Rely-Guarantee Is Coinductive
– A Proof-Centered Investigation of Inductively Approximated Coinduction –

Abstract. We make the case that the foundation for Rely-Guarantee reasoning can be fruitfully delivered by a coinductive semantics. Using insight from an Isabelle formalization, via a proof analysis we show that the coinductive semantics tends to simplify the proof development; in particular it enables more direct proofs for the soundness of the Rely-Guarantee rules. The comparison between inductive and coinductive proofs also suggests inductive counterparts of coinductive "up-to" enhancements. On the way, we fill a gap in the literature, by showing that three previously defined inductive semantics for Rely-Guarantee are equivalent. Underlying our transformation of an inductive into a coinductive semantics is the notion of inductively approximating a coinductive predicate—which, deployed in the opposite direction (from coinduction to induction), is a standard technical tool for approximating process algebra bisimilarities. On the spectrum between the abstract fixpoint theorems and concrete instances, we formalize effective format-based criteria that enable sound approximation.

1 Introduction

The coinduction definition and proof method, which is the categorical dual of induction, has recently emerged as a powerful methodology for specifying and reasoning about systems. Roughly speaking, while induction is most suitable for describing finitary syntax and behavior, coinduction excels in the compact description of infinite system behavior [58,37,26,32,56,43,69].

The Rely-Guarantee method (§2) for the compositional verification of concurrent programs, introduced by Jones at the beginning of the 1980s [29,30], has had a major intellectual and practical influence on formal verification [24]. Several verification tools have incorporated ideas from Rely-Guarantee [47,17], whose application success has been further boosted by their combination with ideas from Separation Logic [49,57,27] which started with the work of Vafeiadis and Parkinson [67,66] on the "marriage" between the two logics.

Here we advocate another useful marriage, namely basing Rely-Guarantee on a coinductive foundation, instead of (or in addition to) its traditional inductive foundation (§3). While the first rigorous semantics for Rely-Guarantee, introduced by Xu et al. [70], employs a heavy inductive definition on interactive computation traces, in subsequent developments this account was simplified to use rely-directed reachability relations [14], and then further simplified to use counting-based inductive safety predicates [67,66]—allowing for increasingly simpler and more transparent proofs of soundness for the Rely-Guarantee rules (§3.1).

Placing ourselves at the end of this simplification spectrum, we further push its boundaries: We show that an induction-to-coinduction shift in foundation can be performed by amending the counting-based semantics (§3.2). We argue that the coinductive semantics is the lightest and in some sense the most natural, and caters for the most direct (though admittedly less elementary) proofs of rule soundness. Furthermore, we formally prove all four semantics to be equivalent.

Then we engage in a proof-centered exploration of what makes the above shift possible: the sound inductive approximation of coinductive predicates, emerging from the properties of fixpoints in lattices via the Knaster-Tarski and Kleene theorems (§4). From our concrete analysis of inductive versus coinductive proofs of soundness for the Rely-Guarantee rules, we extract patterns of proof connections between inductive and coinductive predicates (§5). These allow us to explain the inductive-to-coinductive simplification in general terms, revealing the coinductive proof as the *core* of the inductive one (§5.1). Going deeper on the connection between coinduction and its approximating induction, we identify the inductive counterparts of coinductive "up-to" enhancements (§5.2).

Further exploring the scope of the inductive approximations of coinduction (§6), we formalize format-based criteria (§6.1, §6.3). These are easier to instantiate and to implement in program verifiers, and also cover the cases of (bi)similarity relations from process algebra (§6.2), and of Rely-Guarantee mixed with Separation Logic (§6.4). Capturing bisimilarities and Rely-Guarantee under a common format is particularly desirable, given that what we propose for Rely-Guarantee (switching from a well-established inductive to a coinductive notion) has been widely applied *in reverse* with bisimilarities, namely using induction to approximate a well-established coinductive notion.

Our findings have been driven by our experience of mechanizing in Isabelle the semantics of Rely-Guarantee while trying to simplify the "formalities" (§7). We take inspiration from, and create bridges between previous developments on Rely-Guarantee semantics, rule systems, and coinduction enhancements (§8).

Here is a summary of the novel contributions of this paper. On Rely-Guarantee, we give (1) the first coinductive semantics, and also (2) the first rigorous (and formal) proof of the equivalence of previously introduced inductive semantics. On inductive approximations for coinduction, we provide (3) a detailed comparison of inductive and coinductive proofs that starts with the Rely-Guarantee case study and extrapolates to general proof connections for least and greatest fixpoints of operators, (4) a novel inductive counterpart of coinductive up-to enhancements, shedding some "elementary" light on these enhancements, and (5) a formalization of general effective criteria that ensure ω-continuity (hence the soundness of the inductive approximations).

Notations and conventions. Bool denotes the two-element set of booleans, $\{\mathsf{true}, \mathsf{false}\}$. A predicate on a set A is a function of type $A \to \mathsf{Bool}$, and a relation between sets A and B is a function of type $A \to B \to \mathsf{Bool}$. For a predicate $P : A \to \mathsf{Bool}$ and $a \in A$, we write $P\ a$ instead of $P\ a = \mathsf{true}$, and read it as "P holds for a"; and similarly for relations. We use "RG" as an abbreviation for "Rely-Guarantee".

2 The Rely-Guarantee Reasoning Rules

We consider a simple imperative language with parallel composition whose syntax is given by the following grammar:

$$\begin{array}{rcl} \mathsf{Com} & ::= & \mathsf{done} \mid \mathsf{Atom} \mid \mathsf{seq\ Com\ Com} \mid \mathsf{if\ Test\ Com\ Com} \mid \\ & & \mathsf{while\ Test\ Com} \mid \mathsf{par\ Com\ Com} \end{array}$$

Thus, a command is either done (completed), an atom (atomic command), a sequential composition, an if-branching over a test, a while loop again regulated by a test, or the parallel composition of two commands. We leave the tests and atoms abstract. The small-step semantics, shown in Fig. 1, is quite standard. It indicates how command-state pairs, which we call *configurations*, evolve during a single execution step. Thus, the semantics is a (binary) relation $\Rightarrow$ between configurations. While the syntax is parameterized by the sets Test and Atom of tests and atoms respectively, the semantics is further parameterized by a set State of states, and by functions $\mathsf{evalT} : \mathsf{Test} \to \mathsf{State} \to \mathsf{Bool}$ and $\mathsf{evalA} : \mathsf{Atom} \to \mathsf{State} \to \mathsf{State} \to \mathsf{State}$ that evaluate tests and commands in any given state. Thus, given a state s and a test t, $\mathsf{evalT}\ t\ s$ returns the boolean value obtained by evaluating t in s. Similarly, given an atom a and two states s, s', $\mathsf{evalA}\ a\ s\ s'$ says that evaluating a starting in s can produce s'. The while loops proceed by unfolding into a conditional that nests a sequential composition, and parallel composition proceeds via shared-state concurrency.

$$\frac{s' \in \mathsf{evalA}\ a\ s}{(a, s) \Rightarrow (\mathsf{done}, s')}(\mathsf{Atom}) \qquad \frac{(c_1, s) \Rightarrow (c_1', s')}{(\mathsf{seq}\ c_1\ c_2, s) \Rightarrow (\mathsf{seq}\ c_1'\ c_2, s')}(\mathsf{Seq})$$

$$(\mathsf{seq\ done}\ c, s) \Rightarrow (c, s) \quad (\mathsf{Seq\text{-}Done})$$

$$\frac{\mathsf{evalT}\ t\ s = \mathsf{True}}{(\mathsf{if}\ t\ c_1\ c_2, s) \Rightarrow (c_1, s)}(\mathsf{If\text{-}True}) \qquad \frac{\mathsf{evalT}\ t\ s = \mathsf{False}}{(\mathsf{if}\ t\ c_1\ c_2, s) \Rightarrow (c_2, s)}(\mathsf{If\text{-}False})$$

$$(\mathsf{while}\ t\ c, s) \Rightarrow (\mathsf{if}\ t\ (\mathsf{seq}\ c\ (\mathsf{while}\ t\ c))\ \mathsf{done},\ s) \quad (\mathsf{While})$$

$$\frac{(c_1, s) \Rightarrow (c_1', s')}{(\mathsf{par}\ c_1\ c_2, s) \Rightarrow (\mathsf{par}\ c_1'\ c_2, s')}(\mathsf{Par\text{-}L}) \qquad \frac{(c_2, s) \Rightarrow (c_2', s')}{(\mathsf{par}\ c_1\ c_2, s) \Rightarrow (\mathsf{par}\ c_1\ c_2', s')}(\mathsf{Par\text{-}R})$$

$$(\mathsf{par\ done\ done}, s) \Rightarrow (\mathsf{done}, s) \quad (\mathsf{Par\text{-}Done})$$

Fig. 1: Small-step operational semantics

A *Rely-Guarantee (RG) clause* is a tuple (P, R, G, Q) where $P : \mathsf{State} \to \mathsf{Bool}$ is the *pre-condition*, $R : \mathsf{State} \to \mathsf{State} \to \mathsf{Bool}$ the *rely-condition*, $G : \mathsf{State} \to \mathsf{State} \to \mathsf{Bool}$ the *guarantee-condition*, and $Q : \mathsf{State} \to \mathsf{Bool}$ the *post-condition*. Fig. 2 shows a standard RG proof system for reasoning compositionally about programs. Besides the monotonicity rule (Mono), we have one rule for each language construct.

$$\frac{c \vdash (P', R', G', Q') \qquad P \leq P' \qquad R \leq R' \qquad G' \leq G \qquad Q' \leq Q}{c \vdash (P, R, G, Q)}\,(\mathsf{Mono})$$

$$\frac{\mathsf{stable}\ Q\ R \qquad P \leq Q}{\mathsf{done} \vdash (P, R, G, Q)}\,(\mathsf{DoneRG})$$

$$\frac{\begin{array}{c}\mathsf{stable}\ P\ R \qquad \mathsf{stable}\ Q\ R \\ (\lambda s'.\ \exists s.\ P\ s \wedge \mathsf{evalA}\ a\ s\ s') \leq Q \qquad (\lambda s, s'.\ P\ s \wedge \mathsf{evalA}\ a\ s\ s') \leq G\end{array}}{a \vdash (P, R, G, Q)}\,(\mathsf{AtomRG})$$

$$\frac{c_1 \vdash (P, R, G, P') \qquad c_2 \vdash (P', R, G, Q) \qquad \mathsf{refl}\ G}{\mathsf{seq}\ c_1\ c_2 \vdash (P, R, G, Q)}\,(\mathsf{SeqRG})$$

$$\frac{\begin{array}{c}c_1 \vdash (P \sqcap (\mathsf{evalT}\ t), R, G, Q) \qquad c_2 \vdash (P \sqcap (\neg\ (\mathsf{evalT}\ t)), R, G, Q) \\ \mathsf{stable}\ P\ R \qquad \mathsf{refl}\ G\end{array}}{\mathsf{if}\ t\ c_1\ c_2 \vdash (P, R, G, Q)}\,(\mathsf{IfRG})$$

$$\frac{\begin{array}{c}c \vdash (P \sqcap (\mathsf{evalT}\ t), R, G, P) \qquad P \sqcap (\neg\ (\mathsf{evalT}\ t)) \leq Q \\ \mathsf{stable}\ P\ R \qquad \mathsf{stable}\ Q\ R \qquad \mathsf{refl}\ G\end{array}}{\mathsf{while}\ t\ c \vdash (P, R, G, Q)}\,(\mathsf{WhileRG})$$

$$\frac{\begin{array}{c}c_1 \vdash (P_1, R_1, G_1, Q_1) \qquad c_2 \vdash (P_2, R_2, G_2, Q_2) \\ P \leq P_1 \sqcap P_2 \qquad R \sqcup G_2 \leq R_1 \qquad R \sqcup G_1 \leq R_2 \\ G_1 \sqcup G_2 \leq G \qquad Q_1 \sqcap Q_2 \leq Q \qquad \mathsf{refl}\ G\end{array}}{\mathsf{par}\ c_1\ c_2 \vdash (P, R, G, Q)}\,(\mathsf{ParRG})$$

Fig. 2: RG proof system. We write $\mathsf{refl}\ G$ to express that the relation G is reflexive. $\leq$ denotes the standard orders on predicates and relations, e.g., $P \leq P'$ means $\forall s.\ P\ s \longrightarrow P'\ s$, and similarly for relations. $\sqcap$ and $\sqcup$ denote the infimum and supremum in the lattices of predicates and relations, which are component-wise conjunction and disjunction. $\mathsf{stable}\ P\ R$ says the predicate P is stable w.r.t. the relation R, i.e., $\forall s, s'.\ P\ s \wedge R\ s\ s' \longrightarrow P\ s'$.

3 Four Ways of Making Sense of Rely-Guarantee

In this section we revisit three increasingly light inductive semantics for RG (§3.1), establish their equivalence, and propose our coinductive semantics further up on the lightness spectrum (§3.2).

A configuration (c, s) is *final*, written $\mathsf{final}\,(c, s)$, when there is no (c', s') such that $(c, s) \Rightarrow (c', s')$ (equivalently, when $c = \mathsf{done}$).

3.1 Three inductive ways

We next describe the three main semantics previously proposed for RG.

(i) The trace-based semantics of Xu, de Roever and He [70] was the first rigorous semantics proposed for RG. It uses *labeled configurations*, which are triples (l, c, s) such that (c, s) is a configuration and l is an element of the two-element set $\{\mathsf{C}, \mathsf{E}\}$, where C indicates a command step and E an environment step. An *(interactive computation) trace* is a nonempty list of labeled configurations $[(l^1, c^1, s^1), \ldots, (l^n, c^n, s^n)]$ such that, for all $i \in \{1, \ldots, n-1\}$, $l^{i+1} = \mathsf{C}$ implies $(c^i, s^i) \Rightarrow (c^{i+1}, s^{i+1})$, and $l^{i+1} = \mathsf{E}$ implies $c^{i+1} = c^i$. We write $\mathsf{Trace}(c)$ for the set of traces $[(l^1, c^1, s^1), \ldots, (l^n, c^n, s^n)]$ starting in c, i.e., such that $c^1 = c$.

The *Xu/de-Roever/He satisfaction* of an RG clause by a command, $c \vDash_{\mathsf{XRH}} (P, R, G, Q)$, is defined as follows: For all $tr = [(l^1, c^1, s^1), \ldots, (l^n, c^n, s^n)] \in \mathsf{Trace}(c)$, if $P\ s^1$ (the pre-condition holds at the beginning of tr) and $\forall i \in \{1, \ldots, n-1\}.\ l^{i+1} = \mathsf{E} \longrightarrow R\ s^i\ s^{i+1}$ (the rely-condition holds on all environment steps), then $\forall i \in \{1, \ldots, n-1\}.\ l^{i+1} = \mathsf{C} \longrightarrow G\ s^i\ s^{i+1}$ (the guarantee-condition holds on all computation steps) and $Q\ s^n$ (the post-condition holds at the end).

(ii) The reachability-based semantics of Coleman and Jones [14]. For a relation R on configurations, we define $\mathsf{stepRel}_R$ by the rules:

$$\frac{(c, s) \Rightarrow (c', s')}{\mathsf{stepRel}_R\,(c, s)\,(c', s')} \qquad\qquad \frac{R\ s\ s'}{\mathsf{stepRel}_R\,(c, s)\,(c, s')}$$

The reflexive-transitive closure of this relation, $\mathsf{stepRel}^*_R\,(c, s)\,(c', s')$ says that the configuration (c', s') is reachable from (c, s) through a combination of computation steps, and environment steps respecting the rely relation R. The *Coleman/Jones satisfaction* of an RG clause, $c \vDash_{\mathsf{CJ}} (P, R, G, Q)$, is defined as follows: For all s, c', s' such that $P\ s$ and $\mathsf{stepRel}^*_R\,(c, s)\,(c', s')$, we have that **(1)** $G\ s'\ s''$ for all c'', s'' with $(c', s') \Rightarrow (c'', s'')$, and **(2)** $\mathsf{final}\,(c', s')$ implies $Q\ s'$.

(iii) The counting-based (step-indexed) semantics of Vafeiadis and Parkinson [66,67]. This semantics uses a count of the number of interactive computation steps, and states that any number of such steps is "safe". While RG clauses are quadruples (P, R, G, Q), let us call any triple of the form (R, G, Q) a *reduced RG clause* (where the pre-condition was removed, i.e., retaining only the rely-, guarantee- and post-conditions). The notion of *safety* of a configuration (c, s) with respect to a reduced RG clause (R, G, Q) *within n execution steps*, $\mathsf{safe}_{(R,G,Q)}\,n\,(c, s)$, is defined inductively as shown in Fig. 3. Thus, every

$$\mathsf{safe}_{(R,G,Q)}\ 0\ (c,s)\ \ (\mathsf{Base})$$

$$\frac{\begin{array}{l}1.\ \forall s'.\ R\ s\ s' \longrightarrow \mathsf{safe}_{(R,G,Q)}\ n\ (c,s')\\ 2.\ \mathsf{final}\ (c,s)\ \longrightarrow Q\ s\\ 3.\ \forall c', s'.\ ((c,s) \Rightarrow (c',s')) \longrightarrow G\ s\ s' \wedge \mathsf{safe}_{(R,G,Q)}\ n\ (c',s')\end{array}}{\mathsf{safe}_{(R,G,Q)}\ (n+1)\ (c,s)}(\mathsf{Step})$$

Fig. 3: The inductive-safety predicate safe

command in every state is safe after 0 execution steps, since of course nothing happened yet. For the inductive step, the predicate says that the $(n+1)$-safety of (c,s) (i.e., the safety of executing $n+1$ steps of c from state s) can be concluded from three hypotheses: **(1)** any rely-compliant environment step to s' (i.e., such that $R\ s\ s'$ produces an n-safe configuration (c,s')); **(2)** the post-condition holds (i.e., $Q\ s$) in case (c,s) is final. **(3)** any computation step $(c,s) \Rightarrow (c',s')$ produces an n-safe configuration (c,s') along a guarantee-compliant state change (i.e., such that $G\ s\ s'$).

The *Vafeiadis-Parkinson satisfaction* of an RG clause, $c \vDash_{\mathsf{VP}} (P,R,G,Q)$, is now defined to mean safety for any number of steps, $\forall s \in \mathsf{State},\ \forall n \in \mathbb{N}.\ P\ s \longrightarrow \mathsf{safe}_{(R,G,Q)}\ n\ (c,s)$.

We can show that these three semantics are all equivalent, and the RG rules are sound for them.

Thm 1. Consider the following statements:

(1) $c \vDash_{\mathsf{XRH}} (P,R,G,Q)$; (2) $c \vDash_{\mathsf{CJ}} (P,R,G,Q)$;
(3) $c \vDash_{\mathsf{VP}} (P,R,G,Q)$; (4) $c \vdash (P,R,G,Q)$.

Then (1), (2) and (3) are equivalent, and (4) implies them. Moreover, the equivalences between (1), (2) and (3) are language-independent, i.e., they still hold if we replace the small-step operational semantics of our language with any sets Com and State and relation $\Rightarrow : (\mathsf{Com} \times \mathsf{State}) \to (\mathsf{Com} \times \mathsf{State}) \to \mathsf{Bool}$.

Although equivalent, the three semantics differ in the heaviness of their inductive machinery:

- $\vDash_{\mathsf{XRH}}$ talks about long-distance action, i.e., quantifies over multiple execution steps, via explicit computation traces.
- $\vDash_{\mathsf{CJ}}$ still talks about long-distance action, but keeps the traces implicit, abstracted under the notion of reachability.
- $\vDash_{\mathsf{VP}}$ goes further, abstracting the (long-)distance into a numeric argument n, and only talks about single steps, n to $n+1$.

These differences are reflected in the level of formal bureaucracy involved in the proofs of RG rules soundness (Thm. 1(4)). Take for example the soundness of the rule for sequential composition, (SeqRG) in Fig. 2:

- For $\vDash_{\mathsf{XRH}}$, where we need to prove that $c_1 \vDash_{\mathsf{XRH}} (P,R,G,P')$, $c_2 \vDash_{\mathsf{XRH}} (P',R,G,Q)$ and $\mathsf{refl}\ G$ implies $\mathsf{seq}\ c_1\ c_2 \vDash_{\mathsf{XRH}} (P,R,G,Q)$, we require a lemma characterizing the traces that start in $\mathsf{seq}\ c_1\ c_2$, stating that such a trace:

$$\frac{\begin{array}{l}1.\ \forall s'.\ R\ s\ s' \longrightarrow \mathsf{safeC}\ (c, s')\ (R, G, Q)\\ 2.\ \mathsf{final}\ (c, s)\ \longrightarrow Q\ s\\ 3.\ \forall c', s'.\ ((c, s) \Rightarrow (c', s')) \longrightarrow G\ s\ s' \wedge \mathsf{safeC}\ (c', s')\ (R, G, Q)\end{array}}{\mathsf{safeC}_{(R,G,Q)}\ (c, s)}\ (\mathsf{StepC})$$

Fig. 4: The coinductive-safety predicate safeC

- either is obtained by wrapping $\mathsf{seq}\ \_\ c_2$ around a trace starting in c_1,
- or consists of a trace obtained by wrapping $\mathsf{seq}\ \_\ c_2$ around a trace starting in c_1 and ending in done, followed by a trace starting in c_2.

– For $\vDash_{\mathsf{CJ}}$ (where we must prove the same as above but with $\vDash_{\mathsf{CJ}}$), we require a lighter inversion lemma for multistep reachability: stating that, if $\mathsf{stepRel}^*_R(\mathsf{seq}\ c_1\ c_2, s)(c', s')$, then
  - either c' has the form $\mathsf{seq}\ c_1'\ c_2$ for some c_1' such that $\mathsf{stepRel}^*_R\ (c_1, s)\ (c_1', s')$,
  - or $\mathsf{stepRel}^*_R\ (c_1, s)\ (\mathsf{done}, s'')$ for some s'', and $\mathsf{stepRel}^*_R\ (c_2, s'')\ (c', s')$.
– Finally, for $\vDash_{\mathsf{VP}}$, we only require the native inversion lemma stemming from the inductive definition of the small-step semantics: stating that, if $(\mathsf{seq}\ c_1\ c_2, s) \Rightarrow (c', s')$, then
  - either c' has the form $\mathsf{seq}\ c_1'\ c_2$ for some c_1' such that $(c_1, s) \Rightarrow (c_1', s')$,
  - or $(c_1, s) \Rightarrow (\mathsf{done}, s'')$ for some s'', and $(c_2, s'') \Rightarrow (c', s')$.

So, proof bureaucracy decreases as we move along the above spectrum: from trace decomposition, to inversion for multi-steps, to inversion for single steps.

3.2 A fourth way: coinductive-safety semantics

Next, we discuss a further simplification from the lightest of the above semantics, the counting-based one of Vafeiadis and Parkinson. Namely, taking advantage of the implicit (co)iterative nature of coinduction, we remove the explicit counting of the number n of steps from Fig. 3's inductive-safety predicate safe, and retain only the (Step) case. This leads us to the coinductive-safety predicate safeC defined by the rule schema in Fig. 4. The definition is now interpreted not inductively but coinductively [52,51,32], as indicated by a double-line in the rule. And the *coinductive satisfaction* of an RG clause by a command, $c \vDash_{\mathsf{C}} (P, R, G, Q)$, is defined as $\forall s \in \mathsf{State}.\ P\ s \longrightarrow \mathsf{safeC}_{(R,G,Q)}\ (c, s)$ (so similarly to how the counting-based semantics $\vDash_{\mathsf{VP}}$ is defined from safe, but of course without the numeric index).

The coinductive semantics is equivalent to the counting-based one (hence, according to Thm. 1, to all three inductive semantics).

Thm 2. $c \vDash_{\mathsf{VP}} (P, R, G, Q)$ is equivalent to $c \vDash_{\mathsf{C}} (P, R, G, Q)$. Moreover, this equivalence is again language-independent (in the sense defined in Thm. 1).

The proof of the theorem relies on a lemma about safe versus safeC.

Lemma 3. The following are equivalent:
(i) $\forall n \in \mathbb{N}.\ \mathsf{safe}_{(R,G,Q)}\ n\ (c, s)$; (ii) $\mathsf{safeC}_{(R,G,Q)}\ (c, s)$.

The lemma says that a $\mathbb{N}$-indexed family of inductive predicates, $\mathsf{safe}_{(R,G,Q)}$, forms a convergent approximation of a coinductive predicate, $\mathsf{safeC}_{(R,G,Q)}$.

4 Coinduction Approximated Inductively in General

Next we recall the abstract phenomenon behind the inductive approximation of coinduction—as background for our proof-based investigation to follow.

Let $(L, \leq)$ be a complete lattice, where we write $\sqcap$ and $\sqcup$ for the binary infima and suprema, and $\bigsqcap$ and $\bigsqcup$ for infima and suprema of arbitrary families of elements in L. A *decreasing ω-chain* in L is a family $(l_i)_{i \in \mathbb{N}}$ such that $l_i \geq l_{i+1}$ for all $i \in \mathbb{N}$. Let $F : L \to L$ be a monotonic operator. An element $k \in L$ is said to be a *fixpoint* for F if $F\ k = k$, a *pre-fixpoint* for F if $F\ k \leq k$, and a *post-fixpoint* for F if $k \leq F\ k$. We recall the Knaster-Tarski theorem [64].

Thm 4. If L is a complete lattice and $F : L \to L$ a monotonic operator, then:
(1) There exists a unique least fixpoint lfp_F for F, which is also the least pre-fixpoint. **(2)** Dually, there exists a unique greatest fixpoint gfp_F for F, which is also the greatest post-fixpoint.

This enables an induction proof principle for lfp_F: to show $\mathsf{lfp}_F \leq k$, it suffices to show that k is a pre-fixpoint. Dually, we have a coinduction proof principle: to show $k \leq \mathsf{gfp}_F$, it suffices to show that k is a post-fixpoint. A small enhancement called *strong (co)induction* includes lfp_F and gfp_F too in the (co)inductive proofs:

Corollary 5. Under the hypotheses of Thm. 4:
(1) $F\ (\mathsf{lfp}_F \sqcap k) \leq k$ implies $\mathsf{lfp}_F \leq k$ for all $k \in L$, and
(2) $k \leq F\ (\mathsf{gfp}_F \sqcup k) \leq k$ implies $k \leq \mathsf{gfp}_F$ for all $k \in L$.

Example 6. The inductive definition of $\mathsf{safe}_{(R,G,Q)}$ comes from applying the Knaster-Tarski theorem for least fixpoints. Namely, fixing (R, G, Q), we have that $\mathsf{safe}_{(R,G,Q)}$ is the least fixpoint of an operator on the lattice $\mathbb{N} \to (\mathsf{Com} \times \mathsf{State}) \to \mathsf{Bool}$, temporarily refer to it as H, defined by taking $H\ K\ n\ (c, s)$ to be

$$\begin{array}{l} n = 0\ \vee\ (\exists m.\ n = m + 1\ \wedge \\ \qquad (\mathsf{final}\ (c, s) \longrightarrow Q\ s)\ \wedge \\ \qquad (\forall c', s'.\ ((c, s) \Rightarrow (c', s')) \longrightarrow G\ s\ s' \wedge K\ (c', s')\ m)\ \wedge \\ \qquad (\forall s'.\ R\ s\ s' \longrightarrow K\ (c, s')\ m)) \end{array}$$

And the coinductive definition of $\mathsf{safeC}_{(R,G,Q)}$ comes from applying Knaster-Tarski for greatest fixpoints: $\mathsf{safeC}_{(R,G,Q)} = \mathsf{gfp}_F$ where the lattice L is $(\mathsf{Com} \times \mathsf{State}) \to \mathsf{Bool}$, and $F : L \to L$ is defined by taking $F\ K\ (c, s)$ to be

$$\begin{array}{l} (\mathsf{final}\ (c, s) \longrightarrow Q\ s)\ \wedge \\ (\forall c', s'.\ ((c, s) \Rightarrow (c', s')) \longrightarrow G\ s\ s' \wedge K\ (c', s'))\ \wedge \\ (\forall s'.\ R\ s\ s' \longrightarrow K\ (c, s')) \end{array}$$

We will denote by $\mathsf{L}_{(R,G,Q)}$ and $\mathsf{F}_{(R,G,Q)}$ the above particular lattice L and particular operator F underlying the coinductive definition of $\mathsf{safeC}_{(R,G,Q)}$. $\square$

To capture abstractly the connection between safeC and safe we need a more fine-grained description of greatest fixpoints, via upper approximations. This is offered by Kleene's theorem [15], which we recall next (in the dual form, targeting

greatest fixpoints). We say that $F : L \to L$ is *ω-cocontinuous* if it commutes with the infima of decreasing ω-chains $(a_i)_{i\in\mathbb{N}}$, in that $F\,(\bigsqcap_{i\in\mathbb{N}} a_i) = \bigsqcap_{i\in\mathbb{N}}(F\,a_i)$. Note that ω-cocontinuity is a strengthening of monotonicity. We write F^n for the n'th iteration of F, i.e., the composition with itself n times.

