A Sequent Calculus For
Trace Formula Implication

Niklas Heidler[0009−0001−9944−7587] and Reiner Hähnle[0000−0001−8000−7613]

Technical University of Darmstadt, Germany
`<firstName>.<lastName>@tu-darmstadt.de`

Abstract. Specification languages are essential in deductive program verification, but they are usually based on first-order logic, hence less expressive than the programs they specify. Recently, trace specification logics with fixed points that are at least as expressive as their target programs were proposed. This makes it possible to specify not merely pre- and postconditions, but the whole trace of even recursive programs. Previous work established a sound and complete calculus to determine whether a program satisfies a given trace formula. However, the applicability of the calculus and prospects for mechanized verification rely on the ability to prove consequence between trace formulas. We present a sound sequent calculus for proving implication (i.e. trace inclusion) between trace formulas. To handle fixed point operations with an unknown recursive bound, fixed point induction rules are used. We also employ contracts and μ-formula synchronization. While this does not yet result in a complete calculus for trace formula implication, it is possible to prove many non-trivial properties.

Keywords: Program specification, fixed point logic, μ-calculus

1 Introduction

There exist a variety of ways to specify and verify program properties in a mechanized fashion. In *Model Checking* [4], temporal logic, such as Linear Temporal Logic (LTL) or Computation Tree Logic (CTL) is used to specify program behaviour. During verification, a model of the given program and its temporal logic specification are finitely unwound, typically by automata constructions. *Deductive Verification* [7] uses first-order logic (FOL) to formalize procedure contracts in Hoare calculus [12] or in program logic [2] to prove that a given first-order postcondition holds in any state reachable by executing the given procedure, assuming that a precondition held in the start state.

It is interesting to note that—with few exceptions [14, 18]— specification languages in deductive verification are weaker in expressiveness than the programs they are supposed to specify. Moreover, nearly all deductive verification techniques are based on reasoning about intermediate states, i.e. before and after a procedure call. In this sense, model checking is more natural, because there is a direct correspondence between the program model and its specification. However,

LTL and CTL, certain extensions [1] notwithstanding, cannot express modular verification over contracts and they target *models* of programs. In consequence, an obvious question arises: Is there a logic that permits trace-based *and* contract-based specification of imperative programs with recursive procedures that has a natural correspondence between program and specification?

This was recently answered affirmatively in the form of a trace specification logic with smallest fixed points. Here, trace formulas Φ specify a (possibly infinite) set of finite computation traces generated by a program S from a simple imperative language `Rec` with recursive procedure declarations. Judgments take the form $S : \Phi$ and mean: Any possible execution trace of S is contained in the set of traces characterized by Φ. Gurov & Hähnle [6] provide a sound, complete, and *compositional* proof calculus for judgments of the form $S : \Phi$, where "compositional" means that the rule premises do not introduce intermediate formulas not present in the conclusion. However *weakening* of trace formulas (i.e. prove $S : \Psi$ instead of $S : \Phi$ provided that Ψ implies Φ) is still necessary.

Soundness and completeness of the calculus rest on a strong correspondence between programs and trace formulas: For any `Rec` program S, there exists a *strongest trace formula stf(S)* that characterizes *exactly* the traces generated by S.[1] Hence, $S : \Phi$ is valid iff the traces specified by $stf(S)$ are included in the traces specified by Φ. This implies one can verify a judgment $S : \Phi$ by simply proving the trace formula consequence $stf(S) \models \Phi$. Alternatively, one can use the rules of the calculus to prove $S : \Phi$ directly. Thus, the correspondence between programs and trace formulas creates the opportunity to verify judgments with a program calculus *or* by trace formula consequence. It is also possible to mix both styles, of course. In either case, weakening is needed for completeness, so implication between trace formulas is a crucial ingredient. This requires a separate proof system and such a calculus was considered as future work in [6]. It is the main objective of the present paper.

The consequence relation between formulas in a fixed point logic is a difficult problem—because trace formulas are as expressive as recursive programs it is highly undecidable. Therefore, our investigation into how far one can get with such a calculus, is interesting in its own right. Existing literature has little to say about the topic. The central challenge in the design of a calculus for implication of trace formulas is the handling of fixed point formulas, i.e. formulas with a leading fixed point operator μ. We propose increasingly complex strategies of how to eliminate fixed point formulas, without reaching completeness yet:

1. Straightforward *unfolding* of μ-formulas is sufficient to deal with executions that have concrete bounds (Section 4.2).
2. Fixed point *induction* lets one prove trace inclusion of recursive executions with an unknown (or very high) bound (Section 4.3).

---

[1] The paper [6] even proves the reverse direction: For any trace formula Φ there is a *canonical program* S having exactly the same traces as Φ, establishing a Galois connection between programs and trace formulas. However, this result is not relevant for the present paper.

3. To capture the execution state *after* a fixed point formula we equip the calculus with Hoare-style state-based procedure contracts. The logic and calculus is expressive enough to prove such contracts and to propagate them inside the proofs, without the need to refer to meta theorems (Section 5.1).
4. When proving the consequence relation between two μ-formulas, one often encounters the problem that the execution of their bodies is not *synchronized*. We equip the calculus with μ-formula synchronization rules (Section 5.2) that are able to synchronize recursive variables inside fixed point operations in many, but not in all cases. This is one source of incompleteness.

The paper is structured as follows: In Section 2, we introduce `Rec` programs. Trace formulas are defined in Section 3. Section 4 proposes a basic calculus for trace implication, which is the core of this paper. Section 5 extends the basic calculus with method contracts and μ-formula synchronization. Section 6 refers to related work, while Section 7 concludes the paper and proposes future work. As noted, completeness is elusive at the moment, however, we are able to prove a range of interesting and non-trivial properties, see Appendix A.

2   The Rec Language

We define a simple imperative programming language `Rec` [6] with (recursive) procedure calls.

Definition 1 (Rec Program). *A `Rec` Program is a pair (S, T), where S is a `Rec` Statement generated by the grammar*

$$S ::= \textbf{skip} \mid x := a \mid S; S \mid \textbf{if } b \textbf{ then } S \textbf{ else } S \mid m()$$

and T is a possibly empty sequence M^* of procedure declarations, where each M declares a parameter-less procedure $M \equiv m\{S\}$ consisting of procedure name m and procedure body S. Schema variables a and b range over side-effect free arithmetic and boolean expressions, respectively, that are not further specified.

A program *trace* σ is a, possibly empty, finite sequence of execution *states* s, partial mappings from program variables x to integer values. Regarding the semantics of a program in terms of its finite $traces(S)$ of statements S, we refer to the standard definitions in the literature [6].

Example 1. The factorial `Rec` Program (S_{fac}, T_{fac}) is given by the statement $S_{fac} \equiv y := 1;\ factorial()$ and the procedure table

$T_{fac} \equiv factorial\{\textbf{if } x = 1 \textbf{ then } skip \textbf{ else } y := y * x;\ x := x - 1;\ factorial()\}$

By convention, sequential composition binds stronger that the conditional, i.e. the final three statements form the **else** block. For any start state $s = [x \mapsto i]$ with $i > 0$, the program computes the factorial of x and stores the result in y, i.e. the program terminates in a state s' where $s'(y) = x!$.

$$\llbracket p \rrbracket_{\mathbb{V}} = \{s \mid s \models p\} \cup \{s \cdot \sigma \mid s \models p \wedge \sigma \in State^+\} \qquad \llbracket R \rrbracket_{\mathbb{V}} = \{s \cdot s' \mid R(s, s')\}$$

$$\llbracket \Phi_1 \wedge \Phi_2 \rrbracket_{\mathbb{V}} = \llbracket \Phi_1 \rrbracket_{\mathbb{V}} \cap \llbracket \Phi_2 \rrbracket_{\mathbb{V}} \qquad\qquad\qquad \llbracket \Phi_1 \vee \Phi_2 \rrbracket_{\mathbb{V}} = \llbracket \Phi_1 \rrbracket_{\mathbb{V}} \cup \llbracket \Phi_2 \rrbracket_{\mathbb{V}}$$

$$\llbracket \Phi_1 \frown \Phi_2 \rrbracket_{\mathbb{V}} = \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in \llbracket \Phi_1 \rrbracket_{\mathbb{V}} \wedge s \cdot \sigma' \in \llbracket \Phi_2 \rrbracket_{\mathbb{V}}\} \qquad \llbracket X \rrbracket_{\mathbb{V}} = \mathbb{V}(X)$$

$$\llbracket \mu X.\Phi \rrbracket_{\mathbb{V}} = \bigcap \{\gamma \subseteq State^+ \mid \llbracket \Phi \rrbracket_{\mathbb{V}[X \mapsto \gamma]} \subseteq \gamma\}$$

Fig. 1: Semantics of trace formulas

3   Trace Formulas

We define the *trace formula logic*. Like for `Rec` programs, the semantics of its formulas is given as a set of program traces.

Definition 2 (Trace Formula Syntax). *The grammar of* trace formulas *is*

$$\Phi ::= p \mid R \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \Phi \frown \Phi \mid X \mid \mu X.\Phi$$

where p ranges over first-order state predicates Pred, R ranges over binary relations between states, and X ranges over recursion variables RVar. The binary operator $\frown$ is called chop.[2] We assume R contains at least the relations

$$Id := \{(s, s) \in State^2\} \text{ and } Sb_x^a := \{(s, s') \in State^2 \mid s' = s[x \mapsto \mathbb{A}\llbracket a \rrbracket(s)]\} .$$

Relation Id models a skip and Sb_x^a an assignment. $\mathbb{A}\llbracket a \rrbracket(s)$ refers to the evaluation of arithmetic expression a in state s. Observe that the logic is not closed under negation: only smallest fixed point formulas are permitted.

Definition 3 (Trace Formula Semantics). *Each trace formula Φ evaluates to a set of finite traces. Given a valuation function $\mathbb{V} : RVar \rightarrow P(State^+)$ that maps recursion variables to sets of traces, the semantics of a trace formula Φ under valuation $\mathbb{V}$, denoted $\llbracket \Phi \rrbracket_{\mathbb{V}}$, is defined by the equations in Figure 1. $\llbracket \Phi \rrbracket$ abbreviates $\llbracket \Phi \rrbracket_{\mathbb{V}}$ when $\mathbb{V}$ does not affect the result.*

Observe that $\llbracket \mu X.\Phi \rrbracket_{\mathbb{V}}$ maps to the least fixed point of Φ in the powerset lattice $(P(State^+), \subseteq)$. This is justified by monotonicity of $\lambda\gamma.\llbracket \Phi \rrbracket_{\mathbb{V}[X \mapsto \gamma]}$ and the Knaster-Tarski theorem.

Theorem 1 (Strongest Trace Formula [6]). *For each `Rec` Program (S, T) there exists a closed strongest trace formula Φ with $traces(S) = \llbracket \Phi \rrbracket$.*

The strongest trace formula can be effectively constructed from a given `Rec` program. The details of the construction and the proof are in [6]. The theorem implies that trace formulas are at least as expressive as the `Rec` language.

Example 2. Trace formula Φ_{fac} is the strongest trace formula for (S_{fac}, T_{fac}):

$$\Phi_{fac} \equiv Sb_y^1 \frown Id \frown \Phi_m, \text{ where}$$

$$\Phi_m \equiv \mu X_{fac}.((x = 1 \wedge Id \frown Id) \vee (x \neq 1 \wedge Id \frown Sb_y^{y*x} \frown Sb_x^{x-1} \frown Id \frown X_{fac}))$$

---

[2] It is inspired by Interval Temporal Logic [10] and its use in specification by [16].

Definition 4 (Satisfiability). *A* `Rec` *program S satisfies a trace formula Φ (write $S : \Phi$) iff $traces(S) \subseteq \llbracket \Phi \rrbracket$.*

As noted in the introduction, a sound and complete compositional proof calculus for $S : \Phi$ is given in [6], but its applicability relies on weakening, i.e. the semantic entailment oracle $\Phi \models \Psi$, of which this paper presents the first formal investigation.

4   A Proof Calculus for Trace Formula Consequence

4.1   Sequents

Definition 5 (Sequents). *A sequent in our calculus has the shape $\xi \diamond \Gamma \vdash \Delta$, where $\xi \subseteq RVar \times Pred \times RVar$ and Γ, Δ are sets of trace formulas. A triple $(X, p, X') \in \xi$ is written $(X|p, X')$ as syntactic sugar.*

The purpose of ξ is to specify constraints on the recursion variables occurring in a valuation $\mathbb{V}$. We write $\Gamma \vdash \Delta$ as an abbreviation for $\emptyset \diamond \Gamma \vdash \Delta$ in case ξ is empty or irrelevant. ξ is always empty for a top-level sequent.

Definition 6 (Validity of Sequents). *A sequent $\xi \diamond \Gamma \vdash \Delta$ is* valid, *if for all valuations $\mathbb{V}$ with $\llbracket X \wedge p \rrbracket_{\mathbb{V}} \subseteq \llbracket X' \rrbracket_{\mathbb{V}}$ for all $(X|p, X') \in \xi$, it is the case that $\llbracket \bigwedge \Gamma \rrbracket_{\mathbb{V}} \subseteq \llbracket \bigvee \Delta \rrbracket_{\mathbb{V}}$.*

Example 3. Let X_1 and X_2 be recursion variables. Then

$$(X_1|_{x \geq 0}, X_2) \diamond x = 0, X_1 \vdash X_2$$

is a (trivially) valid sequent, because $(X_1|_{x \geq 0}, X_2)$ already assumes trace inclusion between X_1 and X_2, whenever $x \geq 0$.

4.2   Base Rules

Definition 7 (Program State). *To extract the current state from the antecedent Γ of a sequent, we define $P_\Gamma := \{p \in \Gamma \mid p \in Pred\}$ as the set of all first-order state predicates occurring in Γ.*

First-order Rules. Standard axioms such as `CLOSE`, `TRUE` and `FALSE`, as well as the usual rules of the first-order sequent calculus are not separately listed. They are all valid in our setting.

Rules for Predicates and Binary Relations (Figure 2). The rule `CUT` performs a case distinction on predicate p. In contrast to trace formulas, first-order formulas are closed under negation. Rule `PRED` infers information from the program state in its first premise and adds it to the antecedent of its second premise.

Axiom `REL` handles trace inclusion between binary relations. Observe that the current program state P_Γ further restricts relation R in the antecedent,

$$\text{CUT } \frac{\xi \diamond \Gamma, p \vdash \Delta \quad \xi \diamond \Gamma, \overline{p} \vdash \Delta}{\xi \diamond \Gamma \vdash \Delta} \qquad \text{REL } \frac{}{\xi \diamond \Gamma, R \vdash R', \Delta \underbrace{\{(s,s') \in R \mid s \models P_\Gamma\}}_{R|_{P_\Gamma}} \subseteq R'}$$

$$\text{PRED } \frac{P_\Gamma \vdash q \quad \xi \diamond \Gamma, q \vdash \Delta}{\xi \diamond \Gamma \vdash \Delta} \qquad \text{RVAR } \frac{P_\Gamma \vdash p}{\xi, (X_1|_p, X_2) \diamond \Gamma, X_1 \vdash X_2, \Delta}$$

$$\text{CH-ID } \frac{\xi \diamond P_\Gamma, Id \vdash \Psi_1 \quad \cdots \quad \xi \diamond P_\Gamma, Id \vdash \Psi_n \quad \xi \diamond P_\Gamma, \Phi \vdash \Psi'_1, \ldots, \Psi'_n}{\xi \diamond \Gamma, Id \frown \Phi \vdash \Psi_1 \frown \Psi'_1, \ldots, \Psi_n \frown \Psi'_n, \Delta}$$

$$\text{CH-UPD } \frac{\xi \diamond P_\Gamma, Sb_x^a \vdash \Psi_1 \quad \cdots \quad \xi \diamond P_\Gamma, Sb_x^a \vdash \Psi_n \quad \xi \diamond spc_{x:=a}(P_\Gamma), \Phi \vdash \Psi'_1, \ldots, \Psi'_n}{\xi \diamond \Gamma, Sb_x^a \frown \Phi \vdash \Psi_1 \frown \Psi'_1, \ldots, \Psi_n \frown \Psi'_n, \Delta}$$

Fig. 2: Calculus rules for predicates and relations

$$\vdots$$
$$\text{RVAR } \frac{P_\Gamma^4 \vdash \bigwedge P_\Gamma^1}{(X_1|_{\bigwedge P_\Gamma^1}, X_2) \diamond P_\Gamma^4, X_1 \vdash X_2}$$

$$\text{REL } \frac{}{P_\Gamma^2, Sb_y^{y*x} \vdash R_{inc}^y} Sb_y^{y*x}|_{\mathbf{P_\Gamma^2}} \subseteq R_{inc}^y \qquad \vdots$$
$$\text{CH-UPD } \frac{(X_1|_{\bigwedge P_\Gamma^1}, X_2) \diamond P_\Gamma^3, Sb_x^{x-1} \frown X_1 \vdash R_{inc}^y \frown X_2}{(X_1|_{\bigwedge P_\Gamma^1}, X_2) \diamond P_\Gamma^2, Sb_y^{y*x} \frown Sb_x^{x-1} \frown X_1 \vdash R_{inc}^y \frown R_{inc}^y \frown X_2}$$

Fig. 3: Demonstration of predicate and relation rules

abbreviated as $R|_{P_\Gamma}$. Rule RVAR characterizes trace inclusion between recursion variables based on ξ, and needs to prove the corresponding restricting predicate in its premise.

