\sigma) \). Apply the induction hypothesis.

\[ \text{Lemma 42. Let } s \in RL, t \in \text{Tid}, \text{ r } \in \text{Res}, s \subseteq r \cap \text{Local}_l \text{, and } \forall \tilde{t} \in \text{Tid} \setminus \{ t \} : s \cap \text{Local}_l = \emptyset. \text{ Then } \forall \sigma \in \mu, v \in Y(\sigma) : r \notin \text{rsc}(v(pc_{t})) \Rightarrow \exists w \in Y(\sigma) : v \sim_{w} w \wedge r \text{ is available in } w. \]

\[ \text{Proof. We’ll use transfinite induction. Let } \sigma \in \mu \text{ and } \forall \sigma < \sigma \forall v \in Y(\sigma) : r \notin \text{rsc}(v(pc_{t})) \Rightarrow \exists w \in Y(\sigma) : v \sim_{w} w \wedge r \text{ is available in } w. \text{ Let } v \in Y(\sigma) \text{ such that } r \notin \text{rsc}(v(pc_{t})). \text{ Then } \sigma > 0. \\]

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

Case \( \tau = t \). From \( \tau \neq \tilde{t} \) and (1) we obtain \( \text{rsc}(z(pc_{\tilde{t}})) = \text{rsc}(y(pc_{\tilde{t}})) \), thus \( r \in \text{rsc}(z(pc_{\tilde{t}})) \).

From \( z \in \text{ConsState} \) we get \( r \notin \text{rsc}(z(pc_{\tilde{t}})) \). Thus \( r \in \text{Res} \setminus (\text{rsc}(z(pc_{\tilde{t}})) \cup \text{rsc}(y(pc_{\tilde{t}}))) \).

From (2) we get \( z \sim_{t} y \), so \( z \sim_{y} y \), so \( z \sim_{v} v \). Apply the induction hypothesis to \( \sigma \) and \( z \).

Case \( \tau \neq \tilde{t} \). From (1) we obtain \( z \sim_{\text{Local}} y \), so \( z \sim_{y} y \), so \( z \sim_{v} v \). Also \( z(pc_{\tilde{t}}) = y(pc_{\tilde{t}}) \), thus \( r \notin \text{rsc}(z(pc_{\tilde{t}})) \). Apply the induction hypothesis to \( \sigma \) and \( z \).

Case \( \sigma \) is a limit ordinal. Then there is \( \sigma < \sigma \) such that \( v \in Y(\sigma) \). Apply the induction hypothesis to \( \sigma \) and \( v \).

\[ \text{Lemma 43. Let } t \in \text{Tid}, v \in Q, r \in \text{Res} \setminus \text{rsc}(v(pc_{t})), s \in RL, s \subseteq r \cap \text{Local}_l. \text{ Then } \exists w \in Q : v \sim_{w} w \wedge r \text{ is available in } w. \]

\[ \text{Proof. There is some } \sigma \in \mu \text{ such that } v \in Y(\sigma). \text{ Case there is } \tilde{t} \in \text{Tid} \setminus \{ t \} \text{ such that } s \subseteq \text{Local}_{\tilde{t}}. \text{ By Lemma 40 there is some } w \in \text{init} \subseteq Q \text{ such that } v \sim_{w} w. \text{ Notice that all resources are available in } w. \text{ Case } \forall \tilde{t} \in \text{Tid} \setminus \{ t \} : s \cap \text{Local}_l = \emptyset. \text{ Lemma 42 provides a suitable } w \in Y(\sigma) \subseteq Q. \]
Lemma 44. Let $t \in \text{Tid}$, $v \in Q$, $r \in \text{Res} \setminus \text{rsc}(v(pc_3))$. Then there is $w \in Q$ such that $r$ is available in $w$ and $w \sim_{\text{Local} \setminus \text{rsc}(w(pc_3))} v$.

Proof. If $r$ is available in $w$, take $w := v$. Otherwise there is $\bar{t} \in \text{Tid} \setminus \{t\}$ such that $r \in \text{rsc}(v(pc_3))$. Lemma 41 implies the existence of some $u \in \text{init}$ such that $u \sim_{\text{Local} \setminus \text{Local}\setminus \text{Res}} v$. Let

$$w : \text{Var} \rightarrow \text{Val}, \quad x \mapsto \begin{cases} u(x), & \text{if } x \in (\text{Local}\setminus \text{Res}) \cup ((\text{Local} \setminus \text{rsc}(v(pc_3))) \setminus \text{Local}_l), \\ v(x), & \text{otherwise}. \end{cases}$$

We’ll show:

1. $\text{rsc}(v(pc_3)) \cap \text{rsc}(w(\text{PCVar})) = \emptyset$: $\text{rsc}(v(pc_3)) \cap \text{rsc}(w(\text{PCVar})) = \text{rsc}(v(pc_3)) \cap (\text{rsc}(w(pc_2)) \cup \bigcup_{\bar{r} \in \text{Tid} \setminus \{t\}} \text{rsc}(w(pc_3))) = \emptyset$ (since $v \in \text{ConsState}$).

2. $w \sim_{\text{Local} \setminus \text{Res}(w(pc_3))} v'$: Let $x \in \text{Local}_l \setminus \text{rsc}(w(pc_3))$. There are two cases.

   Case $x \in \text{Local}_l \setminus \text{Res}$. Then $w(x) = u(x)$.

   Case $x \in \text{Local}_l \setminus \text{Res}$. Then $w(x) = v(x)$, so $w(x) = v(x)$.

3. $w \in Q$: We’ll show first that $w$ is in $\rho(Q)$ as follows.

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

   For all $t \in \text{Tid}$ we have $w(pc_3) \in \{u(pc_3), v(pc_3)\} \subseteq \text{PL}$ and $\text{rsc}(w(pc_2)) = \emptyset \subseteq \text{rsc}(v(pc_2))$. Thus for $\bar{t} \neq \bar{t}_1$ in $\text{Tid}$ we have $\text{rsc}(w(pc_1)) \cap \text{rsc}(w(pc_2)) \subseteq \text{rsc}(v(pc_1)) \cap \text{rsc}(v(pc_2)) = \emptyset$. Thus $w \in \text{ConsState}$.

   Next, let $s \in \bar{R}$ and $\bar{r} \in \text{Res}$ such that $s \subseteq \bar{r}$. We will show that

   \[ (\bar{r} \not\in \text{rsc}(w(\text{PCVar})) \Rightarrow \exists \bar{w} \in Q : w \sim_{\bar{r}} \bar{w} \wedge \bar{r} \not\in \text{rsc}(w(\text{PCVar}))) \]

   \[ \wedge (\forall \bar{t} \in \text{Tid} : \bar{r} \not\in \text{rsc}(w(pc_3)) \Rightarrow \exists \bar{w} \in Q : w \sim_{\text{Local}_l \setminus \text{Res}, \bar{r}} \bar{w}). \]

   Notice that $s \cap (\text{Local}_l \setminus \text{Res}) = \emptyset$.

   - Assume for a moment that $\bar{r} \not\in \text{rsc}(w(\text{PCVar}))$. We distinguish two cases.

     Case $\bar{r} \in \text{rsc}(v(pc_3))$. By part 3 of Lemma 35 one of the following cases holds.

     Case $s \subseteq \text{Local}_l$. Then $s \cap (\text{rsc}(v(pc_3)) \setminus \text{Local}_l) = \emptyset$, so $w \sim_s \bar{r}$. From $v \in \text{ConsState}$ we get $\bar{r} \not\in \text{rsc}(v(pc_3))$. By Lemma 43 there is $\bar{w} \in Q$ such that $v \sim_s \bar{w}$ and $\bar{r}$ is available in $\bar{w}$. Then $w \sim_s \bar{w}$.

     Case $s \cap \text{Local}_l = \emptyset$. Then $w \sim_s \bar{r}$. Take $\bar{w} := u$.

     Case $\bar{r} \not\in \text{rsc}(v(pc_3))$. Then $w \sim_s \bar{r}$. From $\bar{r} \not\in \text{rsc}(v(pc_3))$ we get $\bar{r} \not\in \text{rsc}(w(pc_3))$.

   - Let $\bar{r} \in \text{Tid}$ such that $\bar{r} \in \text{rsc}(w(pc_3))$. From $\bar{r} \not\in \text{rsc}(w(pc_3))$ we get $\bar{r} \neq \bar{t}$, so $\bar{r} \not\in \text{rsc}(v(pc_3))$. From consistency of $v$ we get $\bar{r} \not\in \text{rsc}(v(pc_3))$. Thus $s \cap \text{rsc}(v(pc_3)) = \emptyset$. So $s \cap (\text{Local}_l \setminus \text{Res}) \cup ((\text{rsc}(v(pc_3)) \setminus \text{Local}_l) = \emptyset$, thus $w \sim_s \bar{r}$. Now let $x \in \text{Local}_l \setminus \text{Res}$. Then $x \not\in (\text{rsc}(v(pc_3)) \setminus \text{Local}_l)$. 37
We claim: Let proof.

At last, we’ll show. We have shown.

Take $\tilde{\varphi}$. Initially, we’ll show that $\tilde{\varphi}$ is available in $w$. Then there is

Next, let $\tilde{\varphi}$. We have shown $w$. Take $\tilde{\varphi} = \varphi$.

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

Reformulation of Lemma 15. Let $t \in \mathit{Tid}$, $(v, v') \rightarrow_r T = rsc(v(pc_1), v' = rsc(v(pc_1), v \in Local_i \cup R v \sim_r v'$. Then there is $w' \in Q$ such that $w' \sim_{Local_i \cup R} v'$.

Proof. Notice that $r' \notin rsc(v(pc_1), v' = rsc(v(pc_1)$). Applying Lemma 44 to $v$ we obtain some $\tilde{v} \in Q$ such that $r$ is available in $\tilde{v}$ and $\tilde{v} \sim_{Local_i \cup R} v$.

Let $w: \forall r \rightarrow \mathit{val}, x \mapsto \begin{cases} \tilde{v}(x), & \text{if } x \in r, \\ \varphi(x), & \text{if } x \notin r. \end{cases}$

We claim:

“$r$ is available in $w$”: Follows from $rsc(w(PCV \varphi)) = rsc(\tilde{v}(PCV \varphi))$.

“$w \sim_{Local_i \cup R} v$”: From (5) we obtain $Local_i \cup R = (Local_i \setminus r) \cup r \cup \mathit{R}$. Then $w \sim_{Local_i \cup R} v$ by definition of $\tilde{\varphi}(Q)$.

“I $w \in Q$”: First, we’ll show that $w$ is in $\tilde{\varphi}(Q)$.

Initially, we’ll show that $w$ is consistent. For all $r \in \mathit{Tid}$ we have $w(pc_2) = \tilde{v}(pc_2) \in PL$. So for $t \in \mathit{Tid}$ we have $rsc(w(pc_1)) = rsc(\tilde{v}(pc_1))$.

Next, let $s \in R$, $r \in Res$, $s \subseteq t$. We have to show two statements.

“$\tilde{v} \notin rsc(w(PCV \varphi)) \Rightarrow \exists \tilde{v} \in Q: w \sim \tilde{v} \wedge t \notin rsc(\tilde{v}(PCV \varphi))$” Assume for a moment that $\tilde{v}$ is available in $w$. We distinguish two cases.

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

Case $t \neq r$. Then $t \cap r = \emptyset$. Thus, $w \sim_r \tilde{v}$. Since $rsc(w(PCV \varphi)) = rsc(\tilde{v}(PCV \varphi))$, we obtain that $\tilde{v}$ is available in $\tilde{v}$. Take $\tilde{w} = \tilde{v}$.

“$\forall r \in \mathit{Tid}, \forall \tilde{v} \in Q: w \sim_{Local_i \cup R(\tilde{v})} \tilde{v}$”. Let $\tilde{r} \in r$ such that $\tilde{r} \in rsc(w(pc_2))$. Since $r$ is available in $w$, we have $r \neq \tilde{r}$, so $w \sim_{Local_i \cup R(\tilde{r})} \tilde{v}$.

Take $\tilde{w} = \tilde{v}$.

At last, we’ll show:

“$\forall r \in \mathit{Tid}, \exists \tilde{w} \in Q: w \sim_{Local_i \cup R(\tilde{v})} \tilde{w}$”. Let $\tilde{r} \in r$ be given. Then $w \sim_{Local_i \cup R(\tilde{r})} \tilde{v}$. Take $\tilde{w} = \tilde{v}$.

We have shown $w \in \tilde{\varphi}(Q)$. From $\tilde{\varphi}(Q) = Q$ we obtain $w \in Q$. 38
By (4) and Remark 23 there is \( w' \) such that \( w \rightarrow r, w' \) and \( w' \sim_{\text{Local} \cup J R} v' \). Notice that \( w' \in \text{post}(Q) \subseteq Q \).

Let \( \mathcal{O} \) be the lattice of Owicki-Gries-core proofs of \( T \).