Thm 7. (Kleene's Theorem) If L is a complete lattice (or at least a copointed ω-cocomplete lattice) where $\top$ denotes its top element and the operator $F : L \to L$ is ω-cocontinuous, then $\mathsf{gfp}_F = \bigsqcap_{n\in\mathbb{N}} F^n \top$.

We recall the proof idea for future reference: The ω-cocontinuity of F ensures that $\bigsqcap_{n\in\mathbb{N}} F^n \top$ is a fixpoint. To show that it is the greatest, let k be a fixpoint. By induction on n it follows that $\forall n.\, k \le F^n \top$, i.e., $k \le \bigsqcap_{n\in\mathbb{N}} F^n \top$. In the inductive step, $k \le F^n \top$ implies, also using F's monotonicity, that $k = F\,k \le F\,(F^n \top) = F^{n+1} \top$. □

For any operator $F : L \to L$, we define its *approximating operator* $F^\sharp : (\mathbb{N} \to L) \to (\mathbb{N} \to L)$ as follows, for all $f : \mathbb{N} \to L$ and $n \in \mathbb{N}$:

$$F^\sharp\, f\, n = \begin{cases} \top, & \text{if } n = 0 \\ F\,(f\,(n-1)) & \text{otherwise} \end{cases}$$

The definition of $F^\sharp$ from F represents the pattern of how $\mathsf{safe}_{(R,G,Q)}$ *could have been* obtained from $\mathsf{safeC}_{(R,G,Q)}$—this is because $\mathsf{safeC}_{(R,G,Q)}$ is $\mathsf{gfp}_{\mathsf{F}_{(R,G,Q)}}$ and $\mathsf{safe}_{(R,G,Q)}$ is $\mathsf{lfp}_{\mathsf{F}^\sharp_{(R,G,Q)}}$. Lemma 3, connecting $\mathsf{safe}_{(R,G,Q)}$ with $\mathsf{safeC}_{(R,G,Q)}$, is therefore an instance of the following consequence of Kleene's theorem:

Thm 8. Assume that L is a complete lattice and $F : L \to L$ is an ω-cocontinuous operator. Then $\mathsf{gfp}_F = \bigsqcap_{n\in\mathbb{N}} \mathsf{lfp}_{F^\sharp} n$.

Thus, our discussed approximation phenomenon is abstractly the coincidence between the greatest fixpoint gfp_F and the infimum of the applications of the approximating operator's fixpoint $\mathsf{lfp}_{F^\sharp}\, n$; and this coincidence is enabled by the ω-cocontinuity of the underlying operator F.

5 Proof Development Perspective

The above discussion shows that, even though we derived $\mathsf{safeC}_{(R,G,Q)}$ from $\mathsf{safe}_{(R,G,Q)}$, it is more natural to see $\mathsf{safe}_{(R,G,Q)}$ as being derived from $\mathsf{safeC}_{(R,G,Q)}$—suggesting that $\mathsf{safeC}_{(R,G,Q)}$ is the simpler of the two. However, one is inductive and the other is coinductive, resulting in different styles of proofs, which we will compare next. We start with a concrete comparison, taking as a case study the soundness of the rule for sequential composition, then we extract general patterns in terms of the abstract operators $F^\sharp$ versus F (§5.1). Then we compare inductive with coinductive enhancements, starting with the soundness proofs of the RG rule for while (susceptible to "up-to" enhancement due to its reliance on other constructs), and again generalizing to abstract operators (§5.2).

Preliminaries

We assume $\mathsf{refl}\ G$, $c_1 \vDash_{\mathsf{VP}} (P, R, G, P')$ and $c_2 \vDash_{\mathsf{VP}} (P', R, G, Q)$ and must show $\mathsf{seq}\ c_1\ c_2 \vDash_{\mathsf{VP}} (P, R, G, Q)$. Expanding the definition of $\vDash_{\mathsf{VP}}$ in the first assumption and the conclusion, the goal is reduced to the following, for all s and n:

Main part

We assume **(1)** $\mathsf{refl}\ G$ and **(2)** $\forall s', m.\ P'\ s' \longrightarrow \mathsf{safe}_{(R,G,Q)}\ m\ (c_2, s')$, and must prove $\forall c_1, s.\ (\forall m.\ \mathsf{safe}_{(R,G,P')}\ m\ (c_1, s)) \longrightarrow \mathsf{safe}_{(R,G,Q)}\ n\ (\mathsf{seq}\ c_1\ c_2, s)$, which we do by induction on n.

The base case is trivial, since the conclusion $\mathsf{safe}_{(R,G,Q)}\ 0$ is vacuously true.

For the induction step, we assume

(IH) $\forall c_1, s.\ (\forall m.\ \mathsf{safe}_{(R,G,P')}\ m\ (c_1, s)) \longrightarrow \mathsf{safe}_{(R,G,Q)}\ n\ (\mathsf{seq}\ c_1\ c_2, s)$

and must prove

$\forall c_1, s.\ (\forall m.\ \mathsf{safe}_{(R,G,P')}\ m\ (c_1, s)) \longrightarrow \mathsf{safe}_{(R,G,Q)}\ (n+1)\ (\mathsf{seq}\ c_1\ c_2, s)$.

To this end, we fix c_1 and s, and assume $\forall m.\ \mathsf{safe}_{(R,G,P')}\ m\ (c_1, s)$, in particular **(3)** $\forall m.\ \mathsf{safe}_{(R,G,P')}\ (m+1)\ (c_1, s)$, which by the definition of $\mathsf{safe}_{(R,G,P')}$ yields the following:

(3.i) $\forall m.\ \forall s'.\ R\ s\ s' \longrightarrow \mathsf{safe}_{(R,G,P')}\ m\ (c_1, s')$
(3.ii) $\mathsf{final}\ (c_1, s) \longrightarrow P'\ s$
(3.iii) $\forall m.\ \forall c_1', s'.\ ((c_1, s) \Rightarrow (c_1', s')) \longrightarrow G\ s\ s' \wedge \mathsf{safe}_{(R,G,P')}\ m\ (c_1', s')$

We must prove $\mathsf{safe}_{(R,G,Q)}\ (n+1)\ (\mathsf{seq}\ c_1\ c_2, s)$, which by the definition of $\mathsf{safe}_{(R,G,Q)}$ amounts to:

(i) $\forall s'.\ R\ s\ s' \longrightarrow \mathsf{safe}_{(R,G,Q)}\ n\ (\mathsf{seq}\ c_1\ c_2, s')$
(ii) $\mathsf{final}\ (\mathsf{seq}\ c_1\ c_2, s) \longrightarrow Q\ s$
(iii) $\forall c', s'.\ ((\mathsf{seq}\ c_1\ c_2, s) \Rightarrow (c', s')) \longrightarrow G\ s\ s' \wedge \mathsf{safe}_{(R,G,Q)}\ n\ (c', s')$

Now, (i) follows from (3.i) and (IH), and (ii) follows by virtue of $\mathsf{final}\ (\mathsf{seq}\ c_1\ c_2, s)$ being false. To prove (iii), fix c' and s' and assume $(\mathsf{seq}\ c_1\ c_2, s) \Rightarrow (c', s')$. We must prove $G\ s\ s' \wedge \mathsf{safe}_{(R,G,Q)}\ n\ (c', s')$. We have two cases:

Case I: $c_1 = \mathsf{done}$. Then $(c', s') = (c_2, s)$, hence $G\ s\ s'$, i.e., $G\ s\ s$, follows from (1). Moreover, since $\mathsf{final}\ (c_1, s)$ holds, from (3.ii) we obtain $P'\ s$, i.e., $P'\ s'$. This together with (2) implies $\mathsf{safe}_{(R,G,Q)}\ n\ (c_2, s')$, i.e., $\mathsf{safe}_{(R,G,Q)}\ n\ (c', s')$.

Case II: $c_1 \neq \mathsf{done}$. Then there exists c_1' such that $(c_1, s) \Rightarrow (c_1', s')$ and $c' = \mathsf{seq}\ c_1'\ c_2$, which together with (3.iii) implies $G\ s\ s'$ (as desired) and $\forall m.\ \mathsf{safe}_{(R,G,P')}\ m\ (c_1', s')$. Using the latter and (IH), we obtain $\mathsf{safe}_{(R,G,Q)}\ n\ (\mathsf{seq}\ c_1'\ c_2, s')$, i.e., $\mathsf{safe}_{(R,G,Q)}\ n\ (c', s')\ (R, G, Q)$. □

(a) For the counting-based semantics

Preliminaries

We assume $\mathsf{refl}\ G$, $c_1 \vDash_{\mathsf{C}} (P, R, G, P')$ and $c_2 \vDash_{\mathsf{C}} (P', R, G, Q)$ and must show $\mathsf{seq}\ c_1\ c_2 \vDash_{\mathsf{C}} (P, R, G, Q)$. Expanding the definition of $\vDash_{\mathsf{C}}$ in the first assumption and the conclusion, the goal is reduced to the following, for all s:

Main part

We assume **(1)** $\mathsf{refl}\ G$ and **(2)** $\forall s'.\ P'\ s' \longrightarrow \mathsf{safeC}_{(R,G,Q)}\ (c_2, s')$, and must prove $(\exists c_1.\ c = \mathsf{seq}\ c_1\ c_2 \wedge \mathsf{safeC}_{(R,G,P')}(c_1, s)) \longrightarrow \mathsf{safeC}_{(R,G,Q)}\ (c, s)$. We prove this by strong coinduction on the definition of $\mathsf{safeC}_{(R,G,Q)}$. Thus, we assume the coinductive invariant $\exists c_1.\ c = \mathsf{seq}\ c_1\ c_2 \wedge \mathsf{safeC}_{(R,G,P')}(c_1, s)$, which amounts to fixing c_1 and assuming $c = \mathsf{seq}\ c_1\ c_2$ and **(3)** $\mathsf{safeC}_{(R,G,P')}\ (c_1, s)$, the latter yielding the following from the definition of $\mathsf{safeC}_{(R,G,P')}$:

(3.i) $\forall s'.\ R\ s\ s' \longrightarrow \mathsf{safeC}_{(R,G,P')}\ (c_1, s')$
(3.ii) $\mathsf{final}\ (c_1, s) \longrightarrow P'\ s$
(3.iii) $\forall c_1', s'.\ ((c_1, s) \Rightarrow (c', s')) \longrightarrow G\ s\ s' \wedge \mathsf{safeC}_{(R,G,P')}\ (c_1', s')$

We must show the following:

(i) $\forall s'.\ R\ s\ s' \longrightarrow \mathsf{safeC}_{(R,G,Q)}\ (\mathsf{seq}\ c_1\ c_2, s') \vee (\exists c_1'.\ \mathsf{seq}\ c_1\ c_2 = \mathsf{seq}\ c_1'\ c_2 \wedge \mathsf{safeC}_{(R,G,P')}(c_1', s))$
(ii) $\mathsf{final}\ (\mathsf{seq}\ c_1\ c_2, s) \longrightarrow Q\ s$
(iii) $\forall c', s'.\ ((\mathsf{seq}\ c_1\ c_2, s) \Rightarrow (c', s')) \longrightarrow G\ s\ s' \wedge (\mathsf{safeC}_{(R,G,Q)}\ (c', s') \vee (\exists c_1'.\ c' = \mathsf{seq}\ c_1'\ c_2 \wedge \mathsf{safeC}_{(R,G,P')}(c_1', s')))$

Now, (i) follows from (3.i) taking $c_1' = c_1$, and (ii) follows by virtue of $\mathsf{final}\ (\mathsf{seq}\ c_1\ c_2, s)$ being false. To prove (iii), fix c' and s' and assume $(\mathsf{seq}\ c_1\ c_2, s) \Rightarrow (c', s')$. We must prove $G\ s\ s' \wedge (*)$ where $(*)$ is $\mathsf{safeC}_{(R,G,Q)}\ (c', s')) \vee (\exists c_1'.\ c' = \mathsf{seq}\ c_1'\ c_2 \wedge \mathsf{safeC}_{(R,G,P')}(c_1', s'))$. We have two cases:

Case I: $c_1 = \mathsf{done}$. Then $(c', s') = (c_2, s)$, hence $G\ s\ s'$, i.e., $G\ s\ s$, follows from (1). Moreover, since $\mathsf{final}\ (c_1, s)$ holds, from (3.ii) we obtain $P'\ s$, i.e., $P'\ s'$. This together with (2) implies $\mathsf{safeC}_{(R,G,Q)}\ (c_2, s')$, i.e., $\mathsf{safeC}_{(R,G,Q)}\ (c', s')$, the first disjunct of $(*)$.

Case II: $c_1 \neq \mathsf{done}$. Then there exists c_1' such that $(c_1, s) \Rightarrow (c_1', s')$ and $c' = \mathsf{seq}\ c_1'\ c_2$, which with (3.iii) implies $G\ s\ s'$ (as desired) and $\mathsf{safeC}_{(R,G,P')}\ (c_1', s')$. Using the latter together with $c' = \mathsf{seq}\ c_1'\ c_2$, we obtain the second disjunct of $(*)$. □

(b) For the coinductive semantics

Fig. 5: Soundness proofs for the RG seq rule

5.1 Inductive versus coinductive proofs

Recall from §3.1 that the counting-based semantics is at the lightweight end of the spectrum of semantics for RG, with consequences on the simplicity of the soundness proofs. Next we illustrate how shifting from this to the coinductive semantics (i.e., from $\vDash_{\mathsf{VP}}$ which is based on safe to $\vDash_{\mathsf{C}}$ which is based on safeC), incurs a further degree of proof simplification. We again consider the rule for sequential composition ((SeqRG) in Fig. 2). The proofs of its soundness w.r.t. the counting-based and coinductive semantics are shown in Fig. 5, side by side. We have arranged the layout and the notations to emphasize the similarity between their structures. Ignoring the specific formats required for starting (co)induction (namely the universal quantification for induction and the existential one for coinduction) and the vacuous base case present for induction only, the two proofs consist of very similar "face-offs" between what we know, labeled (3.i)–(3.iii), and what we must prove, labeled (i)–(iii). The differences are highlighted in gray in the figure:

- in the inductive proof, in addition to (3.i)–(3.iii) we also know (and of course need) the inductive hypothesis (IH);
- in the coinductive proof, in the goals (i) and (iii) we have the extra slack offered by a disjunct with the coinductive invariant.

The coinductive proof is seen to be conceptually more direct. Indeed, for inferring (i)–(iii) from (3.i)–(3.iii), the inductive version involves some back and forth between the premise and the conclusion of the fact to be proved, mediated by the inductive hypothesis. For example, when proving (iii) while in Case II, the inductive argument obtains $\forall m.\ \mathsf{safe}_{(R,G,P')}\ m\ (c_1', s')$ which is fed as the premise of the inductive hypothesis to produce the desired $\mathsf{safe}_{(R,G,Q)}\ n\ (\mathsf{seq}\ c_1'\ c_2, s')$. (Note also that the premise cannot be decoupled from the fact to be proved by induction, because the state in the premise can change during the proof.) By contrast, in the coinductive version one obtains the corresponding fact $\mathsf{safeC}_{(R,G,P')}(c_1', s')$ which is fed directly to (iii) via the coinductive invariant component.

This conceptual simplification offered by the coinductive semantics can be traced back to the abstract operators F and $F^\sharp$, generalizing $\mathsf{safeC}_{(R,G,Q)}$ and $\mathsf{safe}_{(R,G,Q)}$ to gfp_F and $\mathsf{lfp}_{F^\sharp}$. The nontrivial part of an inductive proof of $k \le \mathsf{gfp}_F = (\bigsqcap_{n\in\mathbb{N}} \mathsf{lfp}_{F^\sharp}\ n)$, i.e., of $\forall n \in \mathbb{N}.\ k \le \mathsf{lfp}_{F^\sharp}\ n$, is to show that $k \le \mathsf{lfp}_{F^\sharp}\ n$ implies $k \le \mathsf{lfp}_{F^\sharp}(n+1)$. And oftentimes, in particular for the soundness proofs of all RG rules (including the Fig. 5 one), the reason why this implication holds is that $k \le F\ (\mathsf{gfp}_F \sqcup k)$—exactly what a proof of $k \le \mathsf{gfp}_F$ by strong coinduction entails. Indeed, we infer $k \le \mathsf{lfp}_{F^\sharp}(n+1)$ from $k \le \mathsf{lfp}_{F^\sharp}\ n$ by plugging $k \le F\ (\mathsf{gfp}_F \sqcup k)$ into the following chain of (in)equalities, where we also use that F is monotonic and $\mathsf{gfp}_F \le \mathsf{lfp}_{F^\sharp}(n)$:

$$k \le F\ (\mathsf{gfp}_F \sqcup k) \le F\ (\mathsf{gfp}_F \sqcup \mathsf{lfp}_{F^\sharp}\ n) = F\ (\mathsf{lfp}_{F^\sharp}\ n) = \mathsf{lfp}_{F^\sharp}(n+1)$$

Thus, the coinductive proof corresponds to the core of the inductive proof; and the rest of the proof is boilerplate, essentially repeating the argument for Kleene's theorem (Thm. 7)!

	Safety preservation needed for inductive soundness	Enhanced version
done	$\dfrac{(1)\ Q\ s \qquad (2)\ \mathsf{stable}\ Q\ R}{\forall n.\ \mathsf{safe}_{(R,G,Q)}\ n\ \mathsf{done}}$	None needed, this is just rephrasing
seq	$\dfrac{(1)\ \mathsf{refl}\ G \quad (2)\ \forall m\ .\ \mathsf{safe}_{(R,G,P)}\ m\ (c_1, s) \quad (3)\ \forall s'.\ P\ s' \longrightarrow \forall n.\ \mathsf{safe}_{(R,G,Q)}\ n\ (c_2, s')}{\forall n.\ \mathsf{safe}_{(R,G,Q)}\ n\ (\mathsf{seq}\ c_1\ c_2, s)}$	for all n ... $\dfrac{(1)\ \mathsf{refl}\ G \quad (2)\ \mathsf{safe}_{(R,G,P)}\ n\ (c_1, s) \quad (3)\ \forall s'.\ P\ s' \longrightarrow \mathsf{safe}_{(R,G,Q)}\ n\ (c_2, s')}{\mathsf{safe}_{(R,G,Q)}\ n\ (\mathsf{seq}\ c_1\ c_2, s)}$
if	$\dfrac{\begin{array}{l}(1)\ \mathsf{refl}\ G \\ (2)\ \forall s'.\ R^*\ s\ s'\ \longrightarrow \\ (\mathsf{evalT}\ t\ s' \longrightarrow \forall n.\ \mathsf{safe}_{(R,G,Q)}\ n\ (c_1, s')) \wedge \\ (\neg\ \mathsf{evalT}\ t\ s' \longrightarrow \forall n.\ \mathsf{safe}_{(R,G,Q)}\ n\ (c_2, s'))\end{array}}{\forall n.\ \mathsf{safe}_{(R,G,Q)}\ n\ (\mathsf{seq}\ c_1\ c_2, s)}$	for all n ... $\dfrac{\begin{array}{l}(1)\ \mathsf{refl}\ G \\ (2)\ \forall s'.\ R^*\ s\ s'\ \longrightarrow \\ (\mathsf{evalT}\ t\ s' \longrightarrow \mathsf{safe}_{(R,G,Q)}\ n\ (c_1, s')) \wedge \\ (\neg\ \mathsf{evalT}\ t\ s' \longrightarrow \mathsf{safe}_{(R,G,Q)}\ n\ (c_2, s'))\end{array}}{\mathsf{safe}_{(R,G,Q)}\ n\ (\mathsf{seq}\ c_1\ c_2, s)}$

Fig. 6: Inductive enhancements

So in our concrete example from Fig. 5, gfp_F is $\mathsf{safeC}_{(R,G,Q)}$, $\mathsf{lfp}_{F^\sharp}$ is $\mathsf{safe}_{(R,G,Q)}$ (so F is $\mathsf{F}_{(R,G,Q)}$), and k is expressed by the coinductive invariant. To see more clearly that our abstract discussion actually matches the concrete Fig. 5 situation, we concretize one notch of the abstract discussion, assuming L is $A \to \mathsf{Bool}$ for some set A and $k : A \to \mathsf{Bool}$ is existentially quantified over parameters from a set Prm, i.e., $k\ a = (\exists p.\ k'\ p\ a)$ for some $k' : Prm \to A \to \mathsf{Bool}$. Then:

- The inductive version proves $k \leq \mathsf{gfp}_F = \bigsqcap_n \mathsf{lfp}_{F^\sharp}\ n$, i.e., $\forall n, a.\ k\ a \longrightarrow \mathsf{lfp}_{F^\sharp}\ n\ a$, i.e., $\forall n, a.\ (\exists p.\ k'\ p\ a) \longrightarrow \mathsf{lfp}_{F^\sharp}\ n\ a$, i.e., $\forall n, a, p.\ k'\ p\ a \longrightarrow \mathsf{lfp}_{F^\sharp}\ n\ a$. Therefore, the induction step is: $\forall a, p.\ k'\ p\ a \longrightarrow \mathsf{lfp}_{F^\sharp}\ n\ a$ implies $\forall a, p.\ k'\ p\ a \longrightarrow \mathsf{lfp}_{F^\sharp}(n+1)\ a$.
- The coinductive version proves $k \leq \mathsf{gfp}_F$, i.e., $\forall a.\ k\ a \longrightarrow \mathsf{gfp}_F\ a$, i.e., $\forall a.\ (\exists p.\ k'\ p\ a) \longrightarrow \mathsf{gfp}_F\ a$. Therefore, coinductively, the goal is the following: $\exists p.\ k'\ p\ a$ implies $F\ (\mathsf{gfp}_F \sqcup (\exists p.\ k'\ p))\ a$.

In our example from Fig. 5, A is $\mathsf{Com} \times \mathsf{State}$, Prm is Com, and k' is defined by $k'\ c_1\ (c, s) \longleftrightarrow (c = \mathsf{seq}\ c_1\ c_2 \wedge \mathsf{safeC}_{(R,G,P')}(c_1, s))$.

5.2 Inductive versus coinductive enhancements

As seen in Fig. 5, the soundness of the RG rule for seq w.r.t. the inductive (counting-based) and coinductive semantics amounts to seq preserving the safety predicates. The same is true for the other language constructs. For example, Fig. 6(left) shows the safety preservation properties needed for the soundness of the RG rules for done, seq and if w.r.t. the inductive semantics.

Now let us look at the soundness proofs for the RG rule of the while construct ((WhileRG) from Fig. 2), which is interesting because its operational semantics

relies on other constructs, namely done, seq and if. Fig. 9 shows the soundness proofs relative to the two semantics, side by side. In the inductive case, we end up having to prove the implication shown in Fig. 9(a)'s box, which amounts to $\mathsf{safe}_{(R,G,Q)}\, n\, (\mathsf{while}\ t\ c, s)$ implies $\mathsf{safe}_{(R,G,Q)}\, n\, (\mathsf{if}\ t\ (\mathsf{seq}\ c\ (\mathsf{while}\ t\ c))\ \mathsf{done}, s)$. Here, safety preservation properties for if, seq and done would of course be handy. But the ones from Fig. 6(left) for if and seq, which *were* sufficient for proving the inductive soundness of the rules for these operators, are not strong enough for helping out with the proof for while. This is because now we need to work index-wise with a fixed n. Fortunately, the index-wise versions of these properties, shown in Fig. 6(right), are also provable—for example, we can prove that safety holds for $\mathsf{seq}\ c_1\ c_2$ in the same number of steps n as the assumed safety for c_2. And indeed, as shown in Fig. 9(a) after the box, these stronger, index-wise versions are just what we need to complete the proof for while. In summary, since while depends on other operators, its safety preservation property needed for the soundness of its RG rule required an index-wise enhancement of the safety preservation properties needed for the soundness of these operators. But what is the corresponding phenomenon within the coinductive semantics?

Before answering this question, let us first rephrase and then generalize the preservation properties for inductive safety. All the original preservation properties from Fig. 6(left) can be stated (under side conditions) as

$$U\,(\bigsqcap_n \mathsf{safe}_{R,G,Q}\, n, \ldots, \bigsqcap_n \mathsf{safe}_{R,G,Q}\, n) \;\leq\; \bigsqcap_n \mathsf{safe}_{R,G,Q}\, n$$

where U is an m-ary operator on the lattice $\mathsf{L}_{(R,G,Q)}$ (in our case with $m \in \{0,1,2\}$). Similarly, their index-wise enhancements from Fig. 6(right) can be stated (under the same conditions) as

$$\forall n.\, U(\mathsf{safe}_{R,G,Q}\, n, \ldots, \mathsf{safe}_{R,G,Q}\, n) \leq \mathsf{safe}_{R,G,Q}\, n\,.$$

For example, the operators for done, seq, if are $\mathsf{U}^{\mathsf{done}}_{Q}$, $\mathsf{U}^{\mathsf{seq}}_{(R,G,P,c_2)}$, $\mathsf{U}^{\mathsf{if}}_{(R,t,c_1,c_2)}$, shown in Fig. 7; they were extracted from the soundness preservation properties.

So in general, for an ω-cocomplete operator $F : L \to L$ on a complete lattice L, where we have $\mathsf{gfp}_F = \bigsqcap_n \mathsf{lfp}_{F^\sharp}\, n$, a gfp_F-preservation property has the form

$$(*) \qquad U\,(\mathsf{gfp}_F, \ldots, \mathsf{gfp}_F) \leq \mathsf{gfp}_F$$

where $U : L^m \to L$; and an inductive enhancement of it has the form

$$(**) \qquad \forall n.\, U\,(\mathsf{lfp}_{F^\sharp}\, n, \ldots, \mathsf{lfp}_{F^\sharp}\, n) \leq \mathsf{lfp}_{F^\sharp}\, n\,.$$

And indeed, (**) is stronger than (*) if U itself is ω-cocontinuous (which *is* the case with our safety preservation operators).

Next, we will infer the coinductive version of enhancement by analyzing the typical pattern of a proof of the inductive enhancement (**) by induction on n, and (similarly to what we did in §5.1) trying to identify its coinductive core. For the inductive step, assuming $U\,(\mathsf{lfp}_{F^\sharp}\, n, \ldots, \mathsf{lfp}_{F^\sharp}\, n) \leq \mathsf{lfp}_{F^\sharp}\, n$, one typically proves $U\,(\mathsf{lfp}_{F^\sharp}\,(n+1), \ldots, \mathsf{lfp}_{F^\sharp}\,(n+1)) \leq \mathsf{lfp}_{F^\sharp}\,(n+1)$ via reasoning that amounts to the following chain of (in)equalities:

$$U(\mathsf{lfp}_{F^\sharp}(n+1), \ldots, \mathsf{lfp}_{F^\sharp}\,(n+1)) = U(\,F(\mathsf{lfp}_{F^\sharp} n)\,, \ldots, \,F(\mathsf{lfp}_{F^\sharp} n)\,)$$
$$\leq F(\,\mathsf{lfp}_{F^\sharp} n \sqcup U(\,\mathsf{lfp}_{F^\sharp} n\,, \ldots, \,\mathsf{lfp}_{F^\sharp} n\,)) = F(\mathsf{lfp}_{F^\sharp} n) = \mathsf{lfp}_{F^\sharp}(n+1)$$

$$\mathsf{U}^{\mathsf{done}}_{Q}\,(c,s) = c = \mathsf{done} \wedge Q\,s$$
$$\mathsf{U}^{\mathsf{seq}}_{(R,G,P,c_2)}\,K\,(c,s) = \exists c_1.\ c = \mathsf{seq}\ c_1\ c_2 \wedge \mathsf{safe}_{(R,G,P)}(c_1,s) \wedge (\forall s'.\ P\,s' \longrightarrow K\,(c_2,s'))$$
$$\mathsf{U}^{\mathsf{if}}_{(R,t,c_1,c_2)}\,(K_1,K_2)\,(c,s) = (c = \mathsf{if}\ t\ c_1\ c_2 \wedge \forall s'.\ R^*\ s\ s' \longrightarrow (\mathsf{evalT}\ t\ s' \longrightarrow K_1\,(c_1,s')) \wedge (\neg\,\mathsf{evalT}\ t\ s' \longrightarrow K_2\,(c_2,s')))$$

Fig. 7: Operators underlying done, seq and if (more details in App. D)

In concrete cases, the highlighted inequality typically holds regardless of whether we have $F\,(\mathsf{lfp}_{F^\sharp}\ n)$ on the left and $\mathsf{lfp}_{F^\sharp}\ n$ on the right, except for the fact that $F\,(\mathsf{lfp}_{F^\sharp}\ n) \leq \mathsf{lfp}_{F^\sharp}\ n$; in other words, we could equally well prove that

$$(***) \qquad U\,(k_1,\ldots,k_m) \leq F\,((k'_1 \sqcup \ldots \sqcup k'_m) \sqcup U\,(k_1,\ldots,k_m))$$

whenever $k_i \leq k'_i$ and $k_i \leq F\ k'_i$ for all $i \in \{1,\ldots,m\}$.