Rules CH-ID and CH-UPD handle the case where a binary relation occurs at the beginning of the current chop sequence in the antecedent. In both rules, the first n premises ensure that the leading relation of the antecedent infers the leading formulas of corresponding chop operations in the succedent. The inference between the remaining trace formula composites occurs in the final premise. As the leading binary relation in the antecedent may change program variables, the program state may need to be adapted to reflect those changes. For this reason we restrict ourselves to relations Id and Sb_x^a in the antecedent which is sufficient to define strongest trace formulas (the rules can be easily extended to support other binary relations in the antecedent). The program state for the remaining trace is preserved when the leading relation is Id. In case of Sb_x^a, however, the program state P_Γ needs to be updated to its strongest postcondition [5] relative to state update $x := a$, indicated by $spc_{x:=a}(P_\Gamma)$.

Example 4. Consider the following four state predicates $P_\Gamma^1 \equiv \{x \geq 1, y \geq 1\}$, $P_\Gamma^2 \equiv \{x > 1, y \geq 1\}$, $P_\Gamma^3 \equiv \{x > 1, y \geq x\}$ and $P_\Gamma^4 \equiv \{x \geq 1, y > x\}$, and define a new binary relation $R_{inc}^y := \{(s, s') \mid s(y) \leq s'(y)\}$, expressing that program

$$\text{UNFL} \ \frac{\xi \diamond \Gamma, \Phi[\mu X.\Phi/X] \vdash \Delta}{\xi \diamond \Gamma, \mu X.\Phi \vdash \Delta} \qquad\qquad \text{UNFR} \ \frac{\xi \diamond \Gamma \vdash \Psi[\mu X.\Psi/X], \Delta}{\xi \diamond \Gamma \vdash \mu X.\Psi, \Delta}$$

$$\text{LENL} \ \frac{\xi \diamond \Gamma, \mu X.repeat_i(\Phi) \vdash \Delta}{\xi \diamond \Gamma, \mu X.\Phi \vdash \Delta} \ i \geq 1 \qquad \text{LENR} \ \frac{\xi \diamond \Gamma \vdash \mu X.repeat_i(\Psi), \Delta}{\xi \diamond \Gamma \vdash \mu X.\Psi, \Delta} \ i \geq 1$$

Fig. 4: Calculus rules for unfoldings and lengthenings

$$\text{ARB1} \ \frac{\xi \diamond \Gamma \vdash \Psi, \Delta}{\xi \diamond \Gamma \vdash true \frown \Psi, \Delta} \qquad \text{ARB2} \ \frac{\xi \diamond \Gamma, \Phi_1 \frown \Phi_2 \vdash \Phi_1 \frown true \frown \Psi, \Delta}{\xi \diamond \Gamma, \Phi_1 \frown \Phi_2 \vdash true \frown \Psi, \Delta}$$

Fig. 5: Calculus rules for arbitrary traces

variable y does not decrease. An example derivation is in Figure 3. It proves that for the constraints on valuations expressed in P_Γ^1, P_Γ^2, P_Γ^3, P_Γ^4, the sequence of state updates $y := y * x$; $x := x - 1$ can be approximated by non-decreasing predicates of program variable y.

Rules for Unfolding and Lengthening (Figure 4). The rules UNFL and UNFR *unfold* a fixed point formula Φ in the antecedent and succedent, respectively. This is sound, because $\mu X.\Phi$ is the least fixed point, implying that an additional recursive application does not change its semantic evaluation.

Rules LENL and LENR *lengthen* fixed point formula Φ in the antecedent and succedent respectively. Let the repetition of fixed point formulas be defined as

$$repeat_0(\Phi) := \Phi \text{ and } repeat_i(\Phi) := \Phi[repeat_{i-1}(\Phi)/X]) \text{ for } i \geq 1 \ .$$

The rules are sound, because for any recursive procedure m, procedure m with n recursive calls inlined has the same least fixed point as m itself.

Example 5. Let $\Phi \equiv \mu X.(R \vee R \frown X)$ be the fixed point formula modeling transitive closure of a binary relation R. Then its unfolding is $R \vee R \frown \Phi$, while its lengthening by a factor of one is $\mu X.(R \vee R \frown (R \vee R \frown X))$.

Rules for Arbitrary Traces (Figure 5). According to Figure 1, chop sequences $true \frown \Psi$ indicate an arbitrary finite trace, represented by $true$, eventually ending with a desired result Ψ. This closely resembles the *eventually* operator of LTL. Rule ARB1 assumes the situation that Ψ already holds in the current state, while ARB2 assumes Ψ does not hold yet, allowing us to skip the leading formula.

Additional Rules. Rules deemed not necessary to understand the central concept behind the calculus can be found in Appendix B.

4.3   Fixed Point Induction

When encountering a fixed point operation $\mu X.\Phi$ in the antecedent, one possible derivation strategy is repeated usage of rule `UNFL` until the recursion terminates based on the current program state. However, not only does a high recursion bound blow up the proof tree size, recursion with an unknown bound may not terminate at all. This may cause the derivation strategy to be unusable, motivating an alternative approach.

Example 6. Trace formula $Sb_x^{10} \frown \Phi_{fac}$ can be handled by a derivation strategy with repeated unfolding. However, this does not work for just Φ_{fac}, because x then has an unknown value, causing the recursion to have an unknown bound.

In the remaining paper we assume a convention giving a unique name to each recursion variable.

Theorem 2 (Fixed Point Induction).   *For recursion variables X_1, X_2, a predicate I, a valuation $\mathbb{V}$, and trace formulas $\mu X_1.\Phi$, $\mu X_2.\Psi$:*

If $[\![I \wedge X_1]\!]_{\mathbb{V}} \subseteq [\![X_2]\!]_{\mathbb{V}}$ implies $[\![I \wedge \Phi]\!]_{\mathbb{V}} \subseteq [\![\Psi]\!]_{\mathbb{V}}$ then $[\![I \wedge \mu X_1.\Phi]\!]_{\mathbb{V}} \subseteq [\![\mu X_2.\Psi]\!]_{\mathbb{V}}$.

Proof. Let recursion variables X_1, X_2, predicate I, valuation $\mathbb{V}$ and trace formulas $\mu X_1.\Phi$, $\mu X_2.\Psi$ be arbitrary, but fixed. Since $[\![I \wedge X_1]\!]_{\mathbb{V}} = [\![I]\!]_{\mathbb{V}} \cap \mathbb{V}(X_1)$:

$$[\![I \wedge X_1]\!]_{\mathbb{V}} \subseteq [\![X_2]\!]_{\mathbb{V}} \text{ implies } [\![I \wedge \Phi]\!]_{\mathbb{V}} \subseteq [\![\Psi]\!]_{\mathbb{V}}$$
$$\Longleftrightarrow \forall \gamma_1, \gamma_2. \ [\![I]\!]_{\mathbb{V}} \cap \gamma_1 \subseteq \gamma_2 \text{ implies } [\![I]\!]_{\mathbb{V}} \cap [\![\Phi]\!]_{\mathbb{V}[X_1 \mapsto \gamma_1]} \subseteq [\![\Psi]\!]_{\mathbb{V}[X_2 \mapsto \gamma_2]}$$

We define the following γ-sequences:

$$(\gamma_1^i, \gamma_2^i)_{i \geq 0} \text{ with } (\gamma_1^0, \gamma_2^0) = (\varnothing, \varnothing), \ \gamma_1^{i+1} = [\![\Phi]\!]_{\mathbb{V}[X_1 \mapsto \gamma_1^i]}, \ \gamma_2^{i+1} = [\![\Psi]\!]_{\mathbb{V}[X_2 \mapsto \gamma_2^i]}$$

We prove by natural induction over i that $[\![I]\!]_{\mathbb{V}} \cap \gamma_1^i \subseteq \gamma_2^i$ for every $i \geq 0$. In the case $i = 0$ we have $[\![I]\!]_{\mathbb{V}} \cap \gamma_1^0 = [\![I]\!]_{\mathbb{V}} \cap \varnothing = \varnothing \subseteq \gamma_2^0$.

Assume as the induction hypothesis that $[\![I]\!]_{\mathbb{V}} \cap \gamma_1^i \subseteq \gamma_2^i$ for a fixed $i \geq 0$. Using our premise, this implies $[\![I]\!]_{\mathbb{V}} \cap [\![\Phi]\!]_{\mathbb{V}[X_1 \mapsto \gamma_1^i]} \subseteq [\![\Psi]\!]_{\mathbb{V}[X_2 \mapsto \gamma_2^i]}$. Then also

$$[\![I]\!]_{\mathbb{V}} \cap \gamma_1^{i+1} = [\![I]\!]_{\mathbb{V}} \cap [\![\Phi]\!]_{\mathbb{V}[X_1 \mapsto \gamma_1^i]} \subseteq [\![\Psi]\!]_{\mathbb{V}[X_2 \mapsto \gamma_2^i]} = \gamma_2^{i+1}.$$

Both sequences must —after possibly infinitely many steps— reach their least fixed points. This means that $[\![I]\!]_{\mathbb{V}} \cap [\![\mu X_1.\Phi]\!]_{\mathbb{V}} \subseteq [\![\mu X_2.\Psi]\!]_{\mathbb{V}}$ must hold. This is equivalent to our proof obligation $[\![I \wedge \mu X_1.\Phi]\!]_{\mathbb{V}} \subseteq [\![\mu X_2.\Psi]\!]_{\mathbb{V}}$.   □

Fixed Point Induction Rule (Figure 6). Rule `FPI` makes use of the theorem above to infer trace inclusion between fixed point formulas. Invariant I allows us to preserve program state information for the derivation of an arbitrary recursive iteration. The first premise establishes that the invariant holds initially. The second premise then takes the shape of the fixed point induction assumption as in Theorem 2, representing an arbitrary recursive iteration. Note that this

$$\text{FPI} \ \frac{P_\Gamma \vdash I \qquad \xi, (X_1|_I, X_2) \diamond I, \Phi \vdash \Psi}{\xi \diamond \Gamma, \mu X_1.\Phi \vdash \mu X_2.\Psi, \Delta}$$

Fig. 6: Fixed point induction rule

$$\text{FPI} \ \frac{\text{CLOSE} \ \dfrac{}{x \geq 0, y = 1 \vdash \bigwedge P_\Gamma^1} \qquad (X_{fac}|_{\bigwedge P_\Gamma^1}, X_{inc}) \diamond \bigwedge P_\Gamma^1, \Phi'_{fac} \vdash repeat_3(\Phi'_{inc})}{\begin{array}{c} \vdots \\ \text{LENR} \ \dfrac{x \geq 0, y = 1, \mu X_{fac}.\Phi'_{fac} \vdash \mu X_{inc}.repeat_3(\Phi'_{inc})}{x \geq 0, y = 1, \mu X_{fac}.\Phi'_{fac} \vdash \mu X_{inc}.\Phi'_{inc}} \ 3 \geq 1 \\ \vdots \\ x \geq 0, Sb_y^1 \frown Id \frown \mu X_{fac}.\Phi'_{fac} \vdash Sb_y^1 \frown \mu X_{inc}.\Phi'_{inc} \\ \vdots \\ Sb_y^1 \frown Id \frown \mu X_{fac}.\Phi'_{fac} \vdash Sb_y^1 \frown \mu X_{inc}.\Phi'_{inc} \vee x < 0 \end{array}}$$

Fig. 7: Demonstration of fixed point induction

premise also enforces the invariant to be preserved, as the derivation between recursion variables X_1, X_2 can only be proven if the invariant holds in the program state before X_1 (see rule RVAR). An alternative fixed point rule can be found in Appendix B.

Example 7. A derivation using rule FPI is in Figure 7: We prove that the factorial program S_{fac} never decreases variable y after its initialization, or else x is initialized with a negative value. For better readability, we use abbreviations:

$$\Phi'_{fac} \equiv ((x = 1 \wedge Id \frown Id) \vee (x \neq 1 \wedge Id \frown Sb_y^{y*x} \frown Sb_x^{x-1} \frown Id \frown X_{fac}))$$

$$\Phi'_{inc} \equiv R_{inc}^y \vee R_{inc}^y \frown X_{inc}$$

Before usage of FPI, trace lengthening is needed to synchronize trace lengths and positions of recursion variable occurrences. Lengthening Φ'_{inc} by a factor of three yields $R_{inc}^y \frown R_{inc}^y \frown R_{inc}^y \frown R_{inc}^y \frown X_{inc}$ as its chop sequence, which synchronizes with the right disjunct in Φ'_{fac}. The left disjunct also synchronizes due to the occurrence of $R_{inc}^y \frown R_{inc}^y$.

Theorem 3 (Soundness).   *The calculus rules presented in this section are sound, implying that only valid sequents are derivable.*

Due to its length, the soundness proof has been moved to Appendix B.

5   Calculus Extensions

5.1   Contracts

The base rules of the calculus we established so far expose a major source of incompleteness: If in an antecedent the fixed point operation or the recursion

variable occurs *non-tail recursively*, such as in $X^\frown\Phi$ or $(\mu X.\Psi)^\frown\Phi$, then there is no rule to continue a derivation. The root cause is that the effect that a fixed point or a recursion variable has on the execution state is unknown. For this reason, all the rules dealing with fixed points so far permit only a single formula in the antecedent. The standard solution in deductive verification to deal with such a situation are *contracts* [7] that summarize the execution state after a complex statement.

Definition 8 (Procedure Contract). *A state-based procedure contract for a given trace formula Φ is a pair $(pre, post)$ of precondition $pre \in Pred$ and postcondition post $\in Pred$. Postconditions may contain fresh program variables x_{old} containing the value of variables x in Φ in the execution state before Φ is evaluated.*

While contracts may approximate *any* kind of trace formula, we kept the attribute "procedure", because the trace formula of a contract can be thought of as the body of a procedure declaration and this is also how we use contracts. Intuitively, a procedure contract $(pre, post)$ is *valid* for a trace formula Φ, if the postcondition is satisfied in the execution state after evaluation of Φ, assuming the precondition is satisfied in the execution state before evaluation of Φ.

Example 8. A valid procedure contract for trace formula Φ_m in Example 2 is

$$(x \geq 1, \ y = y_{old} * x_{old}! \wedge x = 1) \ .$$

We *encode* the intuitive validity of a procedure contract formally as trace inclusion.