**Restatement of Theorem 18.** The strongest property provable by abstract interpretation with \( \rho \) is provable by the Owicki-Gries core. Formally:

\[
\gamma_{\text{OG}}(\inf \mathcal{O}) \subseteq \text{lpf}(\lambda x. \rho(\text{init} \cup \text{post}(x))).
\]

**Proof.** For each \( r \in \text{Res} \) let

\[
I_r = \{ u \in (\text{ProofVar}(r) \rightarrow \text{Val}) \mid \exists \hat{v} \in Q : u \sim_r \hat{v} \land r \text{ is available in } \hat{v} \},
\]

and for all \( t \in \text{Tid} \), \( p \in \text{PL} \) let

\[
M_{p,t} = \{ u \in ((\text{LocalDataVar}) \cup \bigcup \text{ProofVar}(\text{rsc}(p))) \rightarrow \text{Val}) \mid \exists \hat{v} \in Q : u \sim_{\text{LocalDataVar} \cup \text{LocalDataVar}(p)} \hat{v} \land \hat{v}(p_c) = p \}.
\]

Let \( P = (I_r)_{r \in \text{Res}}, (M_{p,t})_{p \in \text{PL}, t \in \text{Tid}} \). 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 \( t \in \text{Tid}, (v, v') \in \rightarrow_r, p = v(p_c), p' = v'(p_c), R = \text{rsc}(p), R' = \text{rsc}(p'), v'_l|\text{LocalDataVar}(p) \in M_{p_t} \), and \( \forall r \in R \setminus R' : v'_l|\text{ProofVar}(r) \in I_r \). Then there is \( \hat{v} \in Q \) such that \( v \sim_{\text{LocalDataVar}(p) \cup \text{LocalDataVar}(p')} \hat{v} \) and \( \hat{v}(p_c) = p \). Then \( \hat{v}(p_c) = v(p_c) \).

Thus \( \hat{v} \sim_{\text{LocalDataVar}(p) \cup \text{LocalDataVar}(p')} \). By (5) there are two cases.

Case \( R' \subseteq R \). Then \( \hat{v} \sim_{\text{LocalDataVar}(p) \cup \text{LocalDataVar}(p')} \). From (4) and Remark 23 we obtain some \( \hat{v}' \in \text{ConsState} \) such that \( (\hat{v}, \hat{v}') \in \rightarrow_r \), and \( \hat{v}' \sim_{\text{LocalDataVar}(p) \cup \text{LocalDataVar}(p')} \hat{v} \). Then \( \hat{v}' \in \text{post}(Q) \subseteq Q \). Also \( \hat{v}'(p_c) = v(p_c) \). Therefore, \( \hat{v}' \sim_{\text{LocalDataVar}(p) \cup \text{LocalDataVar}(p')} \).

Now take any \( r \in R \setminus R' \) (by (5) there is one or none). Note that \( r \in \text{rsc}(\hat{v}(p_c)) \).

Since \( \hat{v} \in \text{ConsState} \), we obtain that \( r \notin \bigcup_{t \in \text{Tid}(r)} \text{rsc}(\hat{v}(p_c)) \) = [using (1)] \( \bigcup_{t \in \text{Tid}(r)} \text{rsc}(\hat{v}(p_c)) \).

Moreover, \( r \notin \text{rsc}(\hat{v}(p_c)) \). Thus, \( r \) is available in \( \hat{v}' \). We have \( v' \sim_r [\text{by (3)}] \).

Case \( R' = R \cup \{ r \} \) for some \( r \in \text{Res} \). From assumption \( v|_{\text{ProofVar}(r)} \in I_r \) we obtain some \( \tilde{v} \in Q \) such that \( v \sim_r \tilde{v} \) and \( r \) is available in \( \tilde{v} \). By Lemma 17 there is \( w' \in Q \) such that \( w' \sim_{\text{LocalDataVar}(p) \cup \text{LocalDataVar}(p')} \). From \( w' \in Q, w'(p_c) = p' \) and \( w' \sim_{\text{LocalDataVar}(p) \cup \text{LocalDataVar}(p')} \).

We obtain \( \tilde{v} \sim_{\text{LocalDataVar}(p) \cup \text{LocalDataVar}(p')} \).

Initial condition. Let \( v \in \text{init} \). Then \( v \in \text{ConsState} \), all resources are available in \( v \) and \( v \in Q \). Thus for any \( r \in \text{Res} \) we have \( v|_{\text{ProofVar}(r)} \in I_r \). Now let \( t \in \text{Tid} \). By definition of \( M_{v(p_c), r} \) we obtain \( v|_{\text{LocalDataVar}(p) \cup \text{LocalDataVar}(p')} \in M_{v(p_c), r} \). So \( v \in \gamma_{\text{OG}}(P) \).

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

Let \( s \in \text{RL}, r \in \text{Res}, s \subseteq r \). We’ll show two statements.

\[
\text{"} r \notin \text{rsc}(v(\text{PCVar})) \Rightarrow \exists \hat{v} \in Q : v \sim_r \hat{v} \land r \notin \text{rsc}(v(\text{PCVar})) \text{"}.
\]

Assume for a moment that \( r \notin \text{rsc}(v(\text{PCVar})) \). The definition of \( \gamma_{\text{OG}} \) implies that \( v|_{\text{ProofVar}(r)} \in I_r \). So there is \( \hat{v} \in Q \) such that \( v \sim_r \hat{v} \) and \( r \) is available in \( \hat{v} \). Then \( v \sim_r \hat{v} \).

39
“∀t ∈ Tid : r ∈ rsc(v(pc_i)) ⇒ ∃v ∈ Q : v ∼ Local(\| Res) ∪ \overline{F}”:

Let t ∈ Tid and r ∈ rsc(v(pc_i)). Then v\|LocalDatalit∪\|ProofVar(rsc(v(pc_i))) \in \text{[definition of } γOG]\ M_t(v(pc_i)) t. Then there is \overline{v} ∈ Q such that \overline{v} ∼ LocalDatalit∪\|ProofVar(rsc(v(pc_i))) v and \overline{v}(pc_i) = v(pc_i). Thus \overline{v} ∼ Local(\| Res v and \overline{v} ∼_v v. Take \overline{v} := \overline{v}.

Now we’ll show that

“∀t ∈ Tid ∃v ∈ Q : v ∼ Local(\| Res) v”:

Let t ∈ Tid. The definition of γOG(P) implies v|LocalDatalit∪\|ProofVar(rsc(v(pc_i))) ∈ M_t(v(pc_i)) t. So there is \overline{v} ∈ Q such that \overline{v}(pc_i) = v(pc_i) and \overline{v} ∼ LocalDatalit∪\|ProofVar(rsc(v(pc_i))) v. Then \overline{v} ∼ Local(\| Res v.

We’ve shown that v ∈ \overline{ρ}(Q) ⊆ [since Q is in the image of \overline{ρ} by definition, it is a fixpoint of \overline{ρ}].

Thus any property provable by abstract interpretation with \overline{ρ} has an Owicki-Gries-core proof.

**Corollary 45.** Let lfp(λx. \overline{ρ}_c(init ∪ post(x))) ⊆ safe ∈ D. Then safe has an Owicki-Gries-core proof.

**Proof.** Let S = lfp(λx. \overline{ρ}_c(init ∪ post(x))), S ⊆ safe ∈ D. From Prop. 38 we get \overline{ρ}(init ∪ post(S)) ⊆ \overline{ρ}_c(init ∪ post(S)) = S, so S is a postfix point of λx. \overline{ρ}(init ∪ post(x)) which is by Tarski’s fixpoint theorem greater than or equal to its least fixpoint. Thus lfp(λx. \overline{ρ}(init ∪ post(x))) ⊆ S ⊆ safe. Apply Thm. 18.

**Proposition 46.** Fix a syntactic structure (Tid, PL, Val, Res, (Local_i, pc_i)) ∈ Tid, ProofVar, rsc) from StrSimple and let ConsState be the set of consistent states of this syntactic structure and D its powerset. Let \overline{ρ} and \overline{ρ}_c be defined for StrSimple as in Sections D and E, respectively. Then the following statements hold.

1. \overline{ρ} = \overline{ρ}_c.
2. For all RPL transitions systems (init, (¬r) ∈ Tid) from Simple that have successor map post and the set of Owicki-Gries-core proofs \overline{Ω} we have

γOG(init(\overline{Ω})) = lfp(λx. \overline{ρ}(init ∪ post(x))) = lfp(λx. \overline{ρ}_c(init ∪ post(x))).

**Proof.**

1. Let Q ∈ D. Since |Res| ≤ 1, we distinguish two cases.

Case Res = ∅. Then \overline{ρ}(Q) = {v ∈ ConsState | ∀t ∈ Tid ∃v ∈ Q : v ∼ Local, v = \overline{ρ}(Q).

Case Res = {r}. Then r ≠ ∅. Then \overline{ρ}(Q) =

\begin{align*}
\{v ∈ ConsState \mid & \left( r \notin rsc(v(ProofVar)) \Rightarrow \exists v ∈ Q : \left( r \notin rsc(v(ProofVar)) \land v ∼_r v \right) \right) \\
& \forall t ∈ Tid ∃v ∈ Q : v ∼ Local(\| Res(v(pc_i))) v \}
\end{align*}

Since r ≠ ∅ and is disjoint from all Local_i (t ∈ Tid), we have r ∈ RL, we have r ∈ RL

Thus Q \overline{ρ}(Q) =
2. Let \((\text{init}, (-\gamma)_{\gamma \in \mathcal{T}_\text{init}})\) be an RPL transition system from Simple that obeys the fixed syntactic structure and has a set of Owicki-Gries-core proofs \(\Theta\) and successor map \(\text{post}\). Then \(\text{lfp}(\lambda x. \hat{\rho}(\text{init} \cup \text{post}(x))) \subseteq [\text{Thm. 10}] \cap \{\text{init}\} \subseteq [\text{Thm. 18}] \cap \{\text{post}\}\). 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 \(\text{lfp}(\lambda x. \hat{\rho}(\text{init} \cup \text{post}(x)))\).

**Extension of Example 19.** Consider the RPL transition system Readers-Writers from Example 1.
Notice that \(|\text{Res}| = 1\). Moreover, \(\forall t \in \mathcal{T}_\text{init}: \text{Local}_t \cap \text{control} = \{pc_1\} \cap \{\text{ww, ar, aw}\} = \emptyset\) and \(\text{ProofNar} (\text{control}) = \text{control}\). 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 \(\text{lfp}(\lambda x. \hat{\rho}(\text{init} \cup \text{post}(x)))\).

**Extension of Example 20.** For the RPL transition system Upper from Example 2 we have \(\mathcal{RL} = \{r, r', \text{Local}_1, \text{Local}_2\} = \{\{u, z\}, \{x, y\}, \{x, l, pc_1\}, \{y, z, pc_2\}\}, \mathcal{RL} = \{\{u\}, \{z\}, \{x\}, \{y\}, \{l, pc_1\}, \{pc_2\}\}. Let
\[
C = \{v \in \text{ConsState} \mid v(u), v(z), v(x), v(y), v(l) \in \{0, 1\} \land v(pc_2) = 0\}.
\]
The upper iteration sequence \((X^{(i)})_{i \in \omega}\) for \(\lambda x. \hat{\rho}(\text{init} \cup \text{post}(x))\) starting from \(\emptyset \in \mathcal{D}\) is below.

\[
X^{(0)} = \emptyset,
\]
\[
X^{(1)} = \hat{\rho}(\text{init}) = \{v \in C \mid v(pc_1) = A\},
\]
\[
\text{post}(X^{(1)}) = \{v \in C \mid v(pc_1) = B \land v(l) = v(u)\},
\]
\[
X^{(2)} = \hat{\rho}(\text{init} \cup \text{post}(X^{(1)})) = \{v \in C \mid v(pc_1) = A \land \bigvee (v(pc_1) = B \land v(u) = v(l))\},
\]
\[
\text{post}(X^{(2)}) = \{v \in C \mid v(pc_1) \in \{B, C\} \land v(u) = v(l)\},
\]
\[
X^{(3)} = \hat{\rho}(\text{init} \cup \text{post}(X^{(2)})) = \{v \in C \mid v(pc_1) = A \land \bigvee (v(pc_1) = B \land v(u) = v(l))\},
\]
\[
\text{post}(X^{(3)}) = \{v \in C \mid (v(pc_1) \in \{B, C\} \land v(u) = v(l)) \lor (v(pc_1) = D \land v(u) = v(l) \land v(x) = 0)\},
\]
\[
X^{(4)} = \hat{\rho}(\text{init} \cup \text{post}(X^{(3)})) = \{v \in C \mid v(pc_1) = A \land \bigvee (v(pc_1) = B \land v(u) = v(l))\},
\]
\[
\text{post}(X^{(4)}) = \{v \in C \mid (v(pc_1) \in \{B, C\} \land v(u) = v(l)) \lor (v(pc_1) = D \land v(u) = v(l) \land v(x) = 0)\},
\]
\[
X^{(5)} = \hat{\rho}(\text{init} \cup \text{post}(X^{(4)})) = \{v \in C \mid v(pc_1) = A \land \bigvee (v(pc_1) = B \land v(u) = v(l))\},
\]
\[
\text{post}(X^{(5)}) = \{v \in C \mid (v(pc_1) \in \{B, C\} \land v(u) = v(l)) \lor (v(pc_1) = D \land v(u) = v(l) \land v(x) = 0)\} \subseteq X^{(5)}, \text{ so }
\]
\[
X^{(6)} = \hat{\rho}(\text{init} \cup X^{(5)}) = \hat{\rho}(X^{(5)}) = X^{(5)}.
\]
Thus
\[
\lfp(\lambda x. \hat{\rho}(\text{init} \cup \text{post}(x))) = \left\{ v \in C \left| \begin{array}{l}
v(p_{C_1}) = A \\
v(p_{C_1}) \in \{B, C, E\} \land v(u) = v(l) \\
v(p_{C_1}) = D \land v(u) = v(l) \land v(x) = 0
\end{array} \right. \right\}
\]
is strictly weaker than the strongest Owicki-Gries-core-provable property, e.g., containing the state \([u \rightarrow 0, z \rightarrow 0, x \rightarrow 0, y \rightarrow 0, l \rightarrow 0, p_{C_1} \rightarrow k, p_{C_2} \rightarrow 0] \notin \mathcal{F}_{3G}(\inf \Omega)\).
\[\square\]

**Extension of Example 22.** Fix an RPL transition system from SepThreads. For all \(S \subseteq \text{ConsState}\) we have
\[
\hat{\rho}(S) = \{ v \in \text{ConsState} \mid \forall t \in \text{Tid} \exists \tilde{v} \in S : v \sim_{\text{Local}_t} \tilde{v} \} = \hat{\rho}(S).
\]

Within this example, let \(F := \lambda x. \text{init} \cup \text{post}(x)\) and \(Q := \lfp(F)\). Notice that \(\text{post}\) is a join-morphism, so is \(F\). By Kleene’s fixpoint theorem \(Q = \bigcup_{n \in \mathbb{N}} F^n(\emptyset)\). Let \(v \in \hat{\rho}(Q)\). Then there is a map \(m : \text{Tid} \to \mathbb{N}_0\) such that \(\forall t \in \text{Tid} \exists \tilde{v} \in F^{m(t)}(\emptyset) : v \sim_{\text{Local}_t} \tilde{v}\). Since \(\text{Tid}\) is finite and the sequence \(\{F^n(\emptyset)\}_{n \in \mathbb{N}_0}\) is monotonically increasing, \(\forall t \in \text{Tid} \exists \tilde{v} \in F^{\max(m(\text{Tid}))(\emptyset)} : v \sim_{\text{Local}_t} \tilde{v}\). Thus \(v \in \hat{\rho}(F^{\max(m(\text{Tid}))(\emptyset)})\).