Thus, at the core of an inductive enhancement proof, we discovered property (***), which happens to be a variation of a well-known concept. Let us call an operator $U : L^m \to L$ *weakly F-respectful* if it satisfies (***); and *F-respectful* if it satisfies (***) even without the $\sqcup$ part, i.e., with conclusion $U\,(k_1,\ldots,k_m) \leq F\,(U\,(k'_1,\ldots,k'_m))$. Unary respectful operators were studied by Sangiorgi and Pous [59,54], who proved that they enable *up-to enhancements* of coinduction. This generalizes to weakly respectful m-ary operators:

Thm 9. If $F : L \to L$ is a monotonic operator on a complete lattice and $U : L^m \to L$ is monotonic and weakly F-respectful, then the following *up-to-U coinduction* principle holds: Given $k_1,\ldots,k_m \in L$, for proving $k_i \leq \mathsf{gfp}_F$ for all $i \in \{1,\ldots,m\}$, it suffices to prove $k_i \leq F\,(U(k_1,\ldots,k_m))$ for all $i \in \{1,\ldots,m\}$.

Assuming $m = 1$, the principle allows inferring $k \leq \mathsf{gfp}_F$ from $k \leq F(U\,k)$, which generalizes the Knaster-Tarski coinduction principle: taking $U = 1_L$ we obtain ordinary coinduction, and taking $U = \lambda k.\ \mathsf{gfp}_F \sqcup k$ we obtain strong coinduction; overall, it offers more flexibility in a conductive proof, since $U\,k$ is usually larger than k. The class of (weakly) respectful operators contains many useful basic operators such as the projections. It is closed under composition, in that $V \circ (U_1,\ldots,U_n)$ is (weakly) respectful whenever V and $U_1,\ldots,U_n$ are, and suprema, in that $U_1 \sqcup U_2$ is (weakly) respectful whenever U_1 and U_2 are. This means that we can obtain coinductive enhancements by quite freely combining weakly respectful operators.

Back to our concrete situation, Fig. 8(right) shows the conditions under which the operators $\mathsf{U}^{\mathsf{seq}}_{(R,G,P,c_2)}$ and $\mathsf{U}^{\mathsf{if}}_{(R,t,c_1,c_2)}$ are weakly respectful. Indeed, the (conditional) weak respectfulness of these operators represent the coinductive enhancements we are after. Moreover, the gray highlighting in Fig. 8 shows how the weak respectfulness properties (from the right) can be regarded as a generalization of the coinductive goal arising when proving the corresponding safety preservation property (from the left). For example, in the case of sequential composition, Fig. 8(left) indicates that the safety preservation for seq, rephrased as $\mathsf{U}^{\mathsf{seq}}_{(R,G,P,c_2)}\mathsf{safeC}_{(R,G,Q)} \leq \mathsf{safeC}_{(R,G,Q)}$, is proved by strong

	Safety preservation needed for coinductive soundness	Enhanced version
done	$\dfrac{(1)\ Q\ s \qquad (2)\ \mathsf{stable}\ Q\ R}{\mathsf{safeC}_{(R,G,Q)}\ (\mathsf{done}, s)}$ i.e., $\dfrac{(1)\ Q\ s \quad (2)\ \mathsf{stable}\ Q\ R}{\mathsf{U}^{\mathsf{done}}_{Q} \le \mathsf{safeC}_{(R,G,Q)}}$ proved by strong coinduction in the form: $\dfrac{(1)\ Q\ s \qquad (2)\ \mathsf{stable}\ Q\ R}{\mathsf{U}^{\mathsf{done}}_{Q}\ (c, s) \le \mathsf{F}_{(R,G,Q)}\ (\mathsf{safe}_{(R,G,Q)} \sqcup \mathsf{U}^{\mathsf{done}}_{Q})}$	No enhancement needed for 0-ary operators such as $\mathsf{U}^{\mathsf{done}}_{Q}$.
seq	$\dfrac{(1)\ \mathsf{refl}\ G \qquad (2)\ \mathsf{safeC}_{(R,G,P)}(c_1, s)}{\mathsf{safeC}_{(R,G,Q)}(\mathsf{seq}\ c_1\ c_2, s)}$ i.e., $\dfrac{(1)\ \mathsf{refl}\ G}{\mathsf{U}^{\mathsf{seq}}_{(R,G,P,c_2)}\mathsf{safeC}_{(R,G,Q)} \le \mathsf{safeC}_{(R,G,Q)}}$ proved by strong coinduction in the form: $\dfrac{(1)\ \mathsf{refl}\ G}{\mathsf{U}^{\mathsf{seq}}_{(R,G,P,c_2)}\ \mathsf{safeC}_{(R,G,Q)}\ \le \mathsf{F}_{(R,G,Q)}(\ \mathsf{safe}_{(R,G,Q)}\ \sqcup\ \mathsf{U}^{\mathsf{seq}}_{(R,G,P,c_2)}\ \mathsf{safeC}_{(R,G,Q)}\ )}$	$\dfrac{(1)\ \mathsf{refl}\ G}{K \le K' \qquad K \le \mathsf{F}_{(R,G,Q)}\ K'}$ $\mathsf{U}^{\mathsf{seq}}_{(R,G,P,c_2)}\ K\ \le$ $\mathsf{F}_{(R,G,Q)}(\ K'\ \sqcup \mathsf{U}^{\mathsf{seq}}_{(R,G,P,c_2)}\ K')$
if	$\dfrac{\begin{array}{c}(1)\ \mathsf{refl}\ G \\ (2)\ \forall s'.\ R^*\ s\ s' \longrightarrow (\mathsf{evalT}\ t\ s' \longrightarrow \mathsf{safeC}_{(R,G,Q)}(c_1, s')) \wedge \\ (\neg\ \mathsf{evalT}\ t\ s' \longrightarrow \mathsf{safeC}_{(R,G,Q)}(c_2, s'))\end{array}}{\mathsf{safeC}_{(R,G,Q)}(\mathsf{if}\ t\ c_1\ c_2, s)}$ i.e., $\dfrac{(1)\ \mathsf{refl}\ G}{\mathsf{U}^{\mathsf{if}}_{(R,t,c_1,c_2)}(\mathsf{safeC}_{(R,G,Q)}, \mathsf{safeC}_{(R,G,Q)}) \le \mathsf{safeC}_{(R,G,Q)}}$ proved by strong coinduction in the form: $\dfrac{(1)\ \mathsf{refl}\ G}{\begin{array}{l}\mathsf{U}^{\mathsf{if}}_{(R,t,c_1,c_2)}(\ \mathsf{safeC}_{(R,G,Q)}\ ,\ \mathsf{safeC}_{(R,G,Q)}\ )\ \le \\ \mathsf{F}_{(R,G,Q)}(\ \mathsf{safe}_{(R,G,Q)}\ \sqcup \\ \qquad \mathsf{U}^{\mathsf{if}}_{(R,t,c_1,c_2)}(\ \mathsf{safeC}_{(R,G,Q)}\ ,\ \mathsf{safeC}_{(R,G,Q)}\ ))\end{array}}$	$\dfrac{(1)\ \mathsf{refl}\ G}{\begin{array}{l}K_1 \le K_1' \qquad K_1 \le \mathsf{F}_{(R,G,Q)}\ K_1' \\ K_2 \le K_2' \qquad K_2 \le \mathsf{F}_{(R,G,Q)}\ K_2'\end{array}}$ $\mathsf{U}^{\mathsf{if}}_{(R,t,c_1,c_2)}(\ K_1\ ,\ K_2\ ) \le$ $\mathsf{F}_{(R,G,Q)}(\ K_1'\ \sqcup\ K_2'\ \sqcup$ $\mathsf{U}^{\mathsf{if}}_{(R,t,c_1,c_2)}(\ K_1'\ ,\ K_2'\ ))$

Fig. 8: Coinductive enhancements

coinduction, thus the goal takes the form

$$(****)\quad \mathsf{U}^{\mathsf{seq}}_{(R,G,P,c_2)}\mathsf{safeC}_{(R,G,Q)} \leq \mathsf{F}_{(R,G,Q)}(\mathsf{safe}_{(R,G,Q)} \sqcup \mathsf{U}^{\mathsf{seq}}_{(R,G,P,c_2)}\mathsf{safeC}_{(R,G,Q)}).$$

And Fig. 8(right) generalizes this to weak respectfulness, by replacing in (****) the left occurrence of $\mathsf{safe}_{(R,G,Q)}$ with K and the right one with K'. Moreover, here as well as for all the other constructs, the proof of weak respectfulness can be generalized in a completely systematic way from that of the original preservation property (****): In Fig. 5(b)'s coinductive proof of (****), by writing $\mathsf{safe}_{(R,G,Q)}$ and respectively $\mathsf{safe}'_{(R,G,Q)}$ for the occurrences of $\mathsf{safe}_{(R,G,Q)}$ stemming from the left and respectively right of (****), we see that the only properties of this predicate needed in the proof are $\mathsf{safe}_{(R,G,Q)} \leq \mathsf{safe}'_{(R,G,Q)}$ and $\mathsf{safe}_{(R,G,Q)} \leq \mathsf{F}_{(R,G,Q)}\, \mathsf{safe}'_{(R,G,Q)}$—meaning we can replace $\mathsf{safe}_{(R,G,Q)}$ and $\mathsf{safe}'_{(R,G,Q)}$ with arbitrary K and K' such that $K \leq K'$ and $K \leq \mathsf{F}_{(R,G,Q)}\, K'$, yielding a proof of weak respectfulness.

We now come to the coinductive proof of the soundness preservation for while. Fixing R, G, Q, t, c, we take advantage of the coinductive enhancements for seq and if, and combine them in an operator W that follows the operational semantics of while, namely

$$W = 1_{\mathsf{L}_{(R,G,Q)}} \sqcup (\mathsf{U}^{\mathsf{if}}_{(R,t,\mathsf{seq}\ c\ (\mathsf{while}\ t\ c),\mathsf{done})} \circ (\mathsf{U}^{\mathsf{seq}}_{(R,G,P,\mathsf{while}\ t\ c)}, \lambda K.\ \mathsf{U}^{\mathsf{done}}_{Q}))\,.$$

Combining the weak $\mathsf{F}_{(R,G,Q)}$-respectfulness properties of $U^{\mathsf{done}}_{\_}$, $U^{\mathsf{if}}_{\_}$ and $U^{\mathsf{seq}}_{\_}$ from Fig. 8(right), we obtain that W is weakly respectful whenever $\mathsf{refl}\ G$, $\mathsf{stable}\ Q\ R$ and $\mathsf{stable}\ P\ R$.

The proof by up-to-W coinduction is shown in Fig. 9(b). Comparing it to the inductive proof from Fig. 9(a), similar comments to those in §5.1 apply about the coinductive proof being more direct. As for the enhancement dimension, the benefit of using the inductive hypothesis on the larger term $\mathsf{if}\ t\ (\mathsf{seq}\ c\ (\mathsf{while}\ t\ c))\ \mathsf{done}$ is now achieved by using the coinductive invariant up-to W, i.e., applying W to it in the coinductive goals (4.i) and (4.iii). Admittedly, the more elementary nature of the inductive enhancements can be an advantage. Moreover, the coinductive proof requires anticipation on how the semantics will unfold, to decide which (weakly) respectful operator to use, whereas the inductive proof proceeds analytically, decomposing the goal and using the properties on a need basis—but this can be achieved with the coinductive enhancements too, via the companion (which is also the largest respectful operator) [26,55].

To summarize the enhancement situation abstractly: For operators $F : L \to L$ and $U : L^m \to L$, assume that we were able to prove that U preserves F's greatest fixpoint, i.e., $U\,(\mathsf{gfp}_F, \ldots, \mathsf{gfp}_F) \leq \mathsf{gfp}_F$,

- either inductively, by showing $\forall n.\, U\,(\bigsqcap_m \mathsf{lfp}_{F^\sharp} m) \leq \mathsf{lfp}_{F^\sharp} n$,
- or coinductively, by showing $U\,(\mathsf{gfp}_F, \ldots, \mathsf{gfp}_F) \leq F\,(U\,(\mathsf{gfp}_F, \ldots, \mathsf{gfp}_F))$.

Then the enhancements proceed as follows:

- for induction, by showing the finer property $\forall n.\, U\,(\mathsf{lfp}_{F^\sharp} n, \ldots, \mathsf{lfp}_{F^\sharp} n) \leq \mathsf{lfp}_{F^\sharp} n$, which we will refer to as *F-approximation-preservation*;
- for coinduction, by showing the finer property that U is weakly F-respectful.

Preliminaries

We assume **(1)** $P \sqcap (\neg\ (\mathsf{evalT}\ t)) \leq Q\ \wedge\ \mathsf{stable}\ P\ R\ \wedge\ \mathsf{stable}\ Q\ R\ \wedge\ \mathsf{refl}\ G$. We must prove that $c \vDash_{\mathsf{VP}} (P \sqcap (\mathsf{evalT}\ t), R, G, P)$ implies $\mathsf{while}\ t\ c \vDash_{\mathsf{VP}} (P, R, G, Q)$. Expanding the definition of $\vDash_{\mathsf{VP}}$, this reduces to:

Main part

Assuming **(2)** $\forall m, s'.\ P\ s' \wedge \mathsf{evalT}\ t\ s' \longrightarrow \mathsf{safe}_{(R,G,P)}\ m\ (c, s')$, we must prove $\forall n, s.\ P\ s \longrightarrow \mathsf{safe}_{(R,G,Q)}\ n\ (\mathsf{while}\ t\ c, s)$, which we do by induction on n. The base case is again trivial.

For the inductive step, we assume **(IH)** $\forall s.\ P\ s \longrightarrow \mathsf{safe}_{(R,G,Q)}\ n\ (\mathsf{while}\ t\ c, s)$ and must prove $\forall s.\ P\ s \longrightarrow \mathsf{safe}_{(R,G,Q)}\ (n+1)\ (\mathsf{while}\ t\ c, s)$. To this end, we fix s, assume **(3)** $P\ s$, and must prove **(4)** $\mathsf{safe}_{(R,G,Q)}\ (n+1)\ (\mathsf{while}\ t\ c, s)$, which by the definition of $\mathsf{safe}_{(R,G,Q)}$ amounts to:

- **(4.i)** $\forall s'.\ R\ s\ s' \longrightarrow \mathsf{safe}_{(R,G,Q)}\ n\ (\mathsf{while}\ t\ c, s')$
- **(4.ii)** $\mathsf{final}\ (\mathsf{while}\ t\ c, s) \longrightarrow Q\ s$
- **(4.iii)** $\forall d', s'.\ ((\mathsf{while}\ t\ c, s) \Rightarrow (d', s')) \longrightarrow G\ s\ s' \wedge \mathsf{safe}_{(R,G,Q)}\ n\ (d', s')$,

of which the last, thanks to the fact that by the semantics of while its hypothesis means $s = s' \wedge d' = \mathsf{if}\ t\ (\mathsf{seq}\ c\ (\mathsf{while}\ t\ c))\ \mathsf{done}$, reduces to $G\ s\ s \wedge \mathsf{safe}_{(R,G,Q)}\ n\ (\mathsf{if}\ t\ (\mathsf{seq}\ c\ (\mathsf{while}\ t\ c))\ \mathsf{done}, s)$, which, since $G\ s\ s$ follows from (1), further reduces to

- **(4.iii')** $\mathsf{safe}_{(R,G,Q)}\ n\ (\mathsf{if}\ t\ (\mathsf{seq}\ c\ (\mathsf{while}\ t\ c))\ \mathsf{done}, s)$

Now, from (1), (3) and (IH), we obtain (4.i); and (4.ii) holds because its hypothesis is vacuously false.

Remains to prove (4.iii'), using (IH) and (3)...

Thanks to enhanced preservation for if, (4.iii') reduces to:

- $\mathsf{refl}\ G$, which holds by (1);
- Fixing s', assuming **(5)** $R^*\ s\ s'$, and:
  - Assuming $\neg\ \mathsf{evalT}\ t\ s'$ and showing $\mathsf{safe}_{(R,G,Q)}\ n\ (\mathsf{done}, s')$, which follows from (1), (3), (5) and the preservation for done.
  - Assuming **(6)** $\mathsf{evalT}\ t\ s'$ and showing $\mathsf{safe}_{(R,G,Q)}\ n\ (\mathsf{seq}\ c\ (\mathsf{while}\ t\ c), s')$, which, thanks to the enhanced preservation for seq, reduces to:
    - (again) $\mathsf{refl}\ G$, holding by (1);
    - $\forall m.\ \mathsf{safe}_{(R,G,P)}\ m\ (c, s')$, following from (2), (3), (6);
    - $\forall s.\ P\ s \longrightarrow \mathsf{safe}_{(R,G,Q)}\ n\ (\mathsf{while}\ t\ c, s)$, i.e., (IH). □

(a) For the counting-based semantics

Preliminaries

We assume **(1)** $P \sqcap (\neg\ (\mathsf{evalT}\ t)) \leq Q\ \wedge\ \mathsf{stable}\ P\ R\ \wedge\ \mathsf{stable}\ Q\ R\ \wedge\ \mathsf{refl}\ G$. We must prove that $c \vDash_{\mathsf{C}} (P \sqcap (\mathsf{evalT}\ t), R, G, P)$ implies $\mathsf{while}\ t\ c \vDash_{\mathsf{C}} (P, R, G, Q)$. Expanding the definition of $\vDash_{\mathsf{C}}$, this reduces to:

Main part

Assuming **(2)** $\forall s'.\ P\ s' \wedge \mathsf{evalT}\ t\ s' \longrightarrow \mathsf{safeC}_{(R,G,P)}\ (c, s')$, we must prove $\forall s.\ P\ s \longrightarrow \mathsf{safeC}_{(R,G,Q)}\ (\mathsf{while}\ t\ c, s)$. To this end, we fix s, c, t, d, assume $P\ s$ and $d = \mathsf{while}\ t\ c$, and must prove $\mathsf{safe}_{(R,G,Q)}\ (d, s)$, which we do by up-to-$W$ coinduction, since W is indeed $F_{(R,G,Q)}$-respectful thanks to (1).

Thus, we assume **(3)** $P\ s$ and $d = \mathsf{while}\ t\ c$, and must show
(4) $\mathsf{F}_{(R,G,Q)}(W\ (\lambda(d, s).\ P\ s \wedge d = \mathsf{while}\ t\ c))\ (d, s)$, which amounts to:

- **(4.i)** $\forall s'.\ R\ s\ s' \longrightarrow W\ (\lambda(d, s).\ P\ s \wedge d = \mathsf{while}\ t\ c))\ (d, s)$,
  which, taking the left disjoint in W, reduces to
- **(4.i')** $\forall s'.\ R\ s\ s' \longrightarrow P\ s' \wedge d = \mathsf{while}\ t\ c$,
  i.e., $\forall s'.\ R\ s\ s' \longrightarrow P\ s'$
- **(4.ii)** $\mathsf{final}\ (d, s) \longrightarrow Q\ s$, i.e., $\mathsf{final}\ (\mathsf{while}\ t\ c, s) \longrightarrow Q\ s$
- **(4.iii)** $\forall d', s'.\ ((d, s) \Rightarrow (d', s')) \longrightarrow G\ s\ s' \wedge W\ (\lambda(d, s).\ P\ s \wedge d = \mathsf{while}\ t\ c))\ (d', s')$,
  which, taking the right disjoint in W, reduces to
- **(4.iii')** $\forall d', s'.\ ((d, s) \Rightarrow (d', s')) \longrightarrow$
  - **(a)** $G\ s\ s' \wedge d' = \mathsf{if}\ t\ (\mathsf{seq}\ c\ (\mathsf{while}\ t\ c))\ \mathsf{done} \wedge$
  - **(b)** $(\forall s'.\ R^*\ s\ s' \wedge \mathsf{evalT}\ t\ s' \longrightarrow \mathsf{safe}_{(R,G,P)}(c, s')) \wedge$
  - **(c)** $(\forall s'.\ R^*\ s\ s' \wedge \neg\,\mathsf{evalT}\ t\ s' \longrightarrow Q\ s') \wedge$
  - **(d)** $(\forall s'.\ P\ s' \longrightarrow P\ s' \wedge d = \mathsf{while}\ t\ c)$

Now, from (1) and (3), we obtain (4.i'); and (4.ii) holds because its hypothesis is vacuously false.
Finally, using $d = \mathsf{while}\ t\ c$ and the semantics of while, the hypothesis of (4.iii') means $s = s' \wedge d' = \mathsf{if}\ t\ (\mathsf{seq}\ c\ (\mathsf{while}\ t\ c))\ \mathsf{done}$, which makes (a) true thanks to (1). Moreover:

- (b) is true thanks to (1), (2) and (3),
- (c) is true thanks to (1) and (3),
- (d) is trivially true (since $d = \mathsf{while}\ t\ c$). □

(b) For the coinductive semantics

Fig. 9: Proofs of soundness of the RG rule for while: for the (inductive) counting-based and coinductive semantics.

In concrete cases such as our safety preservation properties, for both induction and coinduction the proof of the finer property can be obtained by systematically generalizing the proof of the original one. The two enhancement types are connected as follows.

Thm 10. Assume L is a complete lattice and $F : L \to L$ and $U : L^m \to L$ are monotonic. Then the following hold:
(1) If U is weakly F-respectful, then it is also F-approximation-preserving.
(2) If F is ω-cocontinuous and U is F-approximation-preserving, then up-to-U coinduction is a sound proof method (in the sense of Thm. 9).

Point (1) of the above theorem says that what traditionally makes coinductive enhancements possible, namely (weak) respectfulness, is stronger than what we have identified as making inductive enhancements possible, namely the approximation-preserving property. And point (2) says that what makes the inductive enhancements possible, although a weaker property, still makes the coinductive ones possible.

6 The Coinduction Approximated Inductively Landscape

Next we look at the scope of coinduction approximated inductively in finer granularity. While so far we only looked at the concrete instances of RG safety operators and a very abstract generalization to operators on lattices, here we will explore a spectrum between these two extremes.

6.1 The rule system format

Thm. 8 captures the general phenomenon behind coinduction approximated inductively, but does not give us a good intuition on its more concrete scope, notably on how it applies to (co)inductive definitions described by sets of rules [2]. To address this, we slightly lower the generality, by focusing on lattices of predicates on a set A, namely $L = (A \to \mathsf{Bool})$ with order defined again by component-wise implication ($K \leq K'$ iff $\forall a \in A.\ K\ a \longrightarrow K'\ a$), and operators given by sets of rules on A, i.e., subsets $Rl \subseteq \mathcal{P}(A) \times A$. So a *rule* on A will be a pair (B, a) where $B \subseteq A$ and $a \in A$; we think of B as the set of the rule's *hypotheses* and of a as the rule's *conclusion*. The associated operator $\mathsf{F}_{Rl} : (A \to \mathsf{Bool}) \to (A \to \mathsf{Bool})$ is defined by applying the rules, namely $\mathsf{F}_{Rl}\ K\ a \longleftrightarrow \exists B.\ (B, a) \in Rl \wedge (\forall b \in B.\ K\ b)$. Thus, the notion of a predicate $K : A \to \mathsf{Bool}$ being closed under the rules, namely $\forall (B, a) \in R.\ (\forall b \in B.\ K\ b) \longrightarrow K\ a$, is the same as K being a pre-fixpoint of F_{Rl}, i.e., $\mathsf{F}_{Rl}\ K \leq K$. And the notion of K being consistent with (i.e., backwards closed under) the rules, namely $\forall (B, a) \in R.\ K\ a \longrightarrow (\forall b \in B.\ K\ b)$, is the same as K being a post-fixpoint of F_{Rl}, i.e. $K \leq \mathsf{F}_{Rl}\ K$. For the least and greatest fixpoints of rule-system operator F_{Rl}, we will write lfp_{Rl} for $\mathsf{lfp}_{\mathsf{F}_{Rl}}$ and gfp_{Rl} for $\mathsf{gfp}_{\mathsf{F}_{Rl}}$.

A first practical observation is that the inductive approximation can be performed intensionally, at the level of rules. Given a set of rules Rl on A, we

define $Rl^\sharp$, called the *approximation rules of* Rl, as a set of rules on $\mathbb{N} \times A$, namely $\{(\varnothing, (0, a)) \mid a \in A\} \cup \bigcup_{n \in \mathbb{N}} \{(\{n\} \times B,\ (n+1, a)) \mid (B, a) \in Rl\}$. Then $\mathsf{F}_{Rl^\sharp}$ and $(\mathsf{F}_{Rl})^\sharp$ are the same (modulo currying), which shifts the discussion from the semantic realm of operators to the (usually) syntactic realm of rules:

Lemma 11. Given $Rl \subseteq \mathcal{P}(A) \times A$ and $K : \mathbb{N} \to A \to \mathsf{Bool}$, we have that $\mathsf{F}_{Rl^\sharp}\ K = (\mathsf{F}_{Rl})^\sharp\ (\lambda n, a.\ K(n, a))$.

And our motivating example predicates are indeed given by rule systems:

Example 12. Both $\mathsf{F}_{(R,G,Q)}$ and $\mathsf{F}^\sharp_{(R,G,Q)}$ are rule-system based. Namely, we have that $\mathsf{F}_{(R,G,Q)} = \mathsf{F}_{Rl_{(R,G,Q)}}$ and $\mathsf{F}^\sharp_{(R,G,Q)} = (\mathsf{F}_{Rl_{(R,G,Q)}})^\sharp = \mathsf{F}_{(Rl_{(R,G,Q)})^\sharp}$, and therefore $\mathsf{safeC}_{(R,G,Q)} = \mathsf{gfp}_{Rl_{(R,G,Q)}}$ and $\mathsf{safe}_{(R,G,Q)} = \mathsf{lfp}_{(Rl_{(R,G,Q)})^\sharp}$, where $Rl_{(R,G,Q)}$ is the set of pairs $(B,\ (c, s))$ such that $B = \{(c', s') \mid ((c, s) \Rightarrow (c', s'))\} \cup \{(c, s') \mid R\ s\ s'\}$ and $(\mathsf{final}\,(c, s) \longrightarrow Q\ s) \wedge (\forall c', s'.\ ((c, s) \Rightarrow (c', s')) \longrightarrow G\ s\ s')$ holds.

We are ready to describe a more effective criterion for sound approximation. We call a set of rules $Rl \subseteq \mathcal{P}(A) \times A$ *backwards-finite* provided that each $a \in A$ has only a finite number of rules that backwards-apply to it, namely $\forall a \in A,\ \mathsf{finite}\,\{B \subseteq A \mid (B, a) \in Rl\}$. This is a sufficient condition for ω-cocontinuity.

Prop 13. If $Rl \subseteq \mathcal{P}(A) \times A$ is backwards-finite, then F_{Rl} is ω-cocontinuous.

This, together with Lemma 11 and Thm. 8, give us:

Thm 14. Assume that $Rl \subseteq \mathcal{P}(A) \times A$ is backwards-finite. Then $\mathsf{gfp}_{Rl}\ a \longleftrightarrow \forall\ n \in \mathbb{N}.\ \mathsf{lfp}_{Rl^\sharp}\ (n, a)$ for any $a \in A$.

Thus, Thm. 14 is a more concrete depiction of the root cause for the inductive approximation soundness: the backwards finiteness of the defining rules.

6.2 Approximations of bisimilarity

The rule system format has limitations. Notably, it fails to cover the prominent instances of coinduction approximated inductively that occur in the theory of (bi)simulations and (bi)similarity going back to Park and Milner [50,41]—where bisimilarity is first defined coinductively, after which an inductive counterpart is introduced as a technical tool (e.g., for proving logical completeness [25] or domain-theoretic properties [1]). The problem is that the many notions of bisimilarity [41,20,19] cannot be defined via rule systems due to their quantification format. Indeed, the operators underlying rule systems are essentially defined using $\exists\forall$ quantification ("there exists a matching rule such that for all its hypotheses ..."), whereas bisimilarities typically have a $\forall\exists$ format ("for all transitions there exists a matching transition ...").

Furthermore, bisimilarities for systems with names and substitutions, such as late bisimilarity for the π-calculus [42,61], even have a $\forall\exists\forall$ format ("for all input transitions there exists a matching transition such that, for all instantiations of the generic name from the transition..."), as shown in the next example.

Example 15. We assume an Act-labeled transition system $\mathcal{L} = (\mathsf{State}, \mathsf{Act}, \Rightarrow)$ with $\Rightarrow \subseteq \mathsf{State} \times \mathsf{Act} \times \mathsf{State}$ such that there is a set Nm of (channel) names and a renaming operation $\_[\_/\_] : \mathsf{State} \times \mathsf{Nm} \times \mathsf{Nm} \to \mathsf{State}$. The elements $s \in \mathsf{State}$ are called processes, and we think of $s[u/v]$ as the process obtained by the capture-free renaming of u to v. Moreover, we assume a subset $\mathsf{IAct} \subseteq \mathsf{Act}$ of *input actions*, where each input actions has the form $u(v)$ for $u, v \in \mathsf{Nm}$. Now, a *(strong) late bisimulation* is a symmetric relation K on State such that, for all s_1, s_2, if $K\ s_1\ s_2$ then the following holds:

$$\begin{array}{l}(\ \forall\ a, s_1'.\ a \in \mathsf{Act} \setminus \mathsf{IAct} \wedge s_1 \overset{a}{\Rightarrow} s_1' \longrightarrow\ \exists\ s_2'.\ s_2 \overset{a}{\Rightarrow} s_2' \wedge K\ s_1'\ s_2') \wedge \\ (\ \forall\ u, v, s_1'.\ u, v \in \mathsf{Nm} \wedge s_1 \overset{u(v)}{\Rightarrow} s_1' \longrightarrow \qquad\qquad (*) \\ \qquad \exists\ s_2'.\ s_2 \overset{u(v)}{\Rightarrow} s_2' \wedge (\ \forall\ w \in \mathsf{Nm}. K\ (s_1'[w/v])\ (s_2'[w/v]))\end{array}$$

The *late bisimilarity* is the largest late bisimulation, i.e., greatest fixpoint of the operator F on relations defined as $F\ K\ (s_1, s_2) \longleftrightarrow K\ (s_2, s_1)\ \wedge\ (*)$.