Definition 9 (Contract Encoding). *Let $(v^i)_{1 \leq i \leq n}$ be all program variables occurring in Φ and $(v^i_{old})_{1 \leq i \leq n}$ fresh program variables. A procedure contract $(pre, post)$ is* valid *for Φ in $\mathbb{V}$ iff*

$$\underbrace{[\![\bigwedge v^i_{old} = v^i \wedge pre \wedge \Phi^\frown true]\!]_{\mathbb{V}}}_{\langle pre(\Phi)\rangle} \subseteq \underbrace{[\![\Phi^\frown post]\!]_{\mathbb{V}}}_{\langle post(\Phi)\rangle} \ .$$

In the following, we use abbreviations $\langle pre(\Phi)\rangle$, $\langle post(\Phi)\rangle$ for the encoding of the pre- and postcondition, respectively, as indicated above. The encoding expresses: Assuming precondition *pre* holds and the information about the execution state *before the evaluation* of Φ is memorized using fresh variables v^i_{old}, then after evaluating Φ we reach a state in the antecedent that implies *post* in the succedent. Observe that to model this as a trace inclusion formula, we have to copy the formula Φ into the succedent to ensure that the traces match.

Theorem 4 (Fixed Point Induction on Contracts). *For any recursion variable X, trace formula Φ, valuation $\mathbb{V}$, and procedure contract $(pre, post)$, if the validity of $(pre, post)$ for X in $\mathbb{V}$ implies its validity for Φ in $\mathbb{V}$, then it must also be valid for $\mu X.\Phi$ in $\mathbb{V}$.*

$$\text{MC}\quad \frac{\begin{array}{cc} v_{old}^i \in fresh(Var) & \mathbb{C}' = \mathbb{C}[m \mapsto (pre, post)] \\ \xi \diamond \langle pre(\varPhi) \rangle \vdash_{\mathbb{C}'} \langle post(\varPhi) \rangle & \xi \diamond \varGamma, \mu X_{m.}\varPhi \vdash_{\mathbb{C}'} \varDelta \end{array}}{\xi \diamond \varGamma, \mu X_{m.}\varPhi \vdash_{\mathbb{C}} \varDelta}$$

$$\text{CH-MC}\quad \frac{\begin{array}{cc} v_{old}^i \in fresh(Var) & \mathbb{C}' = \mathbb{C}[m \mapsto (pre, post)] \\ \xi \diamond \langle pre(\varPhi_1) \rangle \vdash_{\mathbb{C}'} \langle post(\varPhi_1) \rangle & \xi \diamond \varGamma, (\mu X_{m.}\varPhi_1)^\frown \varPhi_2 \vdash_{\mathbb{C}'} \varDelta \end{array}}{\xi \diamond \varGamma, (\mu X_{m.}\varPhi_1)^\frown \varPhi_2 \vdash_{\mathbb{C}} \varDelta}$$

Fig. 8: Calculus rules for procedure contract validity

The proof for this theorem is in Appendix C.

To integrate contracts into the calculus rules presented in Section 4, we need to remodel sequents so they include information about procedure contracts.

Definition 10 (Sequent with Contract). *A* procedure contract table *is a partial function* $\mathbb{C} : ProcName \rightharpoonup Pred \times Pred$, *assigning each procedure of a program P a possible contract. $\mathbb{C}$ is called* valid *in $\mathbb{V}$ iff for all $m \in dom(\mathbb{C})$, $\mathbb{C}(m)$ is valid for $\mu X_{m.}\varPhi$ in $\mathbb{V}$, where $\mu X_{m.}\varPhi$ is the subformula of $\varGamma$ corresponding to procedure m. A* sequent (with contract) *has the form $\xi \diamond \varGamma \vdash_{\mathbb{C}} \varDelta$, where a procedure contract table $\mathbb{C}$ is added as an index to $\vdash$.*

Note that procedure contracts in our sequents are only available for fixed point formulas $\mu X_{m.}\varPhi$ generated by procedures m via $stf(P)$, which is sufficient for proving sequents of the form $stf(P) \vdash_{\mathbb{C}} \varPsi$.

Definition 11 (Validity of Sequent with Contract). *A sequent $\xi \diamond \varGamma \vdash_{\mathbb{C}} \varDelta$ is* valid*, if for all valuations $\mathbb{V}$, contract table $\mathbb{C}$ valid in $\mathbb{V}$, and $[\![X \wedge p]\!]_{\mathbb{V}} \subseteq [\![X']\!]_{\mathbb{V}}$ holding for all $(X|p, X') \in \xi$ implies $[\![\bigwedge \varGamma]\!]_{\mathbb{V}} \subseteq [\![\bigvee \varDelta]\!]_{\mathbb{V}}$.*

The contract table $\mathbb{C}$ is always empty in a top-level sequent of a derivation. Procedure contracts are added to $\mathbb{C}$ on demand by the calculus rules during a derivation. The rules ensure that all added contracts are proven valid.

Example 9. Continuing Example 8, let $\mathbb{C}(fac) \equiv (x \geq 1, y = y_{old} * x_{old}! \wedge x = 1)$.

$$P_\varGamma^2, \varPhi_m^\frown Sb_x^{x-1} \vdash_{\mathbb{C}} true^\frown x = 0$$

is a valid sequent, because the postcondition guarantees that fac terminates with $x = 1$ before eventually being reduced to $x = 0$.

Procedure Contract Validity Rules (Figure 8). Rules MC and CH-MC prove the validity of a procedure contract for the leading fixed point formula and add it to the procedure contract table $\mathbb{C}$, as can be seen in the right premise. The left premise assumes the procedure contract holds for the internal recursion variable X_m and proves that it hence must also be valid for $\varPhi, \varPhi_1$. Theorem 4 justifies the validity of the contract for the whole fixed point formula $\mu X_{m.}\varPhi$. The proof uses contract table $\mathbb{C}'$ that already assumes the contract for m, because this contract may be assumed to handle recursive calls to m in $\varPhi, \varPhi_1$.

$$\mathbb{C}(m) = (pre, post)$$

$$\text{CH-RVAR} \quad \frac{P_\Gamma \vdash_\mathbb{C} p \wedge pre \qquad \xi \diamond P_\Gamma[v_{old}^i/v^i], post, \Phi \vdash_\mathbb{C} \Psi}{\xi, (X_m|_p, X) \diamond \Gamma, X_m \frown \Phi \vdash_\mathbb{C} X \frown \Psi, \Delta}$$

$$\mathbb{C}(m) = (pre, post)$$

$$\text{CH-FPI} \quad \frac{P_\Gamma \vdash_\mathbb{C} I \wedge pre \qquad \xi, (X_m|_I, X) \diamond I, \Phi_1 \vdash_\mathbb{C} \Psi_1 \qquad \xi \diamond P_\Gamma[v_{old}^i/v^i], post, \Phi_2 \vdash_\mathbb{C} \Psi_2}{\xi \diamond \Gamma, (\mu X_m.\Phi_1) \frown \Phi_2 \vdash_\mathbb{C} (\mu X.\Psi_1) \frown \Psi_2, \Delta}$$

Fig. 9: Calculus rules for procedure contract application

$$\text{CLOSE} \quad \frac{}{y \geq 1, x = 0, y > x \vdash_\mathbb{C} y > x}$$

$$\text{PRED} \quad \frac{}{y \geq 1, x = 0 \vdash_\mathbb{C} y > x}$$

$$\vdots$$

$$\text{CH-FPI} \quad \frac{x_{old} \geq 1, y_{old} \geq 1, y = y_{old} * x_{old}! \wedge x = 1, Sb_x^{x-1} \vdash_\mathbb{C} Sb_x^{x-1} \frown y > x}{P_\Gamma^1, \mu X_{fac}.\Phi'_{fac} \frown Sb_x^{x-1} \vdash_\mathbb{C} \mu X_{inc}.\Phi'_{inc} \frown Sb_x^{x-1} \frown y > x}$$

$$\text{CH-MC} \quad \frac{}{P_\Gamma^1, \mu X_{fac}.\Phi'_{fac} \frown Sb_x^{x-1} \vdash_\varnothing \mu X_{inc}.\Phi'_{inc} \frown Sb_x^{x-1} \frown y > x}$$

Fig. 10: Demonstration of calculus with procedure contracts

Procedure Contract Application Rules (Figure 9). Rule `CH-RVAR` handles the occurrence of a recursion variable X_m in a non-tail recursive setting. In addition to rule `RVAR`, it looks up the procedure contract $(pre, post)$ of m, as indicated by the side condition. Since the recursion variable of procedure m is uniquely named as X_m, the correct procedure is used. The left premise additionally proves the precondition *pre*. The right premise takes the current program state, substitutes every occurrence of variable v^i with variable v_{old}^i, as determined in the contract, and adds the postcondition *post*. This modified program state is then used to continue the derivation of the remaining trace. Rule `CH-FPI` behaves similarly, guaranteeing the derivation of non-tail recursive fixed point formula occurrences.

It is future work to extend the calculus to support multiple contracts for procedures by applying contracts in a hierarchical fashion. This necessitates a modification of the contract table definition and the calculus rules.

Example 10. The calculus with procedure contracts is illustrated by an example in Figure 10. We use the abbreviations from Example 7, $\mathbb{C} := [fac \mapsto (pre, post)]$, $P_\Gamma^1 \equiv \{x \geq 1, y \geq 1\}$ and $(pre, post) \equiv (x \geq 1, y = y_{old} * x_{old}! \wedge x = 1)$. For readability, the derivation only follows the rightmost premises.

Theorem 5 (Soundness of the Calculus with Procedure Contracts).
The calculus rules presented in this section are sound, implying that only valid sequents are derivable.

Due to its length, the soundness proof has been moved to Appendix C.

$$
\text{CH-UPD} \cfrac{P_\Gamma^1, Sb_y^{y*x} \vdash X_{inc} \qquad \cfrac{\begin{array}{c} \textit{not derivable} \\ \vdots \\ (X_{fac}|_{\bigwedge P_\Gamma^1}, X_{inc}) \diamond P_\Gamma^4, X_{fac} \vdash R_{inc}^y \\ \vdots \end{array}}{(X_{fac}|_{\bigwedge P_\Gamma^1}, X_{inc}) \diamond P_\Gamma^3, Sb_x^{x-1} \frown X_{fac} \vdash R_{inc}^y \frown R_{inc}^y}}{(X_{fac}|_{\bigwedge P_\Gamma^1}, X_{inc}) \diamond P_\Gamma^2, Sb_y^{y*x} \frown Sb_x^{x-1} \frown X_{fac} \vdash X_{inc} \frown R_{inc}^y \frown R_{inc}^y}
$$

with the left branch above $P_\Gamma^1, Sb_y^{y*x} \vdash X_{inc}$ being:

$$
\begin{array}{c} \textit{not derivable} \\ \vdots \end{array}
$$

Fig. 11: Demonstration of recursion variable synchronization problem

5.2 Synchronization

To successfully perform a fixed point induction, the trace lengths and positions of the recursion variable occurrences must align in antecedent and succedent. This is not always the case, and it motivates the following synchronization rules.

Example 11. In fixed point formula $\Phi_{inc} := \mu X_{inc}.(R_{inc}^y \vee X_{inc} \frown R_{inc}^y)$, the recursion variable X_{inc} does not occur tail recursively. So any synchronizing formula must have its recursion variable as a leading formula in its chop sequence. This issue is demonstrated in Figure 11: The second disjunct in Φ_{inc} is expanded to $X_{inc} \frown R_{inc}^y \frown R_{inc}^y$, so that in the initial sequent of Figure 11 the positions of recursion variables X_{fac}, X_{inc} misalign.

Definition 12 (Chop Formula). *Let relation R and recursion variable X be fixed.* Primitive chop formulas *are a subclass of trace formulas consisting of chop sequences containing exclusively R or X, specified by the grammar*

$$
\Psi_{(R,X)} ::= R \mid X \mid \Psi_{(R,X)} \frown \Psi_{(R,X)} \ .
$$

The chop formulas $CF_{(R,X)}$ *with fixed R and X are defined as disjunctions over primitive chop formulas, specified by the grammar*

$$
\Phi_{(R,X)} ::= \Psi_{(R,X)} \mid \Psi_{(R,X)} \vee \Phi_{(R,X)} \ .
$$

All recursion variables X occurring in a chop formula are not bound.

Example 12. $\Phi_{sub} \equiv Id \vee Id \frown X \frown Id \frown X \vee Id \frown Id \frown Id$ is a chop formula, i.e. $\Phi_{sub} \in CF_{(Id,X)}$. The subformula $Id \frown X \frown Id \frown X$ is a primitive chop formula.

Let $\Phi \in CF_{(R,X)}$ be a chop formula. Then there exists a natural mapping $gr : CF_{(R,X)} \to G(\{X\}, \{R\}, \delta, X)$ from $\Phi = \bigvee_{1 \leq i \leq n} \varphi_i$ to a context-free grammar with *non-terminal X, terminal R, production rules δ* and *initial non-terminal X*, where production rules δ are defined as $X \to grammatize(\varphi_i)$ for $1 \leq i \leq n$. The function *grammatize* maps each primitive chop formula to a sequence over terminal R and non-terminal X. It is defined by

$$
grammatize(S_1 \frown S_2 \frown \ldots \frown S_n) := S_1 S_2 \cdots S_n \text{ for } S_i \in \{R, X\} \ .
$$

$$\text{SYNC } \frac{\xi \diamond \Gamma \vdash \mu X.\Psi', \Delta}{\xi \diamond \Gamma \vdash \mu X.\Psi, \Delta} \; L(gr(\Psi')) \subseteq L(gr(\Psi))$$

Fig. 12: Calculus rule for μ-formula synchronization

$$\textit{See Figure 3 (cf. Figure 11)}$$
$$\vdots$$
$$(X_{fac}|_{\bigwedge P_\Gamma^1}, X_{inc}) \diamond P_\Gamma^2, Sb_y^{y*x} \frown Sb_x^{x-1} \frown X_{fac} \vdash R_{inc}^y \frown R_{inc}^y \frown X_{inc}$$
$$\vdots$$
$$\text{SYNC } \frac{\Phi_m \vdash \mu X_{inc.}(R_{inc}^y \vee R_{inc}^y \frown X_{inc})}{\Phi_m \vdash \mu X_{inc.}(R_{inc}^y \vee X_{inc} \frown R_{inc}^y)}$$

Fig. 13: Demonstration of μ-formula synchronization

This construction ensures that every $\Phi \in CF_{(R,X)}$ has a *unique grammar representation* $gr(\Phi)$. There is exactly one terminal symbol in $gr(\Phi)$, so we may use Parikh's theorem [17] to deduce that its specified language is regular.

Definition 13. *The* regular trace language *of a chop formula Φ is $L(gr(\Phi))$.*

Example 13. The context-free grammar $gr(\Phi_{sub})$ of the formula from Example 12 is: $X \to Id \mid Id\,X\,Id\,X \mid Id\,Id\,Id$. Now consider the chop formula $\Phi'_{sub} \equiv Id \vee Id \frown Id \frown X \frown X \vee Id \frown Id \frown Id$. Its context-free grammar $gr(\Phi'_{sub})$ has the production rules: $X \to Id \mid Id\,Id\,X\,X \mid Id\,Id\,Id$. The induced regular trace languages are identical, i.e. $L(\Phi_{sub}) = L(\Phi'_{sub})$, implying that both chop formulas generate the exact same traces.

Synchronization Rule (Figure 12). Rule SYNC permits to realign problematic fixed point formulas to synchronize with the antecedent. This requires the trace language of the premise to be smaller than or equal to the trace language of the conclusion. We cannot apply the synchronization rule when the fixed point formula in the premise is not a chop formula (for example, in the case of nested fixed point formulas), which is a limitation to completeness.

Example 14. A derivation with μ-formula synchronization is in Figure 13.

Theorem 6 (Soundness of the Calculus with Synchronization). *The* SYNC *rule is sound, implying that only valid sequents are derivable.*

Due to its length, the soundness proof has been moved to Appendix D.