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

Case \(i = 0\).

Case \(\text{Tid} = \emptyset\). Then \(\text{ConsState} = \{ v \in \text{Val} \mid \forall t \in \text{Tid} : \ldots \} = (\forall v \rightarrow \text{Val}) = \{ \emptyset \} = \hat{\rho}(\emptyset)\). By definition of SepThreads, \(\forall v \in \text{ConsState} : ((\forall t \in \text{Tid} : \ldots) \Rightarrow v \in \text{init})\). Thus \(\text{init} = \{ \emptyset \}\). Then \(\hat{\rho}(F^0(\emptyset)) = \text{init} \subseteq Q\).

Case \(\text{Tid} \neq \emptyset\). Then \(\hat{\rho}(F^0(\emptyset)) = \emptyset \subseteq Q\).

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

Case \(i > 1\). Let \(v \in \hat{\rho}(F^i(\emptyset))\), i.e., \(v \in \hat{\rho}(\text{init} \cup \text{post}(F^{i-1}(\emptyset)))\). Let \(I = \{ t \in \text{Tid} \mid \exists \tilde{v} \in \text{init} : v \sim_{\text{Local}_t} \tilde{v}\}\). Then for all \(t \in \text{Tid} \setminus I\) there is \(\tilde{v} \in \text{post}(F^{i-1}(\emptyset))\) such that \(v \sim_{\text{Local}_t} \tilde{v}\). Let \(J = \{ t \in \text{Tid} \mid \exists \tilde{v} \in F^{i-1}(\emptyset) : v \sim_{\text{Local}_t} \tilde{v}\}\) and \(K = \text{Tid} \setminus (I \cup J)\). For each \(t \in K\) there is some \(\tilde{v} \in F^{i-1}(\emptyset)\) such that there is some \(\tilde{v} \in \text{post}(F^{i-1}(\emptyset))\) such that \(v \sim_{\text{Local}_t} \tilde{v}\) from \(t \notin J\) we obtain \(\tilde{v} \sim_{\text{Local}_t} \tilde{v}\), so \(\tilde{v} \sim_{\text{Local}_t} \tilde{v}\) and (1) implies \(\tilde{v} \sim_{\text{Local}_t} \tilde{v}\). Thus there are maps \(f, f' : \text{ConsState} \to \text{ConsState}\) such that \(\forall t \in K : f(t) \in F^{i-1}(\emptyset) \land f(t) \sim_{\text{Local}_t} f'(t) \sim_{\text{Local}_t} v\). Let \(C = \bigcup_{t \in I \cup J} \text{Local}_t \cap \text{Local}_t\). There is a map \(h : (\forall t \in C) \to \text{Tid}\) such that \(\forall x \in \forall t \in C : x \in \forall t \in [C]\). Such a map \(h\) is unique. Let \(n = |K|\), and fix some enumeration \(\tau_1, \ldots, \tau_n\) of \(K\). For each \(k \in \mathbb{N}_0 \cap \{0, n\}\) let
\[
w^{(k)} : \forall t \to \text{Val}, x \mapsto \begin{cases}v(x), & \text{if } x \in C \cup \bigcup_{t \in I \cup J} \text{Local}_t \cup \bigcup_{j=1}^k \text{Local}_{\tau_j}, \\f(h(x))(x), & \text{if } x \in \bigcup_{j=k+1}^n \text{Local}_{\tau_j} \setminus C.
\end{cases}
\]

To show that each \(w^{(k)}\) is well-defined \((0 \leq k \leq n)\), notice that
- \(\forall \subseteq C \cup \bigcup_{t \in I \cup J} \text{Local}_t \cup \bigcup_{j=1}^k \text{Local}_{\tau_j} \cup \bigcup_{j=k+1}^n \text{Local}_{\tau_j} \setminus C\);
- \((I \cup J) \cap K = \emptyset\) 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 \(\text{Local}_a \cap (\text{Local}_b \setminus C) = \emptyset\).
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 \( \bar{\rho}(F^{-1}(\emptyset)) \). Let \( t \in \mathbb{T}d \).

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

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

Case \( t \in K \). Let \( x \in \mathbb{Local}_t \). There is some \( j \in \mathbb{N}^* \cap [1,n] \) such that \( t = \tau_j \); then \( x \in \mathbb{Local}_{\tau_j} \).

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

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

Thus \( w^{(0)}(x) = f(t)(x) \) for all \( x \in \mathbb{Local}_t \). Thus \( w^{(0)} \sim_{\mathbb{Local}_t} f(t) \).

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

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

1. \( ^w(f(\tau_k)) \sim_{\mathbb{Local}_q} w^{(k-1)} \): Let \( x \in \mathbb{Local}_{\tau_k} \).

   Case \( x \in C \). Then \( f(\tau_k)(x) = [\text{part 1 of Lemma 39}] \).

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

2. \( ^w(w^{(k-1)} \rightarrow_{\tau_q} w^{(k)}) \): From (4) and \( f(\tau_k) \rightarrow_{\tau_q} f'(\tau_k) \) we obtain some \( w' \) such that \( ^w(w^{(k-1)} \rightarrow_{\tau_q} w') \).

We'll show now that \( w' \) is equal to \( w^{(k)} \). So let \( x \in \mathbb{Var} \) be arbitrary.

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

Case \( x \in \mathbb{Local}_{\tau_k} \). Then \( w'(x) = f'(\tau_k)(x) = v(x) = w^{(k)}(x) \).

Case \( x \in \mathbb{Local}_{\tau_k} \) for some \( j \neq k \). Then \( w'(x) = [\text{by (1)}] \).

Since \( x \) was arbitrary, \( w' = w^{(k)} \). Thus, \( ^w(w^{(k-1)} \rightarrow_{\tau_q} w^{(k)}) \).

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

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

We have proven by induction that \( \forall \in \mathbb{N}_0 : \bar{\rho}(F^{i}(\emptyset)) \subseteq Q \). 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 \( Q = \text{lfp}(F) \subseteq \text{lfp}(\bar{\rho} \circ F) \subseteq Q \).

\[
\text{lfp}(\lambda x. \bar{\rho}(\text{init} \cup \text{post}(x))) = \text{lfp}(\lambda x. \text{init} \cup \text{post}(x)).
\]
By Theorems 10 and 18 we also get

\[
\operatorname{lfp}(\lambda x. p(\text{init} \cup \text{post}(x))) = \gamma_{\mathcal{C}}(\inf \mathcal{D}) = \operatorname{lfp}(\lambda x. p(\text{init} \cup \text{post}(x))).
\]

\[\square\]

### F Absence of equivalent semantic characterization in abstract interpretation

Given a complete lattice \( D \), a \textit{semantic transformation} over \( D \) is a map \( F^\# : (D \to D) \to D \to D \) such that

- for all join morphisms \( \tau \in D \to D \), the map \( F^\# : D \to D \) is monotone and
- for all join morphisms \( \tau_1, \tau_2 \in D \to D \) we have

\[
\operatorname{lfp}(\tau_1) = \operatorname{lfp}(\tau_2) \Rightarrow \operatorname{lfp}(F^\#(\tau_1)) = \operatorname{lfp}(F^\#(\tau_2)).
\]

**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

\[
F^\# : (D \to D) \to D \to D, \quad \tau \mapsto \lambda x. f \left( \left( \begin{array}{c} \operatorname{lfp}(\tau), \quad \text{if } \tau \text{ is monotone,} \\ \sup D, \quad \text{otherwise} \end{array} \right), x \right)
\]

is a semantic transformation.

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

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

\[\square\]

**Example 48.** Given a complete lattice \( D \), the following maps are semantic transformations (where infimum over the empty set is the top of \( D \)):

- \( \lambda \tau \in D \to D. \lambda x \in D. \sup D \);
- \( \lambda \tau \in D \to D. \lambda x \in D. \inf \{ y \in D \mid \tau(y) = y \} \);
- \( \lambda \tau \in D \to D. \lambda x \in D. \sup \{ \inf \{ y \in D \mid \tau(y) = y \} \} \).

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

\[
\text{Trace} = \text{ConsState}^*, \quad \tilde{D} = \mathcal{P}(\text{Trace}) \quad \text{and} \quad \bar{\alpha} : \tilde{D} \to D, \quad t \mapsto \bigcup_{\delta \in T} \pi_2(\delta),
\]

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 \( \pi_2(\delta) = \{ y \mid \exists x : (x, y) \in \delta \} \) is the set of states occurring in the trace \( \delta \).

Given a program \((\text{init}, (\to_t)_{t \in \text{Tid}})\) that obeys the above syntactic structure, its set of \textit{initial traces} is \( \text{init} = \{(1, v) \mid v \in \text{init}\} \) and its \textit{trace extension operator} is

\[
\overline{\text{post}} : \tilde{D} \to \tilde{D}, \quad t \mapsto \{ \delta w' \mid \delta v \in T \wedge \exists t \in \text{Tid} : (v, v') \in \to_t \}.
\]
Restatement of Theorem 6. There is a syntactic structure 
(Tid, PL, Val, Res, (Local, pc)∈Tid, ProofVar, rsc) 
such that for ConsState, D, D, α defined as above, all the following statements hold: 
1. There is no semantic transformation $F^#$ over $D$ such that for any RPL transition 
 system (init,...) that obeys the syntactic structure, has successor map $post$ and the 
least Owicki-Gries-core proof $P$, we have 
\[ \gamma_{OG}(P) = \text{lfp}(F^#_{\lambda x \cdot \text{init} \cup \text{post}(x)}). \]

2. There is no monotone operator $\rho : D \to D$ such that for any RPL transition system 
 (init,...) that obeys the syntactic structure, has successor map $post$ and the 
least Owicki-Gries-core proof $P$, we have 
\[ \gamma_{OG}(P) = \text{lfp}(\lambda x \cdot \rho(\text{init} \cup \text{post}(x))). \]

3. There is no monotone operator $\tilde{\rho} : \tilde{D} \to \tilde{D}$ such that for any RPL transition system 
that obeys the syntactic structure, has a set of initial traces $\tilde{\text{init}}$, has a trace extension 
operator $\text{post}$ and has the least Owicki-Gries-core proof $P$, we have 
\[ \gamma_{OG}(P) = \tilde{\alpha}(\text{lfp}(\lambda x \cdot \tilde{\rho}(\tilde{\text{init}} \cup \tilde{\text{post}}(x)))). \]

Proof. Consider the syntactic structure $S$ given by 
- $\text{Tid} = \{1, 2\}$; 
- $\text{PL} = \{A, B, C, \ldots, V, W\}$; 
- $\text{Val} = \text{PL} \cup \{0, 1\}$; 
- $\text{Res} = \{r, r'\}$ where $r = \{u, z\}$, $r' = \{x, y\}$; 
- $\text{Local}_1 = \{l, x, pc_1\}$, $\text{Local}_2 = \{y, z, pc_2\}$; 
- $\text{ProofVar}(r) = r$ and $\text{ProofVar}(r') = r'$.
- $\text{rsc} : \text{PL} \to \mathcal{P}(\text{Res})$ given by 
\[
\begin{array}{ccc}
rsc(p) & \emptyset & \{r\} & \{r'\} & \{r, r'\}
\end{array}
\]

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

\[ \text{Local}_1 \ldots \begin{array}{c} l \end{array} \begin{array}{c} pc_1 \end{array} \]
\[ \text{Local}_2 \ldots \begin{array}{c} y \end{array} \begin{array}{c} pc_2 \end{array} \]
\[ \begin{array}{c} r' \end{array} \begin{array}{c} u \end{array} \]

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 $A$. 