6.3 An alternating quantifier format

As we discuss next, what makes bisimilarity-defining operators ω-cocontinuous, hence bisimilarities inductively approximable, is the finiteness of the space for existential quantification. Obeying this, any quantifier alternation can be allowed.

We model this by the following predicate. Given a set A and a set of sets Π, of whose elements $P \in \Pi$ we think of as sets of parameters, we define the *existentially-finite alternating quantifier format* predicate $\mathsf{alter} : ((A \to \mathsf{Bool}) \to (A \to \mathsf{Bool})) \to \mathsf{Bool}$, (thus, a predicate on operators in $A \to \mathsf{Bool}$) inductively:

$$\frac{K' : A \to \mathsf{Bool}}{\mathsf{alter}\ (\lambda K.\ K')}(\mathsf{Const}) \qquad \frac{f : A \to A}{\mathsf{alter}\ (\lambda K.\ K \circ f)}(\mathsf{Comp})$$

$$\frac{\begin{array}{c}F : P \to (A \to \mathsf{Bool}) \to (A \to \mathsf{Bool}) \quad T : P \to (A \to \mathsf{Bool}) \\ P \in \Pi \qquad \forall p \in P.\ \mathsf{alter}\ (F\ p)\end{array}}{\mathsf{alter}\ (\lambda\ K, a.\ \forall p \in P.\ T\ p\ a \to F\ p\ K\ a)}(\mathsf{Forall})$$

$$\frac{\begin{array}{c}F : P \to (A \to \mathsf{Bool}) \to (A \to \mathsf{Bool}) \quad T : P \to (A \to \mathsf{Bool}) \\ \forall a.\ \mathsf{finite}\ \{p \mid T\ p\ a\} \qquad P \in \Pi \qquad \forall p \in P.\ \mathsf{alter}\ (F\ p)\end{array}}{\mathsf{alter}\ (\lambda\ K, a.\ \exists p \in P.\ T\ p\ a \wedge F\ p\ K\ a)}(\mathsf{Exists})$$

Thus, $\mathsf{alter}\ F$ says that the operator F (on predicates on A) is obtained by starting with operators that either are constant or just compose their argument predicates with a function on A, and applying (in any order) a sequence of universal and existential quantifiers over parameters, relative to predicates T connecting parameters with the elements of A. Highlighted in the case of existential quantification is that this takes place within a finite space, i.e., the relativization predicate T has finite extension. Note that alter depends on A and Π, and we write $\mathsf{alter}_{A,\Pi}$ when we want to emphasize this dependency.

Since alter is also closed under conjunction and disjunction (as a particular case of closure under quantification), we can obtain operators satisfying alter by

freely combining positive logical operators, as long as the existentials are finite. All such operators are guaranteed to be ω-cocontinuous:

Thm 16. If $\mathsf{alter}_{A,\Pi}\ F$ holds, then F is ω-cocontinuous, in particular $\mathsf{gfp}_F\ a \longleftrightarrow \forall n \in \mathbb{N}.\ \mathsf{lfp}_{F^\sharp}\ n\ a$ for any $a \in A$.

This criterion, which generalizes the rule system format, captures the image-finiteness condition for bisimilarity approximations. In particular, it applies to Example 15, since the π-calculus yields an image-finite labeled transition system. (Note that the additional nested universal quantification in Example 15 takes place in an infinite space (since Nm is usually infinite), but universal quantification is not constrained by the format.) It also generalizes a syntactic criterion implemented in the Dafny program verifier [33] by Leino and Moskal [34]. Their formulation refers to the syntax of predicates definable in Dafny, and involves a syntactic check on a predicate's negation normal form, namely that existential quantification happens over finite types. But their proof (in a technical report cited from [34]) already contains the general apparatus in terms of ω-continuity that we use here.

$$\mathsf{safeSep}_{(R,G,Q)}\ 0\ (c, s_1, s_2)\ \ \text{(Base)}$$

$$\frac{\begin{array}{l}1.\ \forall s_2', \alpha.\ R\ s_2\ s_2' \longrightarrow \mathsf{safeSep}_{(R,G,Q)}\ n\ (c, s_1, s_2')\\ 2.\ \mathsf{final}\ (c, s_1, s_2) \longrightarrow Q\ (s_1, s_2)\\ 3.\ \forall c', t_1, s_1', s_2', \alpha.\ \alpha \neq \tau \wedge s_1 \preceq t_1 \wedge\\ \qquad (c, t_1, s_2) \overset{\alpha}{\Rightarrow} (c', s_1', s_2') \longrightarrow G\ s\ s'\\ 4.\ (c, s_1, s_2) \overset{\alpha}{\Rightarrow} (c', s_1', s_2') \longrightarrow \mathsf{safeSep}_{(R,G,Q)}\ n\ (c, s_1', s_2')\\ 5.\ ((c, t_1 + u_1, s_2) \overset{\alpha}{\Rightarrow} (c', s_1', s_2')) \longrightarrow\\ \qquad (\exists t_1'.\ t_1' \# u_1 \wedge s_1' = t_1' + u_1\ \wedge (\alpha = \tau \longrightarrow t_1' = t_1) \wedge\\ \qquad\quad \mathsf{safeSep}_{(R,G,Q)}\ n\ (c, t_1', s_2'))\end{array}}{\mathsf{safeSep}_{(R,G,Q)}\ (n+1)\ (c, s_1, s_2)}\ \text{(Step)}$$

Fig. 10: The inductive counting-based predicate $\mathsf{safeSep}$. Its coinductive counterpart $\mathsf{safeSepC}$ is obtained (like before) by removing the items highlighted and interpreting the (Step) rule coinductively.

6.4 Generic RG Separation Logic

We now go back to our starting topic, RG reasoning. The counting-based semantics for RG, which served as our motivating example, is actually a trimmed down version of a semantics for a combination of RG with Separation Logic [66,67].

GenRGSep has a process algebra syntax. What is relevant here is that there is a set of actions Act (ranged over by α) which contains a special silent action τ. Moreover, the states s are pairs (s_1, s_2) where s_1 is the local part of the state

s_2 is the part shared with the environment (hence subject to change according to the rely relation R). Differently from RG, the execution steps in GenRGSep are labelled by actions, so they have the form $(c, s_1, s_2) \overset{\alpha}{\Rightarrow} (c', s_1', s_2')$. Moreover, local states form *permission algebras* [13], where + denotes the partial semi-group operator and # denotes the definedness predicate for this operator. We think of $s \# t$ as saying that s and t are non-overlapping (or consistently overlapping) and of $s + t$ as putting together two such non-overlapping components.

The formulation of safety for GenRGSep, predicate $\mathsf{safeSep}$, is shown in Fig. 10. The base case and the inductive hypotheses for the post- and rely-conditions (hypotheses 1 and 2) are essentially the same as those for the RG predicate safe (Fig. 3), but factoring in the distinction between the local and environment-exposed parts of the state. On the other hand, the hypotheses involving steps taken by the command (hypotheses 3–5) are more complex, quantifying universally over extensions of the local state ($s_1 \preceq t_2$ in hypothesis 4) and existentially over non-overlapping partitions of the local state ($t_1' \# u_1$ in the frame-safety hypothesis 5). Details on the rationale for these are provided in [28].

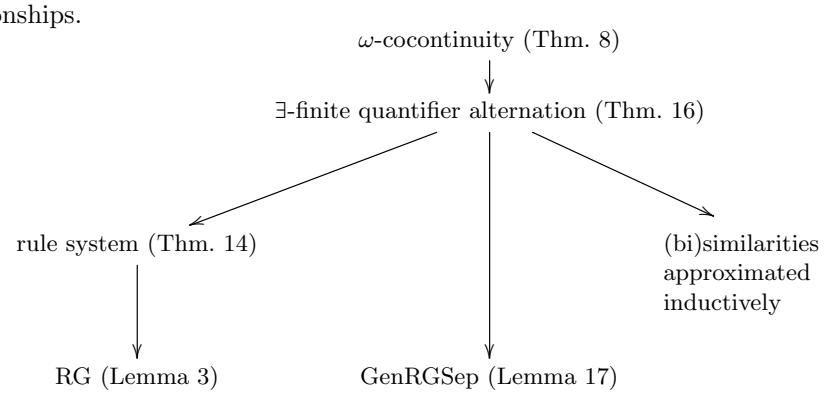
Because hypothesis 5 introduces nested existential quantification (over the component t_1'), this case does not fall under the rule-system format (Thm. 14). However, since all quantification (including the nested one) takes place in a finite space, Thm. 16 applies and produces a coinductive semantics based on a coinductive counterpart $\mathsf{safeSepC}$ of $\mathsf{safeSep}$, which is provably equivalent to the inductive semantics—via corresponding versions of Lemma 3 and Thm. 2:

Lemma 17. We have that $\forall n \in \mathbb{N}.\ \mathsf{safeSep}_{(R,G,Q)} n\,(c, s_1, s_2)$ holds if and only if $\mathsf{safeSepC}_{(R,G,Q)}(c, s_1, s_2)$ holds.

Thus, considering the inductive and coinductive satisfaction relations for GenRGSep, $\vDash_{\mathsf{Sep,VP}} (P, R, G, Q)$ and $c \vDash_{\mathsf{Sep,C}}$, defined from $\mathsf{safeSep}$ and $\mathsf{safeSepC}$ just like for RG, we obtain:

Thm 18. We have that $c \vDash_{\mathsf{Sep,VP}} (P, R, G, Q)$ iff $c \vDash_{\mathsf{Sep,C}} (P, R, G, Q)$.

Summary. Below is the hierarchy of the results we discussed pertaining to coinduction approximated inductively, with arrows representing instantiation relationships.



7 Isabelle-Supported Explorations

The starting point of this work was the goal of formalizing in Isabelle different semantics for Rely-Guarantee and proving them equivalent. The coinductive semantics came from the realization that safety w.r.t. any number of steps in the counting-based semantics is an instance of Kleene-style convergence. Isabelle was instrumental in our proof-mining-like explorations of this realization:

(1) Developing soundness proofs in parallel in the counting-based and the coinductive semantics suggested the coinductive core of an inductive argument, which we could then formulate abstractly in terms of a greatest fixpoint of an operator and its approximations (as in §5.1). This brought both concrete and abstract evidence for the more direct nature of the coinductive semantics.

(2) The above pattern of simplification when switching to a coinductive semantics seemed to be contradicted by the soundness of the rule for while, where the inductive proof prevailed over the coinductive one, since the former worked on the original goal whereas the latter had to be generalized to a larger coinductive invariant. We noticed that the inductive proof was effectively using (what we later called) approximation-preservation to delve through a larger context built around while by unfolding its operational semantics, which is traditionally a specialty of coinductive up-to enhancements. This led us to the general connection between inductive and coinductive enhancements.

(3) While the notion of **(A)** U being respectful, is known to imply that **(B)** U preserves the greatest fixpoint gfp_F (i.e., $U\ \mathsf{gfp}_F \leq \mathsf{gfp}_F$), working with formal proofs in Isabelle allowed us to identify a stronger connection, undocumented in the literature: Not only that (A) implies (B), but the proof of (A) can be obtained by essentially copying the coinductive proof of (B) and replacing the occurrences of gfp_F with arbitrary predicates k and k', depending on whether the occurrences come from the left or right of the initial coinductive goal. However, since for convenience in the original proofs we used not raw but strong coinduction (Coro. 5), we realized that copying the proofs as they are gives us not exactly respectfulness but what we called weak respectfulness (a minor variation that still enables coinduction up-to). This is in line with other case studies [26,53,55], which show that in practice respectfulness needs to be weakened.

(4) Owing to working fully formally hence not having the freedom to gloss over details when instantiating results to particular cases, we aimed at lowering the abstract-to-concrete instantiation distance, leading to the formalization of the format-based criteria from §6.

The Isabelle scripts supporting this paper are provided as supplementary material. The general results on coinduction (see §4) and coinduction up-to (see §5.2), including the format-based criteria above in (4), are top-level theories. The remaining formalization is broken down into seven directories: one for each of our four RG semantics, the separation logic extension from §6.4, the equivalence proofs (i.e., Thm. 1 and Thm. 2), and the wider landscape (see §6.2). In particular, we note each RG semantics includes a theory dedicated to soundness proofs (used for observations (1) and (2)), including a separate theory for the respectful variation of the coinductive semantics (see (3)). App. F gives details.

8 Related Work

Formal semantics for Rely-Guarantee. Nieto [47] formalized the trace-based semantics in Isabelle; Coleman and Jones [14] developed the reachability based semantics informally, and listed the comparison with Nieto's formal semantics as future work. Vafeiadis [66] informally, then Jackson et al. formally in Isabelle [28] developed the counting-based semantics for Separation Logic extensions. We seem to be the first to provide a coinductive semantics of RG, and also to prove the equivalence between previous semantics.

We mostly focused on a simple while language and the basic RG logic, but the coinductive transformation we applied to RG also fits richer languages and logics, as we have illustrated with GenRGSep. We restricted ourselves to partial correctness; and also for expository simplicity to unary post-conditions, though our formalization covers the binary post-conditions, a feature that Jones has advocated from the early days [29]. (Apps. E and F give details.) Moreover, we focused on Hoare-style RG definitions, rather than algebraic or refinement approaches, which also have a rich literature [7,8,23]. Several RG semantics variations are reviewed by Van Staden [63] who, taking a trace-based view, classifies them based on whether the guarantee and rely conditions are decoupled.

Rule systems and formats. Aczel introduced rule systems in his study of inductive definitions [2], considering coinductive definitions only briefly via duality with induction. He singled out deterministic rule systems (where each item can be the conclusion of at most one rule), because these enable defining the least fixpoint by (possibly transfinite) well-founded recursion; deterministic systems are particular cases of backwards-finite rule systems, so they would also ensure ω-cocontinuity. Aczel, and previously Moschovakis [44], were interested in abstract inductive definitions for studying recursively enumerable sets and Gödel encodings. In their theory, inductive predicates specified by positive logic formats play a central role—our alternating quantifier format (generalizing the one implemented in Dafny) is a refinement of the positive logic format. While, in process algebra, formats for operational semantics rules ensuring desirable properties of (bi)similarity have been studied extensively [45], we seem to be the first to look at the formats of the (bi)similarity definitions themselves.

Coinduction approximated inductively. Approximating coinduction by induction is useful for implementing coinduction in theorem provers that initially only support induction. We already mentioned Danfy's coinduction, certified by the Leino-Moskal syntactic criterion. More recently, Mastorou et al. [39] implemented coinduction in Liquid Haskell [68], also relying on inductive approximations. So far their examples do not involve existential quantification, but in general ensuring the relevance of inductive reasoning for the coinductive counterparts will seem to require a criterion similar to Dafny's. Alternatively, if a prover supports transfinite recursion (as is already partially the case with Dafny [35]), then any coinductive predicate can be approximated inductively (without restriction), via a generalization of Kleene's theorem (App. C).

While we focused on (co)inductive predicates, similar phenomena occur with datatypes versus codatatypes and recursion versus corecursion. Barr showed

that a codatatype can be regarded as a Cauchy completion of a datatype [10,11], and this enables defining functions on codatatypes as convergent approximations of recursive definitions. Matthews [40], and Di Gianantonio and Miculan [18] have built definitional frameworks around these ideas using order-theoretic approximation structures, implemented in Isabelle and Coq respectively. These works have some of their roots in domain theory and metric spaces, which have been used for many years to give semantics to programming languages and process algebras. When switching from sets to domains, often the initial algebra and the final coalgebra are isomorphic [62,4].

Usually in the literature (e.g., process algebra bisimilarities, and the aforementioned implementations of coinduction in verifiers) one starts with the goal of capturing coinductive definitions, and then thinks of how to represent / approximate these inductively. By contrast, here we started with a well-established definition of Rely-Guarantee semantics based on step-indexing, and argued for its "coinductivization". This approach can be applied more generally to step-indexed logical relations [5,3,36], where the numeric index counts the number of steps for which a certain desirable property (such as typing, or in our case RG contract compliance) is guaranteed not to be falsified. Indeed, Nakano's "later" modality [46] used in reasoning with step-based logical relations [16,6] corresponds to our approximating operators $F^\sharp$, and Löb's rule [12] for this modality corresponds to inductive reasoning over approximations; and the approximations are here coinductively sound thanks to the finite branching (and in fact often the determinism) of the transition relation. It would be interesting to explore the connection between Löb's rule and coinduction up-to along the lines of our discussion in §5.2. However, such a coinductive abstraction for step-indexed logical relations would fail as soon as the indexing involves not only numbers (of execution steps) but also other "world" data such as resources *that are mutually dependent* with these numbers—as is the case, for example, when reasoning in higher-order separation logic frameworks such as Iris [31,65].

(Co)inductive enhancements. Notwithstanding the prominently up-to feature of the while construct, the literature on verification based on coinduction up-to did not seem to handle while programs directly like we do, although imperative features were shown to be in the scope of the technique (e.g., [38]).

Our established connection between up-to enhancements of coinduction and corresponding index-wise enhancements of the greatest fixpoint's inductive approximations recommends approximation-preserving operators as a potentially interesting class of operators that are sound for coinduction up-to, which are both compositional and extremely lightweight. In future work, we will study their interaction with parameterized coinduction [26] and second-order reasoning [55] when restricted to ω-cocontinuous operators. Sangiorgi [60] considered a different type of inductive-coinductive connection, adapting up-to techniques to behavioral relations, where coinduction becomes structural induction over the observation contexts. A similar connection has been studied by Goguen and Malcolm in the context of hidden algebra [22].

References

1. Abramsky, S.: A domain equation for bisimulation. Inf. Comput. 92(2), 161–218 (1991), https://doi.org/10.1006/inco.1991.9999
2. Aczel, P.: An introduction to inductive definitions. In: Barwise, J. (ed.) Handbook of Mathematical Logic, Studies in Logic and the Foundations of Mathematics, vol. 90, pp. 739–782. Elsevier (1977), https://www.sciencedirect.com/science/article/pii/S0049237X08711200
3. Ahmed, A.: Step-indexed syntactic logical relations for recursive and quantified types. In: European Symposium on Programming (ESOP). Lecture Notes in Computer Science, vol. 3924, pp. 69–83. Springer (2006), https://doi.org/10.1007/11693024_6
4. Amadio, R.M., Curien, P.: Domains and lambda-calculi, Cambridge tracts in theoretical computer science, vol. 46. Cambridge University Press (1998)
5. Appel, A.W., McAllester, D.A.: An indexed model of recursive types for foundational proof-carrying code. ACM Trans. Program. Lang. Syst. 23(5), 657–683 (2001), https://doi.org/10.1145/504709.504712
6. Appel, A.W., Melliès, P., Richards, C.D., Vouillon, J.: A very modal model of a modern, major, general type system. In: Hofmann, M., Felleisen, M. (eds.) Proceedings of the 34th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2007, Nice, France, January 17-19, 2007. pp. 109–122. ACM (2007), https://doi.org/10.1145/1190216.1190235
7. Armstrong, A., Gomes, V.B.F., Struth, G.: Algebraic principles for rely-guarantee style concurrency verification tools. In: Jones, C., Pihlajasaari, P., Sun, J. (eds.) FM 2014: Formal Methods. pp. 78–93. Springer International Publishing, Cham (2014)
8. Armstrong, A., Gomes, V.B.F., Struth, G.: Algebras for program correctness in isabelle/hol. In: Höfner, P., Jipsen, P., Kahl, W., Müller, M.E. (eds.) Relational and Algebraic Methods in Computer Science - 14th International Conference, RAMiCS 2014, Marienstatt, Germany, April 28-May 1, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8428, pp. 49–64. Springer (2014), https://doi.org/10.1007/978-3-319-06251-8_4
9. Ballarin, C.: Tutorial to Locales and Locale Interpretation. In: Contribuciones Científicas en Honor de Mirian Andrés Gómez, pp. 123–140. University of Rioja, Spain (2010), online at https://dialnet.unirioja.es/servlet/articulo?codigo=3216664
10. Barr, M.: Terminal coalgebras in well-founded set theory. Theor. Comput. Sci. 114(2), 299–315 (1993), https://doi.org/10.1016/0304-3975(93)90076-6
11. Barr, M.: Additions and corrections to "terminal coalgebras in well-founded set theory". Theor. Comput. Sci. 124(1), 189–192 (1994), https://doi.org/10.1016/0304-3975(94)90060-4
12. Boolos, G.S.: The Logic of Provability. Cambridge University Press, Cambridge and New York (1993), https://www.cambridge.org/core/books/logic-of-provability/F1549530F91505462083CE2FEB6444AA
13. Calcagno, C., O'Hearn, P.W., Yang, H.: Local action and abstract separation logic. In: 22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10-12 July 2007, Wroclaw, Poland, Proceedings. pp. 366–378. IEEE Computer Society (2007), https://doi.org/10.1109/LICS.2007.30
14. Coleman, J.W., Jones, C.B.: A structural proof of the soundness of rely/guarantee rules. J. Log. Comput. 17(4), 807–841 (2007), https://doi.org/10.1093/logcom/exm030

15. Cousot, P., Cousot, R.: Constructive versions of Tarski's fixed point theorems. Pacific Journal of Mathematics 82(1), 43 – 57 (1979)
16. Dreyer, D., Ahmed, A., Birkedal, L.: Logical step-indexed logical relations. Log. Methods Comput. Sci. 7(2) (2011), https://doi.org/10.2168/LMCS-7(2:16)2011
17. Gavran, I., Niksic, F., Kanade, A., Majumdar, R., Vafeiadis, V.: Rely/guarantee reasoning for asynchronous programs. In: Aceto, L., de Frutos-Escrig, D. (eds.) 26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1.4, 2015. LIPIcs, vol. 42, pp. 483–496. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2015), https://doi.org/10.4230/LIPIcs.CONCUR.2015.483
18. Gianantonio, P.D., Miculan, M.: A unifying approach to recursive and co-recursive definitions. In: Geuvers, H., Wiedijk, F. (eds.) Types for Proofs and Programs, Second International Workshop, TYPES 2002, Berg en Dal, The Netherlands, April 24-28, 2002, Selected Papers. Lecture Notes in Computer Science, vol. 2646, pp. 148–161. Springer (2002), https://doi.org/10.1007/3-540-39185-1_9
19. van Glabbeek, R.J.: The linear time - branching time spectrum II. In: Best, E. (ed.) CONCUR '93, 4th International Conference on Concurrency Theory, Hildesheim, Germany, August 23-26, 1993, Proceedings. Lecture Notes in Computer Science, vol. 715, pp. 66–81. Springer (1993), https://doi.org/10.1007/3-540-57208-2_6
20. van Glabbeek, R.J.: The linear time - branching time spectrum I. In: Bergstra, J.A., Ponse, A., Smolka, S.A. (eds.) Handbook of Process Algebra, pp. 3–99. North-Holland / Elsevier (2001), https://doi.org/10.1016/b978-044482830-9/50019-9
21. van Glabbeek, R.J., Weijland, W.P.: Branching time and abstraction in bisimulation semantics (extended abstract). In: Ritter, G.X. (ed.) Information Processing 89, Proceedings of the IFIP 11th World Computer Congress, San Francisco, USA, August 28 - September 1, 1989. pp. 613–618. North-Holland/IFIP (1989)
22. Goguen, J.A., Malcolm, G.: Hidden coinduction: behavioural correctness proofs for objects. Math. Struct. Comput. Sci. 9(3), 287–319 (1999), http://journals.cambridge.org/action/displayAbstract?aid=44821
23. Hayes, I.J.: Generalised rely-guarantee concurrency: an algebraic foundation. Form. Asp. Comput. 28(6), 1057–1078 (Nov 2016), https://doi.org/10.1007/s00165-016-0384-0
24. Hayes, I.J., Jones, C.B.: A guide to rely/guarantee thinking. In: Bowen, J.P., Liu, Z., Zhang, Z. (eds.) Engineering Trustworthy Software Systems - Third International School, SETSS 2017, Chongqing, China, April 17-22, 2017, Tutorial Lectures. Lecture Notes in Computer Science, vol. 11174, pp. 1–38. Springer (2017), https://doi.org/10.1007/978-3-030-02928-9_1
25. Hennessy, M., Milner, R.: Algebraic laws for nondeterminism and concurrency. J. ACM 32(1), 137–161 (1985), https://doi.org/10.1145/2455.2460
26. Hur, C., Neis, G., Dreyer, D., Vafeiadis, V.: The power of parameterization in coinductive proof. In: Giacobazzi, R., Cousot, R. (eds.) The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013. pp. 193–206. ACM (2013), https://doi.org/10.1145/2429069.2429093
27. Ishtiaq, S.S., O'Hearn, P.W.: BI as an assertion language for mutable data structures. In: Hankin, C., Schmidt, D. (eds.) Conference Record of POPL 2001: The 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, London, UK, January 17-19, 2001. pp. 14–26. ACM (2001), https://doi.org/10.1145/360204.375719

28. Jackson, V., Murray, T., Rizkallah, C.: A generalised union of rely-guarantee and separation logic using permission algebras. In: Bertot, Y., Kutsia, T., Norrish, M. (eds.) 15th International Conference on Interactive Theorem Proving, ITP 2024, September 9-14, 2024, Tbilisi, Georgia. LIPIcs, vol. 309, pp. 23:1–23:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2024), https://doi.org/10.4230/LIPIcs.ITP.2024.23
29. Jones, C.B.: Development Methods for Computer Programs including a Notion of Interference. Ph.D. thesis, Oxford University (Jun 1981), http://www.cs.ox.ac.uk/files/9025/PRG-25.pdf, printed as: Programming Research Group, Technical Monograph 25
30. Jones, C.B.: Specification and design of (parallel) programs. In: Mason, R.E.A. (ed.) Information Processing 83, Proceedings of the IFIP 9th World Computer Congress, Paris, France, September 19-23, 1983. pp. 321–332. North-Holland/IFIP (1983)
31. Jung, R., Swasey, D., Sieczkowski, F., Svendsen, K., Turon, A., Birkedal, L., Dreyer, D.: Iris: Monoids and invariants as an orthogonal basis for concurrent reasoning. In: Rajamani, S.K., Walker, D. (eds.) Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015. pp. 637–650. ACM (2015), https://doi.org/10.1145/2676726.2676980
32. Kozen, D., Silva, A.: Practical coinduction. Math. Struct. Comput. Sci. 27(7), 1132–1152 (2017), https://doi.org/10.1017/S0960129515000493
33. Leino, K.R.M.: Dafny: An automatic program verifier for functional correctness. In: Clarke, E.M., Voronkov, A. (eds.) Logic for Programming, Artificial Intelligence, and Reasoning - 16th International Conference, LPAR-16, Dakar, Senegal, April 25-May 1, 2010, Revised Selected Papers. Lecture Notes in Computer Science, vol. 6355, pp. 348–370. Springer (2010), https://doi.org/10.1007/978-3-642-17511-4_20
34. Leino, K.R.M., Moskal, M.: Co-induction simply - automatic co-inductive proofs in a program verifier. In: Jones, C.B., Pihlajasaari, P., Sun, J. (eds.) FM 2014: Formal Methods - 19th International Symposium, Singapore, May 12-16, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8442, pp. 382–398. Springer (2014), https://doi.org/10.1007/978-3-319-06410-9_27
35. Leino, K.R.M., Tristan, J.B.: Dafny power user: Working with coinduction, extreme predicates, and ordinals (2023), technical report. https://leino.science/papers/krml285.html
36. Leroy, X.: Step carefully: Step-indexing techniques. Lecture Notes, Collège de France (Jan 2019), https://xavierleroy.org/CdF/2018-2019/8.pdf
37. Leroy, X., Grall, H.: Coinductive big-step operational semantics. Inf. Comput. 207(2), 284–304 (2009), https://doi.org/10.1016/j.ic.2007.12.004
38. Madiot, J., Pous, D., Sangiorgi, D.: Modular coinduction up-to for higher-order languages via first-order transition systems. Log. Methods Comput. Sci. 17(3) (2021), https://doi.org/10.46298/lmcs-17(3:25)2021
39. Mastorou, L., Papaspyrou, N., Vazou, N.: Coinduction inductively: mechanizing coinductive proofs in liquid haskell. In: Polikarpova, N. (ed.) Haskell '22: 15th ACM SIGPLAN International Haskell Symposium, Ljubljana, Slovenia, September 15 - 16, 2022. pp. 1–12. ACM (2022), https://doi.org/10.1145/3546189.3549922
40. Matthews, J.: Recursive function definition over coinductive types. In: Bertot, Y., Dowek, G., Hirschowitz, A., Paulin-Mohring, C., Théry, L. (eds.) Theorem Proving in Higher Order Logics, 12th International Conference, TPHOLs'99, Nice, France,

September, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1690, pp. 73–90. Springer (1999), https://doi.org/10.1007/3-540-48256-3_6
41. Milner, R.: Communication and concurrency. Prentice Hall (1989)
42. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, II. Inf. Comput. 100(1), 41–77 (1992), https://doi.org/10.1016/0890-5401(92)90009-5
43. Momigliano, A., Pientka, B., Thibodeau, D.: A case study in programming coinductive proofs: Howe's method. Math. Struct. Comput. Sci. 29(8), 1309–1343 (2019), https://doi.org/10.1017/S0960129518000415
44. Moschovakis, Y.N.: Elementary induction on abstract structures (Studies in logic and the foundations of mathematics). American Elsevier Pub. Co (1974)
45. Mousavi, M.R., Reniers, M.A., Groote, J.F.: SOS formats and meta-theory: 20 years after. Theor. Comput. Sci. 373(3), 238–272 (2007), https://doi.org/10.1016/j.tcs.2006.12.019
46. Nakano, H.: A modality for recursion. In: 15th Annual IEEE Symposium on Logic in Computer Science, Santa Barbara, California, USA, June 26-29, 2000. pp. 255–266. IEEE Computer Society (2000), https://doi.org/10.1109/LICS.2000.855774
47. Nieto, L.P.: The Rely-Guarantee method in Isabelle/HOL. In: Degano, P. (ed.) Programming Languages and Systems, 12th European Symposium on Programming, ESOP 2003, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2003, Warsaw, Poland, April 7-11, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2618, pp. 348–362. Springer (2003), https://doi.org/10.1007/3-540-36575-3_24
48. Nipkow, T., Paulson, L., Wenzel, M.: Isabelle/HOL — A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer (2002)
49. O'Hearn, P.W., Reynolds, J.C., Yang, H.: Local reasoning about programs that alter data structures. In: Fribourg, L. (ed.) Computer Science Logic, 15th International Workshop, CSL 2001. 10th Annual Conference of the EACSL, Paris, France, September 10-13, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2142, pp. 1–19. Springer (2001), https://doi.org/10.1007/3-540-44802-0_1
50. Park, D.M.R.: Concurrency and automata on infinite sequences. In: Deussen, P. (ed.) Theoretical Computer Science, 5th GI-Conference, Karlsruhe, Germany, March 23-25, 1981, Proceedings. Lecture Notes in Computer Science, vol. 104, pp. 167–183. Springer (1981), https://doi.org/10.1007/BFb0017309
51. Paulson, L.C.: A fixedpoint approach to implementing (co)inductive definitions. In: Bundy, A. (ed.) Automated Deduction - CADE-12, 12th International Conference on Automated Deduction, Nancy, France, June 26 - July 1, 1994, Proceedings. Lecture Notes in Computer Science, vol. 814, pp. 148–161. Springer (1994), https://doi.org/10.1007/3-540-58156-1_11
52. Pierce, B.C.: Types and Programming Languages. MIT Press (2002)
53. Pohjola, J.Å., Parrow, J.: Bisimulation up-to techniques for psi-calculi. In: Avigad, J., Chlipala, A. (eds.) Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs, Saint Petersburg, FL, USA, January 20-22, 2016. pp. 142–153. ACM (2016), https://doi.org/10.1145/2854065.2854080
54. Pous, D.: Complete lattices and up-to techniques. In: Shao, Z. (ed.) Programming Languages and Systems, 5th Asian Symposium, APLAS 2007, Singapore, November 29-December 1, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4807, pp. 351–366. Springer (2007), https://doi.org/10.1007/978-3-540-76637-7_24

55. Pous, D.: Coinduction all the way up. In: Grohe, M., Koskinen, E., Shankar, N. (eds.) Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016. pp. 307–316. ACM (2016), https://doi.org/10.1145/2933575.2934564
56. Pous, D.: Coinduction: Automata, formal proof, companions (invited paper). In: Roggenbach, M., Sokolova, A. (eds.) 8th Conference on Algebra and Coalgebra in Computer Science, CALCO 2019, June 3-6, 2019, London, United Kingdom. LIPIcs, vol. 139, pp. 4:1–4:4. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019), https://doi.org/10.4230/LIPIcs.CALCO.2019.4
57. Reynolds, J.C.: Separation logic: A logic for shared mutable data structures. In: 17th IEEE Symposium on Logic in Computer Science (LICS 2002), 22-25 July 2002, Copenhagen, Denmark, Proceedings. pp. 55–74. IEEE Computer Society (2002), https://doi.org/10.1109/LICS.2002.1029817
58. Rutten, J.J.M.M.: Universal coalgebra: a theory of systems. Theor. Comput. Sci. 249(1), 3–80 (2000), https://doi.org/10.1016/S0304-3975(00)00056-6
59. Sangiorgi, D.: On the proof method for bisimulation (extended abstract). In: Wiedermann, J., Hájek, P. (eds.) Mathematical Foundations of Computer Science 1995, 20th International Symposium, MFCS'95, Prague, Czech Republic, August 28 - September 1, 1995, Proceedings. Lecture Notes in Computer Science, vol. 969, pp. 479–488. Springer (1995), https://doi.org/10.1007/3-540-60246-1_153
60. Sangiorgi, D.: An abstract account of up-to techniques for inductive behavioural relations. In: Margaria, T., Steffen, B. (eds.) Leveraging Applications of Formal Methods, Verification and Validation. REoCAS Colloquium in Honor of Rocco De Nicola - 12th International Symposium, ISoLA 2024, Crete, Greece, October 27-31, 2024, Proceedings, Part I. Lecture Notes in Computer Science, vol. 15219, pp. 62–74. Springer (2024), https://doi.org/10.1007/978-3-031-73709-1_5
61. Sangiorgi, D., Walker, D.: The Pi-Calculus - a theory of mobile processes. Cambridge University Press (2001)
62. Smyth, M.B., Plotkin, G.D.: The category-theoretic solution of recursive domain equations. SIAM J. Comput. 11(4), 761–783 (1982), https://doi.org/10.1137/0211062
63. van Staden, S.: On rely-guarantee reasoning. In: Hinze, R., Voigtländer, J. (eds.) Mathematics of Program Construction - 12th International Conference, MPC 2015, Königswinter, Germany, June 29 - July 1, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9129, pp. 30–49. Springer (2015), https://doi.org/10.1007/978-3-319-19797-5_2
64. Tarski, A.: A lattice-theoretical fixpoint theorem and its applications. Pacific Journal of Mathematics 5, 285–309 (1955), https://api.semanticscholar.org/CorpusID:13651629
65. Timany, A., Krebbers, R., Dreyer, D., Birkedal, L.: A logical approach to type soundness. J. ACM 71(6), 40:1–40:75 (2024), https://doi.org/10.1145/3676954
66. Vafeiadis, V.: Modular fine-grained concurrency verification. Tech. Rep. UCAM-CL-TR-726, University of Cambridge, Computer Laboratory (Jul 2008), https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-726.pdf
67. Vafeiadis, V., Parkinson, M.J.: A marriage of rely/guarantee and separation logic. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR 2007 - Concurrency Theory, 18th International Conference, CONCUR 2007, Lisbon, Portugal, September 3-8, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4703, pp. 256–271. Springer (2007), https://doi.org/10.1007/978-3-540-74407-8_18

68. Vazou, N., Seidel, E., Vytiniotis, D., Peyton Jones, S., Jhala, R.: Liquid types. In: Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. pp. 159–170. ACM (2014)
69. Xia, L., Zakowski, Y., He, P., Hur, C., Malecha, G., Pierce, B.C., Zdancewic, S.: Interaction trees: representing recursive and impure programs in coq. Proc. ACM Program. Lang. 4(POPL), 51:1–51:32 (2020), https://doi.org/10.1145/3371119
70. Xu, Q., de Roever, W.P., He, J.: The rely-guarantee method for verifying shared variable concurrent programs. Formal Aspects Comput. 9(2), 149–174 (1997), https://doi.org/10.1007/BF01211617

APPENDIX

This Appendix provides further technical detail on concepts presented in the main paper:

- §A provides further technical details concerning the approximation landscape discussed in §6.
- §B provides informal proof sketches for our results.
- §C outlines a technical extension to the coinduction approximated inductively phenomenon, considering inductive approximations indexed by an arbitrary ordinals instead of $\mathbb{N}$.
- §D provides further supporting details on up-to operators (as presented in §5.2).
- §E presents a technical extension considering relational post-conditions for all concepts given in §2–§3.
- §F gives a detailed outline and discussion of the supporting Isabelle formalization, expanding on §7.

A More on the Coinduction Approximated Inductively Landscape

Here we give more details about the concepts from §6. Namely, we show how the rule-system criterion fails to precisely capture the sound approximation of (bi)similairties. Then we detail how the alternating format criterion fixes the problem, and how it generalizes the rule-system format.

Let us illustrate the notion of inductive approximations of (bi)similarities on a simple paradigmatic example: the similarity relation between two labeled transition systems. Recall that a *labeled transition system* (*LTS*) is a triple $\mathcal{L} = (\mathsf{State}, \mathsf{Act}, \Rightarrow)$ where: State, ranged over by s, s' etc., is a set of (items called) states, Act, ranged over by a, is a set of actions; and $\Rightarrow\colon (\mathsf{State}\times\mathsf{Act}\times\mathsf{State}) \to \mathsf{Bool}$ a transition relation; we will write $s \overset{a}{\Rightarrow} s'$ instead of $\Rightarrow\ (s, a, s')$. Given two LTSs $\mathcal{L}_1 = (\mathsf{State}_1, \mathsf{Act}, \Rightarrow_1)$ and $\mathcal{L}_2 = (\mathsf{State}_2, \mathsf{Act}, \Rightarrow_2)$ (having the same action set A), a *simulation* between them is a relation $R : (\mathsf{State}_1 \times \mathsf{State}_2) \to \mathsf{Bool}$ such that the following property holds: For all s_1 and s_2 such that $R\,(s_1, s_2)$ and all a, s_1' such that $s_1 \overset{a}{\Rightarrow}_1 s_1'$, there exists s_2' such that $s_2 \overset{a}{\Rightarrow}_2 s_2'$ and $R\,(s_1', s_2')$. The *similarity relation* between $\mathcal{L}_1$ and $\mathcal{L}_2$, denoted by $\succeq$, is defined as the largest simulation. This is the same as defining $\succeq : (\mathsf{State}_1 \times \mathsf{State}_2) \to \mathsf{Bool}$ coinductively by the following rule:

$$\frac{\forall a, s_1'.\ s_1 \overset{a}{\Rightarrow}_2 s_1' \longrightarrow (\exists s_2'.\ s_1 \overset{a}{\Rightarrow}_2 s_1' \wedge s_1' \succeq s_2')}{s_1 \succeq s_2}$$

Following Hennessy and Milner [25] and Abramsky [1], (who work with a bisimulation rather than a simulation), we define the sequence of relations $\succeq_n$:

$(\mathsf{State}_1 \times \mathsf{State}_2) \to \mathsf{Bool}$ *inductively* by the following rules:

$$s_1' \succeq_0 s_2' \qquad\qquad \frac{\forall a, s_1'.\ s_1 \stackrel{a}{\Rightarrow}_2 s_1' \longrightarrow (\exists s_2'.\ s_1 \stackrel{a}{\Rightarrow}_2 s_1' \wedge s_1' \succeq_n s_2')}{s_1 \succeq_{n+1} s_2}$$

An LTS $\mathcal{L} = (\mathsf{State}, \mathsf{Act}, \Rightarrow)$ is called *image-finite* provided that the number of states that a state can transit to via any fixed action is finite, namely $\forall s, a.\ \mathsf{finite}\ \{s' \mid s \stackrel{a}{\Rightarrow} s'\}$. The following result (or a variation of it via changing the concept of image-finiteness to fit the style of LTS and/or (bi)simulation considered) is frequently invoked to connect coinductive (bi)similarities with their approximations:

Thm 19. (E.g., [25, Thm. 2.1]) If $\mathcal{L}_2$ is image-finite, then $\succeq$ is the intersection of the $\succeq_n$'s, namely $s_1 \succeq s_2$ iff $\forall n.\ s_1 \succeq_n s_2$.

This is an instance of Thm. 8, since the operator $\mathsf{F}_{\succeq}$ underlying the greatest fixpoint definition of $\succeq$ (which takes any relation R between State_1 and State_2 to the relation $\mathsf{F}_{\succeq}R$ that holds for s_1 and s_2 just in case $\forall a, s_1'.\ s_1 \stackrel{a}{\Rightarrow}_2 s_1' \longrightarrow (\exists s_2'.\ s_1 \stackrel{a}{\Rightarrow}_2 s_1' \wedge R\ s_1'\ s_2')$ holds) is ω-cocontinuous whenever $\mathcal{L}_2$ is image-finite. This connection with ω-cocontinuity is a piece of folklore in process algebra, e.g., Hennessy and Milner [25] invoke it explicitly.

One can now ask whether Thm. 19 is also an instance of the rule-system criterion Thm. 14. The existential quantification in the definition of $\succeq$ prevents it from immediately falling under its scope. However, we can massage it to fit. Namely, for any LTS $\mathcal{L} = (\mathsf{State}, \mathsf{Act}, \Rightarrow)$, for any $s \in \mathsf{State}$ and $a \in \mathsf{Act}$ we define the set of continuations $\mathsf{Cont}\ (s, a) = \{s' \mid s \stackrel{a}{\Rightarrow} s'\}$; and for any $s \in S$, the set $\mathsf{ACont}\ s = \{(a, s') \mid s \stackrel{a}{\Rightarrow} s'\}$ of action-tagged continuations. Note that image-finiteness means $\forall s, a.\ \mathsf{finite}\,(\mathsf{Cont}\ (s, a))$; on the other hand, $\forall s.\ \mathsf{finite}\,(\mathsf{ACont}\ s)$ is a stronger property often called *finite branching* [1] (which would be ensured by image-finiteness plus the finiteness of Act). Then the rule defining $\succeq$ is equivalent to

$$\frac{\exists f \in \prod_{(a,s_1') \in \mathsf{ACont}\ s_1} \mathsf{Cont}\ (s_2, a).\ \forall (a, s_1') \in \mathsf{ACont}\ s_1.\ s_1' \succeq f\ (a, s_1')}{s_1 \succeq s_2}$$

We can therefore see that $\mathsf{F}_{\succeq}$ is given by a rule system, let us call it $R_{\succeq}$, consisting of: for each function $f \in \prod_{(a,s_1') \in \mathsf{ACont}\ s_1} \mathsf{Cont}\,(s_2, a)$, a rule of conclusion (s_1, s_2) and hypotheses $\{(s_1',\ f\,(a, s_1')) \mid (a, s_1') \in \mathsf{ACont}\ s_1\}$. Nevertheless, Thm. 19 cannot be obtained as an instance of our rule system Thm. 14, because the latter would not give a result that is precise enough: Even if $\mathcal{L}_2$ is image-finite (which is enough for Thm. 19), unless $\mathcal{L}_1$ is finite-branching the above rule system $R_{\succeq}$ is not in general backwards-finite (as required by Thm. 14). In conclusion,the rule-system criterion, Thm. 14, is not strong enough to capture the case of (bi)similarities at the desired level of precision.

On the other hand, Thm. 19 is an instance of Thm. 16:

Example 20. $\mathsf{alter}_{\mathsf{State}\times\mathsf{State},\Pi}\ \mathsf{F}_{\geq}$ holds, where $\Pi = \{\mathsf{State}, \mathsf{Act}\}$. Indeed, since $\mathsf{F}_{\geq}\ K\ (s_1, s_2)$ means $\forall a, s_1'.\ (a, s_1') \in \mathsf{Cont}_1\ s_1 \longrightarrow (\exists s_2'.\ s_2 \in \mathsf{ACont}_2(a, s_2').\ K\ s_1'\ s_2')$, we obtain $\mathsf{alter}_{\mathsf{State}\times\mathsf{State},\Pi}\ \mathsf{F}_{\geq}$ by backwards-applying the following sequence of rules: (Forall), (Exists) and (Comp). And the image-finiteness condition for the LTS $\mathcal{L}_2$ gives exactly the required finiteness condition for the existential quantifier, here $\forall a, s.\ \mathsf{finite}\ \mathsf{ACont}_2(a, s_2')$. □

And Thm. 16 applies not only to the similarity relation we defined as an example, but also the many similarities and bisimilarities defined in the literature, including strong and weak bisimularities for CCS [41] and for different variants of π-calculus [42,61], rooted bisimulation [21] etc. Van Glabbeek [20,19] surveys a large number of these relations. In each of these cases, the condition ensuring ω-continuity can be obtained by only looking at the space of the existential quantifier and requiring that to be finite—in other words, the criterion can be applied without any understanding of the (intended) semantics of the particular notion. Below we give another example:

Example 21. Given an LTS $\mathcal{L} = (\mathsf{State}, \mathsf{Act}, \Rightarrow)$ with an emphasized silent action $\tau \in \mathsf{Act}$, we write $\overset{\epsilon}{\Rightarrow}$ for the transitive closure of $\overset{\tau}{\Rightarrow}$. A *rooted bisimulation* is a symmetric relation K on State such that, for all s_1, s_2, if $K\ s_1\ s_2$ then the following holds:

$$\forall a, s_1'.\ s_1 \overset{a}{\Rightarrow} s_1' \longrightarrow \exists s_2', s_2''.\ s_2 \overset{a}{\Rightarrow} s_2'' \wedge s_2'' \overset{\epsilon}{\Rightarrow} s_2' \wedge K\ s_1\ s_2'' \wedge K\ s_1'\ s_2' \qquad (*)$$

The *rooted bisimilarity* is defined as the largest bisimulation, or equivalently as the fixed point of the operator F defined by $F\ K\ (s_1, s_2) \longleftrightarrow K\ (s_2, s_1) \vee (*)$. To apply our criterion (enabling sound inductive approximations for rooted bisimilarity), we need that the existential quantification happens in a finite space, i.e., that the set $\{(s_2', s_2'') \mid s_2 \overset{\epsilon}{\Rightarrow} s_2'' \wedge s_2'' \overset{a}{\Rightarrow} s_2'\}$ is finite. □

Finally, Thm. 16 straightforwardly generalizes Thm. 14:

Example 22. If Rl is a set of rules over a set A, then $\mathsf{alter}_{A,\Pi}\ \mathsf{F}_{Rl}$ holds, where $\Pi = \{A, \mathcal{P}(A)\}$. Indeed, since $\mathsf{F}_{Rl}\ K\ a$ means $\exists B \in \mathcal{P}(A).\ (B, a) \in Rl \wedge (\forall b.\ b \in B \longrightarrow K\ b)$, we obtain $\mathsf{alter}_{A,\Pi}\ \mathsf{F}_{Rl}$ by backwards-applying the following sequence of rules from the inductive definition of $\mathsf{alter}_{A,\Pi}$: (Exists), (Forall) and (Comp). And the backwards-finiteness condition gives exactly the required finiteneness condition for the existential quantifier, here $\forall a.\ \mathsf{finite}\ \{B \mid (B, a) \in Rl\}$. □

B Proof sketches

This section provides proof sketches for all our new results. Further detail is available in the formalization (App. F).

Proof idea of Thm. 1.
The proof proceeds by showing (1) $c \vDash_{\mathsf{XRH}} (P, R, G, Q)$ is equivalent to both (2) $c \vDash_{\mathsf{CJ}} (P, R, G, Q)$ and (3) $c \vDash_{\mathsf{VP}} (P, R, G, Q)$. We break this down by first showing the right-to-left implications (i.e. "(3) implies (1)" and "(2) implies (1)", which follow a similar pattern.

We begin by unfolding the definitions and then apply induction over the length of the trace with the premises that (i) the trace is valid (therefore non-empty), (ii) the rely-condition holds, and (iii) either reachability or safety hold by the pre-condition and (2) or (3) respectfully. The base case is trivial as the trace is required to be non-empty by (i).

The inductive case considers an $n+1$-length trace which is composed of an arbitrary head element t followed by an n-length trace tr. If $n = 0$, the singleton trace t satisfies the guarantee- and post-conditions by (ii) and (iii). Assuming $n > 0$ (i.e tr is non-empty thus satisfying (i)) then we have that t transitions to the first element of tr. We then show, (iv) if the transition is a command step, it preserves the guarantee from (iii) and (v) if it is an environment step, the rely is preserved (by (ii)). Combining these with the inductive hypothesis implies the guarantee- and post-condition hold for the $n+1$-length trace.

For the converse, we consider separately (1) implies (2) and (1) implies (3): To prove (1) implies (2), from $\mathsf{stepRel}^*_R\,(c, s)\,(c', s')$ we can obtain a trace tr such that the rely-condition holds on all environment steps, thus the guarantee condition holds for all $(c', s') \Rightarrow (c'', s'')$ and the post-condition Qs' holds.

The proof of (1) implies (3), follows by induction on the step-counter n. The base case is trivial, and the inductive step applies the Step rule for $\mathsf{safe}_{(R,G,Q)}\,n+1\,(c, s)$. Prefixing either an environment step $R\,s\,s'$ or a command step $(c, s) \Rightarrow (c', s')$ preserves safety by choosing an appropriate pre-condition to extend the trace obtained from (1).

This completes the equivalence between (1), (2) and (3).

To establish soundness of the proof system, it suffices to prove that (4) implies one of (1), (2), or (3). For (4) implies (2), detailed proofs for the seq and while constructs are provided in Figs. 5(a) and 9(a), and we otherwise refer to the formalization.

□

Note that in our formalization, the rule soundness proofs of the above are done for each semantics in isolation of each other (i.e. "(4) implies (1)", "(4) implies (2)", and "(4) implies (3)"), as otherwise proofs could be easily completed using our equivalence results when our aim was to compare the proof engineering required for each. App. F gives more details.

Proof sketch of Lemma 3.
This is an instance of Thm. 8. □

Proof of Thm. 2.
Follows immediately from Lemma 3 after expanding the definitions of $\vDash_{\mathsf{VP}}$ and $\vDash_{\mathsf{C}}$. □

Proof of Thm. 8.
Since, by Kleene's theorem, $\mathsf{gfp}_F = \bigsqcap_{i\in\mathbb{N}}(F^n\,\top)$, it suffices to prove $\forall n.\ \mathsf{lfp}_{F^\sharp}\ n = F^n\,\top$, which we do by induction on n.

The base case: From the fact that $\mathsf{lfp}_{F^\sharp}$ is a fixpoint of $F^\sharp$ and the definition of $F^\sharp$, we obtain $\mathsf{lfp}_{F^\sharp}\ 0 = F^\sharp\ \mathsf{lfp}_{F^\sharp}\ 0 = \top = F^0\,\top$.

The inductive step: Using the fact that $\mathsf{lfp}_{F^\sharp}$ is a fixpoint of $F^\sharp$, the definition of $F^\sharp$, and the inductive hypothesis, we obtain
$\mathsf{lfp}_{F^\sharp}\ (n+1) = F^\sharp\ \mathsf{lfp}_{F^\sharp}\ (n+1) = F\ (\mathsf{lfp}_{F^\sharp}\ n)F\ (F^n\,\top) = F^{n+1}\,\top$.

Incidentally, notice that we only used that $\mathsf{lfp}_{F^\sharp}$ is a fixpoint of $F^\sharp$, and not that it is the least fixpoint. In fact, it is easily proved by induction that $\mathsf{lfp}_{F^\sharp}$ is the unique fixpoint of $F^\sharp$. □

Proof sketch of Thm. 9.
First we prove the unary case (for $m = 1$), then we generalize to arbitrary m.

The unary case: We have $U : L \to L$, and, assuming U is weakly F-respectful, we must prove that, for all $k \in L$, $k \le F(U\ k)$ implies $k \le \mathsf{gfp}_F$.

Let $V : L \to L$ be $\lambda k.\ U\ k \sqcup k$.

We show that V is F-respectful. To this end, assume $k \le k'$ and $k \le F\ k'$. By the weak F-respectfulness of U, we have $U\ k \le F\ (U\ k' \sqcup k')$. Moreover, from $k \le F\ k'$ and the monotonicity of F, we obtain $k \le F\ (U\ k' \sqcup k')$. These give us $U\ k \sqcup k \le F\ (U\ k' \sqcup k')$, i.e., $V\ k \le F\ (V\ k')$.

Now we are ready to prove the desired fact. Assume $k \le F\ (U\ k)$, in particular $k \le F\ (V\ k)$. Using the known result about respectful operators [54], we obtain $k \le \mathsf{gfp}_F$, as desired.

(This is a particular case of a general phenomenon, namely that not only respectful operators, but also any operators smaller than them validate up-to coinduction.)

The general case: Now we assume $U : L^m \to L$. We let L' be the lattice L^m (with component-wise ordering), $G : L^m \to L^m$ be defined by $G\ \overline{k} = (F\ k_1, \ldots, F\ k_m)$ for all $\overline{k} = (k_1, \ldots, k_m) \in L^m$, and $V : L^m \to L^m$ defined by $V\ \overline{k} = (U\ \overline{k}, \ldots, U\ \overline{k})$ for all $\overline{k} \in L^m$. It is easy to check that V is weakly G-respectful because U is weakly F-respectful. Hence, using the result in the unary case, we obtain $\forall \overline{k} \in L^m.\ \overline{k} \le G\ (V\ \overline{k}) \longrightarrow \overline{k} \le \mathsf{gfp}_G$. Finally, using that $\mathsf{gfp}_G = (\mathsf{gfp}_F, \ldots, \mathsf{gfp}_F)$ and expanding the definitions of V and G, we obtain $\forall \overline{k} \in L^m.\ (\forall i \in \overline{1, m}.\ k_i \le F\ (U\ \overline{k})) \longrightarrow (\forall i \in \{1, \ldots, m\}.\ k_i \le \mathsf{gfp}_F)$, as desired. □

Proof of Thm. 10.
(1): We prove $U\ (\mathsf{lfp}_{F^\sharp} n, \ldots, \mathsf{lfp}_{F^\sharp} n) \le \mathsf{lfp}_{F^\sharp} n$ by induction on n. The base case is trivial. For the inductive case, assume $U\ (\mathsf{lfp}_{F^\sharp} n, \ldots, , \mathsf{lfp}_{F^\sharp} n) \le \mathsf{lfp}_{F^\sharp} n$. Then $U\ (\mathsf{lfp}_{F^\sharp}(n+1), \ldots, \mathsf{lfp}_{F^\sharp}(n+1)) \le \mathsf{lfp}_{F^\sharp}(n+1)$ is proved as follows:

$$\begin{array}{c} U(\mathsf{lfp}_{F^\sharp}(n+1), \ldots, \mathsf{lfp}_{F^\sharp}(n+1)) \le \\ F(\mathsf{lfp}_{F^\sharp} n \vee U(\mathsf{lfp}_{F^\sharp} n, \ldots, \mathsf{lfp}_{F^\sharp} n)) = \\ F(\mathsf{lfp}_{F^\sharp} n) = \mathsf{lfp}_{F^\sharp}(n+1) \end{array}$$

Above, the inequality follows from F-respectfulness by taking, for each $i \in \{1,\ldots,m\}$, that $k_i = \mathsf{lfp}_{F^\sharp}(n+1)$, $k'_i = \mathsf{lfp}_{F^\sharp} n$, and the $\mathsf{lfp}_{F^\sharp}(n+1) \leq \mathsf{lfp}_{F^\sharp} n$. The first part of the inequality follows from the inductive hypothesis.

(2): Assume $\forall i \in \{1,\ldots,m\}.\ k_i \leq F\,(U\,(k_1,\ldots,k_m))$. We show $\forall i \in \{1,\ldots,m\}.\, k_i \leq \mathsf{gfp}_F = \bigsqcap_n(\mathsf{lfp}_{F^\sharp}\, n)$, i.e., $\forall n.\, \forall i \in \{1,\ldots,m\}.\, k_i \leq \mathsf{lfp}_{F^\sharp}\, n$, by induction on n. The base case is trivial. For the inductive case, assume $\forall i \in \{1,\ldots,m\}.\, k_i \leq \mathsf{lfp}_{F^\sharp} n$ and let $i \in \{1,\ldots,m\}$. Then $k \leq \mathsf{lfp}_{F^\sharp}(n+1)$ is proved as follows:

$$\begin{array}{c} k_i \leq F\,(U\,(k_1,\ldots,k_m)) \leq F\,(U\,(\mathsf{lfp}_{F^\sharp}\, n,\ldots,\mathsf{lfp}_{F^\sharp}\, n)) \\ \leq F(\mathsf{lfp}_{F^\sharp}\, n) = \mathsf{lfp}_{F^\sharp}\,(n+1) \end{array}$$

Above, in the second inequality we used the inductive hypothesis and the monotonicity of F and U, and in the third inequality we used the F-approximation-preservation of U and the monotonicity of F. □

Proof of Lemma 11.
Let $n \in \mathbb{N}$ and $a \in A$. We have the following chain of equivalences:

$$\begin{array}{c} \mathsf{F}_{Rl^\sharp}\ K\ (n,a) \\ \longleftrightarrow \text{ (by the definition of } \mathsf{F}_{\_}) \\ \exists B'.\ (B',(n,a)) \in Rl^\sharp \ \wedge\ (\forall (n',b) \in B'.\ K\ (n',b)) \\ \longleftrightarrow \text{ (by the definition of } Rl^\sharp) \\ \exists B'.\ ((n=0 \wedge B = \varnothing) \vee (\exists n', B.\ n = n'+1 \wedge B' = \{n'\} \times B \wedge \\ \qquad (B,a) \in Rl)) \ \wedge\ (\forall (n',b) \in B'.\ K\ (n',b)) \\ \longleftrightarrow \text{ (by logic)} \\ n = 0 \ \vee\ (\exists n', B.\ n = n'+1 \wedge (B,a) \in Rl \wedge (\forall b \in B.\ K\ (n',b))) \\ \longleftrightarrow \text{ (by the definition of } \mathsf{F}_{Rl}) \\ n = 0 \vee (\exists n'.\ n = n'+1 \wedge \mathsf{F}_{Rl}\ (\lambda a.\ K\ (n',a))\ a) \\ \longleftrightarrow \text{ (by the definition of } (\mathsf{F}_{Rl})^\sharp) \\ (\mathsf{F}_{Rl})^\sharp\ (\lambda n, a.\ K(n,a))\ n\ a \end{array}$$

This proves $\mathsf{F}_{Rl^\sharp}\ K\ n\ a = (\mathsf{F}_{Rl})^\sharp\ (\lambda n, a.\ K(n,a))\ n\ a$, as desired. □

Proof of Prop. 13.
This follows from Thm. 16 because, as shown in Example 22, the backwards-finite rule system format is a particular case of the quantifier alternation format. □

Proof sketch of Thm. 14
Since, by Lemma 13, F_{Rl} is ω-cocontinuous, from Thm. 8 we obtain $\mathsf{gfp}_{Rl} = \bigsqcap_{n\in\mathbb{N}}(\mathsf{lfp}_{(F_{Rl})^\sharp}\, n)$, hence $(*)\ \mathsf{gfp}_{Rl}\ a \longleftrightarrow \forall_{n\in\mathbb{N}}(\mathsf{lfp}_{(F_{Rl})^\sharp}\, n\ a)$.