6   Related Work

Lange et al. [13] analyze the model checking problem over finite transition systems using a modal μ-calculus logic enriched with a chop operator. They focus

on providing a model checker for this extended logic and prove its soundness and completeness. The paper presents a tableau calculus that lets one verify whether a transition system T satisfies a corresponding formula Φ. Formula consequence is *not* addressed.

Walukiewicz [19] extends propositional modal logic with fixpoint operations, resulting in the common μ-calculus. An axiomatization is provided to syntactically infer sequents $\Gamma \vdash \Delta$ that semantically correspond to the implication between μ-calculus formulas. The presented calculus is proven to be *sound* and *complete*. In contrast to the present paper, the logic syntax contains modal connectives, but neither relations nor the chop operator.

Müller-Olm [15] extends the classical modal μ-calculus with chop, which is semantically interpreted using *predicate transformers*. The paper focuses on proving that any context-free process has a characteristic formula up to bisimulation or simulation. The paper further analyzes decidability and expressiveness of this logic, but reasoning about formula consequence is *not* discussed.

7   Conclusion

We designed a sound calculus to prove formula consequence in a trace logic with smallest fixed points, chop, and binary relations. The significance of the logic derives from the fact that it can characterize the behavior of imperative programs with recursive procedures. To prove the judgment $S : \Phi$ that a program S conforms to a trace formula specification Φ, it is necessary to infer consequence relations $\Phi \models \Psi$ of trace formulas [6].

The calculus presented here predictably uses fixed point induction as its central inference rule, but in its standard form this turns out not to be very useful. The reason is the presence of the chop operator which (i) necessitates to approximate the state *after* evaluation of the first constituent in a chop formula and (ii) may cause misalignment among the bodies of smallest fixed point formulas. We added *contracts* for fixed point formulas and grammar-based realignment, respectively, to mitigate these issues. We have not seen such mechanisms in the literature on proof systems related to μ-calculus and believe these ideas constitute an interesting and viable approach to make such calculi more complete.

At the same time, both presented solutions are clearly incomplete: Regarding (i), consequence between fixed points with unbounded iterations and a formula like $true \frown \Phi$ cannot be proven: This requires to track state changes *during* the fixed point evaluation, between iterations. Related to (ii), μ-formula synchronization was defined for a specific subclass of trace formulas. Direct generalization of grammar-based alignment leads to the inclusion problem of context-free grammars which is undecidable.

In the future we want to investigate how the novel concepts—contracts and grammar-based alignment—can be generalized towards completeness and how they can be employed in automated proof search. It is also interesting to analyze the practicality of an integration of this calculus with related calculi relying on trace-based judgments [8, 9].

References

1. Alur, R., Etessami, K., Madhusudan, P.: A temporal logic of nested calls and returns. In: Jensen, K., Podelski, A. (eds.) Tools and Algorithms for the Construction and Analysis of Systems, 10th Intl. Conf., TACAS, Barcelona, Spain. LNCS, vol. 2988, pp. 467–481. Springer (2004). https://doi.org/10.1007/978-3-540-24730-2_35
2. Beckert, B., Bubel, R., Drodt, D., Hähnle, R., Lanzinger, F., Pfeifer, W., Ulbrich, M., Weigl, A.: The Java verification tool KeY: A tutorial. In: Platzer, A., Rozier, K.Y., Pradella, M., Rossi, M. (eds.) Proc. 26th Intl. Symp. on Formal Methods, Milan, Italy. LNCS, vol. 14934, pp. 597–623. Springer, Cham (Sep 2024). https://doi.org/10.1007/978-3-031-71177-0_32
3. Börger, E.: Dijkstra Edsger W. and Scholten Carel S. Predicate calculus and program semantics. The Journal of Symbolic Logic **59**, 673–678 (Jun 2014). https://doi.org/10.2307/2275420
4. Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press (2001)
5. Dijkstra, E.W.: A Discipline of Programming. Prentice Hall, Inc (1976)
6. Gurov, D., Hähnle, R.: An expressive trace logic for recursive programs. In: Fernandez, M. (ed.) Proc. 10th Intl. Conf. on Formal Structures for Computation and Deduction, Birmingham, UK. LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2025), pre-print available at doi.org/10.48550/arXiv.2411.13125
7. Hähnle, R., Huisman, M.: Deductive software verification: From pen-and-paper proofs to industrial tools. In: Steffen, B., Woeginger, G. (eds.) Computing and Software Science: State of the Art and Perspectives. LNCS, vol. 10000, pp. 345–373. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-91908-9_18
8. Hähnle, R., Kamburjan, E., Scaletta, M.: Context-aware trace contracts. In: De Boer, F., Damiani, F., Hähnle, R., Johnsen, E.B., Kamburjan, E. (eds.) Active Object Languages: Current Research Trends. LNCS, vol. 14360, pp. 292–325. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-51060-1_11
9. Hähnle, R., Scaletta, M., Kamburjan, E.: Herding CATs. In: Ferreira, C., Willemse, T. (eds.) 21st Intl. Conf. on Software Engineering and Formal Methods, SEFM, Eindhoven, The Netherlands. LNCS, vol. 14323, pp. 1–6. Springer, Cham (2023)
10. Halpern, J.Y., Manna, Z., Moszkowski, B.C.: A hardware semantics based on temporal intervals. In: Díaz, J. (ed.) Automata, Languages and Programming, 10th Colloquium, Barcelona, Spain. LNCS, vol. 154, pp. 278–291. Springer, Heidelberg (1983). https://doi.org/10.1007/BFb0036915
11. Heidler, N.: A Calculus for Trace Formula Implication (Sep 2024), https://doi.org/10.26083/tuprints-00029959
12. Hoare, C.A.R.: An axiomatic basis for computer programming. Comm. of the ACM **12**(10), 576–580, 583 (Oct 1969)
13. Lange, M., Stirling, C.: Model checking fixed point logic with chop. In: Nielsen, M., Engberg, U. (eds.) Foundations of Software Science and Computation Structures. pp. 250–263. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
14. McGuire, H., Manna, Z., Waldinger, R.J.: Annotation-based deduction in temporal logic. In: Gabbay, D.M., Ohlbach, H.J. (eds.) Temporal Logic, First Intl. Conf., ICTL, Bonn, Germany. LNCS, vol. 827, pp. 430–444. Springer, Berlin, Heidelberg (1994). https://doi.org/10.1007/BFB0014003
15. Müller-Olm, M.: A modal fixpoint logic with chop. In: Meinel, C., Tison, S. (eds.) STACS. pp. 510–520. Springer, Berlin, Heidelberg (1999)

16. Nakata, K., Uustalu, T.: Trace-based coinductive operational semantics for While. In: Theorem Proving in Higher Order Logics (TPHOLs). LNCS, vol. 5674, pp. 375–390. Springer, Berlin Heidelberg (2009). https://doi.org/10.1007/978-3-642-03359-9_26

17. Parikh, R.J.: On context-free languages. J. ACM **13**(4), 570–581 (Oct 1966). https://doi.org/10.1145/321356.321364

18. Sprenger, C., Dam, M.: On global induction mechanisms in a μ-calculus with explicit approximations. Theoretical Informatics and Applications **37**(4), 365–391 (2003), http://www.edpsciences.org/articles/ita/pdf/2003/04/ita0317.pdf

19. Walukiewicz, I.: On completeness of the mu-calculus. In: Proc. Eighth Annual Symp. on Logic in Computer Science (LICS), Montreal, Canada. pp. 136–146. IEEE Computer Society (1993). https://doi.org/10.1109/LICS.1993.287593

A   Additional Examples

In addition to the running example used throughout the paper, we succeeded to prove several non-trivial, interesting properties of programs. The proofs are executed in the calculus for judgments $S : \Phi$ in [6], while necessary weakening steps were proven in the calculus presented here. The derivations can be found in [11].

1. Let program S_{down} be a program that decreases a variable x by 2 until x reaches the value 0. Afterwards, it further decreases variable x by 1. Whether the recursion is entered depends on the initial value of x.

   $S_{down} \equiv down()$ with
   $$down\{\textbf{if } x = 0 \textbf{ then } x := x - 1 \textbf{ else } x := x - 2; down()\}$$

   The following properties of this program were proven:
   (a) Variable x never increases through the program execution:

   $$\mu X_{dec.} R^x_{dec} \vee R^x_{dec} \frown X_{dec}$$

   with $R^x_{dec} := \{(s, s') \in State \times State \mid \mathbb{A}[\![x]\!](s) \geq \mathbb{A}[\![x]\!](s')\}$.
   (b) If x is *even* and *non-negative*, then x will eventually reach value 0. Afterwards, x will eventually reach value -1:

   $$\overline{even(x)} \vee x < 0 \vee true \frown x = 0 \frown x = -1$$

2. Let Program S_{fac} compute the factorial of 10 and store the result of the computation in variable y.

   $S_{fac} \equiv x := 10; y := 1; factorial()$ with
   $$factorial\{\textbf{if } x = 1 \textbf{ then } skip \textbf{ else } y := y * x; x := x - 1; factorial()\}$$

   The following property of this program was proven:

   Variable y will eventually map to 10!: $true \frown y = 10!$

3. Let program S_{pow} compute the power y^x and store the result in variable z. This is a program with mutually recursive procedures.

   $S_{pow} \equiv z := 1; pow()$ with
   $$pow\{\textbf{if } x = 1 \textbf{ then } skip \textbf{ else } z := z * y; subtract()\} \text{ and}$$
   $$subtract\{x := x - 1; pow()\}$$

   The following property of this program was proven:

   Either variable z never changes after its initialization or variable x will eventually change:

   $$(Sb^1_z \frown \mu X_{zstat.} (R^z_{stat} \vee R^z_{stat} \frown X_{zstat})) \vee$$
   $$(\mu X_{xstat.} R^x_{stat} \vee R^x_{stat} \frown X_{xstat}) \frown R^x_{change} \frown true$$

   with $R^x_{stat} := \{(s, s') \mid s(x) = s'(x)\}$, $R^x_{change} := \{(s, s') \mid s(x) \neq s'(x)\}$.

4. Let $S_{contract}$ be a program behaving as follows. If x is 0, the program terminates. If $x > 0$, then x is decreased by 1, before the method is called recursively and ev is set to 0. If $x < 0$, then x is increased by 1, before the method is called recursively and ev is set to 1. This is an example of a non-linear, non-tail recursive program with unbounded behavior.

$$S_{contract} \equiv main() \text{ with}$$
$$main\{\textbf{if } x = 0 \textbf{ then } skip \textbf{ else}$$
$$\textbf{if } x > 0$$
$$\textbf{then } x := x - 1; main(); ev := 0$$
$$\textbf{else } x := x + 1; main(); ev := 1\}$$

The following property of this program was proven:

At some point a state is reached where ev is 0 or ev is 1 and x is 0 assuming x is initialized with $x \neq 0$:

$$x = 0 \vee true \frown (ev = 0 \vee ev = 1) \wedge x = 0$$

B   Additional Material Relating to Section 4

B.1   Additional Base Rules

$$\texttt{CH-PREDL}\ \frac{\xi \diamond \Gamma, p, p \frown \Phi \vdash \Delta}{\xi \diamond \Gamma, p \frown \Phi \vdash \Delta} \qquad\qquad \texttt{CH-PREDR}\ \frac{\xi \diamond \Gamma, q \vdash true \frown \Psi, \Delta}{\xi \diamond \Gamma, q \vdash q \frown \Psi, \Delta}$$

$$\texttt{END-ID}\ \frac{P_\Gamma, Id \vdash \Psi_1 \qquad P_\Gamma \vdash \Psi_2}{\xi \diamond \Gamma, Id \vdash \Psi_1 \frown \Psi_2} \qquad \texttt{END-UPD}\ \frac{P_\Gamma, Sb_x^a \vdash \Psi_1 \qquad spc_{x:=a}(P_\Gamma) \vdash \Psi_2}{\xi \diamond \Gamma, Sb_x^a \vdash \Psi_1 \frown \Psi_2}$$

$$\texttt{CH-}\vee\texttt{L}\ \frac{\xi \diamond \Gamma, \Phi_1 \frown \Phi_3 \vdash \Delta \quad \xi \diamond \Gamma, \Phi_2 \frown \Phi_3 \vdash \Delta}{\xi \diamond \Gamma, (\Phi_1 \vee \Phi_2) \frown \Phi_3 \vdash \Delta} \quad \texttt{CH-}\wedge\texttt{L}\ \frac{\xi \diamond \Gamma, \Phi_1 \frown \Phi_3, \Phi_2 \frown \Phi_3 \vdash \Delta}{\xi \diamond \Gamma, (\Phi_1 \wedge \Phi_2) \frown \Phi_3 \vdash \Delta}$$

$$\texttt{CH-}\wedge\texttt{R}\ \frac{\xi \diamond \Gamma \vdash \Psi_1 \frown \Psi_3, \Delta \quad \xi \diamond \Gamma \vdash \Psi_2 \frown \Psi_3, \Delta}{\xi \diamond \Gamma \vdash (\Psi_1 \wedge \Psi_2) \frown \Psi_3, \Delta} \quad \texttt{CH-}\vee\texttt{R}\ \frac{\xi \diamond \Gamma \vdash \Psi_1 \frown \Psi_3, \Psi_2 \frown \Psi_3, \Delta}{\xi \diamond \Gamma \vdash (\Psi_1 \vee \Psi_2) \frown \Psi_3, \Delta}$$

$$\texttt{CH-UNFL}\ \frac{\xi \diamond \Gamma, (\Phi[\mu X.\Phi/X]) \frown \Phi' \vdash \Delta}{\xi \diamond \Gamma, (\mu X.\Phi) \frown \Phi' \vdash \Delta} \qquad \texttt{CH-UNFR}\ \frac{\xi \diamond \Gamma \vdash (\Psi[\mu X.\Psi/X]) \frown \Psi', \Delta}{\xi \diamond \Gamma \vdash (\mu X.\Psi) \frown \Psi', \Delta}$$

$$\texttt{CH-LENL}\ \frac{\xi \diamond \Gamma, (\mu X.repeat_i(\Phi)) \frown \Phi' \vdash \Delta}{\xi \diamond \Gamma, (\mu X.\Phi) \frown \Phi' \vdash \Delta}\ i \geq 1 \quad \texttt{CH-LENR}\ \frac{\xi \diamond \Gamma \vdash (\mu X.repeat_i(\Psi)) \frown \Psi', \Delta}{\xi \diamond \Gamma \vdash (\mu X.\Psi) \frown \Psi', \Delta}\ i \geq 1$$

Fig. 14: Additional base rules

B.2   Alternative Fixed Point Induction Rule

$$\texttt{FPI-ALT}\ \frac{\xi \diamond P_\Gamma \vdash I \qquad \xi, (X|_I, \Psi) \diamond I, \Phi \vdash \Psi}{\xi \diamond \Gamma, \mu X.\Phi \vdash \Psi, \Delta}$$

Fig. 15: Alternative fixed point induction rule

This rule requires ξ to accept not only recursion variables, but arbitrary fixed point formulas as its third triple composite. This makes the calculus even more general, covering a wider range of derivable sequents. A exemplary sequent that is derivable with `FPI-ALT`, but *not* with `FPI` could be

$$P_\Gamma^2, \Phi_m \vdash true \frown x = 1$$

B.3   Theorems Needed in the Proof of Theorem 3

Theorem 7 (Distributivity of Disjunction/Conjunction with Chop).
For any trace formulas Φ_1, Φ_2 and Φ_3 and any valuation $\mathbb{V}$, it holds that

$$\llbracket (\Phi_1 \vee \Phi_2) \frown \Phi_3 \rrbracket_{\mathbb{V}} = \llbracket \Phi_1 \frown \Phi_3 \vee \Phi_2 \frown \Phi_3 \rrbracket_{\mathbb{V}}$$

and

$$\llbracket (\Phi_1 \wedge \Phi_2) \frown \Phi_3 \rrbracket_{\mathbb{V}} = \llbracket \Phi_1 \frown \Phi_3 \wedge \Phi_2 \frown \Phi_3 \rrbracket_{\mathbb{V}}$$