45
We are going to show an Owicki-Gries-core proof of property \( \text{init} \) for UpperA. Let
\( I_r = \{ [u \mapsto 0, z \mapsto 1], [u \mapsto 1, z \mapsto 0] \} \), \( I'_r = \{ [x \mapsto 0, y \mapsto 0], [x \mapsto 1, y \mapsto 1] \} \),
\( M_{A,1} = \{ [x \mapsto 0, l \mapsto 0], [x \mapsto 1, l \mapsto 1] \} \),
\( M_{0,2} = \{ [y \mapsto 0, z \mapsto 0], [y \mapsto 1, z \mapsto 1] \} \) and
\( M_{x,t} = \emptyset \) for \((x,t) \in (PL \times Tid) \setminus \{(A,1),(0,2)\}\).

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

By definition, \( P_{UpperA} \models (\Psi(\text{ProofVar}(r) \rightarrow \text{Val}) \times \Psi(\text{ProofVar}(r') \rightarrow \text{Val})) \times
(\Psi(\text{LocalDataVar}_1 \rightarrow \text{Val}) \times \Psi(\text{LocalDataVar}_2 \rightarrow \text{Val}) \times
\Psi((\text{LocalDataVar}_1 \cup \text{ProofVar}(r)) \rightarrow \text{Val}) \times
\ldots) \),
where the components are indexed by \( r, r' \) in the first line and \((A,1), (0,2), (8,1)\) and so on in the remainder. Thus \( P_{UpperA} \models PA \). We will show:

- Sequential consistency. Let \( t \in Tid \) with \((v,v') \in r \rightarrow r' \). By \( R = \text{rsc}(v(pc)) \), \( R' = \text{rsc}(v'(pc)) \), as well as \( v|_\text{LocalDataVar}_1, v|\text{ProofVar}(R) \in M_{pc} \) and \( \forall \tilde{r} \in R'\setminus R : v|\text{ProofVar}(\tilde{r}) \in I_r \).
  - Case \( v(pc) \in \{A,\emptyset\} \). Then thread \( r \) can’t proceed from \( v \), a contradiction.
  - Case \( v(pc) \notin \{A,\emptyset\} \). Then \( v|\text{LocalDataVar}_1, v|\text{ProofVar}(R) \in \emptyset \), a contradiction.

- Initial condition. We have
  \( \gamma_{OG}(P_{UpperA}) = \{ [u \mapsto 1, z \mapsto 0, x \mapsto 0, y \mapsto 0, l \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto 0],
  [u \mapsto 0, z \mapsto 1, x \mapsto 1, y \mapsto 1, l \mapsto 1, pc_1 \mapsto A, pc_2 \mapsto 0] \} \).

Notice that \( \text{init} \subseteq \gamma_{OG}(P_{UpperA}) \).

Since \( \gamma_{OG}(P_{UpperA}) \subseteq \text{init} \), \( P_{UpperA} \) is an Owicki-Gries-core proof of property \( \text{init} \) 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 \textit{init}.

Now we’ll show the claims of the theorem.

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

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

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

   Notice that $\lambda x. \text{init} \cup \text{post}U(x)$ and $\lambda x. \text{init} \cup \text{post}UA(x)$ are join-morphisms on the same lattice and their least fixpoints coincide. By the assumption, for UpperA with successor map $\text{post}UA$, the strongest Owicki-Gries-core-provable property is $\text{lfp}(\lambda x. \text{init} \cup \text{post}UA(x))$. Thus, $\text{init} = \text{lfp}(\lambda x. \text{init} \cup \text{post}UA(x)) = \text{[property of } F^\#$] $\text{lfp}(\lambda x. \text{init} \cup \text{post}UA(x))$.

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

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

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

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

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

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

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

### 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, \((D^\#, \subseteq)\) a complete lattice, \((\alpha, \gamma)\) a pair of adjoint maps, \(\text{init} \in D, \text{post} : D \rightarrow D\) a complete join morphism and \(\text{post}^\# : D^\# \rightarrow D^\#\) a monotonic operator such that \(\forall Y \in D^\# : \alpha \text{post}_Y \subseteq \text{post}^\# Y\).

Intuitively, \(D\) is the powerset of program states, \(\text{init}\) is the set of initial states, \(\text{post}\) maps a set of states to the set of their direct successors, \(D^\#\) is an abstract lattice used for analysis and \(\text{post}^\#\) 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

\[ T^{(0)} = \alpha \text{init}, \]

\[ T^{(i+1)} = \begin{cases} 
\text{post}^\# T^{(i)}, & \text{if } \text{post}^\# T^{(i)} \not\subseteq \bigcup_{j=0}^{i} T^{(j)}, \\
T^{(i)} \cup \text{post}^\# T^{(i)}, & \text{if } \text{post}^\# T^{(i)} \subseteq \bigcup_{j=0}^{i} T^{(j)}. 
\end{cases} \]

Let \(k = \min\{i \mid T^{(i+1)} \subseteq T^{(i)}\} \in \mathbb{N}_0 \cup \{\infty\}\) (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 \(\subseteq\).

**Proposition 50.** If the height \(h\) of \((D^\#, \subseteq)\) is finite, then \(k \leq h(h - 1)\).
Proof. Let $h$ be finite. The elements of the monotone sequence $(\bigcup_{i=0}^{k'} T^{(i)})_{k' \geq 0}$ build a chain, so this sequence stabilizes for some $k' \geq 0$, i.e., $T^{(j)} \subseteq \bigcup_{i=0}^{k'} T^{(i)}$ for all $j \geq k'$. Take the smallest such $k'$. For all $j \geq k'$ we have $T^{(j+1)} = post^h T^{(j)}$ and thus $post^h T^{(j)} \subseteq \bigcup_{i=0}^{k'} T^{(i)}$. By definition of the algorithm, the sequence $(T^{(j)})_{j \geq k'}$ is monotonically increasing. The domain has finite height, so there is a $k \geq k'$ with $T^{(k+1)} = T^{(k)}$. Take the smallest such $k$.

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

Case $post^h T^{(0)} \subseteq T^{(0)}$. Then $post^h T^{(0)} \subseteq \bigcup_{i=0}^{k} T^{(i)}$, so $T^{(1)} = T^{(0)}$. Then $k = 0 \leq h(h - 1)$.

Case $post^h T^{(0)} \nsubseteq T^{(0)}$. Then $post^h T^{(0)} \nsubseteq \bigcup_{i=0}^{k} T^{(i)}$, so $T^{(1)} = post^h T^{(0)} \nsubseteq T^{(0)}$. Therefore $\bigcup_{i=0}^{k} T^{(i)} \neq \bigcup_{i=0}^{k-1} T^{(i)}$ (otherwise $T^{(1)}$ were less than or equal to $T^{(0)}$).

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

$$\bigcup_{i=0}^{a-1} T^{(i)} \neq \bigcup_{i=0}^{a} T^{(i)}$$

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

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

The sequence $(T^{(c)})_{a \leq c \leq b}$ 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 $(T^{(i)})_{i \geq c}$ were constant for some $c < b$, contradicting the choice of $b$). The maximal chain length gives $b - a \leq h$ 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 $k' - 1 \leq b(h - 2)$. Thus $k' \leq h^2 - 2h + 1$ and $k \leq k' + h - 1 \leq h^2 - h$.

\[\square\]

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

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

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

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

$$\subseteq \bigcup_{i \in \mathbb{N}_0} \gamma post^h T^{(i)} \subseteq \bigcup_{i \in \mathbb{N}_0} \gamma T^{(i+1)} \subseteq \bigcup_{i \in \mathbb{N}_0} \gamma T^{(i)}.$$

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

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

Proof. By definition of $k$ we have $T^{(k+1)} \subseteq T^{(k)}$. If for some $j > k$ we have $T^{(j)} \subseteq T^{(k)}$, then $T^{(j+1)} \subseteq T^{(j)} \cup \text{post}^{#} T^{(j)} \subseteq \bigcup_{i=0}^{k} \gamma T^{(i)} \cup \text{post}^{#} T^{(j)} \subseteq T^{(k+1)} \subseteq T^{(k)}$ [by definition of $k$]. By induction $T^{(j)} \subseteq T^{(k)}$ for all $j \geq k$. Thus $\gamma T^{(j)} \subseteq \gamma T^{(k)}$ for all $j \geq k$, implying $\bigcup_{i=0}^{k} \gamma T^{(i)} = \bigcup_{i=0}^{k} \gamma T^{(i)}$.

As a corollary, if $(D^{#}, \subseteq)$ 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 $(\text{Tid, PL, Val, Res, (Local, pc)}_{\text{r} \in \text{Tid}}, \text{ProofVar, rsc})$ be an arbitrary but fixed syntactic structure, $D$ the powerset of consistent states, $D^{#}$ the set of program annotations equipped with componentwise order, $(\text{init, } (\rightarrow)_{\text{r} \in \text{Tid}})$ an RPL program that obeys the syntactic structure, $\text{post}$ its successor map, $\mathcal{D}$ the lattice of Owicki-Gries-core proofs, and

$$p_{\text{OG}}^{#} : \text{PA} \rightarrow \text{PA}, \quad (I,M) \mapsto \left\{ \begin{array}{l}
\left. u \in \text{ProofVar}(r) \rightarrow \text{Val} \right| \\
\quad \begin{array}{l}
\exists t \in \text{Tid}, (v,v') \in \rightarrow; \\
\quad r \not\in \text{rsc}(v'(pc)) \wedge \\
\quad \forall \text{LocalDVar}_{r} \cup \text{ProofVar}(\text{rsc}(v'(pc))) \in \text{M}_{\text{pc}}(t) \\
\quad (r \not\in \text{rsc}(v(pc)) \wedge u \in I) \vee \\
\quad (r \in \text{rsc}(v(pc)) \wedge u \not\in \text{M}_{\text{pc}}(t))
\end{array}
\end{array} \bigg\} \quad \bigg( r \in \text{Res} \bigg) \bigg\},
\end{array} \right\}
$$

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

$$\begin{align*}
\text{init} & = \{ \mathcal{v} | \text{init} \} \cup \{ \mathcal{v} | \text{in} \} \cup \{ \mathcal{v} | \text{pc} = \text{init} \}\big\} \quad \bigg( r \in \text{Res} \bigg), \\
\text{M}_{\text{init}} & = \{ \mathcal{v} | \text{LocalDVar}_{r} \cup \text{ProofVar}(\text{rsc}(pc)) \in \text{M}_{\text{pc}}(t) \} \quad \quad \bigg( r \in \text{Res} \bigg) \bigg),
\end{align*}$$

\(^8\) Using $\text{post}^{#}_{\text{OG}}$ would not give us more precision: the generated inductive invariant would be equal to $\gamma_{\text{OG}}(\ell \hspace{1pt} \text{it}(\theta))$. We cannot use $\text{next}^{#}_{\text{OG}}$, as it does not overapproximate $\text{OG} \circ \text{post} \circ \gamma_{\text{OG}}$ in general. We are going to suggest an operator in between: it will overapproximate $\text{OG} \circ \text{post} \circ \gamma_{\text{OG}}$ and underapproximate $\text{post}^{#}_{\text{OG}}$, allowing more precision in the relaxed frontier search.
and the improved Owicki-Gries-core abstract transformer be
\[ G^8_{OG} : PA \rightarrow PA, \]
\[ (I, M) \mapsto (I^{init}, M^{init}) \cup p^8_{OG}(I, M). \]

**Proposition 53.** \( p^8_{OG} \) is monotone.

**Proof.** Let \((I, M) \subseteq (\hat{I}, \hat{M})\) be program annotations, \( p^8_{OG}(I, M) = (I', M'), p^8_{OG}(\hat{I}, \hat{M}) = (\hat{I}', \hat{M}'). \)

\("\forall r \in Res : I' \subseteq \hat{I}'": \) 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 \( v|_{\text{LocalData}} \cup \text{\text{Var}}(\text{PCVar}) \subseteq M(v_{\text{pc}})_t \), which is a subset of \( \hat{M}(v_{\text{pc}})_t \), \( r \not\in \text{rsc}(v'(p_{\text{pc}})) \) and either \( r \not\in \text{rsc}(v(p_{\text{pc}})) \land w = v_{\text{post}} \). Then \( v|_{\text{LocalData}} \cup \text{\text{Var}}(\text{PCVar}) \subseteq M(v_{\text{pc}})_t \); also either \( r \not\in \text{rsc}(v(p_{\text{pc}})) \land w = v_{\text{post}} \). 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 \( t \in Tid \setminus \{t\} \) and \( (v, v') \in \rightarrow_t \) such that \( v|_{\text{LocalData}} \cup \text{\text{Var}}(\text{PCVar}) \subseteq M(v_{\text{pc}})_t \) and \( r \not\in \text{rsc}(v_{\text{pc}}) \cup \text{rsc}(v(p_{\text{pc}})) \cup \text{rsc}(v'(p_{\text{pc}})) \). Notice that \( M_{p,t} \subseteq M_{p,t} \) and \( M_{p,t} \subseteq M_{p,t} \). Thus \( v|_{\text{LocalData}} \cup \text{\text{Var}}(\text{PCVar}) \subseteq M(v_{\text{pc}})_t \) and \( w \in M_{p,t} \). So \( w \in \hat{M}_{p,t} \).