On the other hand, from Lemma 11, by an easy induction we obtain that $\mathsf{lfp}_{(F_{Rl})^\sharp} = \lambda n, a.\, \mathsf{lfp}_{Rl^\sharp}\,(n,a)$. This, together with (*), gives us $\mathsf{gfp}_{Rl}\ a \longleftrightarrow \forall\ n \in \mathbb{N}.\, \mathsf{lfp}_{Rl^\sharp}\,(n,a)$, as desired. □

Proof sketch of Thm. 16.
In the case of the lattice of predicates $A \to \mathsf{Bool}$, the ω-cocontinuity of an operator F means that, for all decreasing chains $(K_n)_{n\in\mathbb{N}}$ of predicates $K_n : A \to \mathsf{Bool}$, $F\ (\forall n.\ K_n)$ implies $\forall n \in \mathbb{N}.\ F\ (K_n)$. So, if the expression defining F is of the form $\forall\exists\forall\exists\ldots$ one must be able to push the innermost $\forall n \in \mathbb{N}$ from

$F\ (\forall n \in \mathbb{N}.\ K_n)$ all the way to the top of the expression. While $\forall$ (sub)commutes unproblematically with $\forall$, it does not in general (sub)commute with $\exists$, in that in general an $\forall n \in \mathbb{N}.\ \exists p \in P$ statement does not imply its quantifier-permuted, $\exists p \in P.\ \forall n \in \mathbb{N}$ version. However, this implication is true provided the existential quantification is in a finite space, and the quantified body is decreasing in n. This is because, by the pigeonhole principle, from the $\forall n \in \mathbb{N}.\ \exists p \in P$ statement we find an existential witness p_0 that works for an infinite set of numbers n; and due to decreasingness in n, working for a set of numbers also means working for all numbers.

The above forms the basis for a proof of the ω-cocontinuity of F by rule induction on the definition of $\mathsf{alter}\ F$ (after also proving by induction that F is monotonic). □

Proof sketch of Lemma 17.
This is an instance of Thm. 16. □

Proof of Thm. 18.
Follows immediately from Lemma. 17 after expanding the definitions of $\vDash_{\mathsf{Sep,VP}}$ and $\vDash_{\mathsf{Sep,C}}$. □

C Beyond finitarily approximated predicates

If we relax the requirement that the inductive approximations are indexed by $\mathbb{N}$ to an arbitrary ordinal κ, then the the inductively approximated coinductive phenomenon becomes truly pervasive. This results in a small technical extension of the criteria discussed in §4 and §6.

Indeed, let us say that $F : L \to L$ is *κ-cocontinuous* provided it commutes with the infima of decreasing κ-chains $(a_i)_{i \in \kappa}$, in that $F\ (\bigsqcap_{i \in \kappa} a_i) = \bigsqcap_{i \in \kappa} (F\ a_i)$. Similarly to the case of ω-continuity, any monotonic operator $F : L \to L$ already supra-commutes with any decreasing κ-chain. Then any monotonic operator $F : L \to L$ on a complete lattice is κ-cocontinuous for some ordinal $\kappa \leq |L|$. (A further extension could even relax the completeness of L, assuming $(L, \leq)$ is a κ-cocomplete partial order, i.e. has an infima of κ-chains.)

In what follows, we assume κ to be an infinite ordinal. We generalize the definition of the inductive approximation structure defining $F^\sharp : (\kappa \to L) \to (\kappa \to L)$ by:

- $F^\sharp\ f\ 0 = \top$;
- $F^\sharp\ f\ (\gamma + 1) = F\ (f\ \gamma)$ (for successor ordinals);
- $F^\sharp\ f\ (\bigsqcup_{\alpha < \gamma}) = \bigsqcap_{\alpha < \gamma}\ (f\ \alpha)$ (for limit ordinals).

As usual, $\mathsf{lfp}_{F^\sharp}$ denotes the least fixpoint of $F^\sharp$ (a monotonic operator on the complete lattice $\kappa \to L$). Now, the transfinite generalization of Thm. 8 follows from Cousot and Cousot's constructive version of Tarski's theorem [15].

Thm 23. If L is a complete lattice and $F : L \to L$ is κ-cocontinuos, $\mathsf{gfp}_F = \bigsqcap_{n \in \mathbb{N}} \mathsf{lfp}_{F^\sharp} n$ is the greatest fixpoint of F.

To also consider the rule system extensions of §6, we generalize the notion of backwards finiteness for a set of rules $Rl \subseteq \mathcal{P}(A) \times A$ as follows: We call Rl *backwards-κ-small* provided that each $a \in A$ has only a $|\kappa|$ number of rules that backwards-apply to it, namely $\forall a \in A,\ |\{B \subseteq A \mid (B, a) \in Rl\}| \leq |\kappa|$ (noting that for any cardinal c, the conditions $c \leq |\kappa|$ and $c \leq \kappa$ are equivalent). Thus Prop. 13 and Thm. 14 generalize as follows:

Thm 24. If $Rl \subseteq \mathcal{P}(A) \times A$ is backwards-$|\kappa|$-small, F_{Rl} is κ-cocontinuous, hence $\forall a \in A. \mathsf{gfp}_{Rl}\ a \longleftrightarrow \forall\ \gamma \in \kappa.\ \mathsf{lfp}_{Rl^\sharp}\ n\ a$.

Moreover, the predicate $\mathsf{alter}_{\kappa,A,\Pi}\ F$ from §6.3 is generalized from finiteness to κ-smallness by replacing, in the (Exists) rule, the condition $\forall a.\ \mathsf{finite}\ \{p \mid L\ p\ a\}$ with $\forall a.\ |\{p \mid L\ p\ a\}| \leq |\kappa|$. Thus, Thm. 16 generalizes as follows:

Thm 25. Assume that A, Π and $F : (A \to \mathsf{Bool}) \to (A \to \mathsf{Bool})$ are such that $\mathsf{alter}_{\kappa,A,\Pi}\ F$ holds. Then F is κ-cocontinuous, hence $\forall a \in A. \mathsf{gfp}_F\ a \longleftrightarrow \forall\ \gamma \in \kappa.\ \mathsf{lfp}_{F^\sharp}\ \gamma\ a$.

In particular, Thm. 25 instantiates to the transfinite approximation of (bi)simulations (as discussed in App. A). Namely, we generalize the definition of the sequence of relations $(\succeq_n)_{n\in\mathbb{N}}$ to a κ-transfinite sequence $(\succeq_\gamma)_{\gamma\in\mathbb{N}}$ by adding an inductive rule for limit ordinals $\gamma < \kappa$: $\dfrac{\forall \alpha < \gamma.\ s_1 \succeq_\alpha s_2}{s_1 \succeq_\gamma s_2}$. Then the Thm. 19 instance of Thm. 16 generalizes to transition systems that are not image-finite (as originally given in §6.3):

Thm 26. If $\mathcal{L}_2 = (\mathsf{State}_2, \mathsf{Act}, \Rightarrow_2)$ is image-$|\kappa|$-small (i.e., $\forall s, a.\ |\{s' \mid s \overset{a}{\Rightarrow}_2 s'\}| < |\kappa|$), then $\succeq$ is the intersection of the $\succeq_\gamma$'s, namely $s_1 \succeq s_2$ iff $\forall \gamma < \kappa.\ s_1 \succeq_\gamma s_2$.

D More Details on the Up-To Operators

In §5.2 we discussed up-to coinduction, both generally and in the context of RG reasoning. This section presents further technical details on defining respectful up-to operators, including incorporating greater flexibility beyond what was required of the up-to coinductive while proof.

D.1 On the choice of the operators

Looking at the main paper's Fig. 7, one may notice some discrepancies when comparing $\mathsf{U}^{\mathsf{seq}}_{(R,G,P,c_2)}$, the operator for seq, and $\mathsf{U}^{\mathsf{if}}_{(R,t,c_1,c_2)}$, the operator for if.

Specifically, $\mathsf{U}^{\mathsf{seq}}_{(R,G,P,c_2)}$ has c_2 as a parameter and quantifies existentially over c_1, whereas $\mathsf{U}^{\mathsf{if}}_{(R,t,c_1,c_2)}$ has both c_1 and c_2 as parameters. Furthermore, $\mathsf{U}^{\mathsf{if}}_{(R,t,c_1,c_2)}$ is a binary operator, which is as expected given if takes two commands as input, and its safety preservation rule has as hypotheses the soundness for

both c_1 and c_2. On the other hand, $\mathsf{U}^{\mathsf{seq}}_{(R,G,P,c_2)}$ is unary in spite of its safety preservation having similar features, i.e., one hypothesis for c_1 and one for c_2.

There are two reasons for this. Firstly, an operator is more convenient to use when fixing as many parameters as possible, so we optimize our definitions for these characteristics.[1] In the case of seq the operator would also not be weakly respectful if c_1 was fixed—reflecting the fact that c_1 needs to change in the coinductive proof of safety preservation as given in Fig. 5 (from which we are mining the (weak) respectfulness proof as discussed in §5.2).

Secondly, unlike for $\mathsf{U}^{\mathsf{if}}_{(R,t,c_1,c_2)}$, here the hypothesis for c_1 does not have the same parameters as the conclusion and c_2 hypothesis—in other words we assume $\mathsf{safe}_{(R,G,\,P\,)}\,c_1$, while we assume $\mathsf{safe}_{(R,G,\,Q\,)}\,c_2$ and conclude $\mathsf{safe}_{(R,G,\,Q\,)}\,(\mathsf{seq}\ c_1\ c_2)$. Consequently, if we wish to work with a safety-underlying operator $\mathsf{F}_{\_,\_,\_}$ with fixed parameters (the most sensible choice for proofs), we must: take our up-to operator to be unary, be up-to for $\mathsf{F}_{R,G,Q}$, and integrate the $\mathsf{safe}_{(R,G,P)}\,c_1$ hypothesis as part of its definition. This necessarily yields the current $\mathsf{U}^{\mathsf{seq}}_{(R,G,P,c_2)}$ rule.

An alternative would have been to work with a variable P, noting that $\mathsf{safeC}_{(R,G,\_)}$ would be the greatest fixpoint of $\mathsf{F}_{(R,G)}$, the operator that takes any $K : (\mathsf{State} \to \mathsf{Bool}) \to \mathsf{Com} \times \mathsf{State} \to \mathsf{Bool}$, and any Q such that $\mathsf{F}_{(R,G,Q)}\,(K\,Q)$. Then we would have been able to prove that the binary operator $\mathsf{V}^{\mathsf{seq}}_{(R,G,P,c_2)}$ defined by

$$\begin{aligned}\mathsf{V}^{\mathsf{seq}}_{(R,G,P,c_2)}\,(K_1,K_2)\,Q\,(c,s)\ = \exists c_1.\ c = \mathsf{seq}\ c_1\ c_2\ \wedge\\ K_1\,P\,(c_1,s)\ \wedge\ (\forall s'.\ P\,s' \longrightarrow K_2\,Q\,(c_2,s'))\end{aligned}$$

is $\mathsf{F}_{(R,G)}$-respectful provided $\mathsf{refl}\ G$ holds. However, losing the uniformity in the third parameter would have prevented us from producing a proof by up-to coinduction for the safety preservation of while, hence our preference of the original rule.

D.2 A generalization of (weak) respectfulness

A better and more general alternative is to abandon the single-lattice / single-operator view, and extend the notion of (weak) respectfulness to multiple lattices.

Namely, let L and M be two complete lattices and let $F : L \to L$ and $G : M \to M$ be two monotonic operators. We say an operator $U : L \to M$ is (F,G)-*respectful* when, for all $k, k' \in L$, $k \leq k'$ and $k \leq F\ k'$ implies $U\ k \leq G\,(U\ k')$.

Similarly to the case of (weak) F-respectfulness, (F,G)-respectfulness generalizes the notion of proving the preservation of greatest fixpoints,

[1] In fact, it is easy to check that the (weak) respectfulness of an operator with fixed parameters implies (weak) respectfulness of the operator's modification with the parameters turned into existentials. For example, the fact that $\mathsf{U}^{\mathsf{if}}_{(R,t,c_1,c_2)}$ is weakly $\mathsf{F}_{(R,G,Q)}$-respectful implies that the operator $\mathsf{V}^{\mathsf{if}}_{(R,t)}$ defined by $\mathsf{V}^{\mathsf{if}}_{(R,t)}\,(K_1,K_2)\,(c,s) = (\exists c_1, c_2.\ \mathsf{U}^{\mathsf{if}}_{(R,t,c_1,c_2)}\,(K_1,K_2)\,(c,s))$ is also weakly $\mathsf{F}_{(R,G,Q)}$-respectful.

$U\ \mathsf{gfp}_F \leq \mathsf{gfp}_G$, by (strong) coinduction, i.e. (F, G)-respectfulness generalizes the property $U\ \mathsf{gfp}_F \leq G\ (U\ \mathsf{gfp}_F)$.

Note that (F, G)-respectfulness instantiates to unary F-respectfulness by taking $M = L$ and $G = F$, and to unary weak F-respectfulness by taking $M = L$ and $G = \lambda k.\ k \sqcup F\ k$ (and similarly for m-ary respectfulness). Moreover, (F, G)-respectfulness enjoys the same nice properties as (weak) F-respectfulness, including closure under composition and suprema. Now, with the freedom to use multiple lattices and operators, we can have a binary version of the sequential composition operator, defining $\mathsf{U}^{\mathsf{seq}}_{(R,G,P,c_2)} : \mathsf{L}_{(R,G,P)} \times \mathsf{L}_{(R,G,Q)} \to \mathsf{L}_{(R,G,Q)}$ as follows: $\mathsf{U}^{\mathsf{seq}}_{(R,G,P,c_2)}\ (K_1, K_2)\ (c, s) = \exists c_1.\ c = \mathsf{seq}\ c_1\ c_2 \wedge K_1\ (c_1, s) \wedge (\forall s'.\ P\ s' \longrightarrow K_2\ (c_2, s'))$

Then $\mathsf{U}^{\mathsf{seq}}_{(R,G,P,c_2)}$ is $(\langle \mathsf{F}_{(R,G,P)}, \mathsf{F}_{(R,G,Q)} \rangle, \mathsf{F}_{(R,G,Q)})$-respectful whenever G is reflexive, where $\langle \mathsf{F}_{(R,G,P)}, \mathsf{F}_{(R,G,Q)} \rangle : \mathsf{L}_{(R,G,P)} \times \mathsf{L}_{(R,G,Q)} \to \mathsf{L}_{(R,G,P)} \times \mathsf{L}_{(R,G,Q)}$ denotes the operator $\lambda(K_1, K_2).\ (\mathsf{F}_{(R,G,P)}\ K_1, \mathsf{F}_{(R,G,Q)}\ K_2)$.

D.3 Operators for the other language constructs

Fig. 11 shows the up-to operators corresponding to the language constructs not covered by the main paper's Fig. 7, namely atom, while and par. Moreover, Fig. 12 summarizes the coinductive enhancements for these operators, extracted from their safety preservation proofs (along the lines we described in §5.2). Note that the operator for while is respectful up-to W, where W is the operator used in the safety preservation proof of while (namely $W = 1_{\mathsf{L}_{(R,G,Q)}} \sqcup (\mathsf{U}^{\mathsf{if}}_{(R,t,\mathsf{seq}\ c\ (\mathsf{while}\ t\ c),\mathsf{done})} \circ (\mathsf{U}^{\mathsf{seq}}_{(R,G,P,\mathsf{while}\ t\ c)}, \lambda K.\ \mathsf{U}^{\mathsf{done}}_Q)))$.

$$\begin{aligned}
\mathsf{U}^{\mathsf{Atom}}_{(R,G,Q)}\ (c, s) &= c \in \mathsf{Atom} \wedge (\forall s', s''.\ R^*\ s\ s' \wedge R^*\ (\mathsf{evalA}\ c\ s')\ s'' \longrightarrow Q\ s'') \wedge \\
&\quad\ (\forall s'.\ R^*\ s\ s' \longrightarrow G\ s'\ (\mathsf{evalA}\ c\ s')) \\
\mathsf{U}^{\mathsf{while}}_{(P,t,c)}\ K\ (c', s) &= c' = \mathsf{while}\ t\ c \wedge P\ s \wedge (\forall s'.\ P\ s' \wedge \mathsf{evalT}\ t\ s' \longrightarrow K\ (c, s')) \\
\mathsf{U}^{\mathsf{par}}\ K_1\ K_2\ (c, s) &= \exists c_1, c_2.\ c = \mathsf{par}\ c_1\ c_2 \wedge K_1\ (c_1, s) \wedge K_2\ (c_2, s)
\end{aligned}$$

Fig. 11: Operators underlying atoms, while and par

E Extending the Formal RG Framework with Relational Post-Conditions

This appendix provides details on the adaption of §2, §3 and §3.2 to consider relational post-conditions. We first provide background on the relational variant of the RG proof system, before giving an overview of the changes to the RG semantics, including our new coinductive approach. We then discuss the impact of this change on the soundness proofs and the proof comparison analysis.

	Safety preservation needed for coinductive soundness	Enhanced version
$a \in \mathsf{Atom}$	$\dfrac{\begin{array}{l}(1)\ \mathsf{stable}\ Q\ R \\ (2)\ \forall s', s''.\ R^*\ s\ s' \wedge R^*\ (\mathsf{evalA}\ a\ s')\ s'' \longrightarrow Q\ s'' \\ (3)\ \forall s'.\ R^*\ s\ s' \longrightarrow G\ s'\ (\mathsf{evalA}\ a\ s')\end{array}}{\mathsf{safeC}_{(R,G,Q)}\ (a, s)}$ i.e., $\dfrac{(1)\ \mathsf{stable}\ Q\ R}{\mathsf{U}^{\mathsf{Atom}}_{(R,G,Q)} \le \mathsf{safeC}_{(R,G,Q)}}$ proved by strong coinduction in the form: $\dfrac{(1)\ \mathsf{stable}\ Q\ R}{\mathsf{U}^{\mathsf{Atom}}_{(R,G,Q)} \le \mathsf{F}_{(R,G,Q)}\ (\mathsf{U}^{\mathsf{Atom}}_{(R,G,Q)} \sqcup \mathsf{safe}_{(R,G,Q)})}$	No enhancement needed for 0-ary operators such as $\mathsf{U}^{\mathsf{Atom}}_{(R,G,Q)}$.
while	$\dfrac{\begin{array}{l}(1)\ P \sqcap (\neg\ (\mathsf{evalT}\ t)) \le Q\ \wedge\ \mathsf{stable}\ P\ R\ \wedge\ \mathsf{stable}\ Q\ R\ \wedge\ \mathsf{refl}\ G \\ (2)\ P\ s \\ (3)\ \forall s.\ P\ s \wedge \mathsf{evalT}\ t\ s \longrightarrow \mathsf{safeC}_{(R,G,P)}(c, s)\end{array}}{\mathsf{safeC}_{(R,G,Q)}\ (\mathsf{while}\ t\ c, s)}$ i.e., $\dfrac{(1)\ P \sqcap (\neg\ (\mathsf{evalT}\ t)) \le Q\ \wedge\ \mathsf{stable}\ P\ R\ \wedge\ \mathsf{stable}\ Q\ R\ \wedge\ \mathsf{refl}\ G}{\mathsf{U}^{\mathsf{while}}_{(P,t,c)}\ \mathsf{safeC}_{(R,G,Q)} \le \mathsf{safeC}_{(R,G,Q)}}$ proved by U-up-to coinduction in the form: $\dfrac{(1)\ P \sqcap (\neg\ (\mathsf{evalT}\ t)) \le Q\ \wedge\ \mathsf{stable}\ P\ R\ \wedge\ \mathsf{stable}\ Q\ R\ \wedge\ \mathsf{refl}\ G}{\mathsf{U}^{\mathsf{while}}_{(P,t,c)}\ \mathsf{safeC}_{(R,G,Q)}\ \le\ \mathsf{F}_{(R,G,Q)}(W\ (\mathsf{U}^{\mathsf{while}}_{(P,t,c)}\ \mathsf{safeC}_{(R,G,Q)}))}$	$\dfrac{\begin{array}{l}(1)\ P \sqcap (\neg\ (\mathsf{evalT}\ t)) \le Q\ \wedge\ \mathsf{stable}\ P\ R \wedge \\ \mathsf{stable}\ Q\ R\ \wedge\ \mathsf{refl}\ G \\ K \le K' \qquad K \le \mathsf{F}_{(R,G,Q)}\ K'\end{array}}{\mathsf{U}^{\mathsf{while}}_{(P,t,c)}\ K \le \mathsf{F}_{(R,G,Q)}(W\ (\mathsf{U}^{\mathsf{while}}_{(P,t,c)}\ K'))}$
par	$\dfrac{\begin{array}{c}(1)\ R \sqcup G_2 \le R_1 \wedge R \sqcup G_1 \le R_2 \wedge G_1 \sqcup G_2 \le G \wedge Q_1 \sqcap Q_2 \le Q \wedge\ \mathsf{refl}\ G \\ (2)\ \mathsf{safeC}_{(R_1,G_1,Q_1)}(c_1, s) \\ (3)\ \mathsf{safeC}_{(R_2,G_2,Q_2)}(c_2, s)\end{array}}{\mathsf{safeC}_{(R,G,Q)}\ (\mathsf{par}\ c_1\ c_2, s)}$ i.e., $\dfrac{(1)\ R \sqcup G_2 \le R_1 \wedge R \sqcup G_1 \le R_2 \wedge G_1 \sqcup G_2 \le G \wedge Q_1 \sqcap Q_2 \le Q \wedge\ \mathsf{refl}\ G}{\mathsf{U}^{\mathsf{par}}(\mathsf{safeC}_{(R_1,G_1,Q_1)}, \mathsf{safeC}_{(R_2,G_2,Q_2)}) \le \mathsf{safeC}_{(R,G,Q)}}$ proved by strong coinduction in the form: $\dfrac{(1)\ R \sqcup G_2 \le R_1 \wedge R \sqcup G_1 \le R_2 \wedge G_1 \sqcup G_2 \le G \wedge Q_1 \sqcap Q_2 \le Q \wedge\ \mathsf{refl}\ G}{\mathsf{U}^{\mathsf{par}}(\ \mathsf{safeC}_{(R_1,G_1,Q_1)}\ ,\ \mathsf{safeC}_{(R_2,G_2,Q_2)}\ )\ \le\ \mathsf{F}_{(R,G,Q)}(\mathsf{safe}_{(R,G,Q)}\ \sqcup\ \mathsf{U}^{\mathsf{par}}\ (\ \mathsf{safeC}_{(R_1,G_1,Q_1)}\ ,\ \mathsf{safeC}_{(R_2,G_2,Q_2)}\ ))}$	$\dfrac{\begin{array}{l}(1)\ R \sqcup G_2 \le R_1 \wedge R \sqcup G_1 \le R_2 \wedge \\ G_1 \sqcup G_2 \le G \wedge Q_1 \sqcap Q_2 \le Q \wedge\ \mathsf{refl}\ G \\ K_1 \le K'_1 \qquad K_1 \le \mathsf{F}_{(R,G,Q)}\ K'_1 \\ K_2 \le K'_2 \qquad K_2 \le \mathsf{F}_{(R,G,Q)}\ K'_2\end{array}}{\mathsf{U}^{\mathsf{par}}\ (\ K_1\ ,\ K_2\ ) \le \mathsf{F}_{(R,G,Q)}(\mathsf{safe}_{(R,G,Q)}\ \sqcup \mathsf{U}^{\mathsf{par}}\ (\ K'_1\ ,\ K'_2\ ))}$

Fig. 12: Coinductive enhancements for atoms, while and par

$$\frac{c \vdash (P', R', G', Q') \quad P \le P' \quad R \le R' \quad G' \le G \quad Q' \le Q}{c \vdash (P, R, G, Q)}\ (\mathsf{Mono})$$

$$\frac{\mathsf{stable}_2\ Q\ R \qquad (\lambda s s'.Ps) \le Q}{\mathsf{done} \vdash (P, R, G, Q)}\ (\mathsf{DoneRG})$$

$$\frac{\mathsf{stable}\ P\ R \qquad \mathsf{stable}_2\ Q\ R \qquad (\lambda s, s'.\ P\ s \wedge \mathsf{evalA}\ a\ s\ s') \le Q \qquad (\lambda s, s'.\ P\ s \wedge \mathsf{evalA}\ a\ s\ s') \le G}{a \vdash (P, R, G, Q)}\ (\mathsf{AtomRG})$$

$$\frac{c_1 \vdash (P, R, G, Q_1) \qquad c_2 \vdash (P', R, G, Q_2) \qquad (\lambda s'.\exists s.P\ s \wedge Q_1\ s\ s') \le P' \qquad Q_1 \circ Q_2 \le Q \qquad \mathsf{refl}\ G}{\mathsf{seq}\ c_1\ c_2 \vdash (P, R, G, Q)}\ (\mathsf{SeqRG})$$

$$\frac{c_1 \vdash (P \sqcap (\mathsf{evalT}\ t), R, G, Q) \qquad c_2 \vdash (P \sqcap (\neg\ (\mathsf{evalT}\ t)), R, G, Q) \qquad \mathsf{stable}\ P\ R \qquad R \circ Q \le Q \qquad \mathsf{refl}\ G}{\mathsf{if}\ t\ c_1\ c_2 \vdash (P, R, G, Q)}\ (\mathsf{IfRG})$$

$$\frac{c \vdash (\ P'\ , R, G, \ Q'\ ) \qquad P \sqcap (\mathsf{evalT}\ t) \le P' \qquad (\lambda s'.\exists s.(P' \sqcap (\mathsf{evalT}\ t))\ s \wedge Q\ s\ s') \le P \qquad P \sqcap (P \sqcap (\mathsf{evalT}\ t) \sqcap Q')^* \sqcap \neg\ (\mathsf{evalT}\ t) \le Q \qquad \mathsf{stable}\ P\ R \qquad \mathsf{stable}_2\ Q'\ R \qquad \mathsf{stable}_2\ Q\ R \qquad \mathsf{refl}\ G}{\mathsf{while}\ t\ c \vdash (P, R, G, Q)}\ (\mathsf{WhileRG})$$

$$\frac{c_1 \vdash (P_1, R_1, G_1, Q_1)\ \ c_2 \vdash (P_2, R_2, G_2, Q_2) \qquad P \le P_1 \sqcap P_2\ \ R \sqcup G_2 \le R_1\ \ R \sqcup G_1 \le R_2\ \ G_1 \sqcup G_2 \le G\ \ Q_1 \sqcap Q_2 \le Q\ \ \mathsf{refl}\ G}{\mathsf{par}\ c_1\ c_2 \vdash (P, R, G, Q)}\ (\mathsf{ParRG})$$

Fig. 13: Relational RG proof system

E.1 A relational proof system

A RG clause (P, R, G, Q) with a *relational post-condition*, requires the post-condition $Q : \mathsf{State} \to \mathsf{State} \to \mathsf{Bool}$ to be a binary predicate, thus considers both the first and final state in a program. As discussed in §3.1, this was the approach taken by Jones in his seminal work [29,30], as well as in the original Coleman-Jones paper [14].

In Fig. 13 we present the relational post-condition variant of the RG proof system from Fig. 2, with the differences highlighted in gray. The majority of modifications are small, however some have a notable impact on the complexity of soundness proofs. Notably:

- The seq rule must differentiate between the binary post-condition of the first command Q_1 and the unary pre-condition of the second P', thus the hypotheses must now relate Q_1 and P as well as Q_1, Q_2 and Q.
- The if rule must introduce a composition assumption between Q and R to mitigate the impact of any initial environment steps on the initial state used by Q.