Proof. Let us assume trace formulas Φ_1, Φ_2 and Φ_3 are arbitrary, but fixed. Let us also assume valuation $\mathbb{V}$ is arbitrary, but fixed. Then also

$$
\begin{aligned}
&\llbracket (\Phi_1 \vee \Phi_2) \frown \Phi_3 \rrbracket_{\mathbb{V}} \\
&= \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in \llbracket \Phi_1 \rrbracket_{\mathbb{V}} \cup \llbracket \Phi_2 \rrbracket_{\mathbb{V}} \wedge s \cdot \sigma' \in \llbracket \Phi_3 \rrbracket_{\mathbb{V}}\} \\
&= \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in \llbracket \Phi_1 \rrbracket_{\mathbb{V}} \wedge s \cdot \sigma' \in \llbracket \Phi_3 \rrbracket_{\mathbb{V}}\} \\
&\quad \cup \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in \llbracket \Phi_2 \rrbracket_{\mathbb{V}} \wedge s \cdot \sigma' \in \llbracket \Phi_3 \rrbracket_{\mathbb{V}}\} \\
&= \llbracket \Phi_1 \frown \Phi_3 \vee \Phi_2 \frown \Phi_3 \rrbracket_{\mathbb{V}}
\end{aligned}
$$

$$
\begin{aligned}
&\llbracket (\Phi_1 \wedge \Phi_2) \frown \Phi_3 \rrbracket_{\mathbb{V}} \\
&= \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in \llbracket \Phi_1 \rrbracket_{\mathbb{V}} \cap \llbracket \Phi_2 \rrbracket_{\mathbb{V}} \wedge s \cdot \sigma' \in \llbracket \Phi_3 \rrbracket_{\mathbb{V}}\} \\
&= \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in \llbracket \Phi_1 \rrbracket_{\mathbb{V}} \wedge s \cdot \sigma' \in \llbracket \Phi_3 \rrbracket_{\mathbb{V}}\} \\
&\quad \cap \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in \llbracket \Phi_2 \rrbracket_{\mathbb{V}} \wedge s \cdot \sigma' \in \llbracket \Phi_3 \rrbracket_{\mathbb{V}}\} \\
&= \llbracket \Phi_1 \frown \Phi_3 \wedge \Phi_2 \frown \Phi_3 \rrbracket_{\mathbb{V}}
\end{aligned}
$$

$\square$

Theorem 8 (Equivalence of Repetitions inside Fixed Point). *For every recursion variable X, trace formula Φ, valuation $\mathbb{V}$ and every positive natural number $n \geq 1$, it holds that $\llbracket \mu X.\Phi \rrbracket_{\mathbb{V}} = \llbracket \mu X.repeat_n(\Phi) \rrbracket_{\mathbb{V}}$.*

Proof. Let recursion variable X, trace formula Φ, valuation $\mathbb{V}$ and $n \geq 1$ be arbitrary, but fixed. We define the following γ-sequences:

$$(\gamma_1^i, \gamma_2^i)_{i \geq 0} \text{ s.t. } (\gamma_1^0, \gamma_2^0) = (\varnothing, \varnothing) \wedge \gamma_1^{i+1} = \llbracket \Phi \rrbracket_{\mathbb{V}[X \mapsto \gamma_1^i]} \wedge \gamma_2^{i+1} = \llbracket repeat_n(\Phi) \rrbracket_{\mathbb{V}[X \mapsto \gamma_2^i]}$$

We will now prove $\gamma_1^{n*i} = \gamma_2^i$ for every $i \geq 0$ via natural induction over i. First, let $i = 0$. Then trivially $\gamma_1^0 = \varnothing = \gamma_2^0$. For the induction step, we assume $\gamma_1^{n*i} = \gamma_2^i$ for some fixed $i \geq 0$. Then

$$
\begin{aligned}
\gamma_1^{n*(i+1)} &= \gamma_1^{n*i+n} = \llbracket \Phi \rrbracket_{\mathbb{V}[X \mapsto \gamma_1^{n*i+(n-1)}]} = \llbracket \Phi \rrbracket_{\mathbb{V}[X \mapsto \llbracket \Phi \rrbracket_{\dots \mathbb{V}[X \mapsto \gamma_1^{n*i}]}]} \\
&= \llbracket repeat_n(\Phi) \rrbracket_{\mathbb{V}[X \mapsto \gamma_1^{n*i}]} = \llbracket repeat_n(\Phi) \rrbracket_{\mathbb{V}[X \mapsto \gamma_2^i]} = \gamma_2^{i+1}
\end{aligned}
$$

Due to this result and the monotonicity of the function, we know that both sequences must, after possibly infinitely many steps, at some point have reached their least fixed points. Hence, $\llbracket \mu X.\Phi \rrbracket_{\mathbb{V}} = \llbracket \mu X.repeat_n(\Phi) \rrbracket_{\mathbb{V}}$, which is what needed to be shown in the first place. $\square$

B.4   Proof of Theorem 3 (Soundness of the Base Calculus)

Proof. To prove that only valid sequents are derivable, we establish that all calculus rules are *locally sound*. A calculus rule is called *locally sound* if the conclusion is a valid sequent assuming all premises are valid sequents.

(`CASE`). Let us assume $[\![\bigwedge \Gamma \wedge p]\!]_{\mathbb{V}} \subseteq [\![\bigvee \Delta]\!]_{\mathbb{V}}$ and $[\![\bigwedge \Gamma \wedge \bar{p}]\!]_{\mathbb{V}} \subseteq [\![\bigvee \Delta]\!]_{\mathbb{V}}$. To prove $[\![\bigwedge \Gamma]\!]_{\mathbb{V}} \subseteq [\![\bigvee \Delta]\!]_{\mathbb{V}}$, we perform a case distinction over predicate p. If we assume that p is satisfied in the antecedent, then the first premise trivially concludes the case. In the case that the complement $\bar{p}$ is satisfied in the antecedent, the second premise trivially infers the conclusion.

(`PRED`). Let us assume $[\![\bigwedge P_\Gamma]\!]_{\mathbb{V}} \subseteq [\![q]\!]_{\mathbb{V}}$ and $[\![\bigwedge \Gamma \wedge q]\!]_{\mathbb{V}} \subseteq [\![\bigvee \Delta]\!]_{\mathbb{V}}$. Then also
$[\![\bigwedge \Gamma]\!]_{\mathbb{V}} = [\![\bigwedge \Gamma \wedge \bigwedge P_\Gamma]\!]_{\mathbb{V}} \subseteq [\![\bigwedge \Gamma \wedge q]\!]_{\mathbb{V}} \subseteq [\![\bigvee \Delta]\!]_{\mathbb{V}}$.

(`CH-PREDL`). Let us assume $[\![\bigwedge \Gamma \wedge p \wedge p \frown \Phi]\!]_{\mathbb{V}} \subseteq [\![\bigvee \Delta]\!]_{\mathbb{V}}$. Then also

$$\begin{aligned}
[\![\bigwedge \Gamma \wedge p \frown \Phi]\!]_{\mathbb{V}} &= [\![\bigwedge \Gamma]\!]_{\mathbb{V}} \cap \{ \sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \models p \wedge s \cdot \sigma' \in [\![\Phi]\!]_{\mathbb{V}} \} \\
&= [\![\bigwedge \Gamma]\!]_{\mathbb{V}} \cap \{ s' \cdot \sigma \mid s' \models p \} \cap \{ \sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \models p \wedge s \cdot \sigma' \in [\![\Phi]\!]_{\mathbb{V}} \} \\
&= [\![\bigwedge \Gamma \wedge p \wedge p \frown \Phi]\!]_{\mathbb{V}} \subseteq [\![\bigvee \Delta]\!]_{\mathbb{V}}
\end{aligned}$$

(`CH-PREDR`). Let us assume $[\![\bigwedge \Gamma \wedge q]\!]_{\mathbb{V}} \subseteq [\![true \frown \Psi \vee \bigvee \Delta]\!]_{\mathbb{V}}$. Then also

$$\begin{aligned}
[\![\bigwedge \Gamma \wedge q]\!]_{\mathbb{V}} &= [\![\bigwedge \Gamma \wedge q]\!]_{\mathbb{V}} \cap [\![q]\!]_{\mathbb{V}} \subseteq ([\![true \frown \Psi]\!]_{\mathbb{V}} \cup [\![\bigvee \Delta]\!]_{\mathbb{V}}) \cap [\![q]\!]_{\mathbb{V}} \\
&\subseteq ([\![true \frown \Psi]\!]_{\mathbb{V}} \cap [\![q]\!]_{\mathbb{V}}) \cup [\![\bigvee \Delta]\!]_{\mathbb{V}} \\
&= (\{ s \cdot \sigma \mid s \cdot \sigma \in [\![true \frown \Psi]\!]_{\mathbb{V}} \} \cap \{ s \cdot \sigma \mid s \models q \}) \cup [\![\bigvee \Delta]\!]_{\mathbb{V}} \\
&\subseteq [\![q \frown \Psi \vee \bigvee \Delta]\!]_{\mathbb{V}}
\end{aligned}$$

(`REL`). Let us assume the side condition $R|_{P(\Gamma)} \subseteq R'$ holds, implying that $[\![R]\!]_{\mathbb{V}} \cap [\![\bigwedge P_\Gamma]\!]_{\mathbb{V}} \subseteq [\![R']\!]_{\mathbb{V}}$. Based on this, we conclude

$$[\![\bigwedge \Gamma \wedge R]\!]_{\mathbb{V}} \subseteq [\![R]\!]_{\mathbb{V}} \cap [\![\bigwedge P(\Gamma)]\!]_{\mathbb{V}} \subseteq [\![R']\!]_{\mathbb{V}} \subseteq [\![R' \vee \bigvee \Delta]\!]_{\mathbb{V}}$$

(`RVAR`). Let ξ be arbitrary, but fixed, such that $(X_1|_I, X_2) \in \xi$. As such, $[\![X_1 \wedge I]\!]_{\mathbb{V}} \subseteq [\![X_2]\!]_{\mathbb{V}}$. Let us assume $[\![\bigwedge P_\Gamma]\!]_{\mathbb{V}} \subseteq [\![I]\!]_{\mathbb{V}}$. Then also

$$[\![\bigwedge \Gamma \wedge X_1]\!]_{\mathbb{V}} \subseteq [\![\bigwedge P_\Gamma \wedge X_1]\!]_{\mathbb{V}} \subseteq [\![I \wedge X_1]\!]_{\mathbb{V}} \subseteq [\![X_2]\!]_{\mathbb{V}} \subseteq [\![X_2 \vee \bigvee \Delta]\!]_{\mathbb{V}}$$

(`CH-ID`). Let us assume $[\![\bigwedge P_\Gamma \wedge Id]\!]_{\mathbb{V}} \subseteq [\![\Psi_i]\!]_{\mathbb{V}}$ for all i with $1 \leq i \leq n$ and $[\![\bigwedge P_\Gamma \wedge \Phi_2]\!]_{\mathbb{V}} \subseteq [\![\bigvee_{1 \leq i \leq n} \Psi_i']\!]_{\mathbb{V}}$. We trivially know that for any $(s, s') \in Id$, it

must hold that $s = s'$. As such,

$$\llbracket \bigwedge \Gamma \wedge Id \frown \Phi_2 \rrbracket_\mathbb{V} \subseteq \llbracket \bigwedge P_\Gamma \rrbracket_\mathbb{V} \cap (\llbracket Id \rrbracket_\mathbb{V} \frown \llbracket \Phi_2 \rrbracket_\mathbb{V})$$

$$= \{ s \cdot s' \cdot \sigma \mid s \cdot s' \in \llbracket \bigwedge P_\Gamma \wedge Id \rrbracket_\mathbb{V} \wedge s' \cdot \sigma \in \llbracket \Phi_2 \rrbracket_\mathbb{V} \}$$

$$= \{ s \cdot s' \cdot \sigma \mid s \cdot s' \in \llbracket \bigwedge P_\Gamma \wedge Id \rrbracket_\mathbb{V} \wedge s' \cdot \sigma \in \llbracket \bigwedge P_\Gamma \wedge \Phi_2 \rrbracket_\mathbb{V} \}$$

$$\subseteq \bigcap_{1 \leq i \leq n} \{ s \cdot s' \cdot \sigma \mid s \cdot s' \in \llbracket \Psi_i \rrbracket_\mathbb{V} \wedge s' \cdot \sigma \in \llbracket \bigvee_{1 \leq i \leq n} \Psi_i' \rrbracket_\mathbb{V} \}$$

$$= \bigcup_{1 \leq i \leq n} \{ s \cdot s' \cdot \sigma \mid s \cdot s' \in \llbracket \bigwedge_{1 \leq i \leq n} \Psi_i \rrbracket_\mathbb{V} \wedge s' \cdot \sigma \in \llbracket \Psi_i' \rrbracket_\mathbb{V} \}$$

$$\subseteq \bigcup_{1 \leq i \leq n} \{ s \cdot s' \cdot \sigma \mid s \cdot s' \in \llbracket \Psi_i \rrbracket_\mathbb{V} \wedge s' \cdot \sigma \in \llbracket \Psi_i' \rrbracket_\mathbb{V} \} \subseteq \llbracket \bigvee_{1 \leq i \leq n} \Psi_i \frown \Psi_i' \vee \bigvee \Delta \rrbracket_\mathbb{V}$$

(CH-UPD). Let us assume $\llbracket \bigwedge P_\Gamma \wedge Sb_x^a \rrbracket_\mathbb{V} \subseteq \llbracket \Psi_i \rrbracket_\mathbb{V}$ for all i with $1 \leq i \leq n$ and $\llbracket \bigwedge spc_{x:=a}(P_\Gamma) \wedge \Phi_2 \rrbracket_\mathbb{V} \subseteq \llbracket \bigvee_{1 \leq i \leq n} \Psi_i' \rrbracket_\mathbb{V}$. We know that for any $(s, s') \in Sb_x^a$ with $s \models P_\Gamma$ for some predicate set P_Γ, it is guaranteed that $s' \models spc_{x:=a}(P_\Gamma)$, which is based on the principle of strongest postconditions [3]. As such,

$$\llbracket \bigwedge \Gamma \wedge Sb_x^a \frown \Phi_2 \rrbracket_\mathbb{V} \subseteq \llbracket \bigwedge P_\Gamma \rrbracket_\mathbb{V} \cap (\llbracket Sb_x^a \rrbracket_\mathbb{V} \frown \llbracket \Phi_2 \rrbracket_\mathbb{V})$$

$$= \{ s \cdot s' \cdot \sigma \mid s \cdot s' \in \llbracket \bigwedge P_\Gamma \wedge Sb_x^a \rrbracket_\mathbb{V} \wedge s' \cdot \sigma \in \llbracket \Phi_2 \rrbracket_\mathbb{V} \}$$

$$= \{ s \cdot s' \cdot \sigma \mid s \cdot s' \in \llbracket \bigwedge P_\Gamma \wedge Sb_x^a \rrbracket_\mathbb{V} \wedge s' \cdot \sigma \in \llbracket \bigwedge spc_{x:=a}(P_\Gamma) \wedge \Phi_2 \rrbracket_\mathbb{V} \}$$

$$\subseteq \bigcap_{1 \leq i \leq n} \{ s \cdot s' \cdot \sigma \mid s \cdot s' \in \llbracket \Psi_i \rrbracket_\mathbb{V} \wedge s' \cdot \sigma \in \llbracket \bigvee_{1 \leq i \leq n} \Psi_i' \rrbracket_\mathbb{V} \}$$

$$= \bigcup_{1 \leq i \leq n} \{ s \cdot s' \cdot \sigma \mid s \cdot s' \in \llbracket \bigwedge_{1 \leq i \leq n} \Psi_i \rrbracket_\mathbb{V} \wedge s' \cdot \sigma \in \llbracket \Psi_i' \rrbracket_\mathbb{V} \}$$

$$\subseteq \bigcup_{1 \leq i \leq n} \{ s \cdot s' \cdot \sigma \mid s \cdot s' \in \llbracket \Psi_i \rrbracket_\mathbb{V} \wedge s' \cdot \sigma \in \llbracket \Psi_i' \rrbracket_\mathbb{V} \} \subseteq \llbracket \bigvee_{1 \leq i \leq n} \Psi_i \frown \Psi_i' \vee \bigvee \Delta \rrbracket_\mathbb{V}$$