- Or there is a transition \( v \rightarrow_{t} v' \) of the same thread \( t \) such that \( v'(p_{\text{pc}}) = p \), \( w = v' \) and \( v|_{\text{LocalData}} \cup \text{\text{Var}}(\text{PCVar}) \subseteq M(v_{\text{pc}})_t \) and \( \forall r \in \text{rsc}(p) \cup \text{rsc}(v(p_{\text{pc}})) \cup \text{rsc}(v'(p_{\text{pc}})) : v_{\text{post}}(r) \subseteq I_r \). Notice that \( M_{p,t} \subseteq M_{p,t} \) and for all \( r \in Res \) we have \( I_r \subseteq \hat{I}_r \). Then \( M_{p,t} \subseteq M_{p,t} \) and \( \forall r \in \text{rsc}(p) \cup \text{rsc}(v(p_{\text{pc}})) \cup \text{rsc}(v'(p_{\text{pc}})) : v_{\text{post}}(r) \subseteq \hat{I}_r \). Then \( w \in M_{p,t} \).

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:

\[ \Omega = \text{postfp}(G^8_{OG}). \]

**Proof.** "\( \subseteq \):" Let \((I, M) \in \Omega \) and \((\hat{I}, \hat{M}) = p^8_{OG}(I, M)\). We will show that \((I', M') := G^8_{OG}(I, M)\) 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 \( w \in I^{init}_r \), there is some \( v \in \text{init} \) such that \( v_{\text{post}} = w \). By the initial condition, \( v \in \gamma_{OG}(I_r) \). Since every resource is available in an initial state, \( r \in Res \cup \text{rsc}(v_{\text{PCVar}}) \). Thus \( v_{\text{post}} \subseteq I_r \).

Otherwise \( w \in \hat{I}_r \). So there is some \( t \in Tid \) and some transition \( v \rightarrow_{t} v' \) such that \( v|_{\text{LocalData}} \cup \text{\text{Var}}(\text{PCVar}) \subseteq M(v_{\text{pc}})_t \), \( r \not\in \text{rsc}(v'(p_{\text{pc}})) \) and either \( r \not\in \text{rsc}(v(p_{\text{pc}})) \land w \in I_r \) or \( r \in \text{rsc}(v(p_{\text{pc}})) \land w = v_{\text{post}} \). Assume for the purpose of contradiction that \( w \not\in I_r \). Then \( r \in \text{rsc}(v(p_{\text{pc}})) \cup \text{rsc}(v'(p_{\text{pc}})) \land w = v_{\text{post}} \). By (5) we have \( \text{rsc}(v(p_{\text{pc}})) \cup \text{rsc}(v'(p_{\text{pc}})) \subseteq \text{rsc}(v(p_{\text{pc}})) \). Sequential consistency implies \( v_{\text{post}} \subseteq I_r \). Contradiction!

We have shown that in both cases \( w \in I_r \). So \( I'_r \subseteq I_r \).
“∀ t ∈ Tid, p ∈ PL: M^p_t ⊆ M^p_t.” Let t ∈ Tid, p ∈ PL. Let w ∈ M^p_t.

If w ∈ M^p_t, there is some v ∈ init such that v(pc_t) = p and w = v|_{LocalData_p}.

By the initial condition, v ∈ ωOG(I, M). From definition of ωOG and rsc(v(pc_t)) = [v ∈ init] Φ we get v|_{LocalData_p} ∈ M^p_t.

Otherwise w ∈ M^p_t. Assume for the purpose of contradiction that w /∈ M^p_t. By definition of ωOG there is a transition v→v’ such that w = v|_{LocalData_p} ↓ ProofVar(rsc(p)) = v’(pc_t) = p and ∀ r ∈ rsc(p) \ rsc(v’(pc_t)).

v|_{ProofVar(r)} ∈ I. From sequential consistency we get v’|_{LocalData_p} ↓ ProofVar(rsc(p)) ∈ M^p_t. Contradiction!

In both cases w ∈ M^p_t. So M^p_t ⊆ M^p_t.

“⊇”: Let (I, M) ∈ postfp(G^3OG), i.e., for (I’, M’) = G^3OG(I, M) we have (I’, M’) ⊆ (I, M).

We’ll show that (I, M) is an Owicki-Gries-core proof.

Sequential consistency: Let t ∈ Tid, (v, v’) ∈ →, p = v(pc_t), p’ = v’(pc_t), R = rsc(p), R’ = rsc(p’), v|_{LocalData_p} ↓ ProofVar(rsc(p)) ∈ M^p_t and ∀ r ∈ R’ \ R : v|_{ProofVar(r)} ∈ I. By definition of ωOG we obtain v|_{LocalData_p} ↓ ProofVar(rsc(p)) ∈ M^p_t’ and M^p_t’ ⊆ M^p_t. If any r ∈ R’ \ R is given, the definition of ωOG implies v’|_{ProofVar(r)} = [by (3)] v|_{ProofVar(r)} ∈ I’ ⊆ I.

Initial condition: Let v ∈ init.

Let r ∈ Res \ rsc(v(PCVar)). Then v|_{ProofVar(r)} ∈ I^v_init ⊆ I’ ⊆ I.

Now let t ∈ Tid. Then v|_{LocalData_p} ∈ M^v_init(p). Since initial states don’t have any busy resources, v|_{LocalData_p} ↓ ProofVar(rsc(v’(pc_t))) ∈ M^v_init(p), ⊆ M^v_init(p) ⊆ I.

We have shown v ∈ ωOG(I, M). Thus init ⊆ ωOG(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^3OG) = inf Ω.

Proof. inf Ω = [Prop. 54] inf(postfp(G^3OG)) = [Tarski’s fixpoint theorem] lfp(G^3OG).

 Proposition 56. The map p^#OG overapproximates the best abstract successor map with respect to the pair of adjoint maps (ωOG, ωOG). Formally:

∀ P ∈ PA : ωOG ◦ post ◦ ωOG(P) ⊆ p^#OG(P).

Proof. Let P = (I, M) ∈ PA, (I^2, M^2) = ωOG ◦ post ◦ ωOG(P), (I^R, M^R) = p^#OG(P). We have to show two facts.

“∀ r ∈ Res : I^2_r ⊆ I^R_r.” Let r ∈ Res. Let w ∈ I^2_r. Then there is t ∈ Tid, (v, v’) ∈ →, such that v ∈ ωOG(P), r is available in v’ and v|_{ProofVar(r)} = w. By definition of ωOG we have v|_{LocalData_p} ↓ ProofVar(rsc(v’(pc_t))) ∈ M^v’(pc_t). There are two cases.

Case r ∈ rsc(v’(pc_t)). Then rsc(v’(pc_t)) /∈ rsc(v’(pc_t)). From (3) we conclude v|_{ProofVar(r)} = v’|_{ProofVar(r)} ⊆ I^2_r. By definition of p^#OG we get w ∈ I^2_r.

Case r /∈ rsc(v’(pc_t)). Then r /∈ rsc(v’(pc_t)) ∪ rsc(v’(pc_t)). In particular, r /∈ rsc(v’(pc_t)) \ rsc(v’(pc_t)). If x ∈ ProofVar(r), we have v(x) = v’(x) by definition of ProofVar.

Thus w = v|_{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 \( \gamma_{\text{OG}} \) we get \( v|_{\text{ProofVar}(\cdot)} \in I_r \), thus \( w \in I_R \). By definition of \( p^\theta_{\text{OG}} \), we have \( w \in I^R \).

We have shown \( \Pi^R \subset I^R \).

\( \forall p \in PL, t \in \text{Tid} : \Pi^R \subseteq M^R_p \) : Let \( p \in PL \) and \( t \in \text{Tid} \). Let \( w \in M^L_{p,t} \). Then there is some \( \tilde{t} \in \text{Tid} \), \((v,v') \to t\) such that \( w = v|_{\text{LocalData} \cup \{v|_{\text{ProofVar}(\text{rsc}(p))} \}} \in \gamma_{\text{OG}}(P) \) and \( v'(pc_t) = p \). There are two cases.

Case \( t \neq \tilde{t} \). By (1) we have \( v \sim_{\text{Local}} v' \). Take any \( r \in \text{rsc}(p) \); then \( r \not\in \text{rsc}(v'(pc_t)) \); thus \( v \sim_{\text{ProofVar}(r)} v' \). So \( v|_{\text{LocalData} \cup \{v|_{\text{ProofVar}(\text{rsc}(p))} \}} = w \). From definition of \( \gamma_{\text{OG}} \) we get \( v|_{\text{LocalData} \cup \{v|_{\text{ProofVar}(\text{rsc}(p))} \}} \in M^L_{p,t} \) and \( v|_{\text{LocalData} \cup \{v|_{\text{ProofVar}(\text{rsc}(p))} \}} \in M^R_{p,t} \). Notice that \( v(pc_t) = p \). Thus \( w \in M^L_{p,t} \) and from definition of consistent states we get \( \text{rsc}(p) \cap \text{rsc}(v(pc_t)) = \emptyset \). Thus \( \text{rsc}(p) \cap (\text{rsc}(v(pc_t)) \cup \text{rsc}(v'(pc_t))) = \emptyset \). By definition of \( p^\theta_{\text{OG}} \) we get \( w \in M^R_{p,t} \).

Case \( t = \tilde{t} \). By definition of \( \gamma_{\text{OG}} \) we get \( v|_{\text{LocalData} \cup \{v|_{\text{ProofVar}(\text{rsc}(p))} \}} \in M^R_{p,t} \).

Consider an arbitrary \( r \in \text{rsc}(p) \setminus \text{rsc}(v(pc_t)) \) (there may be none). From \( v' \in \text{ConsState} \) and (1) we get \( r \not\in \bigcup_{t \in \text{Tid} \setminus \{t\}} \text{rsc}(v'(pc_t)) \). Thus \( r \) is available in \( v \).

By definition of \( \gamma_{\text{OG}} \) we get \( v|_{\text{LocalData} \cup \{v|_{\text{ProofVar}(\text{rsc}(p))} \}} \in M^R_{p,t} \). Thus \( w \in M^R_{p,t} \).

We have shown \( M^R_{p,t} \subseteq M^R_{p,t} \).

The last proposition allows using \( p^\theta_{\text{OG}} \) in the relaxed frontier search.

\section*{G.3 Examples}

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

\[
T^{(0)} = \alpha_{\text{OG}}(\text{init}) = \begin{cases} 
\begin{array}{c}
\{ \text{r} \leftrightarrow \{(u \to 0, z \to \text{1}], [u \to 1, z \to \text{0}] \}, \\
\text{r'} \leftrightarrow \{(x \to 0, y \to \text{0}], [x \to \text{1}, y \to \text{1}] \}, \\
(A, 1) \leftrightarrow \{(x \to 0, l \to \text{0}], [x \to \text{1}, l \to \text{1}] \}, \\
(0, 2) \leftrightarrow \{(y \to 0, z \to \text{0}], [y \to \text{1}, z \to \text{1}] \}, \\
\text{everything else} \leftrightarrow \emptyset
\end{array}
\end{cases}
\]

\[
p^\theta_{\text{OG}}(T^{(0)}) = \begin{cases} 
\begin{array}{c}
[\text{r} \to \emptyset, \text{r'} \to \{(x \to 0, y \to \text{0}], [x \to \text{1}, y \to \text{1}] \}], \\
(B, 1) \to \{(u \to 0, z \to 1, x \to 0, l \to 0], [u \to 1, z \to 0, x \to 1, l \to 1] \}, \\
(0, 2) \to \{(y \to 0, z \to 1], [y \to \text{1}, z \to 1] \}, \\
\text{everything else} \to \emptyset
\end{array}
\end{cases}
\]

\[\not\in \bigcup_{j=0}^{0} T^{(j)}, \text{ so } T^{(1)} = p^\theta_{\text{OG}}(T^{(0)}),\]

\[
p^\theta_{\text{OG}}(T^{(1)}) = \begin{cases} 
\begin{array}{c}
[\text{r} \to \emptyset, \text{r'} \to \emptyset], \\
(C, 1) \leftrightarrow \{(u \to 0, z \to 1, x \to 0, y \to 0, l \to 0] \}, \\
(0, 2) \to \{(y \to 0, z \to 0], [y \to \text{1}, z \to \text{1}] \}, \\
\text{everything else} \to \emptyset
\end{array}
\end{cases}
\]
\end{example}
\[
\bigvee_{j=0}^{1} T^{(j)}, \text{ so } T^{(2)} = \rho_{\text{OG}}(T^{(1)}),
\]

\[
\rho_{\text{OG}}(T^{(2)}) = \left\{ \begin{array}{ll}
[r \mapsto \emptyset, r' \mapsto \emptyset], \\
(D, 1) \mapsto \{u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 0, l \mapsto 0\}, \\
(0, 2) \mapsto \{y \mapsto 0, z \mapsto 0\}, [y \mapsto 1, z \mapsto 1]\end{array} \right.
\]

\[
\bigvee_{j=0}^{2} T^{(j)}, \text{ so } T^{(3)} = \rho_{\text{OG}}(T^{(2)}),
\]

\[
\rho_{\text{OG}}(T^{(3)}) = \left\{ \begin{array}{ll}
[r \mapsto \emptyset, r' \mapsto \emptyset],\{(x \mapsto 0, y \mapsto 0)\}, \\
(E, 1) \mapsto \{u \mapsto 0, z \mapsto 1, x \mapsto 0, l \mapsto 0\}, [u \mapsto 1, z \mapsto 0, x \mapsto 0, l \mapsto 1]\end{array} \right.
\]

\[
\bigvee_{j=0}^{3} T^{(j)}, \text{ so } T^{(4)} = \rho_{\text{OG}}(T^{(3)}),
\]

\[
\rho_{\text{OG}}(T^{(4)}) = \left\{ [r \mapsto \emptyset, r' \mapsto \emptyset], \left( (\emptyset)_{p \in \text{PL}, r \in \text{Tal}} \right) \right\} \equiv \bigcup_{j=0}^{4} T^{(j)}, \text{ so }
\]

\[
T^{(5)} = T^{(4)} \sqcup \rho_{\text{OG}}(T^{(4)}) = T^{(4)}.
\]

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

\[
\bigcup_{j=0}^{k} \gamma_{\text{OG}}(T^{(j)}) = \left\{ [u \mapsto 0, z \mapsto 1, x \mapsto 1, y \mapsto 1, l \mapsto 1, pc_1 \mapsto A, pc_2 \mapsto 0], [u \mapsto 1, z \mapsto 0, x \mapsto 0, y \mapsto 0, l \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto 0], [u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 1, l \mapsto 0, pc_1 \mapsto E, pc_2 \mapsto 0], [u \mapsto 1, z \mapsto 0, x \mapsto 0, y \mapsto 0, l \mapsto 1, pc_1 \mapsto E, pc_2 \mapsto 0] \right\}.
\]

Notice that \(\bigcup_{j=0}^{k} \gamma_{\text{OG}}(T^{(j)})\) is strictly stronger than \(\gamma_{\text{OG}}(\inf \Omega)\), e.g., \([u \mapsto 0, z \mapsto 1, x \mapsto 0, y \mapsto 1, l \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto 0] \in \gamma_{\text{OG}}(\inf \Omega) \setminus \left( \bigcup_{j=0}^{k} \gamma_{\text{OG}}(T^{(j)}) \right)\).