– The while rule requires the addition of auxiliary pre- and post-conditions for the inner command c (P' and Q' respectively), and new assumptions that relate these to the overall command conditions. These are used to specify that for Q to hold on the final state, each loop iteration execution must starts and end in a valid state w.r.t. to the initial state (modelled via the transitive closure $(P \sqcap (\mathsf{evalT}\ t) \sqcap Q')^*$).

E.2 Updating the Rely-Guarantee semantics

Small changes are sufficient to adapt each of the RG semantics from §3 to a relational post-condition.

Trace-based semantics: Here, only the definition of *post-satisfy* differs when adapting to the relational setting from the unary case. We now write, that a given trace $tr = [(l^1, c^1, s^1), \ldots, (l^n, c^n, s^n)] \in \mathsf{Trace}(c)$, *post-satisfies* a relation $Q : \mathsf{State} \to \mathsf{State} \to \mathsf{Bool}$, written $tr \vDash_{\mathsf{post}} P$, provided $Q\ s^1\ s^n$.

Reachability-based semantics: Here, the relational definition of satisfaction also closely follows the unary case, with the only difference being the clause for the final states. This now states "for all (c', s') reachable via $\mathsf{stepRel}_R^*$ from (c, s) ... If (c', s') is final, then $Q\ \boxed{s}\ s'$."

Counting-based and coinductive semantics: Due to the (co)inductive nature of the definition of *safety* used by both these semantics, only a single step in the program execution is considered, compared to the long-distance consideration of the entire trace required in the earlier approaches. Hence, it is not sufficient to simply change the aspect of the semantics considering the final state.

Consider the process of establishing the $(n+1)$-safety of a configuration (c, s) where s is the initial state. This would only hold if a step to (c', s') resulted in an n-safe configuration. Notably, assuming the program (c, s) can reach some final state s'', a relational post-condition would require $Q\ s\ s''$ to hold but *not* $Q\ s'\ s''$ to hold. As a result, the definition of safety cannot directly specify the initial state for Q in its current form, but would need an arbitrary initial state variable that is carried unchanged through (co)induction. This, in turn, could unnecessarily complicate (co)inductive invariants.

Instead, we leave the safety definitions *unchanged*, but modify only the definition of RG satisfaction to pass in the correct initial state: i.e. $c \vDash_{\mathsf{VP}} (P, R, G, Q)$ holds iff $\forall s \in \mathsf{State},\ \forall n \in \mathbb{N}.\ P\ s \longrightarrow \mathsf{safe}_{(R,G,\ \boxed{(Q\ s)}\ )}\ n\ (c, s)$ and $c \vDash_{\mathsf{C}} (P, R, G, Q)$ holds iff $\forall s \in \mathsf{State}.\ P\ s \longrightarrow \mathsf{safe}_{(R,G,\ \boxed{(Q\ s)}\ )}\ (c, s)$. Thus, safety considers a unary post-condition $\lambda s'.(Q\ s)\ s'$ with the initial state s effectively fixed as part of the predicate, making the overall change to the semantics to accommodate relative post-conditions very small.

E.3 Analysis of soundness proofs

In our formal development we treat the relational setting independently of the unary setting described in the paper. However, the unary variant of the RG proof

system (Fig. 2) can be considered a special case of the relational post-condition variant of this RG proof system (Fig. 13) can be viewed as generalization of the unary case. Thus, the soundness of the unary variants (e.g. Thm. 1, Fig. 5) could alternately be established as a corollary of their binary counterparts.

As an example, consider the sequential soundness rule. A unary post-condition Q can be *lifted* to a binary predicate, $\mathsf{lift}_2\ Q$ which only considers the second argument. We want to establish $\mathsf{seq}\ c_1\ c_2 \vdash (P, R, G, (\mathsf{lift}_2\ Q))$ using only the assumptions of the unary variant: (i) $c_1 \vdash (P, R, G, (\mathsf{lift}_2\ P'))$ and (ii) $c_2 \vdash (P', R, G, (\mathsf{lift}_2\ Q))$. Letting $Q_1 = (\mathsf{lift}_2\ P')$ and $Q_2 = (\mathsf{lift}_2\ Q)$ the assumptions of the binary counterpart are satisfied as followed:

- $c_1 \vdash (P, R, G, Q_1)$ and $c_2 \vdash (P', R, G, Q_2)$ follow trivially
- $(\lambda s'.\exists s.P\ s \wedge Q_1\ s\ s') \leq P'$ also holds trivially as the image of P through a lifted binary relation such as $Q_1 = (\mathsf{lift}_2\ P')$ is simply equivalent to P'.
- $Q_1 \circ Q_2 \leq Q$ also holds as the composition of two binary relations only considering the second argument is simply equivalent to the unary component of the second, i.e. Q.

Below we discuss the impact of the relational post-conditions on the complexity of the soundness proofs, highlighting that the relative complexity across the different RG semantics remains similar.

Trace-based: Relational post-conditions add another layer of complexity to the trace-based proofs, noting that the previous formalization by Nieto [47] only considered unary post-conditions. Whereas in the unary case it is sufficient to reason simply on the final element of the trace (for each possible trace characterization), in the binary case it is necessary to also reason over the head of the trace and how it changes. This requires more index based reasoning, similar to what was already required to establish the guarantee.

For example, consider the sequential case where the trace is characterized by splitting it into two traces for c_1 and c_2. In the unary case, it is sufficient to first establish the post-condition P' of c_1 (implied by the pre-condition of the overall trace), where P' is also the pre-condition of the trace of c_2, from which we can directly reason on the last element of the c_2 trace to establish the overall post-condition Q. Comparatively, for the binary case it is necessary to establish first establish Q_1, then separately obtain the pre-condition P' for c_2. This enables the establishment of Q_2 but only with respect to the head of the second trace, so to finally establish the post-condition over the whole trace we need to employ the compositionality and rely assumptions, as well as some index-based reasoning to carry the initial state through the proof.

Notably, the same trace characterization lemmas are used by both the relational and unary cases. They cannot be "reduced" in the unary case, as regardless of the post-condition, reasoning about different program executions is integral to RG proofs given the existing relational rely- and guarantee-conditions.

Reachability-based: The reachability setting shares requires a similar change to the trace-based setting. While the reachability criterion does eliminate the need for the more involved index-based reasoning required in trace-based semantics,

it still requires a more intricate inductive proof. For example, in the sequential case, the adjustment for the relational proof strategy remains the same: we must derive the post-condition Q_1, and subsequently the pre-condition P' for c_2. This requires slightly more effort to address these additional steps. Notably, the original Coleman-Jones proofs [14] already considered relational post-conditions.

As in the trace-based setting, here we reuse the same inversion lemmas as the unary case, which, as discussed in §3.1, are much simpler than the characterization lemmas of the trace-based approach as they hide execution details. In both cases, these inversion-like lemmas are responsible for the "heaviness" of the proofs. As a result, the comparative complexity of the soundness proofs between the two approaches remains similar to that of the unary cases.

Counting-based and coinductive: As the relative post-condition variant of both the inductive and coinductive semantics did not directly affect the definitions of safety, much of the existing unary soundness proofs were sufficient. In fact, for every command except for while, it was possible to reuse the existing safety-preservation proofs, such as Fig. 5, passing in the unary post-condition $\lambda s'.\ (Q\ s)\ s'$ (i.e. fixing the initial state).

For example, consider once again our sequential example in the coinductive setting. To show $\mathsf{seq}\ c_1\ c_2 \vDash_{\mathsf{C}} (P, R, G, Q)$, we unfold the definitions resulting in the proof goal of $\mathsf{safeC}_{(R,G,Q\ s)}\ (\mathsf{seq}\ c_1\ c_2)$ given the following assumptions: (i) $\mathsf{refl} G$, (ii) $Q' \circ Q \leq Q$, (iii) $\mathsf{safeC}_{(R,G,(Q'\ s))}(c_1, s)$, and (iv) $\forall s'.\ (Q'\ s)\ s' \longrightarrow \mathsf{safeC}_{(R,G,(Q\ s'))}(c_2, s')$. We now need to establish the three assumptions required for the existing unary safety lemma:

1. $\mathsf{refl}\ G$ holds trivially via (i)
2. $\mathsf{safeC}_{(R,G,(Q'\ s))}(c_1, s)$ holds trivially via (iii)
3. $(Q'\ s)\ s' \longrightarrow \mathsf{safeC}_{(R,G,(Q\ s))}(c_2, s)$ requires a helper lemma with a straightforward coinductive proof to show that from both assumption (ii) and (iv) we can establish the safeC predicate using $(Q\ s)$ rather than $(Q\ s')$.

Notably, the assumptions for done, atom, if, and par could be established even more trivially without the need for an additional coinductive proof. The only reason a coinductive proof is required for assumption (3) of seq is due to the composition of Q' and Q which results in the post-condition initially only being relevant to s' rather than s.

An alternate approach for seq would be to modify the core safety lemma to directly consider the relational nature of the predicates Q and Q' in the assumptions (the proof goal would remain the same). While we can avoid this for seq via the helper lemma, it is necessary to make such a direct change for while. Specifically, it is not possible to establish assumption (2) as the relational variant of the assumption is stronger. Fig. 14 summarizes the differences, giving both the soundness property with expanded definitions and the underlying goal for (co)induction.

While many of these mirror the changes in the overall proof system, in particular, observe the minor difference in the goal. In the safeC predicate (ready

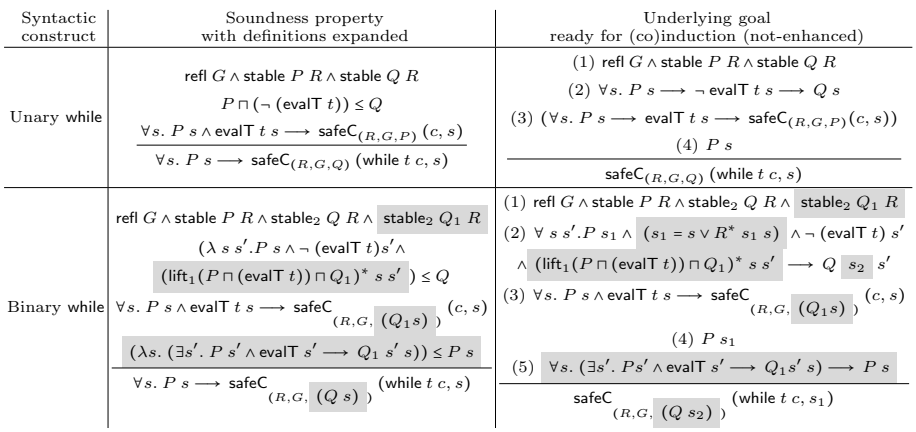


Fig. 14: Coinductive proof goal comparison for unary and binary variants

for (co)induction) we specify the post-condition as $Q\ s_2$, where s_2 is the initial state. Similarly, s_1 also is the initial state when the rule is instantiated (thus $s_1 = s_2$ to begin), however during (co)induction s_1 may change whereas s_2 does not. This results in the need to carefully restate assumption (2) prior to beginning (co)induction. Additionally, in assumption (3) we must also specify the post-condition as $Q_1\ s$ rather than P.

Note that the changes outlined in Fig. 14 apply to both inductive and coinductive variants of the proof. This is a key example of the small but noticeable layer of complexity the relational variant of the soundness proofs can add to each individual proof, however, given the change is carried across both variations it does not affect our inductive-to-coinductive comparison from the main paper.

F Isabelle Mechanization Details

We have mechanized the majority of our results pertaining to Rely-Guarantee (RG) reasoning and the general coinduction approximated inductively phenomenon in the proof assistant Isabelle/HOL [48]. This section will provide an overview of the mechanization with links to the relevant theorems and definitions from the main paper in App. F.1 to App. F.8. It concludes by extending on the discussion from §7 by highlighting key technical insights and contributions from the mechanization, including: a blueprint on using coinduction (as an alternative to induction) in a formal environment (App. F.9), key insights from the first formalization of the reachability-based approach following the original Coleman-Jones paper [14] (App. F.10), and a proof engineering focused comparison of the formal proofs of soundness across our four semantics (App. F.11).

F.1 Formalization overview

Our mechanization, more precisely, includes the following:

- Our basic parallel small step semantics from Fig. 1.
- The trace-based semantics for RG reasoning from §3.1, and soundness proofs w.r.t the proof system in Fig. 2.
- The reachability-based semantics §3.1 and soundness proofs w.r.t the proof system in Fig. 2. This additionally is the first formalization of this approach.
- Our new simplified version of counting-based inductive safety semantics (Fig. 3) from §3.1, and soundness proofs w.r.t the proof system in Fig. 2.
- Our new coinductive safety semantics (Fig. 4) from §3.2 and soundness proofs w.r.t the proof system in Fig. 2.
- The first formal proofs of equivalence of the three previously developed inductive RG semantics, which together with their soundness proofs justify Thm. 1.
- The equivalence of our new coinductive semantics (Thm. 2, Lemma 3).
- The principles behind our general account of coinduction approximated inductively, including the fully general fixpoint reasoning (Thm. 8).
- The general theory on up-to coinduction (including the unary variant of Thm. 9)
- The up-to enhancement version of the soundness proof for while including the formal operator definitions, required respectfulness proofs, and the inductive enhancement counterparts, as showing in Fig. 6, Fig. 7, Fig. 8, and Fig. 9.
- Generalizations the rule system format (Prop. 13, namely Thm. 14, Thm. 16, and (indirectly) Lemma 11).
- The connections with approximations of bisimilarities, including instantiation to LTS (Example. 20).
- The coinductive version of an RG Separation Logic inductive-safety semantics (Fig. 10) and its equivalence (Lemma 17 and Thm. 18), building on a recent Isabelle library for RG Separation Logic reasoning [28].
- The relational post-condition variants (semantics and soundness proofs) for each of the four RG semantics (as discussed in App. E).

We have *not* mechanized pre-existing general results related to fixpoint reasoning such as the Knaster-Tarski or Kleene Theorem (Thm. 4, Corollary 5, Thm. 7), or existing bisimulation results (e.g. Thm. 19).

Of our new claims, we have not formalized the generalized variant of Thm. 9 (as the unary variant was sufficient in our use-case) or the general theory connecting inductive and coinductive enhancements (Thm. 10). Additionally, the results in App. C are generally not formalized and only some examples from App. D have been formalized. In relation to LTS in App. A, we only mechanized Example. 20 and not the extension discussed in Example. 15 (for which the formalization would be similar).

We note our formalization includes a folder containing the Isabelle library for RG Separation Logic by Jackson et al. [28] with some minor restructuring,

which our formalization depends on. This library is publicly available and used freely in accordance with its license.

The mechanization is provided as archive folder (intended to be made publicly available in the future), containing the Isabelle sources as well as a browsable HTML version (with a documented README file). It has the following structure:

- Preliminary work and language semantics as top-level theories: `Prelim`, `Sequential_Par_While_Language`.
- General results on coinduction approximated inductively as top-level theories: `Coinduction_Approximated_Inductively` and `Up_To_Coinduction`.
- A folder for each of our semantics labeled: `RG_Trace`, `RG_Vafeiadis_Parkinson`, `RG_Coleman_Jones`, and `RG_Coinductive`. Each folder contains: the general semantic definitions (theories named `Abstract`); the soundness proofs with respect to our language (theories labeled `Soundness`); the relational post-condition adaptions of both of these (indicated by the `Binary` suffix in theory names); and auxiliary theories with common lemmas specific to the folder's semantics (e.g. `Inversion` theories).
- RG equivalence proofs for both unary and relational variants are in the `RG_Equivalence folder`.
- Further instances of coinduction approximated inductively are in the `Wider_Landscape` folder, i.e. theory `Simulations`.
- Separation logic extensions are in the `RG_Sep` folder.

Fig. 15 shows the overall theory structure in Isabelle/HOL.

F.2 Preliminary work

The theory `Prelim` contains useful auxiliary facts and sets up the Isabelle version of basic notation used in the proof system from Fig. 2 such as `stable`. It additionally contains a locale `Step`. Locales are Isabelle's module system [9]. A locale fixes parameters (with types) and assumptions, providing a local context in which concepts and theorems relative to these entities can be established. Locales additionally support some powerful inheritance mechanisms. In this case, the locale `Step` defines a context with two fixed parameters: `small_step` and `final`. This first is a relation designed to represent a small-step semantics and must have the type $(\mathsf{Com} \times \mathsf{State}) \to (\mathsf{Com} \times \mathsf{State}) \to \mathsf{Bool}$ and the second is a relation representing all `final` states. This represents an abstract small-step semantics.

The theory `Sequential_Par_While_Language` defines the small-step semantics for the simple imperative programming language with parallel composition from Fig. 1 as noted in §2. Where possible, we have kept our syntax naming consistent between the paper and the Isabelle theories. In summary, the main features of the theory include:

- The datatype `com`, parameterized by the types `'atom` and `'test`, which represents our syntax from §2.

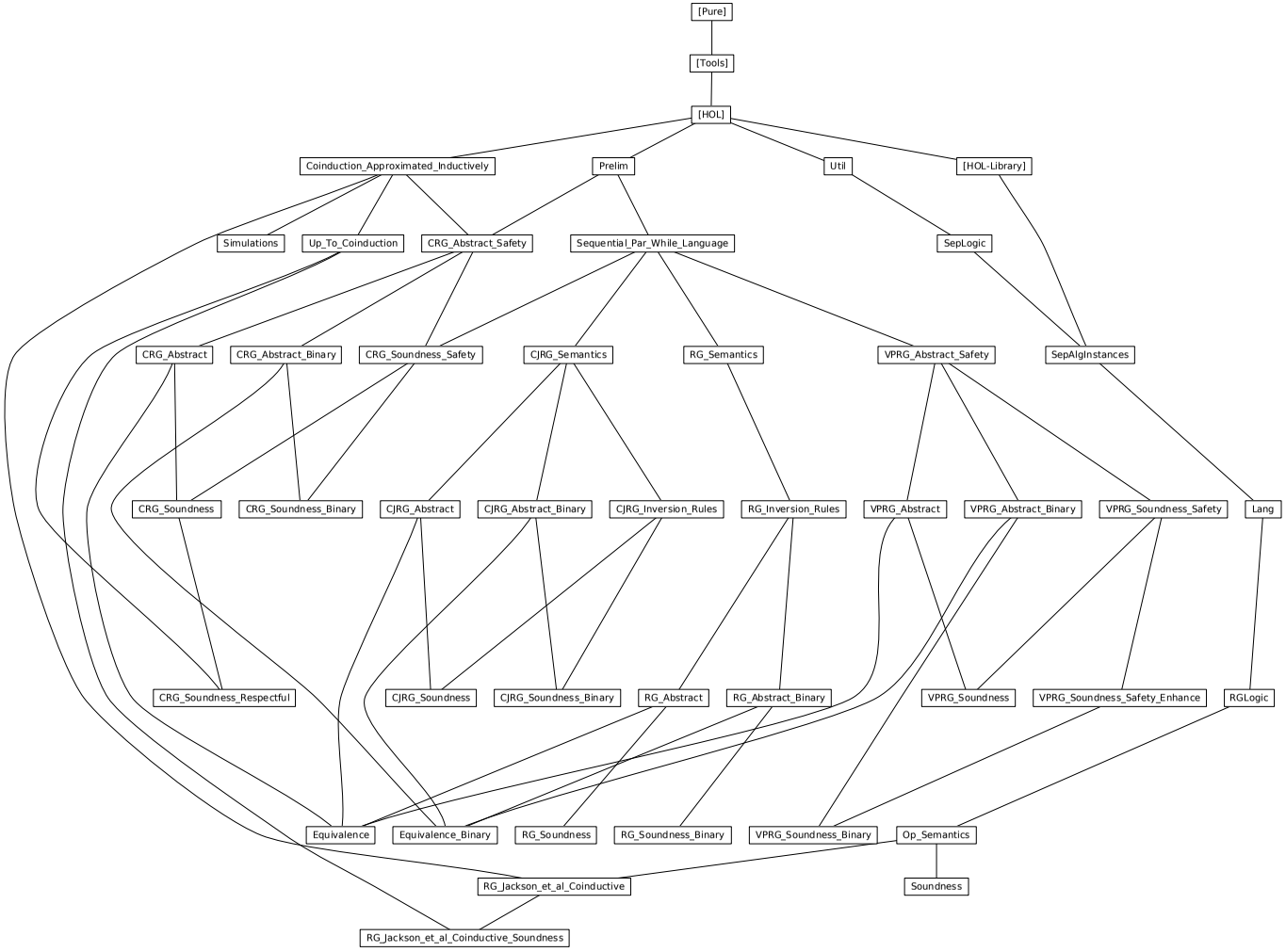


Fig. 15: Theory structure of our abstract Isabelle mechanization

– The locale `SeqParWhileLang` which fixes the functions `evalA`, and `evalT`. Here we use locales again, this time to parameterize our semantics. The remainder of the theory is within the context of this locale.
– The small-step semantics concretely defined as an inductive definition `small_step` and a concrete definition for `final`, as well as supporting lemmas for reasoning on these definitions.
– A sublocale declaration showing that the locale `SeqParWhileLang` and its contained `small_step` and `final` definitions are an instantiation of the abstract `Step` locale.

F.3 Formalization of the general results

The general results (as presented throughout §4, §5, and §6) have been mechanized in the `Coinduction_Approximated_Inductively` and `Up_to_Coinduction` theories.

The `Coinduction_Approximated_Inductively` theory begins with ideas from §4 which reuses Isabelle's existing definitions on lattices and fixpoints to set up the infrastructure for ω-continuity:

– The definition `ococont` for modelling a function on a lattice which is *ω-cocontinuous*.
– The `Mono` locale with a parameter `F` and an assumption re monotonicity of `F`, setting up an environment for reasoning on a monotonic operator on a lattice.
– Local definitions `llfp` and `ggfp` representing the least and greatest fixpoints of `F` respectively.
– Theorem `ggfp_eq_llfp` which formalizes Thm. 8

Next the theory moves to formalizing the rule system account as described in §6.1 and §6.3. It begins by setting up a rule-based context.

– A locale `Mono_Pred`, which is a sublocale of `Mono` requiring the parameter `F` to be on a lattice of predicates.
– The definitions `ilfp` and `cgfp` which define `llfp` and `ggfp` using inductive and coinductive definitions respectively instead of the built-in `lfp` and `gfp` functions, along with equivalence proofs between the definitions. These provide alternate proof approaches for reasoning on `llfp` and `ggfp`.

Following this, the theory sets up the generalization of the rule system context using the `alter` definition:

– The inductive definition `alter` which formalizes the *alter* definition on alternating existential and universal quantifies given in §6.3.
– The lemmas `alter_conj` and `alter_disj` which show alter is closed under conjunction and disjunction as stated in §6.3.
– The proposition `alter_ococont` which formalizes the proof of Thm. 16.
– A locale `Alter` with a parameter F that satisfies the definition `alter`. Within this locale is a sublocale declaration with `Mono_Pred` which establishes the particular case mentioned in Thm. 16.

Finally, the theory adds infrastructure to increase the usability of the rule-based context:

- A locale `Rules` which has a parameter `Rls` that represents a set of rules Rl on a set A. Within is a local definition `F` which represents the operator F_R that can be established generally from the rule set and is proven monotonic.
- A general definition `backFinite` which takes a rule set and determines if it is *backwards-finite*.
- A locale `Rules_BackFinite` which adds an assumption on backwards finiteness to the `Rules` locale.
- A proof that `Rules_BackFinite` is a sublocale of `Alter` such as in Example 22. This resultantly establishes *ω-cocontinuity* as `F` now inherits that property from the locale `Alter` and in turn `Mono`, which means that formal proofs of Thm. 14, Prop. 13 and Lemma 11 are also inherited.

For further usability, the theory defines an alternate definition `injectR` on injectivity of rules instead of backwards-finiteness, alongside a locale `Rules_InjectR` using this assumption instead. This can be shown to be a sublocale of `Rules_BackFinite`, thus inherits all the results.

The `Up_to_Coinduction` theory formalizes the general ideas of up-to coinduction discussed in §5.2 as well as further formal infrastructure. Of particular relevance to this paper are:

- The definition `respect`, which formalizes the idea of *weak F-respectfulness* for a unary operator.
- The theorem `wrespect_upto` and corollary `respect_upto` which formalizes the unary variant of the proof of Thm. 9.
- Variants of `respect` for non-unary cases.
- Locales enabling reasoning on respectfulness over multiple lattices as discussed in App. D.

F.4 Wider landscape

The connection of the general theory to approximations of (bi)similarity discussed in §6.3 and App. A is mechanized in the `Simulations` theory:

- An LTS is formalized via the `LTS` locale.
- Definitions such as `imageFinite` and `finiteBranching` formalize the LTS properties.
- The sublocale declaration `Alter` showing a function F for an image-finite LTS satisfies the requirements of the alter definition (as given in Example. 20)

F.5 RG semantic definitions

All four ways of defining RG semantics as per §3 are included in this mechanization as well as their relational post-condition adaptations from App. E. Each is defined in the context of the generic `Step` locale (i.e. assumes

some arbitrary small-step and final operator, but is *not* specific to our small-step semantics from `Sequential_Par_While_Language`).

Trace-based semantics: Our mechanization of the traditional trace-based semantics from §3.1 has two distinct parts. We first formalize the definitions required to model traces in the theory `RG_Semantics`:

- A datatype `acfg` for our labeled configurations, where each is labeled by either `S` or `E`. Note that in Isabelle we use `S` in place of C as used in the paper.
- The definition `trace` to represent a valid computation.

Using this, the RG semantics are formalized in the theory `RG_Abstract`:

- The definitions `satPre` (for when a computation *pre-satisfies* a predicate), `satPost` (for when a computation *post-satisfies* a relation), `satRely` (for when a computation *rely-satisfies* a relation), and `satGuar`, for when a computation *guarantee-satisfies* a relation.
- The definition `sat` for when a command satisfies a RG clause, i.e. `sat c P Q R G` is equivalent to $c \vDash (P, R, G, Q)$.

The relational post-condition counterpart explored in App.E is available in `RG_Abstract_Binary` with the same structure, reusing the first theory.

Reachability-based: The formalization of the reachability-based semantics (inspired by Coleman and Jones) from §3.1 can be found in the folder `RG_Coleman_Jones` and is split similarly across two theories. The theory `CJRG_Semantics` defines general reachability concepts:

- The inductive definition `step_rel` is defined such that `step_rel R (c,s) (c',s')` is equivalent to $\mathsf{stepRel}_R\,(c, s)\,(c', s')$
- The inductive definition `step_closure`, defined as `star (step_rel R) x y` is equivalent to $\mathsf{stepRel}^*_R\,x\,y$.

`CJRG_Abstract` then introduces the RG semantics:

- The definition `withinG`, in line with the original paper, is equivalent to $\mathsf{stepRel}^*_R\,(c, s)\,(c', s')$, such that $G\ s'\ s''$ for all $c''\ s''$ such that $(c', s') \Rightarrow (c'', s'')$.
- The definition `safeCJ` such that `withinG` and for $\mathsf{stepRel}^*_R\,(c, s)\,(c', s')$, $\mathsf{final}(c', s')$ implies $Q\ s'$.
- The definition `satCJ` (which uses both of the above) for when a command satisfies a RG clause such that `satCJ P R c G Q` is equivalent to $c \ \vDash_{\mathsf{CJ}} (P, R, G, Q)$.

The relational post-condition counterpart `satCJR` is in the theory `CJRG_Abstract_Binary`.

Counting-based: The formalization of our simpler version of the counting-based semantics (inspired by Vafeiadis and Parkinson) given in Fig. 3 (§3.1) defines safety and satisfiability separately.

The `VPRG_Abstract_Safety` theory contains the inductive definition of safety `safe`, such that `safe n (c,s) R G Q` is equivalent to $\mathsf{safe}_{(R,G,Q)}\ n\ (c, s)$, as given in Fig. 3, along with useful helper lemmas on the definition. `VPRG_Abstract` defines `satVP` for when a command satisfies the RG clause, i.e. `satVP c P R G Q` is equivalent to $c \vDash_{\mathsf{VP}} (P, R, G, Q)$. The relational post-condition counterpart `satVPR` is in the theory `VPRG_Abstract_Binary`. Following the discussion in App. E, both the latter theories branch off the `VPRG_Abstract_Safety` theory, given they both utilize the same inductive safety definition.

Coinductive safety: The formalization of our new coinductive semantics given in Fig. 4 (§3.2) follows the same structure as its inductive counterpart. The coinductive definition of safety `safeC`, such that `safeC R G Q(c,s)` is equivalent to $\mathsf{safeC}_{(R,G,Q)}\ (c, s)$, is located in the `CRG_Abstract_Safety` theory. `CRG_Abstract` defines `satC` for when a command satisfies the RG clause, i.e. `satC c P R G Q` is equivalent to $c \vDash_{\mathsf{C}} (P, R, G, Q)$, and its relational counterpart `satCR` is in `CRG_Abstract_Binary`

Additionally, the theory on coinductive safety `CRG_Abstract_Safety` inherits from the `Coinduction_Approximated_Inductively` theory, instantiating it to the coinductive notion of safety, as explored in Examples 6 and 12:

- The functions `hypsOf`, `condsOf` and definition `Rls` define the rule system that can be derived from our coinductive semantics (as given in Example 12)
- The sublocale declaration `safeC_rules` shows that this rule set inherits all the properties of the locale `Rules_InjectR`.
- A direct definition of the function `F` that forms the basis of our coinductive definition (as given in Example 6), and variations `F2` and `F3` respectively which fix R, G and Q in different ways as discussed in App. D.
- In turn, the sublocale declaration `safeC` shows that the `Step` locale with the local definition `F` is a sublocale of the `Mono` locale.