(END-ID). Let us assume $\llbracket \bigwedge P_\Gamma \wedge Id \rrbracket_\mathbb{V} \subseteq \llbracket \Psi_1 \rrbracket_\mathbb{V}$ and $\llbracket \bigwedge P_\Gamma \rrbracket_\mathbb{V} \subseteq \llbracket \Psi_2 \rrbracket_\mathbb{V}$. We trivially know that for any $(s, s') \in Id$, it must hold that $s = s'$. As such,

$$\llbracket \bigwedge \Gamma \wedge Id \rrbracket_\mathbb{V} \subseteq \llbracket \bigwedge P_\Gamma \wedge Id \rrbracket_\mathbb{V} = \{ s \cdot s' \mid s \cdot s' \in \llbracket Id \rrbracket_\mathbb{V} \wedge s \models \bigwedge P_\Gamma \}$$

$$= \{ s \cdot s' \mid s \cdot s' \in \llbracket Id \rrbracket_\mathbb{V} \wedge s \models \bigwedge P_\Gamma \wedge s' \models \bigwedge P_\Gamma \}$$

$$\subseteq \{ s \cdot s' \mid s \cdot s' \in \llbracket \Psi_1 \rrbracket_\mathbb{V} \wedge s' \models \bigwedge P_\Gamma \} \subseteq \llbracket \Psi_1 \frown \Psi_2 \rrbracket_\mathbb{V}$$

(END-UPD). We assume $\llbracket \bigwedge P_\Gamma \wedge Sb_x^a \rrbracket_\mathbb{V} \subseteq \llbracket \Psi_1 \rrbracket_\mathbb{V}$ and $\llbracket \bigwedge spc_{x:=a}(P_\Gamma) \rrbracket_\mathbb{V} \subseteq \llbracket \Psi_2 \rrbracket_\mathbb{V}$. We know that for any $(s, s') \in Sb_x^a$ with $s \models P_\Gamma$ for some predicate set P_Γ, it is guaranteed that $s' \models spc_{x:=a}(P_\Gamma)$, which is based on the principle

of strongest postconditions [3]. As such,

$$\llbracket \bigwedge \Gamma \wedge Sb_x^a \rrbracket_{\mathbb{V}} \subseteq \llbracket \bigwedge P_\Gamma \wedge Sb_x^a \rrbracket_{\mathbb{V}} = \{s \cdot s' \mid s \cdot s' \in \llbracket Sb_x^a \rrbracket_{\mathbb{V}} \wedge s \models \bigwedge P_\Gamma\}$$

$$= \{s \cdot s' \mid s \cdot s' \in \llbracket Sb_x^a \rrbracket_{\mathbb{V}} \wedge s \models \bigwedge P_\Gamma \wedge s' \models \bigwedge spc_{x:=a}(P_\Gamma)\}$$

$$\subseteq \{s \cdot s' \mid s \cdot s' \in \llbracket \Psi_1 \rrbracket_{\mathbb{V}} \wedge s' \models \bigwedge spc_{x:=a}(P_\Gamma)\} \subseteq \llbracket \Psi_1 \frown \Psi_2 \rrbracket_{\mathbb{V}}$$

(CH-∨L). Assume $\llbracket \bigwedge \Gamma \wedge \Phi_1 \frown \Phi_3 \rrbracket_{\mathbb{V}} \subseteq \llbracket \bigvee \Delta \rrbracket_{\mathbb{V}}$ and $\llbracket \bigwedge \Gamma \wedge \Phi_2 \frown \Phi_3 \rrbracket_{\mathbb{V}} \subseteq \llbracket \bigvee \Delta \rrbracket_{\mathbb{V}}$. Using Theorem 7 where marked with *, we then infer

$$\llbracket \bigwedge \Gamma \wedge (\Phi_1 \vee \Phi_2) \frown \Phi_3 \rrbracket_{\mathbb{V}} \overset{*}{=} \llbracket \bigwedge \Gamma \wedge (\Phi_1 \frown \Phi_3) \vee (\Phi_2 \frown \Phi_3)) \rrbracket_{\mathbb{V}}$$

$$= \llbracket \bigwedge \Gamma \rrbracket_{\mathbb{V}} \cap (\llbracket (\Phi_1 \frown \Phi_3) \rrbracket_{\mathbb{V}} \cup \llbracket (\Phi_2 \frown \Phi_3) \rrbracket_{\mathbb{V}})$$

$$= (\llbracket \bigwedge \Gamma \rrbracket_{\mathbb{V}} \cap \llbracket \Phi_1 \frown \Phi_3 \rrbracket_{\mathbb{V}}) \cup (\llbracket \bigwedge \Gamma \rrbracket_{\mathbb{V}} \cap \llbracket \Phi_2 \frown \Phi_3 \rrbracket_{\mathbb{V}})$$

$$= (\llbracket \bigwedge \Gamma \wedge (\Phi_1 \frown \Phi_3) \rrbracket_{\mathbb{V}}) \cup (\llbracket \bigwedge \Gamma \wedge (\Phi_2 \frown \Phi_3) \rrbracket_{\mathbb{V}}) \subseteq \llbracket \bigvee \Delta \rrbracket_{\mathbb{V}}$$

(CH-∧L). Let us assume $\llbracket \bigwedge \Gamma \wedge \Phi_1 \frown \Phi_3 \wedge \Phi_2 \frown \Phi_3 \rrbracket_{\mathbb{V}} \subseteq \llbracket \bigvee \Delta \rrbracket_{\mathbb{V}}$. Using Theorem 7, we then infer

$$\llbracket \bigwedge \Gamma \wedge (\Phi_1 \wedge \Phi_2) \frown \Phi_3 \rrbracket_{\mathbb{V}} \overset{*}{=} \llbracket \bigwedge \Gamma \wedge \Phi_1 \frown \Phi_3 \wedge \Phi_2 \frown \Phi_3 \rrbracket_{\mathbb{V}} \subseteq \llbracket \bigvee \Delta \rrbracket_{\mathbb{V}}$$

(CH-∧R). Let us assume $\llbracket \bigwedge \Gamma \rrbracket_{\mathbb{V}} \subseteq \llbracket (\Psi_1 \frown \Psi_3) \vee \bigvee \Delta \rrbracket_{\mathbb{V}}$, as well as the proposition $\llbracket \bigwedge \Gamma \rrbracket_{\mathbb{V}} \subseteq \llbracket (\Psi_2 \frown \Psi_3) \vee \bigvee \Delta \rrbracket_{\mathbb{V}}$. Using Theorem 7, we then infer

$$\llbracket \bigwedge \Gamma \rrbracket_{\mathbb{V}} \subseteq (\llbracket (\Psi_1 \frown \Psi_3) \vee \bigvee \Delta \rrbracket_{\mathbb{V}}) \cap (\llbracket (\Psi_2 \frown \Psi_3) \vee \bigvee \Delta \rrbracket_{\mathbb{V}})$$

$$= (\llbracket (\Psi_1 \frown \Psi_3) \rrbracket_{\mathbb{V}} \cup \llbracket \bigvee \Delta \rrbracket_{\mathbb{V}}) \cap (\llbracket (\Psi_2 \frown \Psi_3) \rrbracket_{\mathbb{V}} \cup \llbracket \bigvee \Delta \rrbracket_{\mathbb{V}})$$

$$= (\llbracket (\Psi_1 \frown \Psi_3) \rrbracket_{\mathbb{V}} \cap \llbracket (\Psi_2 \frown \Psi_3) \rrbracket_{\mathbb{V}}) \cup \llbracket \bigvee \Delta \rrbracket_{\mathbb{V}}$$

$$= (\llbracket (\Psi_1 \frown \Psi_3) \wedge (\Psi_2 \frown \Psi_3) \rrbracket_{\mathbb{V}}) \cup \llbracket \bigvee \Delta \rrbracket_{\mathbb{V}} \overset{*}{=} \llbracket (\Psi_1 \wedge \Psi_2) \frown \Psi_3 \vee \bigvee \Delta \rrbracket_{\mathbb{V}}$$

(CH-∨R). Let us assume $\llbracket \bigwedge \Gamma \rrbracket_{\mathbb{V}} \subseteq \llbracket (\Psi_1 \frown \Psi_3) \vee (\Psi_2 \frown \Psi_3) \vee \bigvee \Delta \rrbracket_{\mathbb{V}}$. Using Theorem 7, we then infer

$$\llbracket \bigwedge \Gamma \rrbracket_{\mathbb{V}} \subseteq \llbracket (\Psi_1 \frown \Psi_3) \vee (\Psi_2 \frown \Psi_3) \vee \bigvee \Delta \rrbracket_{\mathbb{V}} \overset{*}{=} \llbracket (\Psi_1 \vee \Psi_2) \frown \Psi_3 \vee \bigvee \Delta \rrbracket_{\mathbb{V}}$$

(ARB1). Let us assume $\llbracket \bigwedge \Gamma \rrbracket_{\mathbb{V}} \subseteq \llbracket \Psi \vee \bigvee \Delta \rrbracket_{\mathbb{V}}$. We then conclude

$$\llbracket \bigwedge \Gamma \rrbracket_{\mathbb{V}} \subseteq \llbracket \Psi \vee \bigvee \Delta \rrbracket_{\mathbb{V}} = \{s \cdot \sigma' \mid s \cdot \sigma' \in \llbracket \Psi \rrbracket_{\mathbb{V}}\} \cup \llbracket \bigvee \Delta \rrbracket_{\mathbb{V}}$$

$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in \llbracket true \rrbracket_{\mathbb{V}} \wedge s \cdot \sigma' \in \llbracket \Psi \rrbracket_{\mathbb{V}}\} \cup \llbracket \bigvee \Delta \rrbracket_{\mathbb{V}}$$

$$= \llbracket true \frown \Psi \vee \bigvee \Delta \rrbracket_{\mathbb{V}}$$

(ARB2). Let us assume $[\![\bigwedge \Gamma \wedge \Phi_1 \frown \Phi_2]\!]_{\mathbb{V}} \subseteq [\![\Phi_1 \frown true \frown \Psi \vee \bigvee \Delta]\!]_{\mathbb{V}}$. Then

$$[\![\bigwedge \Gamma \wedge \Phi_1 \frown \Phi_2]\!]_{\mathbb{V}} \subseteq [\![\Phi_1 \frown true \frown \Psi \vee \bigvee \Delta]\!]_{\mathbb{V}}$$

$$= \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![\Phi_1]\!]_{\mathbb{V}} \wedge s \cdot \sigma' \in [\![true \frown \Psi]\!]_{\mathbb{V}}\} \cup [\![\bigvee \Delta]\!]_{\mathbb{V}}$$

$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![true]\!]_{\mathbb{V}} \wedge s \cdot \sigma' \in [\![true \frown \Psi]\!]_{\mathbb{V}}\} \cup [\![\bigvee \Delta]\!]_{\mathbb{V}}$$

$$= \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![true]\!]_{\mathbb{V}} \wedge s \cdot \sigma' \in [\![\Psi]\!]_{\mathbb{V}}\} \cup [\![\bigvee \Delta]\!]_{\mathbb{V}}$$

$$= [\![true \frown \Psi \vee \bigvee \Delta]\!]_{\mathbb{V}}$$

(UNFL). Let us assume $[\![\bigwedge \Gamma \wedge \Phi[\mu X.\Phi/X]]\!]_{\mathbb{V}} \subseteq [\![\bigvee \Delta]\!]_{\mathbb{V}}$. Due to fixed point unfolding, we trivially also know that $[\![\Phi[\mu X.\Phi/X]]\!]_{\mathbb{V}} = [\![\mu X.\Phi]\!]_{\mathbb{V}}$. As such, $[\![\bigwedge \Gamma \wedge \mu X.\Phi]\!]_{\mathbb{V}} = [\![\bigwedge \Gamma \wedge \Phi[\mu X.\Phi/X]]\!]_{\mathbb{V}} \subseteq [\![\bigvee \Delta]\!]_{\mathbb{V}}$.

(UNFR). Let us assume $[\![\bigwedge \Gamma]\!]_{\mathbb{V}} \subseteq [\![\Psi[\mu X.\Psi/X] \vee \bigvee \Delta]\!]_{\mathbb{V}}$. Due to fixed point unfolding, we trivially also know that $[\![\Psi[\mu X.\Psi/X]]\!]_{\mathbb{V}} = [\![\mu X.\Psi]\!]_{\mathbb{V}}$. As such, $[\![\bigwedge \Gamma]\!]_{\mathbb{V}} \subseteq [\![\Psi[\mu X.\Psi/X] \vee \bigvee \Delta]\!]_{\mathbb{V}} = [\![\mu X.\Psi \vee \bigvee \Delta]\!]_{\mathbb{V}}$.

(LENL). Let us assume $[\![\bigwedge \Gamma \wedge \mu X.repeat_i(\Phi)]\!]_{\mathbb{V}} \subseteq [\![\bigvee \Delta]\!]_{\mathbb{V}}$. Using Theorem 8 (marked with †, we now conclude that $[\![\bigwedge \Gamma \wedge \mu X.\Phi]\!]_{\mathbb{V}} \stackrel{\dagger}{=} [\![\bigwedge \Gamma \wedge \mu X.repeat_i(\Phi)]\!]_{\mathbb{V}} \subseteq [\![\bigvee \Delta]\!]_{\mathbb{V}}$ for all $i \geq 1$.

(LENR). Let us assume $[\![\bigwedge \Gamma]\!]_{\mathbb{V}} \subseteq [\![\mu X.repeat_i(\Psi) \vee \bigvee \Delta]\!]_{\mathbb{V}}$. Using Theorem 8, we now conclude that $[\![\bigwedge \Gamma]\!]_{\mathbb{V}} \subseteq [\![\mu X.repeat_i(\Psi) \vee \bigvee \Delta]\!]_{\mathbb{V}} \stackrel{\dagger}{=} [\![\mu X.\Psi \vee \bigvee \Delta]\!]_{\mathbb{V}}$ for all $i \geq 1$.