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:

\[ S \mapsto \{ \bar{v} \mid \forall t \in Tid \exists v \in S : \bar{v} \sim_{\{pc_j\} \cup DataVar} v \} . \]

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 \( D = \mathcal{P}(ConsState) \), namely

\[ \rho_{OG,mc} : D \to D, S \mapsto \{ v \in ConsState \mid \forall t \in Tid \exists \bar{v} \in S : v \sim_{\{pc_j\} \cup DataVar} \bar{v} \}, \]

is usable for RPL.

**Proposition 58.** \( \rho_{OG,mc} : D \to D \) 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 \( v \in \rho_{OG,mc}(Q) \). Let \( t \in Tid \).

Then there is some \( \bar{v} \in Q \) such that \( v \sim_{\{pc_j\} \cup DataVar} \bar{v} \). Notice that \( \bar{v} \in Q' \). Thus \( v \in \rho_{OG,mc}(Q') \). We have shown \( \rho_{OG,mc}(Q) \subseteq \rho_{OG,mc}(Q') \).

**Extensivity.** Let \( Q \in D \). Let \( v \in Q \). Certainly \( \forall t \in Tid : v \sim_{\{pc_j\} \cup DataVar} v \). Thus \( v \in \rho_{OG,mc}(Q) \). We have shown \( Q \subseteq \rho_{OG,mc}(Q) \).

**Idempotence.** Let \( Q \in D \). Let \( v \in \rho_{OG,mc}(\rho_{OG,mc}(Q)) \). Let \( t \in Tid \). Then there is some \( \bar{v} \in \rho_{OG,mc}(Q) \) such that \( v \sim_{\{pc_j\} \cup DataVar} \bar{v} \). Then there is some \( \bar{v} \in Q \) such that \( \bar{v} \sim_{\{pc_j\} \cup DataVar} \bar{v} \). Then \( v \sim_{\{pc_j\} \cup DataVar} \bar{v} \). Thus \( v \in \rho_{OG,mc}(Q) \). We have shown \( \rho_{OG,mc}(\rho_{OG,mc}(Q)) \subseteq \rho_{OG,mc}(Q) \). \( \square \)

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 \( \rho_{OG,mc} \), 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 \( \rho_{OG,mc} \) 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 \( \rho_{OG,mc} \).

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

\[ \{ v \in \text{ConsState} \mid v(ww), v(ar) \in N_0 \land v(aw) \in \{0, 1\} \land \forall i \in N_m : (v(pc_{\text{reader}}) \in \{ \text{xready, read} \mid x \in \{ \text{start, finish} \} \land y \in \{ A, B, C \} ) \land (v(pc_{\text{reader}}) = \text{startreadB} \Rightarrow v(ww) = 0) \land (v(pc_{\text{reader}}) = \text{startreadC} \Rightarrow v(ww) = 0 < v(ar)) \land \forall j \in N_m : (v(pc_{\text{writer}}) \in \{ \text{write, write} \mid x \in \{ \text{ask, start, finish} \} \land y \in \{ A, B, C \} ) \land (v(pc_{\text{writer}}) = \text{askwriteC} \Rightarrow v(ww) > 0) \land (v(pc_{\text{writer}}) = \text{startwriteB} \Rightarrow v(ar) = v(aw) = 0) \land (v(pc_{\text{writer}}) = \text{startwriteC} \Rightarrow v(ar) = 0 < v(aw)) \land (v(pc_{\text{writer}}) = \text{finishwriteC} \Rightarrow v(aw) = 0) \} \].

\( \subseteq \): First we show the right hand side (RHS) is closed under \( \rho_{OG,mc} \). Let \( v \in \rho_{OG,mc}(\text{RHS}) \). By definition of \( \rho_{OG,mc} \) there is some \( w \in \text{RHS} \) such that \( v \sim_{\{ pc_{\text{reader}}, \, ww, \, ar, \, aw \} } w \). Then \( v(ww), v(ar) \in N_0 \) and \( v(aw) \in \{0, 1\} \).
Let \( i \in N_m \). Then there is \( w \in \text{RHS} \) such that \( v \sim_{\{pc_{\text{reader}}, \, ww, \, ar, \, aw\}} w \). Thus \( v(pc_{\text{reader}}) \in \{ \text{startreadA}, \text{startreadB}, \text{startreadC}, \text{read}, \text{finishreadA}, \text{finishreadB}, \text{finishreadC} \} \). If \( v(pc_{\text{reader}}) = \text{startreadB}, \) then \( w(pc_{\text{reader}}) = \text{startreadB} \), so \( w(ww) = 0 \) by definition of \( \text{RHS} \), so \( v(ww) = 0 \). If \( v(pc_{\text{reader}}) = \text{startreadC} \), then \( w(pc_{\text{reader}}) = \text{startreadC} \), so \( w(ww) = 0 < w(ar) \) by definition of \( \text{RHS} \), so \( v(ww) = 0 < v(ar) \). Let \( j \in N_m \). Then there is \( w \in \text{RHS} \) such that \( v \sim_{\{pc_{\text{writer}}, \, ww, \, ar, \, aw\}} w \). Thus \( v(pc_{\text{writer}}) \in \{ \text{askwriteA}, \text{askwriteB}, \text{askwriteC}, \text{startwriteA}, \text{startwriteB}, \text{startwriteC}, \text{write}, \text{finishwriteA}, \text{finishwriteB}, \text{finishwriteC} \} \). If \( v(pc_{\text{writer}}) = \text{askwriteC} \), then \( w(pc_{\text{writer}}) = \text{askwriteC} \), so \( w(ww) > 0 \), so \( v(ww) > 0 \). If \( v(pc_{\text{writer}}) = \text{startwriteB} \), then \( w(pc_{\text{writer}}) = \text{startwriteB} \), so \( w(ar) = w(aw) = 0 \), so \( v(ar) = v(aw) = 0 \). If \( v(pc_{\text{writer}}) = \text{startwriteC} \), then \( w(pc_{\text{writer}}) = \text{startwriteC} \), so \( w(ar) = 0 < w(aw) \), so \( v(ar) = 0 < v(aw) \). If \( v(pc_{\text{writer}}) = \text{finishwriteC} \), then \( w(pc_{\text{writer}}) = \text{finishwriteC} \), so \( w(aw) = 0 \), so \( v(aw) = 0 \).
Thus \( v \in \text{RHS} \). Since \( v \) was arbitrary, \( \rho_{OG,mc}(\text{RHS}) \subseteq \text{RHS} \). Notice that \( \text{init} \cup \text{post}(\text{RHS}) \subseteq \text{RHS} \), thus \( \rho_{OG,mc}(\text{init} \cup \text{post}(\text{RHS})) \subseteq \text{RHS} \), so \( \text{RHS} \) is a postfix point of \( F \), so \( Q \subseteq \text{RHS} \).

\( \supseteq \): Let
\[
\begin{align*}
\text{CritR} & = \{ \text{xready} \mid x \in \{ \text{start, finish} \}, y \in \{B, C\} \}, \\
\text{CritW} & = \{ \text{xwrite} \mid x \in \{ \text{ask, start, finish} \}, y \in \{B, C\} \}, \\
\text{NonCritR} & = \{ \text{read, xreadA} \mid x \in \{ \text{start, finish} \} \}, \\
\text{NonCritW} & = \{ \text{write, xwriteA} \mid x \in \{ \text{ask, start, finish} \} \}.
\end{align*}
\]
(I) First we will show that for all $a, b \in \mathbb{N}_0, c \in \{0, 1\}$, the state $v$ with $(\forall i \in \mathbb{N}_n : v(pc_{\text{reader},i}) = \text{startreadA}) \land (\forall j \in \mathbb{N}_n : v(pc_{\text{writer},j}) = \text{askwriteA}) \land v(ww) = a \land v(ar) = b \land v(aw) = c$ 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 $\text{init} \subseteq Q$. Otherwise $c = 1$. Take the initial state, make 3 steps of $\text{writer}_1$, resulting in a state $v_1 \in Q$ with $v_1(pc_{\text{writer},i}) = \text{startwriteA}$, all other writers are at $\text{askwriteA}$, all readers are at $\text{startreadA}$, $v_1(ww) = 1, v_1(ar) = 0, v_1(aw) = 0$. Making 3 steps of $\text{writer}_2$ from the initial state results in a state $v_2 \in Q$ with $v_2(pc_{\text{writer},j}) = \text{startwriteA}$, all other writers are at $\text{askwriteA}$, all readers are at $\text{startreadA}$, $v_2(ww) = 1, v_2(ar) = 0, v_2(aw) = 0$. Consider the state $v_3$ such that $v_3(pc_{\text{writer},i}) = v_3(pc_{\text{writer},j}) = \text{startwriteA} \land (\forall j \in \mathbb{N}_m \setminus \{1, 2\} : v_3(pc_{\text{writer},j}) = \text{askwriteA}) \land v_3(ww) = v_3(ar) = v_3(aw) = 0$. Making three steps of $\text{writer}_2$ from $v_3$ results in a state $v_5 \in Q$ such that $(\forall i \in \mathbb{N}_n : v_5(pc_{\text{reader},i}) = \text{startreadA}) \land v_5(ww) = v_5(ar) = v_5(aw) = 0 \land v_5(\text{aw}) = 1$. Analogously, starting from $v_3$, making seven steps of $\text{writer}_2$ and three steps of $\text{writer}_1$ results in a state $v_7 \in Q$ such that $(\forall i \in \mathbb{N}_n : v_7(pc_{\text{reader},i}) = \text{startreadA}) \land v_7(ww) = v_7(ar) = v_7(aw) = 1$. Then $v_7$ 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 $(\forall i \in \mathbb{N}_n : v(pc_{\text{reader},i}) = \text{startreadA}) \land (\forall j \in \mathbb{N}_n : v(pc_{\text{writer},j}) = \text{askwriteA}) \land v(ww) = 0 \land v(ar) = b \land v(aw) = c$ 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 $v_1 \in Q$ in which each thread sits at its initial location and $v_1(ww) = 0 \land v_1(ar) = b - 1 \land v_1(aw) = c$. Making three steps of $\text{reader}_1$ from $v_1$ results in a state $v_2 \in Q$ such that $v_2(pc_{\text{reader},i}) = \text{read} \land (\forall i \in \mathbb{N}_n \setminus \{1\} : v_2(pc_{\text{reader},i}) = \text{startreadA}) \land (\forall j \in \mathbb{N}_m : v_2(pc_{\text{writer},j}) = \text{askwriteA}) \land v_2(ww) = 0 \land v_2(ar) = b \land v_2(aw) = c$. Making three steps of $\text{reader}_2$ from $v_1$, we get a state $v_3 \in Q$ such that $v_3(pc_{\text{reader},i}) = \text{read} \land (\forall i \in \mathbb{N}_n \setminus \{2\} : v_3(pc_{\text{reader},i}) = \text{startreadA}) \land (\forall j \in \mathbb{N}_m : v_3(pc_{\text{writer},j}) = \text{askwriteA}) \land v_3(ww) = 0 \land v_3(ar) = b \land v_3(aw) = c$. Since $Q$ is a fixpoint of $\rho_{\text{OG,mc}}$, the state $v$
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 \((\forall i \in \mathbb{N}_n : v(pc_{\text{reader}_i}) = \text{startread}A) \land (\forall j \in \mathbb{N}_m : v(pc_{\text{writer}_j}) = \text{askwrite}A) \land v(ww) = a \land v(ar) = b \land v(aw) = c\) 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 \(v_1 \in Q\) in which each threads sits at its initial location, \(v(ww) = a - 1 \land v(ar) = b \land v(aw) = c\). Making three steps of \(\text{writer}_1\) from \(v_1\) results in a state \(v_2 \in Q\) such that \(v_2(pc_{\text{writer}_j}) = \text{startwrite}A\), all other threads are at their initial locations, \(v(ww) = a \land v(aw) = c\). Making three steps of \(\text{writer}_2\) from \(v_1\) results in a state \(v_3 \in Q\) such that \(v_3(pc_{\text{writer}_j}) = \text{startwrite}A\), all other threads are at their initial locations, \(v(v(ww) = a \land v(aw) = c\). From \(v_2 \in Q, v_3 \in Q\) and the definition of \(\rho_{\text{OG,mc}}\) we obtain that the state \(v\) such that all threads in \(v\) are at their initial locations, \(v(ww) = a \land v(ar) = b \land v(aw) = c\) 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 \(v_1 \in Q\) in which all threads are at their initial locations, \(v_1(ww) = 0 \land v_1(ar) = b \land v_1(aw) = c\). Making three steps by thread \(\text{reader}_i\) results in a state \(v_2 \in Q\) such that \(v_2(pc_{\text{reader}_i}) = \text{read}\), all other threads are at their initial location, \(v_2(ww) = 0 \land v_2(aw) = c\). Since \(n \geq 2\), there is some \(k \in \mathbb{N}_n \setminus \{i\}\). Making three steps from \(v_1\) by thread \(\text{reader}_k\) results in a state \(v_3 \in Q\) such that \(v_3(pc_{\text{reader}_j}) = \text{read}\), all other threads are at their initial locations, \(v_3(ww) = 0 \land v_3(ar) = c\). Combining \(v_2 \in Q, v_3 \in Q\) and using the definition of \(\rho_{\text{OG,mc}}\) we obtain \(v_4 \in Q\) such that \(v_4(pc_{\text{reader}_j}) = v_4(pc_{\text{reader}_j}) = \text{read}\), all threads except \(\text{reader}_i\) and \(\text{reader}_k\) are at their corresponding initial location, \(v_4(ww) = 0, v_4(ar) = b + 1\) and \(v_4(aw) = c\). Making four steps from \(v_4\) by thread \(\text{reader}_k\) results in the state \(v \in Q\) such that \(v(pc_{\text{reader}_j}) = \text{read} \land (\forall i \in \mathbb{N}_n \setminus \{i\} : v(pc_{\text{reader}_j}) = \text{startread}A) \land v(ww) = 0 \land v(ar) = b \land v(aw) = c\).