F.6 Equivalence proofs

The `Equivalence` theory mechanizes the new formal treatment of the equivalence of the three inductive semantics (stated as part of Thm. 1), and the further equivalence of the inductive and coinductive semantics stated in Thm. 2. The relational variations are shown equivalent in `Equivalence_Binary`.

The equivalence proofs are located in the `Step` locale, thus proved for an abstract small-step semantics. The formal proof of the equivalence part of Thm. 1 proceeds by establishing equivalence in a pair-wise fashion as in App. B. This approach is taken over a more circular proof primarily due to the history of the formal proof development. The lemma grouping `sat_equivalence` at the end of the theory (and `satR_equivalence` for the relational variation) brings together all equivalence theorems.

Trace-based and reachability-based: The equivalence between the trace and reachability-based semantics (i.e. (1) iff (2)) requires the following lemmas:

- The auxiliary lemma `trace_step_closure` formalizes the information encoded within $\mathsf{stepRel}_R^*$ in relation to the explicit trace-based semantics. Specifically, $\mathsf{stepRel}_R^*\,(c,s)\,(c',s')$ holds if there exists $tr = [(l^1,c,s),\ldots,(l^n,c',s')]$ such that $tr \vDash_{\mathsf{rely}} R$. This goes through by induction on the transitive closure of $\mathsf{stepRel}_R$.
- The auxiliary lemma `satCJ_singl_PostGuar` proves that a singleton $\mathsf{stepRel}_R^*\,(c,s)\,(c,s)$ trace which preserves the post-condition, is a trace which preserves `satPost` and `satGuar`.
- The lemma `satRG_imp_satCJ` shows one side of the implication (i.e. (1) implies (2)) using `trace_step_closure` to obtain relevant trace properties.
- The lemma `satCJ_imp_satRG` shows the other side of the implication (i.e. (2) implies (1)) and goes through by list induction on the trace computation.

We also extend `trace_step_closure` to an iff lemma `trace_step_closure_eq` which provides a direct equivalence between $\mathsf{stepRel}_R^*$ and the trace-based semantics. This is not used in the equivalence proofs but is useful as a sanity check to ensure the definitions are accurate.

Trace-based and counting-based: The equivalence between the trace and counting-based semantics (i.e. (1) iff (3)) requires the following lemmas:

- The lemma `sat_imp_satVP_aux` performs induction on the number of computation steps n, the crucial part of showing (1) implies (3)
- The lemma `satVP_imp_sat` shows the reverse direction (3) implies (1) via induction on the list representing the trace, i.e. the computation.
- The theorem `sat_iff_satVP` shows the equivalence.

A little careful work is required to adapt both these proofs to relative post-conditions in the `Equivalence_Binary` theory, given safety still uses a unary post-condition. In particular, to enable cleaner reasoning on the initial state, we extend the `RG_Abstract` with the definition `reduced_sat` which takes the trace `sat` definition and partially unwinds it by fixing s and pulling out the pre-condition.

Counting-based and coinductive: The equivalence of the counting-based and coinductive semantics relies on Lemma 3 to establish equivalence between inductive and coinductive safety. This is inherited as `safeC_rules.ggfp_iff_llfp` from the `Coinduction_Approximated_Inductively` theory. However, to use this, it is necessary to finish instantiating our definitions per Example 6:

- The lemma `llfp_safeC` shows that `safe` is equivalent to the least fixpoint definition `llfp` in the `safeC_rules` locale. The proof proceeds on induction in both directions, using the inductive definition `ilfp`.
- The lemma `ggfp_safeC` similarly shows that `safeC` is equivalent to `safeC_rules.ggfp`, using coinduction in both directions.
- The theory `satVP_iff_satC` establishes the equivalence in one line using these lemmas and the inherited `ggfp_iff_llfp`.

Note that as this proof is done primarily on the definitions of safety, a very simple one-line proof also establishes the relational post-condition variant `satVPR_iff_satCR` in `Equivalence_Binary`.

F.7 RG soundness proofs

The soundness proofs of both the previously developed inductive semantics and the new coinductive semantics, discussed through §3.1 and §5, have all been formalized. These theories form the basis for our proof comparisons.

Each soundness theory is structured similarly. For each RG rule from Fig. 2 (i.e. each syntactic command in our language), we have a proposition in a soundness Isabelle theory, named `sat_{Rule}`. For example, the soundness proof for the (SeqRG) rule under our coinductive semantics is a proposition named `satC_Seq` in the `CRG_Soundness` theory. The relational variants are established in separate theories, such as `CRG_Soundness_Binary`.

Some aspects of the soundness proofs are shared between the unary and relational post-condition variants. Thus, some auxiliary theories are defined for each semantics:

- For soundness on traces, the theory `RG_Inversion_Rules` sets out the characterization lemmas for computation traces.
- Similarly, the theory `CJRG_Inversion_Rules` does the same but for $\mathsf{stepRel}^*_R$, deriving the necessary properties of this reachability criterion to ensure the soundness of the syntactic constructs.
- The theories `VPRG_Soundness_Safety` and `CRG_Soundness_Safety` contain the safety preservation proofs on the inductive and coinductive definitions of safety respectively. These theories are the most important in relation to the soundness proof discussions in §5, and include the mechanized version of Fig. 5.

The `VPRG_Soundness_Safety_Enhance` and `CRG_Soundness_Respectful` theories provide variants of the soundness proofs utilizing the inductive and coinductive enhancements respectfully, as presented in §5.2. In particular, we highlight the following which are necessary for the while example discussed:

- The new proofs of `safe_Seq'` and `safe_If'` represent the inductive enhancements in Fig. 6.
- The operators $\mathsf{U}^{\mathsf{done}}_{Q}$, $\mathsf{U}^{\mathsf{seq}}_{(R,G,P,c_2)}$, and $\mathsf{U}^{\mathsf{if}}_{(R,t,c_1,c_2)}$, given in Fig. 7 are defined formally by `U_Done`, `U_Seq`, and `U_If` respectively. `U_forWhile` defines the operator W to be used in the up-to coinductive proof.
- Lemmas beginning with `respect` establish the respectfulness of each of these operators in turn per the criteria given in Fig. 8.
- The proposition `safeC_coind_upto_forWhile` defines the W-up-to coinduction rule for while using the `respect_upto` theorem (i.e. Thm. 9).
- The proposition `safeC_While` establishes the soundness of while using the up-to approach as described in Fig. 9, along with its inductively enhanced counterpart `safe_While'`.

Some further formalization work is included on the different respectfulness operators mentioned in App. D.

F.8 Separation logic extension

In §6.4 we discussed how our results could be extended to a RG Separation Logic. One motivation for basing our analysis on the GenRGSep logic by Jackson et al. [28] is their existing Isabelle library. From a formalization perspective, this has two benefits: **(1)** the formalization for inductive safety is already complete, and **(2)** it provides an opportunity to test the usability and efficiency of our coinductive approach following a different mechanization style.

The mechanization of §6.4 is available in the `RG_Sep` folder. It is not connected to the rest of the development, other than utilizing the general `Coinduction_Approximated_Inductively` and `Up_to_Coinduction` theories. The original Isabelle library published by Jackson et al. [28] is included as a sub-folder labeled `Generic_RG_Sep`, enabling our theories to use the same syntax, language, and general definitions from their development. The definition `safe` in the `Op_Semantics` theory from this library is the mechanized version of the inductive predicate safeSep given in Fig. 10.

Note that we have made a minor refactor to the library as included in our mechanization compared to the original archive published online. Rather than the operational semantics, safety and soundness proofs being grouped together in the `Soundness` theory, the operational semantics and safety definitions are now in a new `Op_Semantics` theory, which the `Soundness` theory then inherits from. This ensures there are no dependencies between our coinductive proofs on soundness (which also require the operational semantics) and the original inductive soundness proofs.

Our contributions are formalized across two theories. Firstly the abstract coinductive definition of safety for RG Separation Logic is formalized in `RG_Jackson_et_al_Coinductive`, specifically:

- The type class `perm_alg'` extends the existing `perm_alg` class with a finiteness assumption. This is a reasonable assumption since it holds for any concrete choice of the notion of state and addition given the states are finite.
- The coinductive definition `safeC` is the formalization of the predicate safeSepC.
- The interpretation `SEP` establishes that the underlying operator F for the coinductive safety definition satisfies the requirements of the `Alter` locale from the `Coinduction_Approximated_Inductively` theory, thus automatically inherits the use of the theorem `ggfp_iff_llfp` (i.e. Thm. 16 in the main paper).
- The proposition `safeC_safe` establishes equivalence between our coinductive definition and Jackson et al.'s inductive definition of safety as per Lemma 17.
- The theorem `satSepC_satSep` is the mechanization of Thm. 18 between the definition of satisfiability using inductive safety (`satSep`) and the definition using our new coinductive safety approach (`satSepC`)

The `RG_Jackson_et_al_Coinductive_Soundness` theory mechanizes the soundness proofs using the coinductive version of safety, including up-to style reasoning for the proof on `Iter`. We note that our approach aims to mirror the tactic-based

style used in Jackson et al.'s library to the greatest possible extent. Thus, for each lemma in the original `Soundness` theory, we provide the coinductive alternate, reusing parts of the proof where possible. Other than benefits directly resulting from using coinduction, we avoided where possible making any other mechanization related improvements to the original proofs. This enables a side by side comparison of the formalizations to clearly demonstrate both: (1) the common aspects of the two proof approaches, and (2) where the coinductive approach has resulted in more direct proofs of soundness.

F.9 A formal blueprint for coinduction approximated inductively

Our paper presents coinduction as an alternative to induction, emphasizing the possible benefits. As a result, this formalization aims to provide a blueprint for future work which could benefit from a coinductive approach in formal environments such as Isabelle.

Firstly, as presented throughout §4, §5, and §6 we have developed several accounts of the coinduction approximated inductively phenomenon in our work. The `Coinduction_Approximated_Inductively` theory includes all of these to maximize usability, each encompassed by a locale. Specifically, as mentioned in §7, some of these approaches were developed explicitly to minimize the abstract-concrete instantiation distance. Thus, it is sufficient to establish that a coinductive definition can be formulated in such a way that its building blocks satisfy the parameters and assumptions of one of these locales. It will then automatically inherit all of our theorems on coinduction approximated inductively for free. For example, as shown by the theory `CRG_Abstract_Safety` it is relatively straightforward to establish a rule-based system inspired by the coinductive safety definition which satisfies the requirements of the locale `Rules_injectR`. Then in the `Equivalence` theory we simply need to show that `safe` is equivalent to the `llfp` generated by this locale instantiation (and similarly for `safeC` and `ggfp`) to use the general equivalence theorem.

Next we consider coinductive proofs, taking as an example the coinductive proof of soundness for sequential composition, as given in Fig. 5(b). In Isabelle, the preliminaries aspect of this proof is done in the corollary `satC_Seq` in the `CRG_Soundness` theory, which unfolds the satisfiability definitions. The reduced assumptions and goals are left for the proposition `safeC_Seq` in the `CRG_Soundness_Safety` theory which contains the formal version of the main part of the proof.

Recall that the core part of a coinductive proof is showing $k \leq F(\mathsf{gfp}_F \sqcup k)$, with $\mathsf{safeC}_{(R,G,Q)} = \mathsf{gfp}_F$ in our example. Thus, to establish $\mathsf{safeC}_{(R,G,Q)}(c, s)$ given an assumption K we must show (1) that $K(c, s)$ holds, and (2) that for any (c, s), $K(c, s) \longrightarrow F(\mathsf{safeC}_{(R,G,Q)} \sqcup K)$. In our formal Isabelle proofs we define K as φ. Note that (1) is trivial, and is implied in the proof in Fig. 5when we mention using coinduction to establish the goal. In Isabelle, (1) must be established to apply coinduction, but its proof goal can be discharged trivially.

While much less automated support exists for coinduction than induction in Isabelle/HOL, the coinductive libraries and tactics have still seen significant

development since their first iteration [51]. A coinductive definition such as `safeC` will automatically provide a coinduction rule (`safeC.coinduct`) which follows the aforementioned structure of a coinductive proof with the definition of F expanded. This rule can be further refined to be more easily applied to proof goals with a desired format. For example, our proofs use the corollary `safeC_coinduct'` (defined in `CRG_Abstract_Safety`) as the coinduction rule. Unlike the automatically generated rule, this minor variant treats c and s as separate parameters (removing the need to constantly unpack a pair datatype in the formal environment), structures the proof by providing a useful case name to the resultant goal, and automatically consumes part (1) of the coinductive proof if it has been established immediately before applying coinduction.

After applying a coinductive tactic, our remaining goal is part (2) of the coinductive proof. Unlike induction in Isabelle, which can automatically derive useful inductive premises, in the coinductive case we must explicitly extract this from our definition of φ, which for the sequential rule enables us to fix c_1 and assume (3) as given in the written proof from Fig. 5(b). After applying basic introduction rules on the conjunction and implication operators, our formal proof now is split into three remaining goals which directly mirror (i), (ii) and (iii) given in the written proof.

Up-to coinductive proofs have significantly less infrastructure available in Isabelle. Thus, much more manual work is required as the existing coinductive tactics are not sufficient. Specifically:

- It is necessary to separately define the underlying F operator (and variants depending on how many parameters need to be fixed) and show it inherits from the `Mono` locale (first defined in the general `Coinduction_Approximated_Inductively` theory), as respectfulness is formally defined inside this locale's context in the up-to theory.
- The U rules for each command must be explicitly defined and shown to be respectful. Note that while the respectfulness proofs can be mined from their coinductive safety preservation counterparts, they do not explicitly use coinductive tactics themselves.
- For any goal requiring up-to coinduction, an up-to coinductive rule must be defined (such as `safeC_coind_upto_forWhile`). This is established using the aforementioned respectfulness proofs and the formal version of Thm. 9, `respect_upto`.
- This rule is then used *directly* on the proof goal in place of a coinduction tactic, as exemplified in the `safeC_While'` lemma.

F.10 Mechanizing the Coleman-Jones approach

Given the Coleman-Jones style reachability-based approach had not been previously formalized, our mechanization provides new insights and clarity on the soundness proofs. We found when formalizing this work that while the intuitive idea of the proofs are along the right lines, concretely, the proofs often missed/skipped key details. The formal development in Isabelle gave insight to

these inaccuracies, which clarifies the proof ideas from the original paper and has the added benefit of removing unnecessary assumptions initially required.

For example, in the proof sketches for respecting guarantee-conditions, termination is assumed to prove soundness for all language constructs. However, in our development, we can show that this is not necessary to obtain soundness. We instead use inversion lemmas on reachability to provide sufficient properties for soundness. The removal of the termination assumption is also in line with the soundness proofs for the other RG semantics. In the original proofs, the termination is assumed to skip possible cases for intermediate states that the command could transition to. However, the inversion lemmas for our multistep reachability can be used to split on cases for these intermediate steps, using the same logic from the original proofs to discharge each case.

To illustrate this, we examine the unary soundness proof of the sequential case. Specifically, consider the process for showing that any command step preserves the guarantee-condition, i.e. given $\mathsf{stepRel}^*_R(\mathsf{seq}\ \ c_1\ \ c_2, s)(c', s')$ and $P\ s$, if $(c', s') \Rightarrow (c'', s'')$, then $G\ s'\ s''$. Coleman and Jones (assuming termination) immediately obtain the intermediate state s_i, such that $Q_1\ s\ s_i$ (since c_1 reduces to final), and thus obtain that c' is within c_2, It then follows from $c_2 \vdash (P', R, G, Q)$ that any command execution ensures the guarantee is stable. While this achieves soundness for the rule, the assumption of termination allows for the intermediate executions within c_1 to be "skipped". However, if we remove this assumption, and instead apply the inversion lemma summarized in §3.1, whenever $\mathsf{stepRel}^*_R(\mathsf{seq}\ c_1\ c_2, s)(c', s')$, either:

- c' has the form $\mathsf{seq}\ c_1'\ c_2$ for some c_1' such that $\mathsf{stepRel}^*_R\ (c_1, s)\ (c_1', s')$ holds;
- or $\mathsf{stepRel}^*_R\ (c_1, s)\ (\mathsf{done}, s'')$ holds for some s'', and $\mathsf{stepRel}^*_R\ (c_2, s'')\ (c', s')$ holds.

We can apply the same logical arguments when c_1 reduces to final, and in the case where c' is within c_1's execution, it similarly follows from $c_1 \vdash (P, R, G, P')$ that the guarantee holds. This enables a soundness proof which does not rely on termination but instead utilizes this inversion lemma. So while on the one hand the proof becomes longer, accounting for the extra execution cases, ultimately the same proof strategy can be discharged, allowing a lighter version than that of the one argued in the original work.

As a consequence of removing the termination assumption in line with our proof system, modifications are required in the soundness proof for while. Specifically, due to the circular nature of the proof, the $\mathsf{stepRel}_R$ predicate needs to be augmented. We introduce a labeled reachability criterion which keeps track of the command/environment steps using numeric counters. This extension enables a proof by complete induction on this numeric counter. Formally $\mathsf{stepRelL}_R\ n\ m\ (c, s)\ (c, s')$ is defined inductively by the rules:

$$\mathsf{stepRelL}_R\ 0\ 0\ (c, s)\ (c, s)\ \ (\mathsf{stepL\text{-}Base})$$

$$\frac{R\ s\ s' \qquad \mathsf{stepRelL}_R\ n\ m\ (c, s')\ (c'', s'')}{\mathsf{stepRelL}_R\ (n+1)\ m\ (c, s)\ (c'', s'')}(\mathsf{stepLR})$$

$$\frac{(c, s) \Rightarrow (c', s') \qquad \mathsf{stepRelL}_R\ n\ m\ (c', s')\ (c'', s'')}{\mathsf{stepRelL}_R\ n\ (m+1)\ (c, s)\ (c'', s'')}(\mathsf{stepLC})$$

This adjustment minimally extends the existing stepRel predicate to include a measure, where the numeric labels provide sufficient information to derive the appropriate induction hypothesis—namely, that the command steps reduce with each loop iteration. By contrast, Coleman and Jones sidestep this issue in their original work by assuming termination in the soundness proofs (since it is already known the command reduces to final) and independently proving termination later using similar well-foundedness criterions. Introducing this concept at the level of the soundness proofs, however, avoids the need to include the termination assumption in the proof system. We note that with the introduction of stepRelL, new inversion rules are required for each syntactic element while can transition to, including itself, if, seq and done. This (necessary) development further emphasizes the improvement offered by the counting-based and coinductive RG semantics, which requires at most only one numeric counter.

Furthermore, the original Coleman and Jones development does not reference safety, however in our formalization, the definition `satCJ` uses a safety definition similar to that of inductive safety. This is defined as `safeCJ` such that `withinG` and for $\mathsf{stepRel}^*_R\ (c, s)\ (c', s')$, (c', s') final implies $Q\ s'$. This definition is equivalent to the original paper's formulation where satisfaction distributes the `withinG` and post-condition proofs, treating them separately (roughly speaking $(\forall s.\ P\ s \longrightarrow X \wedge Y) = ((\forall s.\ P\ s \longrightarrow X)\ \wedge\ (\forall s.\ P\ s \longrightarrow Y))$). In our development, we choose to unify the proof effort by requiring a single implication of the universal quantification over states rather than handling the `withinG` and post-condition in independent proofs. We then apply an introduction rule, obtaining the assumptions of $\mathsf{stepRel}^*_R\ (c, s)\ (c', s')$ and $P\ s$, and utilize Isabelle's Isar proof language to tackle the guarantee proof (with the assumption of $(c', s') \Rightarrow (c'', s'')$) and post-condition proof (with the assumption that (c', s') is final) independently. This reduces the LOC required, as proofs of soundness for any given syntactic construct do not require two separate lemmas, yet still preserve the separate proof structure of the original work.

F.11 Formal proof engineering for RG soundness

As stated in the introduction, the work presented in this paper was initially motivated by our experience mechanizing RG related proofs in Isabelle/HOL. In such a formal environment, the heaviness of the underlying inductive machinery has a much greater impact on the overall complexity of a proof, and resultantly the time and effort required to complete a formalization. Here we provide a technically detailed comparison of the formal proof-engineering efforts required for the RG soundness proofs, highlighting key insights that contributed to the ideas in the paper. This extends on §7 and complements the proof development discussion in §3.1 and §5.

Abstractly, each soundness proof follows the same general structure based on standard RG intuition: Assuming the hypothesis of the given rule, if the pre-condition holds on the initial state, and the rely-condition is satisfied throughout the execution, then any step taken satisfies the guarantee relation, and the post-condition holds on the (possibly existing) final state. By formalizing the soundness of each of the four RG semantics with respect to the same simple language and proof system, the mechanizations can be directly compared.

Inductive RG semantics: When reasoning on $\vDash_{\mathsf{XRH}}$, based on [70], it is necessary to reason over multiple execution steps, i.e. "long distance action". As indicated in §3.1, this requires a characterization lemma which outlines the possible forms a computation trace for a given command could take. Both reasoning over traces generally, and specifically this additional lemma add a significant amount of formal bureaucracy.

A trace is naturally formalized as a list with certain properties between different list elements. This definition alone requires numerous helper lemmas to make it usable. There are over 200 LOC in the theory `RG_Semantics` that are only required for the trace-based approach yet don't even introduce any RG concepts. The characterization lemmas then require careful and involved inductive proofs (using the built-in list induction rules). Interestingly, most of the inductive work is required at this stage of the proof, depending on the command. In fact, the theory containing all the characterization lemmas (`RG_Inversion_Rules`) is a similarly long length to the main soundness theory (`RG_Soundness`)—both at a little over 1400 LOC. This emphasizes the significant effort required before the main proof is even begun.

The soundness proofs require consideration of each possible trace format, for which each case must consider the entire trace, complicating the proof process. For example, to establish the guarantee-condition it is necessary to consider each pair of list indices representing a command step, split these into cases depending on what part of the trace they belong to (based on trace characterization), and then establish the guarantee holds between the pair. This approach obscures aspects of the key intuition behind RG reasoning given the significant formal bureaucracy involved. Considering the process for proving soundness of `seq` alone, this main proof combined with its characterization lemma required approximately 250 LOC.

The approach to proving $\vDash_{\mathsf{CJ}}$ does see some improvement to this transparency. As noted in §3.1, although it is still explicit about long-distance action, it hides the details of the traces. This is reflected in the formalization, with the setup for the reachability definition $\mathsf{stepRel}_R^*$ only requiring 150 LOC. The inversion lemmas additionally clearly result in less formal bureaucracy than characterization lemmas. This is exemplified by only having 380 LOC in the `CJRG_Inversion_Rules` theory in contrast to the 1400 LOC required in the trace based case. Notably, these proofs also all use the same induction rule, thus are significantly more intuitive to obtain compared to their trace equivalents.

The resultant proofs of soundness for this approach are more clearly structured towards establishing the guarantee and post-conditions. While cases

are still required, formal case-based reasoning can typically be pushed to later in the proof, and as the reachability definitions hide trace details it is not necessary to fully consider the underlying case. Looking specifically again at our seq soundness proof, the formalization of this and its inversion lemma is only 42 LOC (not including definitions/lemmas to enable reasoning on reachability).

While lighter, the proofs for $\vDash_{\mathsf{CJ}}$ still follow a similar overall structure to $\vDash_{\mathsf{XRH}}$: first requiring definitions to model the long-distance actions, then detailed inversion lemmas, before finally beginning reasoning on the RG properties. In contrast, by reasoning on a single step as done for $\vDash_{\mathsf{VP}}$, the proof can focus on RG intuition. The "native inversion" lemmas referenced in §3.1 come directly from our small-step semantics theory. As such, the formalization of the counting-based proofs start with reasoning on safety preservation (the core part of the soundness proof) in `VPRG_Soundness_Safety`, this time requiring no helper theories. Our seq example required 34 LOC for the formal safety aspect of the proof, plus 5 additional lines to unfold the definitions in the preliminary part of the proof (in `VPRG_Soundness`). While this is not a significant difference from the reachability approach, the difference is more evident when taking into account the additional general setup required to reason on reachability. Additionally, it can be argued that the counting-based formalization is more direct and transparent for mirroring RG intuition.

We note that in all cases, the soundness proof of while added additional complexity, which compounded the differences resulting from the relative "heaviness" of each approach. In the trace-based approach, the proof requires an additional induction step (over the length of the trace)—which leads to both the base and step cases considering the subcases arising from the characterization lemma. For the reachability-based approach, we use the labeled reachability criterion (see App. F.10) and also required new inversion lemmas for several of the commands, thus adding up to significantly more formal proof infrastructure. Lastly, the counting-based inductive safety approach requires either more cases depending on steps to if and seq commands, or the use of our inductive enhancements outlined in §5.2.

In comparison, despite concurrency being a key aspect of RG reasoning, the proofs of soundness of the parallel rule resulted in little further insight apart from an added complexity in the trace-based proof. To establish soundness of the par rule using this approach, substantial extra work is required to show that the guarantee and rely predicates of the two parallel execution traces are satisfied by the main trace. Following Xu et al.'s proof [70] this involves using a proof by contradiction, before any reasoning can be done on the overall guarantee-condition and post-condition of the entire trace. This "detour" in the proof is not required when using any of the other RG semantics.

Counting-based vs coinductive: The coinductive approach offers very similar benefits to the counting-based approach in a formal environment compared to reachability or trace-based reasoning. The differences between the counting-based and coinductive approach are much more subtle, as explored extensively in §5. As detailed in §7 formalizing the proofs of soundness for

inductive and coinductive safety in parallel was key to the many insights presented in the main paper. Notably, for both proofs the formalization is a similar length to the detailed written proofs provided in the paper such as Fig. 5. This emphasizes the formal engineering benefit over other approaches.

After establishing the patterns in reasoning, our formalization approach typically first focused on the coinductive proof, then adapted this to the inductive proof. Where possible, we directly copied the main body of the proof, then both removed unnecessary coinduction infrastructure and added the necessary inductive infrastructure, including the use of the inductive hypothesis to proof steps. This methodology emphasizes the shared coinductive core of the proofs.

When formalizing the RG Separation Logic extension (discussed in §6.3) we moved in the opposite direction—taking previously completed inductive proofs from the Jackson et al. [28] library and translating them into coinductive proofs. By directly copying the proofs and maintaining the tactic proof style, our earlier observations were further consolidated. Other than the coinductive setup at the start, most of the modifications involved deleting tactic steps using the inductive hypothesis. The tactic style reasoning in fact makes the directness of the coinductive approach even more obvious. Additionally, given the relationship between existential and universally quantified predicates in coinduction and induction respectively (as discussed in §5.1), this process made it easy to see where unnecessary quantification occurred in the original inductive proofs.

From a proof engineering perspective, we observe that although the coinductive proof on paper is more direct with less trivial steps, it is limited in practice by the current formal proof infrastructure available. Induction, particularly on natural numbers, has seen significant development and automation in Isabelle, so more comes "for free". Comparatively, coinduction requires more manual work to use, as outlined in App. F.9, however the removal of the arbitrary n variable does remove the need for some helper lemmas. Thus, the approaches currently require a similar amount of effort, concentrated in slightly different areas. Further support for coinduction in Isabelle would likely maximize the coinductive benefits, providing a slight edge over the inductive safety approach.

As noted in §7 one of the key insights coming from the mechanization was the necessity of up-to coinduction to re-establish the pattern of simplification when moving from the inductive to coinductive proof for reasoning on while. This observation originally came when transforming the existing inductive proofs for RG Separation Logic to coinductive proofs. Unlike the earlier cases, small changes to the inductive proofs were not sufficient to establish a coinductive proof. In fact, the approach used by Jackson et al. [28] (likely unknowingly) leveraged a variant of a form of induction which we call approximation-preserving. It was easy to adapt this approach in the while case for our simplified counting-based semantics, leading to the challenge of developing a coinductive equivalent. Interestingly, our resultant observations on the relationship between inductive and coinductive enhancements made it

clear to see some added redundancy in the original inductive approach, where a monotonicity condition was unnecessarily included on n.

As noted in §5.2 the elementary nature of the inductive enhancements can be an advantage intuitively compared to up-to coinductive proofs. Specifically, as noted in App. F.9, up-to coinduction currently requires a significant amount of manual setup work in Isabelle/HOL, whereas the inductive enhancements follow easily using the existing induction tooling. Thus, from a proof-engineering perspective given the currently available tooling, when up-to coinductive reasoning is required the inductive approach is much more straightforward. However, given the main body of the proof using up-to coinduction is significantly simpler than its inductive counterpart, that further automated support for such coinductive techniques could lead to easier coinductive proofs in the future.