(CH-UNFL). Let us assume $[\![\bigwedge \Gamma \wedge (\Phi[\mu X.\Phi/X]) \frown \Phi']\!]_{\mathbb{V}} \subseteq [\![\bigvee \Delta]\!]_{\mathbb{V}}$. Due to fixed point unfolding, we trivially also know that $[\![\Phi[\mu X.\Phi/X]]\!]_{\mathbb{V}} = [\![\mu X.\Phi]\!]_{\mathbb{V}}$. As such, we can also conclude that

$$[\![\bigwedge \Gamma \wedge (\mu X.\Phi) \frown \Phi']\!]_{\mathbb{V}}$$

$$= [\![\bigwedge \Gamma]\!]_{\mathbb{V}} \cap \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![\mu X.\Phi]\!]_{\mathbb{V}} \wedge s \cdot \sigma' \in [\![\Phi']\!]_{\mathbb{V}}\}$$

$$= [\![\bigwedge \Gamma]\!]_{\mathbb{V}} \cap \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![\Phi[\mu X.\Phi/X]]\!]_{\mathbb{V}} \wedge s \cdot \sigma' \in [\![\Phi']\!]_{\mathbb{V}}\}$$

$$= [\![\bigwedge \Gamma \wedge (\Phi[\mu X.\Phi/X]) \frown \Phi']\!]_{\mathbb{V}} \subseteq [\![\bigvee \Delta]\!]_{\mathbb{V}}$$

(CH-UNFR). Let us assume $[\![\bigwedge \Gamma]\!]_{\mathbb{V}} \subseteq [\![(\Psi[\mu X.\Psi/X]) \frown \Psi' \vee \bigvee \Delta]\!]_{\mathbb{V}}$. Due to fixed point unfolding, we trivially also know that $[\![\Psi[\mu X.\Psi/X]]\!]_{\mathbb{V}} = [\![\mu X.\Psi]\!]_{\mathbb{V}}$. As such, we can also conclude that

$$[\![\bigwedge \Gamma]\!]_{\mathbb{V}} \subseteq [\![(\Psi[\mu X.\Psi/X]) \frown \Psi' \vee \bigvee \Delta]\!]_{\mathbb{V}}$$

$$= \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![\Psi[\mu X.\Psi/X]]\!]_{\mathbb{V}} \wedge s \cdot \sigma' \in [\![\Psi']\!]_{\mathbb{V}}\} \cup [\![\bigvee \Delta]\!]_{\mathbb{V}}$$

$$= \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![\mu X.\Psi]\!]_{\mathbb{V}} \wedge s \cdot \sigma' \in [\![\Psi']\!]_{\mathbb{V}}\} \cup [\![\bigvee \Delta]\!]_{\mathbb{V}}$$

$$= [\![(\mu X.\Psi) \frown \Psi' \vee \bigvee \Delta]\!]_{\mathbb{V}}$$

(CH-LENL). Let us assume $[\![\bigwedge \Gamma \wedge (\mu X.repeat_i(\Phi))^\frown \Phi']\!]_{\mathbb{V}} \subseteq [\![\bigvee \Delta]\!]_{\mathbb{V}}$. Using Theorem 8, we can now conclude, that for any $i \geq 1$

$$
\begin{aligned}
&[\![\bigwedge \Gamma \wedge (\mu X.\Phi)^\frown \Phi']\!]_{\mathbb{V}} \\
&= [\![\bigwedge \Gamma]\!]_{\mathbb{V}} \cap \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![\mu X.\Phi]\!]_{\mathbb{V}} \wedge s \cdot \sigma' \in [\![\Phi']\!]_{\mathbb{V}}\} \\
&\overset{\dagger}{=} [\![\bigwedge \Gamma]\!]_{\mathbb{V}} \cap \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![\mu X.repeat_i(\Phi)]\!]_{\mathbb{V}} \wedge s \cdot \sigma' \in [\![\Phi']\!]_{\mathbb{V}}\} \\
&= [\![\bigwedge \Gamma \wedge (\mu X.repeat_i(\Phi))^\frown \Phi']\!]_{\mathbb{V}} \subseteq [\![\bigvee \Delta]\!]_{\mathbb{V}}
\end{aligned}
$$

(CH-LENR). Let us assume $[\![\bigwedge \Gamma]\!]_{\mathbb{V}} \subseteq [\![(\mu X.repeat_i(\Psi))^\frown \Psi' \vee \bigvee \Delta]\!]_{\mathbb{V}}$. Using Theorem 8, we can now conclude, that for any $i \geq 1$

$$
\begin{aligned}
&[\![\bigwedge \Gamma]\!]_{\mathbb{V}} \subseteq [\![(\mu X.repeat_i(\Psi))^\frown \Psi' \vee \bigvee \Delta]\!]_{\mathbb{V}} \\
&= \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![\mu X.repeat_i(\Psi)]\!]_{\mathbb{V}} \wedge s \cdot \sigma' \in [\![\Psi']\!]_{\mathbb{V}}\} \cup [\![\bigvee \Delta]\!]_{\mathbb{V}} \\
&\overset{\dagger}{=} \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![\mu X.\Psi]\!]_{\mathbb{V}} \wedge s \cdot \sigma' \in [\![\Psi']\!]_{\mathbb{V}}\} \cup [\![\bigvee \Delta]\!]_{\mathbb{V}} \\
&= [\![(\mu X.\Psi)^\frown \Psi' \vee \bigvee \Delta]\!]_{\mathbb{V}}
\end{aligned}
$$

(FPI). Let us then assume the premises are valid, i.e.

(1) $[\![P_\Gamma]\!]_{\mathbb{V}} \subseteq [\![I]\!]_{\mathbb{V}}$

(2) If $[\![I \wedge X_1]\!]_{\mathbb{V}} \subseteq [\![X_2]\!]_{\mathbb{V}}$, then also $[\![I \wedge \Phi]\!]_{\mathbb{V}} \subseteq [\![\Psi]\!]_{\mathbb{V}}$

Using the second premise, as well as Theorem 2, we can now infer the proposition $[\![I \wedge \mu X_1.\Phi]\!]_{\mathbb{V}} \subseteq [\![\mu X_2.\Psi]\!]_{\mathbb{V}}$. Hence, we conclude that

$$
[\![\bigwedge \Gamma \wedge \mu X_1.\Phi]\!]_{\mathbb{V}} \subseteq [\![\bigwedge P_\Gamma \wedge \mu X_1.\Phi]\!]_{\mathbb{V}} \subseteq [\![I \wedge \mu X_1.\Phi]\!]_{\mathbb{V}} \subseteq [\![\mu X_2.\Psi \vee \bigvee \Delta]\!]_{\mathbb{V}}
$$

$\square$

C   Proofs of Contract Rules

C.1   Additional Contract Application Rules

$$\text{CH-RVAR-EQ} \quad \frac{\mathbb{C}(m) = (pre, post) \qquad P_\Gamma \vdash_\mathbb{C} pre \qquad \xi \diamond P_\Gamma[v^i_{old}/v^i], post, \Phi \vdash_\mathbb{C} \Psi}{\xi \diamond \Gamma, X_m \frown \Phi \vdash_\mathbb{C} X_m \frown \Psi, \Delta}$$

$$\text{CH-FPI-ALT} \quad \frac{\mathbb{C}(m) = (pre, post) \qquad P_\Gamma \vdash_\mathbb{C} I \wedge pre \quad \xi, (X_m|_I, \Psi_1) \diamond I, \Phi_1 \vdash_\mathbb{C} \Psi_1 \quad \xi \diamond P_\Gamma[v^i_{old}/v^i], post, \Phi_2 \vdash_\mathbb{C} \Psi_2}{\xi \diamond \Gamma, (\mu X_{m}.\Phi_1) \frown \Phi_2 \vdash_\mathbb{C} \Psi_1 \frown \Psi_2, \Delta}$$

Fig. 16: Additional contract application rules

C.2   Proof of Theorem 4

Proof. Let recursion variable X, trace formula Φ, valuation $\mathbb{V}$, and procedure contract $(pre, post)$ be arbitrary, but fixed. Let us assume the validity of $(pre, post)$ for X in $\mathbb{V}$ implies its validity for Φ in $\mathbb{V}$. This is equivalent to saying that $[\![\langle pre(X)\rangle]\!]_\mathbb{V} \subseteq [\![\langle post(X)\rangle]\!]_\mathbb{V}$ implies that $[\![\langle pre(\Phi)\rangle]\!]_\mathbb{V} \subseteq [\![\langle post(\Phi)\rangle]\!]_\mathbb{V}$. Let

$$P \equiv \bigwedge v^i_{old} = v^i \wedge pre$$

be the predicates of the precondition encoding. Using the information contained in our premise, since X specifies an arbitrary trace γ, we can also say that for any trace γ

$$[\![P]\!]_\mathbb{V} \cap \gamma \frown State^+ \subseteq \gamma \frown [\![post]\!]_\mathbb{V}$$
$$\text{implies}$$
$$[\![P]\!]_\mathbb{V} \cap [\![\Phi]\!]_{\mathbb{V}[X \mapsto \gamma]} \frown State^+ \subseteq [\![\Phi]\!]_{\mathbb{V}[X \mapsto \gamma]} \frown [\![post]\!]_\mathbb{V}$$

We can now construct the following γ-sequence:

$$(\gamma^i)_{i \geq 0} \text{ with } \gamma^0 = \varnothing \wedge \gamma^{i+1} = [\![\Phi]\!]_{\mathbb{V}[X \mapsto \gamma^i]}$$

We prove via natural induction over i that for every γ^i with $i \geq 0$: $[\![P]\!]_\mathbb{V} \cap \gamma^i \frown State^+ \subseteq \gamma^i \frown [\![post]\!]_\mathbb{V}$. Let $i = 0$. Then trivially

$$[\![P]\!]_\mathbb{V} \cap \gamma^0 \frown State^+ = [\![P]\!]_\mathbb{V} \cap \varnothing \frown State^+ = \varnothing \subseteq \gamma^0 \frown [\![post]\!]_\mathbb{V}$$

For the induction hypothesis, let $i \geq 0$ be fixed, such that it is guaranteed that $[\![P]\!]_\mathbb{V} \cap \gamma^i \frown State^+ \subseteq \gamma^i \frown [\![post]\!]_\mathbb{V}$ holds. Using our earlier premise, this is equivalent to saying that

$$[\![P]\!]_\mathbb{V} \cap [\![\Phi]\!]_{\mathbb{V}[X \mapsto \gamma^i]} \frown State^+ \subseteq [\![\Phi]\!]_{\mathbb{V}[X \mapsto \gamma^i]} \frown [\![post]\!]_\mathbb{V}$$

Using this information, we can now complete the induction step by inferring that

$$[\![P]\!]_\mathbb{V} \cap \gamma^{i+1} \frown State^+ = [\![P]\!]_\mathbb{V} \cap [\![\Phi]\!]_{\mathbb{V}[X \mapsto \gamma^i]} \frown State^+$$
$$\subseteq [\![\Phi]\!]_{\mathbb{V}[X \mapsto \gamma^i]} \frown [\![post]\!]_\mathbb{V} = \gamma^{i+1} \frown [\![post]\!]_\mathbb{V}$$

Due to the monotonicity of the function, we know the γ-sequence above must, after possibly infinitely many steps, reach its least fixed point. Hence, we can conclude that also

$$[\![P]\!]_\mathbb{V} \cap [\![\mu X.\Phi]\!]_\mathbb{V} \frown State^+ \subseteq [\![\mu X.\Phi]\!]_\mathbb{V} \frown [\![post]\!]_\mathbb{V}$$

This is again equivalent to $[\![\langle pre(\mu X.\Phi)\rangle]\!]_\mathbb{V} \subseteq [\![\langle post(\mu X.\Phi)\rangle]\!]_\mathbb{V}$, which needed to be shown in the first place.   □

C.3   Application of Procedure Contracts

Lemma 1 (Application of Procedure Contracts). *For any trace formulas Φ, Ψ, recursion variable X, precondition pre, postcondition post, predicate P and valuation $\mathbb{V}$, assuming procedure contract $(pre, post)$ holds for Φ in $\mathbb{V}$, it must also hold that*

$$\{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![P \wedge pre \wedge \Phi]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\Psi]\!]_\mathbb{V}\}$$
$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![\Phi]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![P[v^i_{old}/v^i] \wedge post \wedge \Psi]\!]_\mathbb{V}\}$$

Proof. Let us assume trace formulas Φ, Ψ, recursion variable X, precondition *pre*, postcondition *post*, predicate P and valuation $\mathbb{V}$ are arbitrary, but fixed, such that the procedure contract $(pre, post)$ holds for trace formula Φ in $\mathbb{V}$, i.e. $[\![\langle pre(\Phi)\rangle]\!]_\mathbb{V} \subseteq [\![\langle post(\Phi)\rangle]\!]_\mathbb{V}$. This encoding directly implies that

$$[\![\bigwedge v^i_{old} = v^i \wedge pre \wedge \Phi \frown true]\!]_\mathbb{V} \subseteq [\![\Phi \frown post]\!]_\mathbb{V}$$

To infer the theorem, we first add the conjunctions $v^i_{old} = v^i$ to our left formula, which is allowed, as v^i_{old} are assumed to be *new program variables* not included in the formula yet.

$$\{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![P \wedge pre \wedge \Phi]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\Psi]\!]_\mathbb{V}\}$$
$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![P \wedge \bigwedge v^i_{old} = v_i \wedge pre \wedge \Phi]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\Psi]\!]_\mathbb{V}\}$$

In the next step, we can then modify Φ to $\Phi \frown true$ in order to *match* the formula with the encoding of the precondition $\langle pre(\Phi)\rangle$. Due to our matching encoding, we can then use the *validity of the procedure contract*, as given in the premise, in order to add the encoding of the postcondition $\langle post(\Phi)\rangle$ to the formula. This is demonstrated as follows:

$$\{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![P \wedge \bigwedge v^i_{old} = v_i \wedge pre \wedge \Phi]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\Psi]\!]_\mathbb{V}\}$$
$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![P \wedge \bigwedge v^i_{old} = v_i \wedge pre \wedge \Phi \frown true]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\Psi]\!]_\mathbb{V}\}$$
$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![P \wedge \bigwedge v^i_{old} = v_i \wedge \Phi \frown post]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\Psi]\!]_\mathbb{V}\}$$

In the following step, we substitute every occurrence of v^i in P with v^i_{old}, which is possible, as we know that $v^i_{old} = v^i$ for all i. Considering that $post$ holds in the final state of $\sigma \cdot s$, we hence know that $s \models post$. As such, we can also add $post$ as a condition for the initial state of $s \cdot \sigma'$:

$$\{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![P \wedge \bigwedge v^i_{old} = v_i \wedge \varPhi \frown post]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\varPsi]\!]_\mathbb{V}\}$$
$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![P[v^i_{old}/v^i] \wedge \varPhi \frown post]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\varPsi]\!]_\mathbb{V}\}$$
$$= \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![P[v^i_{old}/v^i] \wedge \varPhi]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![post \wedge \varPsi]\!]_\mathbb{V}\}$$

Considering that v^i_{old} are fresh program variables <u>not</u> occurring in $\varPhi$, we know that they stay unchanged during the execution of $\varPhi$. Hence, all information about the old variables <u>before</u> the execution of $\varPhi$ can simply be transferred intact until <u>after</u> the execution of $\varPhi$, which finally proves the lemma, as can be seen below:

$$\{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![P[v^i_{old}/v^i] \wedge \varPhi]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![post \wedge \varPsi]\!]_\mathbb{V}\}$$
$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![\varPhi]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![P[v^i_{old}/v^i] \wedge post \wedge \varPsi]\!]_\mathbb{V}\}$$

$\square$

C.4   Proof of Theorem 5

Proof. We prove that each new rule is *locally sound*.

(`MC`). Let us assume the premises are valid, i.e.

    (1) $\mathbb{C}'$ is valid in $\mathbb{V}$ implies $[\![\langle pre(\varPhi)\rangle]\!]_\mathbb{V} \subseteq [\![\langle post(\varPhi)\rangle]\!]_\mathbb{V}$

    (2) $\mathbb{C}'$ is valid in $\mathbb{V}$ implies $[\![\bigwedge \varGamma \wedge \mu X_{m.}\varPhi]\!]_\mathbb{V} \subseteq [\![\bigvee \varDelta]\!]_\mathbb{V}$

for $\mathbb{C}' = \mathbb{C}[m \mapsto (pre, post)]$ and $v^i_{old} \in fresh(Var)$.

Using the first premise and Theorem 4, we can infer that $(pre, post)$ is valid for $\mu X_{m.}\varPhi$ in $\mathbb{V}$, i.e.

$$[\![\langle pre(\mu X_{m.}\varPhi)\rangle]\!]_\mathbb{V} \subseteq [\![\langle post(\mu X_{m.}\varPhi)\rangle]\!]_\mathbb{V}$$

This only holds because $\mathbb{C}'$ being valid in $\mathbb{V}$ in this context means that $(pre, post)$ is valid for X_m, as no subformula $\mu X_{m.}\varPsi$ can occur in $\varPhi$. As such, $(pre, post)$ is valid for $\mu X_{m.}\varPhi$ in $\mathbb{V}$. The second premise then tells us that

$$[\![\bigwedge \varGamma \wedge \mu X_{m.}\varPhi]\!]_\mathbb{V} \subseteq [\![\bigvee \varDelta]\!]_\mathbb{V}$$

which is needed to be proven in the first place.

(`CH-MC`). Let us assume the premises are valid, i.e.