Case \(a > 0\). Let \(b \in \mathbb{N}, c \in \{0, 1\}, i \in \mathbb{N}_n\). By assumption hypothesis there is \(v_1 \in Q\) such that \(v_1(pc_{\text{reader}_i}) = \text{read}\), all other threads are at their initial locations, \(v_1(ww) = a - 1 \land v_1(ar) = b \land v_1(aw) = c\). Taking three steps of \(\text{writer}_1\) from \(v_1\) results in a state \(v_2 \in Q\) such that \(v_2(pc_{\text{writer}_j}) = \text{read} \land v_2(pc_{\text{reader}_i}) = \text{startwrite}A\), all other threads are at their initial locations, \(v_2(ww) = a \land v_2(ar) = b \land v_2(aw) = c\). In particular, \(v_2(pc_{\text{writer}_j}) = \text{askwrite}A\). Taking three steps of \(\text{writer}_2\) from \(v_1\) results in a state \(v_3 \in Q\) such that \(v_3(pc_{\text{reader}_j}) = \text{read} \land v_3(pc_{\text{writer}_j}) = \text{startwrite}A\), all other threads are at their initial locations, \(v_3(ww) = a\).
\( \land v_3(ar) = b \land v_3(aw) = c \). In particular, \( v_3(pc_{\text{writer}}) = \text{askwriteA} \).

Combining \( v_2 \in Q \) and \( v_3 \in Q \) we get that the state \( v \) with \( v(pc_{\text{reader}}) = \text{read} \), all other threads are at their initial locations, \( v(ww) = a \land v(ar) = b \land v(aw) = c \) is in \( Q \).

(III) We claim that for every \( a, b, \in N_0, c \in \{0,1\} \), the state \( v \) such that
\[ v(pc_{\text{reader}}) = \text{finishreadA} \land (\forall k \in N_m \setminus \{j\} : v(pc_{\text{reader}}) = \text{startreadA}) \land (\forall j \in N_m : v(pc_{\text{writer}}) = \text{askwriteA}) \land v(ww) = a \land v(ar) = b \land v(aw) = c \]
is in \( Q \). To see that, make one step from a state described in (II).

(IV) We claim that for all \( a, b, \in N_0, c \in \{0,1\}, j \in N_m \), the state \( v \) such that
\[ (\forall i \in N_m : v(pc_{\text{reader}}) = \text{startreadA}) \land v(pc_{\text{writer}}) = \text{startwriteA} \land (\forall k \in N_m \setminus \{j\} : v(pc_{\text{writer}}) = \text{askwriteA}) \land v(ww) = a \land v(ar) = b \land v(aw) = c \]
is in \( Q \). Let \( a \in N_0 \). We case split on \( a \).

Case \( a = 0 \). We’ll induct on \( b \).

Case \( b = 0 \). Let \( j \in N_m, c \in \{0,1\} \). Making from the initial state (which is in \( Q \)) three steps of \( \text{writer}_j \) results in \( v_1 \in Q \) such that \( v_1(pc_{\text{writer}}) = \text{startwriteA} \), all other threads are at their initial locations, \( v_1(ww) = 1 \land v_1(ar) = v_1(aw) = 0 \). Since \( m \geq 2 \), there is some \( k \in N_m \setminus \{j\} \). Taking three steps of \( \text{writer}_k \) from the initial state results in \( v_2 \in Q \) such that \( v_2(pc_{\text{writer}}) = \text{startwriteA} \), all other threads are at their initial locations, \( v_2(ww) = 1 \land v_2(ar) = v_2(aw) = 0 \). Combining \( v_1 \in Q \) and \( v_2 \in Q \), we obtain a state \( v_3 \in Q \) such that \( v_3(pc_{\text{writer}}) = v_3(pc_{\text{reader}}) = \text{startwriteA} \), all other threads are at their initial locations, \( v_3(ww) = 1 \land v_3(ar) = v_3(aw) = 0 \). Making seven steps of \( \text{writer}_k \) from \( v_3 \) results in a state \( v_4 \in Q \) such that \( v_4(pc_{\text{writer}}) = \text{startwriteA} \), all other threads are at their initial locations, \( v_4(ww) = v_4(ar) = v_4(aw) = 0 \).

Case \( c = 0 \). Notice that \( v_4 \) satisfies all the requirements for \( v \).

Case \( c = 1 \). Making seven steps of \( \text{writer}_j \) from \( v_3 \) results in a state \( v_5 \in Q \) such that \( v_5(pc_{\text{writer}}) = \text{startwriteA} \), all other threads are at their initial locations, \( v_5(ww) = v_5(ar) = v_5(aw) = 0 \). Combining \( v_4 \in Q, v_5 \in Q \) and the definition of \( \rho_{\text{G,mcv}} \), results in a state \( v_6 \in Q \) such that \( v_6(pc_{\text{writer}}) = v_6(pc_{\text{reader}}) = \text{startwriteA} \), all other threads are at their initial locations, \( v_6(ww) = v_6(ar) = v_6(aw) = 0 \). Making three steps of \( \text{writer}_k \) from \( v_6 \) results in a state \( v_7 \in Q \) such that \( v_7(pc_{\text{writer}}) = \text{startwriteA} \land v_7(pc_{\text{writer}}) = \text{write} \), all other threads are at their initial locations, \( v_7(ww) = v_7(ar) = 0 \land v_7(aw) = 1 \). By (I) there is some state \( v_8 \in Q \) in which all threads are at their initial locations and \( v_8(ww) = v_8(ar) = 0 \land v_8(aw) = 1 \). Combining \( v_7 \in Q \) and \( v_8 \in Q \), we obtain the state \( v \in Q \) such that \( v(pc_{\text{writer}}) = \text{startwriteA}, \) all other threads are at their initial locations, and \( v(ww) = v(ar) = 0 \land v(aw) = 1 \).

Case \( b > 1 \). Let \( j \in N_m, c \in \{0,1\} \). By induction assumption there is some \( v_1 \in Q \) such that \( v_1(pc_{\text{writer}}) = \text{startwriteA} \), all other
threads are at their initial locations, \( v_1(ww) = 0 \land v_1(ar) = b - 1 \land v_1(aw) = c \). Making three steps of reader, from \( \nu_1 \) results in a state \( \nu_2 \in Q \) such that \( \nu_2(pc_{\text{reader}_1}) = \text{read} \land v_2(pc_{\text{writer}_1}) = \text{startwriteA} \), all other threads are at their initial locations, \( \nu_2(ww) = 0 \land v_2(ar) = b \land v_2(aw) = c \). Making three steps of reader, from \( \nu_1 \) results in a state \( \nu_3 \in Q \) such that \( \nu_3(pc_{\text{reader}_2}) = \text{read} \land v_3(pc_{\text{writer}_2}) = \text{startwriteA} \), all other threads are at their initial locations, \( \nu_3(ww) = 0 \land v_3(ar) = b \land v_3(aw) = c \). Notice that \( \nu_2(pc_{\text{reader}_1}) = \nu_3(pc_{\text{reader}_2}) = \nu_3(pc_{\text{writer}_2}) = \text{startreadA} \). From \( \nu_2 \in Q \), \( \nu_3 \in Q \) we get the state \( \nu \in Q \) such that \( \nu(pc_{\text{writer}_1}) = \text{startwriteA} \), all other threads are at their initial locations, \( \nu(ww) = 0 \land v(ar) = b \land v(aw) = c \).

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 \( \nu_1(ww) = a - 1 \land v_1(ww) = b \land v_1(aw) = c \). Making three steps by \( \text{writer}_j \) results in a state \( \nu \in Q \) such that \( \nu(pc_{\text{writer}_j}) = \text{startwriteA} \), all other threads are at their initial locations, \( \nu(ww) = a \land v(ar) = b \land v(aw) = c \).

(V) We claim that for all \( a, b, c \in \mathbb{N}_0, j \in \{0, 1\} \), \( j \in \mathbb{N}_m \) the state \( \nu \) such that \( (\forall i \in \mathbb{N}_m : v_i(pc_{\text{reader}_i}) = \text{startreadA} \land \nu(pc_{\text{writer}_i}) = \text{write} \land (\forall k \in \mathbb{N}_m \setminus \{j\} : v_k(ww) = a \land v_k(ar) = b \land v_k(aw) = c) \) 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 \( \text{writer}_j \) from the initial state results in a state \( \nu_1 \) in \( Q \) such that \( v_1(pc_{\text{writer}_1}) = \text{write} \), other threads are at their initial locations, \( v_1(ww) = v_1(aw) = 1 \) and \( v_1(ar) = 0 \). Let \( k \in \mathbb{N}_m \setminus \{j\} \). Making six steps of \( \text{writer}_k \) from the initial state results in a state \( \nu_2 \in Q \) such that \( v_2(pc_{\text{writer}_2}) = \text{write} \), other threads are at their initial locations, \( v_2(ww) = v_2(aw) = 1 \) and \( v_2(ar) = 0 \). Combining \( \nu_2 \in Q \), \( \nu_3 \in Q \) and \( k \neq j \), we obtain a state \( \nu_3 \in Q \) such that \( v_3(pc_{\text{writer}_3}) = v_3(pc_{\text{writer}_1}) = v_3(pc_{\text{writer}_2}) = \text{write} \), other threads are at their initial locations, \( v_3(ww) = v_3(aw) = 1 \) and \( v_3(ar) = 0 \). Making four steps of \( \text{writer}_j \) from \( \nu_3 \) results in the state \( \nu \in Q \) such that \( \nu(pc_{\text{writer}_1}) = \text{write} \), all other threads are at their initial locations, \( \nu(ww) = \nu(aw) = \nu(ar) = 0.0 \).

Case \( c = 1 \). By (IV) the state \( \nu_1 \) such that \( v_1(pc_{\text{writer}_1}) = \text{startwriteA} \), all other threads are at their initial locations, \( v_1(ww) = v_1(ar) = v_1(aw) = 0 \) is in \( Q \). Make three steps by thread \( \text{writer}_j \).

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

By induction assumption, there is a state \( \nu_1 \in Q \) such that \( v_1(pc_{\text{writer}_1}) = \text{write} \), other threads are at their initial locations, \( v_1(ww) = 0 \land v_1(ar) = b - 1 \land v_1(aw) = c \). Making three steps of reader, from \( \nu_1 \) results in a state \( \nu_2 \in Q \) such that \( v_2(pc_{\text{reader}_1}) = \text{read} \), \( \nu_2(pc_{\text{reader}_1}) = \text{read} \), all other threads (including reader) are at
their initial locations, $v2(ww) = 0 \land v2(ar) = b \land v2(aw) = c$. Making three steps of reader$_2$ from $v1$ results in a state $v3 \in Q$ such that $v3(pc_{\text{writer}}) = \text{write}, v3(pc_{\text{reader}}) = \text{read}$, all other threads (including reader$_1$) are at their initial locations, $v3(ww) = 0 \land v3(ar) = b \land v3(aw) = c$. Combining $v2 \in Q$ and $v3 \in Q$ we get the state $v \in Q$ such that $v(pc_{\text{writer}}) = \text{write}$, all other threads are at their initial locations and $v(ww) = 0 \land v(ar) = b \land v(aw) = c$.

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 $v1(pc_{\text{writer}}) = \text{write}$, 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 writer$_k$ from $v1$ results in a state $v2 \in Q$ such that $v2(pc_{\text{writer}}) = \text{write}, v2(pc_{\text{writer}})_{k} = \text{startwriteA}, v2(ww) = a, v2(ar) = b, v2(aw) = c$. 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, $v3(pc_{\text{writer}}) = \text{askwriteA}$. From $v2 \in Q$ and $v3 \in Q$ we obtain that the state $v$ such that $v(pc_{\text{writer}}) = \text{write}$, all other threads are at their initial locations, $v(ww) = a \land v(ar) = b \land v(aw) = c$ 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 $(\forall i \in \mathbb{N}_n: v(pc_{\text{reader}}) = \text{startreadA}) \land v(pc_{\text{writer}}) \land (\forall k \in \mathbb{N}_m \setminus \{j\}: v(pc_{\text{writer}}) = \text{askwriteA}) \land v(ww) = a \land v(ar) = b \land v(aw) = c$ 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 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 $(\forall i \in \mathbb{N}_n: v(pc_{\text{reader}}) \in \text{NonCrit}(R)) \land (\forall j \in \mathbb{N}_m: v(pc_{\text{writer}}) \in \text{NonCrit}(W)) \land v(ww) = a \land v(ar) = b \land v(aw) = c$ is in $Q$.

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

(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 $v(pc_{\text{reader}}) = \text{startreadB} \land (\forall k \in \mathbb{N}_m \setminus \{i\}: v(pc_{\text{reader}}) \in \text{NonCrit}(R)) \land (\forall j \in \mathbb{N}_m: v(pc_{\text{writer}}) \in \text{NonCrit}(W)) \land v(ww) = 0 \land v(ar) = b \land v(aw) = c$ 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 (VIII), the state $v1 = v(pc_{\text{reader}}) \rightarrow \text{startreadA}$ is in $Q$. Make one step of reader$_i$.

(IX) We claim that for all $b \in \mathbb{N}_0, c \in \{0, 1\}, i \in \mathbb{N}_n$, any state $v$ such that $v(pc_{\text{reader}}) = \text{startreadC} \land (\forall k \in \mathbb{N}_m \setminus \{i\}: v(pc_{\text{reader}}) \in \text{NonCrit}(R)) \land (\forall j \in \mathbb{N}_m: v(pc_{\text{writer}}) \in \text{NonCrit}(W)) \land v(ww) = 0 \land v(ar) = b \land v(aw) = c$ is in $Q$.

To see that, let $b \in \mathbb{N}_0, c \in \{0, 1\}, i \in \mathbb{N}_n$ and $v$ as stated. By (VIII) the state $v1 = v(pc_{\text{reader}}) \rightarrow \text{startreadB, ar} \rightarrow b - 1$ is in $Q$. Make one step of thread 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 $v(pc_{\text{reader}}) = \text{finishreadB} \land (\forall k \in \mathbb{N}_m \setminus \{i\}: v(pc_{\text{reader}}) \in \text{NonCrit}(R)) \land (\forall j \in \mathbb{N}_m: v(pc_{\text{writer}}) \in \text{NonCrit}(W)) \land v(ww) = a \land v(ar) = b \land v(aw) = c$ is in $Q$.
To see that, let \( a, b \in \mathbb{N}_0, c \in \{0, 1\}, i \in \mathbb{N}_m \), \( v \) as required. By (VII), the state \( v_1 = v[p_{reader} \mapsto finishreadA] \) is in \( Q \). Take one step by reader, from \( v_1 \).

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

To see that, let \( a, b \in \mathbb{N}_0, c \in \{0, 1\}, i \in \mathbb{N}_m \) and \( v \) as above. By (X), the state \( v_1 = v[p_{reader} \mapsto finishreadB, ar \mapsto b + 1] \) is in \( Q \). Take one step by thread reader, \( r \).

(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 \( (\forall i \in \mathbb{N}_m : v(p_{reader}_i) \in NonCritR) \land v(p_{writer}) = askwriteB \land (\forall k \in \mathbb{N}_m \setminus \{j\}) : v(p_{writer}_k) \in NonCritW \land v(ww) = a \land v(ar) = b \land v(aw) = c \) 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 \( v_1 = v[p_{writer} \mapsto askwriteA] \) is in \( Q \). Apply one step of writer, \( w \), to \( v_1 \).

(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 \( (\forall i \in \mathbb{N}_m : v(p_{reader}_i) \in NonCritR) \land v(p_{writer}) = askwriteB \land (\forall k \in \mathbb{N}_m \setminus \{j\}) : v(p_{writer}_k) \in NonCritW \land v(ww) = a \land v(ar) = b \land v(aw) = c \) 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 \( v_1 = v[p_{writer} \mapsto askwriteB, ww \mapsto a - 1] \) is in \( Q \). Apply one step of writer, \( w \), to \( v_1 \).

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

To see that, let \( a \in \mathbb{N}_0, j \in \mathbb{N}_m \), \( v \) as stated. By (VII) the state \( v_1 = v[p_{writer}_j \mapsto startwriteA] \) is in \( Q \). Apply one step of writer, \( w \), to \( v_1 \).

(XV) We claim that for all \( a \in \mathbb{N}_0, j \in \mathbb{N}_m \) any state \( v \) such that \( (\forall i \in \mathbb{N}_m : v(p_{reader}_i) \in NonCritR) \land v(p_{writer}_j) = startwriteC \land (\forall k \in \mathbb{N}_m \setminus \{j\}) : v(p_{writer}_k) \in NonCritW \land v(ww) = a \land v(ar) = 0 \land v(aw) = 1 \) 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 \( v_1 = v[p_{writer}_j \mapsto startwriteB, aw \mapsto 0] \) is in \( Q \). Make one step from \( v_1 \) by thread writer, \( w \).

(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 \( (\forall i \in \mathbb{N}_m : v(p_{reader}_i) \in NonCritR) \land v(p_{writer}_k) = finishwriteB \land (\forall k \in \mathbb{N}_m \setminus \{j\}) : v(p_{writer}_k) \in NonCritW \land v(ww) = a \land v(ar) = b \land v(aw) = c \) 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 \( w = v[p_{writer}_j \mapsto finishwriteA] \) is in \( Q \). Apply one step of writer, \( w \), from \( w \).

(XVII) We claim that for all \( a, b \in \mathbb{N}_0, j \in \mathbb{N}_m \) any state \( v \) such that \( (\forall i \in \mathbb{N}_m : v(p_{reader}_i) \in NonCritR) \land v(p_{writer}_k) = finishwriteC \land (\forall k \in \mathbb{N}_m \setminus \{j\}) : v(p_{writer}_k) \in NonCritW \land v(ww) = a \land v(ar) = b \land v(aw) = 0 \) 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 \( w = v[p_{\text{writer}} \rightarrow \text{finishwriteB}, aw \rightarrow 1] \) is in \( Q \). Apply one step of \text{writer}_j from \( w \).

Let \( v \in \text{RHS} \). If \text{control} is available in \( v \), apply (VII). Otherwise there is some thread at a critical location \( \ldots B \) or \( \ldots C \). If the critical location is in a reader, apply one of (VIII), (IX), (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 \( \rho_{\text{OG,mc}} \) coincide on Readers-Writers.

\[ \square \]

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

Let \( v \in \rho_{\text{OG,mc}}(\text{init}) \). For the left thread we have some \( v_1 \in \text{init} \) such that \( v \sim_{(pc_1, \Sigma, x, y, z)} v_1 \). Thus \( v(pc_1) = A \) and \( v(u) \neq v(z) = v(x) = v(y) = v(l) \). For the right thread we have some \( v_2 \in \text{init} \) such that \( v \sim_{(pc_2, \Sigma, x', y', z')} v_2 \). Thus \( v(pc_2) = 0 \). Then \( v \in \text{init} \).

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

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

\[ \square \]

**Example 61 (SepThreads).** Consider an RPL transition system from \( \text{SepThreads} \). For the syntactic structure \( \text{StrSepThreads} \) and \( Q \in D \) we have \( \rho_{\text{OG,mc}}(Q) = \{v \in \text{ConsState} | \forall t \in \text{Tid} \exists v \in Q : v \sim_{(pc, \Sigma, \text{DataB}} v \} \subseteq \{v \in \text{ConsState} | \forall t \in \text{Tid} \exists v \in Q : v \sim_{\text{Local}} v \} = \rho(Q) \). Thus \( \text{lfp}(\text{lfp}(\lambda x. \rho_{\text{OG,mc}}(\text{init} \cup \text{post}(x)))) \subseteq \text{lfp}(\text{lfp}(\lambda x. \rho(\text{init} \cup \text{post}(x)))) \subseteq \text{init} \). Thus \( \text{lfp}(\lambda x. \rho_{\text{OG,mc}}(\text{init} \cup \text{post}(x))) = \text{init} \).

The precision of Owicki-Gries-core and abstract interpretation with \( \rho_{\text{OG,mc}} \) is the same on RPL transition systems from \( \text{SepThreads} \).

\[ \square \]

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

Fig. 6: RPL transition system progAssign1 with its smallest Owicki-Gries-core proof.
The proof denotes the property

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

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

\[
X^{(0)} = \emptyset,
X^{(1)} = \{[x \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto A]\},
X^{(2)} = \left\{ \begin{array}{l}
[x \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto A], \\
x \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto B], [x \mapsto 0, pc_1 \mapsto B, pc_2 \mapsto A] \end{array} \right\},
\]

\[
\text{init } \cup \text{post}(X^{(2)}) = \left\{ \begin{array}{l}
[x \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto A], \\
x \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto B], [x \mapsto 0, pc_1 \mapsto B, pc_2 \mapsto A], \\
x \mapsto 1, pc_1 \mapsto A, pc_2 \mapsto C], [x \mapsto 1, pc_1 \mapsto C, pc_2 \mapsto A] \end{array} \right\},
\]

\[
X^{(3)} = \left\{ \begin{array}{l}
[x \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto A], \\
x \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto B], [x \mapsto 0, pc_1 \mapsto B, pc_2 \mapsto A], \\
x \mapsto 1, pc_1 \mapsto A, pc_2 \mapsto C], [x \mapsto 1, pc_1 \mapsto C, pc_2 \mapsto A] \end{array} \right\},
\]

\[
\text{init } \cup \text{post}(X^{(3)}) = \left\{ \begin{array}{l}
[x \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto A], \\
x \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto B], [x \mapsto 0, pc_1 \mapsto B, pc_2 \mapsto A], \\
x \mapsto 1, pc_1 \mapsto A, pc_2 \mapsto C], [x \mapsto 1, pc_1 \mapsto C, pc_2 \mapsto A] \end{array} \right\},
\]

\[
X^{(4)} = \left\{ \begin{array}{l}
[x \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto A], \\
x \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto B], [x \mapsto 0, pc_1 \mapsto B, pc_2 \mapsto A], \\
x \mapsto 1, pc_1 \mapsto A, pc_2 \mapsto C], [x \mapsto 1, pc_1 \mapsto C, pc_2 \mapsto A] \end{array} \right\},
\]

\[
\text{init } \cup \text{post}(X^{(4)}) = \left\{ \begin{array}{l}
[x \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto A], \\
x \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto B], [x \mapsto 0, pc_1 \mapsto B, pc_2 \mapsto A], \\
x \mapsto 1, pc_1 \mapsto A, pc_2 \mapsto C], [x \mapsto 1, pc_1 \mapsto C, pc_2 \mapsto A] \end{array} \right\},
\]

\[
X^{(5)} = X^{(4)}.
\]

Notice that \(\gamma_{OG}(\text{inf } D) \subseteq \text{lfp}(\lambda x. \rho_{OG, mc}(\text{init } \cup \text{post}(x)))\).

So abstract interpretation with \(\rho_{OG, mc}\) produces a strictly weaker property than the strongest Owicki-Gries-core proof on this example. \(\square\)

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 \(\text{Tid} = \{1, 2\}, \ PL = \{A, B\}, \ \text{Val} = PL \cup \{0, 1\}, \ \text{Res} = \emptyset, \ \text{Local}_1 = \{pc_1, l_1\}, \ \text{Local}_2 = \{pc_2, l_2\}, \ \text{ProofVar} = \emptyset, \ \text{rsc}(A) = \emptyset, \ \text{rsc}(B) = \emptyset.\)
Then
\[ \gamma_{OG}(\inf \mathcal{D}) = \{ \left[ l_1 \mapsto 0, l_2 \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto A \right], \right. \]
\[ \left. \left[ l_1 \mapsto 0, l_2 \mapsto 1, pc_1 \mapsto A, pc_2 \mapsto A \right], \right. \]
\[ \left. \left[ l_1 \mapsto 1, l_2 \mapsto 0, pc_1 \mapsto A, pc_2 \mapsto A \right], \right. \]
\[ \left. \left[ l_1 \mapsto 1, l_2 \mapsto 1, pc_1 \mapsto A, pc_2 \mapsto A \right] \}, \]
and
\[ \rho_{OG,mc}(\text{init}) = \text{init}, \text{ thus } \text{lfp}(\lambda x. \rho_{OG,mc}(\text{init} \cup \text{post}(x))) = \text{init}. \]
So the strongest Owicki-Gries-core-provable property is strictly weaker than the strongest property provable by abstract interpretation with \( \rho_{OG,mc} \).

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 \( \rho_{OG,mc} \), 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 \( \rho_{OG,mc} \).

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 \( \rho_{OG,mc} \) 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

\[ \rho_c \rightarrow \rho \rightarrow \text{Owicki-Gries core} \rightarrow \rho \rightarrow \text{identity} \]

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 \( \rho_c \) 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 \( \rho_c \), 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 \( \rho_c \)
should be recommended to attempt first.