    (1) $\mathbb{C}'$ is valid in $\mathbb{V}$ implies $[\![\langle pre(\varPhi_1)\rangle]\!]_\mathbb{V} \subseteq [\![\langle post(\varPhi_1)\rangle]\!]_\mathbb{V}$

    (2) $\mathbb{C}'$ is valid in $\mathbb{V}$ implies $[\![\bigwedge \varGamma \wedge (\mu X_{m.}\varPhi_1) \frown \varPhi_2]\!]_\mathbb{V} \subseteq [\![\bigvee \varDelta]\!]_\mathbb{V}$

for $\mathbb{C}' = \mathbb{C}[m \mapsto (pre, post)]$ and $v_{old}^i \in fresh(Var)$.

Using the first premise and Theorem 4, we can infer that $(pre, post)$ is valid for $\mu X_{m.} \Phi_1$ in $\mathbb{V}$, i.e.

$$[\![\langle pre(\mu X_{m.} \Phi_1) \rangle]\!]_\mathbb{V} \subseteq [\![\langle post(\mu X_{m.} \Phi_1) \rangle]\!]_\mathbb{V}$$

This only holds because $\mathbb{C}'$ being valid in $\mathbb{V}$ in this context means that $(pre, post)$ is valid for X_m, as no subformula $\mu X_{m.} \Psi$ can occur in Φ_1. As such, $(pre, post)$ is valid for $\mu X_{m.} \Phi_1$ in $\mathbb{V}$. The second premise then tells us that

$$[\![\bigwedge \Gamma \wedge (\mu X_{m.} \Phi_1) ^\frown \Phi_2]\!]_\mathbb{V} \subseteq [\![\bigvee \Delta]\!]_\mathbb{V}$$

which needed to be proven.

(`CH-RVAR-EQ`). Let us assume the premises are valid, i.e.

(1) $[\![P_\Gamma]\!]_\mathbb{V} \subseteq [\![pre]\!]_\mathbb{V}$

(2) $\mathbb{C}$ being valid in $\mathbb{V}$ implies $[\![\bigwedge P_\Gamma[v_{old}^i/v^i] \wedge post \wedge \Phi]\!]_\mathbb{V} \subseteq [\![\Psi]\!]_\mathbb{V}$

for $\mathbb{C}(m) = (pre, post)$. Since $\mathbb{C}(m) = (pre, post)$, we can assume that $(pre, post)$ holds for X_m in $\mathbb{V}$. Hence, we can apply Lemma 1 to conclude

$$[\![\bigwedge \Gamma \wedge X_m ^\frown \Phi]\!]_\mathbb{V} \subseteq [\![\bigwedge P_\Gamma \wedge X_m ^\frown \Phi]\!]_\mathbb{V}$$
$$= \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![\bigwedge P_\Gamma \wedge X_m]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\Phi]\!]_\mathbb{V}\}$$
$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![\bigwedge P_\Gamma \wedge pre \wedge X_m]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\Phi]\!]_\mathbb{V}\}$$
$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![X_m]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\bigwedge P_\Gamma[v_{old}^i/v^i] \wedge post \wedge \Phi]\!]_\mathbb{V}\}$$
$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![X_m]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\Psi]\!]_\mathbb{V}\} \subseteq [\![X_m ^\frown \Psi \vee \bigvee \Delta]\!]_\mathbb{V}$$

(`CH-RVAR`). Let ξ be arbitrary, but fixed, such that $(X_m|_p, X) \in \xi$. As such, $[\![X_m \wedge p]\!]_\mathbb{V} \subseteq [\![X]\!]_\mathbb{V}$. Let us assume the premises are valid, i.e.

(1) $[\![P_\Gamma]\!]_\mathbb{V} \subseteq [\![p \wedge pre]\!]_\mathbb{V}$

(2) $\mathbb{C}$ being valid in $\mathbb{V}$ implies $[\![\bigwedge P_\Gamma[v_{old}^i/v^i] \wedge post \wedge \Phi]\!]_\mathbb{V} \subseteq [\![\Psi]\!]_\mathbb{V}$

for $\mathbb{C}(m) = (pre, post)$. Since $\mathbb{C}(m) = (pre, post)$, we can assume that $(pre, post)$ holds for X_m in $\mathbb{V}$. Hence, we can apply Lemma 1 to conclude

$$[\![\bigwedge \Gamma \wedge X_m ^\frown \Phi]\!]_\mathbb{V} \subseteq [\![\bigwedge P_\Gamma \wedge X_m ^\frown \Phi]\!]_\mathbb{V}$$
$$= \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![\bigwedge P_\Gamma \wedge X_m]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\Phi]\!]_\mathbb{V}\}$$
$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![p \wedge \bigwedge P_\Gamma \wedge pre \wedge X_m]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\Phi]\!]_\mathbb{V}\}$$
$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![p \wedge X_m]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\bigwedge P_\Gamma[v_{old}^i/v^i] \wedge post \wedge \Phi]\!]_\mathbb{V}\}$$
$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![X]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\Psi]\!]_\mathbb{V}\} \subseteq [\![X ^\frown \Psi \vee \bigvee \Delta]\!]_\mathbb{V}$$

(CH-FPI). Let us assume the premises are valid, i.e.

(1) $[\![P_\Gamma]\!]_\mathbb{V} \subseteq [\![I \wedge pre]\!]_\mathbb{V}$

(2) $(X_m|_I, X) \in \xi$ implies $[\![I \wedge \Phi_1]\!]_\mathbb{V} \subseteq [\![\Psi_1]\!]_\mathbb{V}$.

(3) $\mathbb{C}$ being valid in $\mathbb{V}$ implies $[\![\bigwedge P_\Gamma[v_{old}^i/v^i] \wedge post \wedge \Phi_2]\!]_\mathbb{V} \subseteq [\![\Psi_2]\!]_\mathbb{V}$

for $\mathbb{C}(m) = (pre, post)$. Since $\mathbb{C}(m) = (pre, post)$, we can assume that $(pre, post)$ holds for $\mu X_m.\Phi_1$ in $\mathbb{V}$. Using Lemma 1, we conclude

$$[\![\bigwedge \Gamma \wedge (\mu X_m.\Phi_1)^\frown \Phi_2]\!]_\mathbb{V} \subseteq [\![\bigwedge P_\Gamma \wedge (\mu X_m.\Phi_1)^\frown \Phi_2]\!]_\mathbb{V}$$

$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![\bigwedge P_\Gamma \wedge \mu X_m.\Phi_1]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\Phi_2]\!]_\mathbb{V}\}$$

$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![I \wedge \bigwedge P_\Gamma \wedge pre \wedge \mu X_m.\Phi_1]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\Phi_2]\!]_\mathbb{V}\}$$

$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![I \wedge \mu X_m.\Phi_1]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![P_\Gamma[v_{old}^i/v^i] \wedge post \wedge \Phi_2]\!]_\mathbb{V}\}$$

$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![\mu X.\Psi_1]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\Psi_2]\!]_\mathbb{V}\} \subseteq [\![(\mu X.\Psi_1)^\frown \Psi_2 \vee \bigvee \Delta]\!]$$

(CH-FPI-ALT). Let us assume the premises are valid, i.e.

(1) $[\![P_\Gamma]\!]_\mathbb{V} \subseteq [\![I \wedge pre]\!]_\mathbb{V}$

(2) $(X_m|_I, \Psi_1) \in \xi$ implies $[\![I \wedge \Phi_1]\!]_\mathbb{V} \subseteq [\![\Psi_1]\!]_\mathbb{V}$.

(3) $\mathbb{C}$ being valid in $\mathbb{V}$ implies $[\![\bigwedge P_\Gamma[v_{old}^i/v^i] \wedge post \wedge \Phi_2]\!]_\mathbb{V} \subseteq [\![\Psi_2]\!]_\mathbb{V}$

for $\mathbb{C}(m) = (pre, post)$. Since $\mathbb{C}(m) = (pre, post)$, we can assume that $(pre, post)$ holds for $\mu X_m.\Phi_1$ in $\mathbb{V}$. Using Lemma 1, we conclude

$$[\![\bigwedge \Gamma \wedge (\mu X_m.\Phi_1)^\frown \Phi_2]\!]_\mathbb{V} \subseteq [\![\bigwedge P_\Gamma \wedge (\mu X_m.\Phi_1)^\frown \Phi_2]\!]_\mathbb{V}$$

$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![\bigwedge P_\Gamma \wedge \mu X_m.\Phi_1]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\Phi_2]\!]_\mathbb{V}\}$$

$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![I \wedge \bigwedge P_\Gamma \wedge pre \wedge \mu X_m.\Phi_1]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\Phi_2]\!]_\mathbb{V}\}$$

$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![I \wedge \mu X_m.\Phi_1]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![P_\Gamma[v_{old}^i/v^i] \wedge post \wedge \Phi_2]\!]_\mathbb{V}\}$$

$$\subseteq \{\sigma \cdot s \cdot \sigma' \mid \sigma \cdot s \in [\![\Psi_1]\!]_\mathbb{V} \wedge s \cdot \sigma' \in [\![\Psi_2]\!]_\mathbb{V}\} \subseteq [\![\Psi_1^\frown \Psi_2 \vee \bigvee \Delta]\!]$$

□

D Proof of Synchronization Rule

D.1 Additional Lemmas

Lemma 2 (Equivalence of Fixed Point Representations). *For any fixed relation R, fixed recursion variable X, valuation $\mathbb{V}$ and chop formula $\Psi \in CF_{(R,X)}$ with $\Psi = \bigvee_{1 \le j \le n} \varphi_j$, let the following be a γ-sequence $(\gamma^i)_{i \ge 0}$ induced by fixed point operation $\mu X.\Psi$:*

$$(\gamma^i)_{i \ge 0} \ \text{with} \ \gamma^0 = \varnothing \wedge \gamma^{i+1} = [\![\Psi]\!]_{\mathbb{V}[X \mapsto \gamma^i]} \ .$$

Also let the following be a sequence of sets of primitive chop formulas $(C^i)_{i \ge 0}$ induced by chop formula Ψ:

$$C^0 = \varnothing \ \text{and} \ C^{i+1} = \bigcup_{1 \le j \le n} \{\varphi_j[c^1/X^{(1)}] \cdots [c^z/X^{(z)}]) \mid c^1, \dots, c^z \in C^i\}$$

where $X^{(i)}$ refers to the i-th occurrence of X in a primitive chop formula φ_j. Then $\gamma^i = [\![C^i]\!]_{\mathbb{V}}$ for all $i \ge 0$.

Proof. Let us assume relation R, recursion variable X, valuation $\mathbb{V}$ and chop formula $\Psi = \bigvee_{1 \le j \le n} \varphi_j \in CF_{(R,X)}$ are arbitrary, but fixed. We apply natural induction on $i \ge 0$ to prove that $\gamma^i = [\![C^i]\!]_{\mathbb{V}}$. For that purpose, we first establish that $\gamma^0 = \varnothing = [\![C^0]\!]_{\mathbb{V}}$. For the induction hypothesis, let us assume that $\gamma^i = [\![C^i]\!]_{\mathbb{V}}$ for a fixed $i \ge 0$. Then we can infer

$$\gamma^{i+1} = [\![\bigvee_{1 \le j \le n} \varphi_j]\!]_{\mathbb{V}[X \mapsto \gamma^i]} = \bigcup_{1 \le j \le n} [\![\varphi_j]\!]_{\mathbb{V}[X \mapsto \gamma^i]} = \bigcup_{1 \le j \le n} [\![\varphi_j]\!]_{\mathbb{V}[X \mapsto [\![C^i]\!]_{\mathbb{V}}]}$$

$$= \bigcup_{1 \le j \le n} \{[\![\varphi_j[c^1/X^{(1)}] \cdots [c^z/X^{(z)}]]\!]_{\mathbb{V}} \mid c^1, \dots, c^z \in C^i\}$$

$$= [\![\bigcup_{1 \le j \le n} \{\varphi_j[c^1/X^{(1)}] \cdots [c^z/X^{(z)}] \mid c^1, \dots, c^z \in C^i\}]\!]_{\mathbb{V}} = [\![C^{i+1}]\!]_{\mathbb{V}}$$

We have established that $\gamma^i = [\![C^i]\!]_{\mathbb{V}}$ holds for all $i \ge 0$. $\square$

Lemma 3 (Derivability of Primitive Chop Formulas in Grammar). *For any fixed relation R, fixed recursion variable X, chop formula $\Psi \in CF_{(R,X)}$, assuming the sequence of sets of primitive chop formulas $(C^i)_{i \ge 0}$ with*

$$C^0 = \varnothing \ \text{and} \ C^{i+1} = \bigcup_{1 \le j \le n} \{\varphi_j[c^1/X^{(1)}] \cdots [c^z/X^{(z)}]) \mid c^1, \dots, c^z \in C^i\}$$

then also $\bigcup_{i \ge 0} grammarize(C^i) = L(gr(\Psi))$.

Proof. Let us assume relation R, recursion variable X and corresponding chop formula $\Psi = \bigvee_{1 \le j \le n} \varphi_j \in CF_{(R,X)}$ are arbitrary, but fixed. We now have to deduce that $\bigcup_{i \ge 0} grammarize(C^i) = L(gr(\Psi))$. We split the proof of the equality into a forward- and backward-direction.

$\Rightarrow$: First show that $\bigcup_{i \geq 0} grammarize(C^i) \subseteq L(gr(\Psi))$ via induction over i. The induction base

$$grammarize(C^0) = \varnothing \subseteq L(gr(\Psi))$$

trivially holds. For the induction step, we fix i and assume, as the induction hypothesis, that $grammarize(C^i)$ can be derived in $gr(\Psi)$. We will now show that the words in $grammarize(C^{i+1})$ can also be derived in $gr(\Psi)$. Let us assume $w_{i+1} \in grammarize(C^{i+1})$ is arbitrary, but fixed. Then there exists a $c^{i+1} \in C^{i+1}$ with $grammarize(c^{i+1}) = w_{i+1}$. This means that there exists a φ_j for some j and $c^1, \ldots, c^z \in C^i$, such that $c^{i+1} = \varphi_j[c^1/X^{(1)}] \cdots [c^z/X^{(z)}]$. We can now derive w_{i+1} by applying the derivation rule $X \to \gamma$ with $\gamma = grammarize(\varphi_j)$, where each occurrence $X^{(m)}$ inside γ is again derived by applying the derivation of $grammarize(c^m)$. This derivation must already exist, because $c^m \in C^i$, and as such $grammarize(c^m) \in grammarize(C^i)$, which lies in the domain of our induction hypothesis.

$\Leftarrow$: We need to prove that $L(gr(\Psi)) \subseteq \bigcup_{i \geq 0} grammarize(C^i)$. To this end, let $w \in L(gr(\Psi))$ be arbitrary, but fixed, and have a derivation depth k. We prove via induction over derivation depth k, that also $w \in grammarize(C^k)$. Let us first assume that w has depth 1. Then there exists a derivation rule $X \to grammarize(\varphi_j)$ for some j with $1 \leq j \leq n$, such that $grammarize(\varphi_j) = w$. Since φ_j can only contain relation R, this also implies that $\varphi_j \in C^1$, hence $w \in grammarize(C^1)$.

For the induction step, we fix k and assume, as the induction hypothesis, that any word with a derivation depth of k is included in $grammarize(C^k)$. Then, let us assume that word w has depth $k + 1$. Hence, there must exist a derivation rule $X \to grammarize(\varphi_j)$ for some j with $1 \leq j \leq n$, ensuring that any derivation of its internal non-terminals X must have a derivation depth of k, such that word w can be derived. Using the induction hypothesis, we hence know that all derived words $w^1, \ldots, w^z$ of the internal non-terminals X must be included in $grammarize(C^k)$. Since C^{k+1} includes all φ_j, where all occurrences of its recursion variables X have been replaced by elements of C^k, our word w must also be included in $grammarize(C^{k+1})$, i.e. $w \in grammarize(C^{k+1})$. $\square$

Lemma 4 (Fixed Point Trace Representation in Language). *For any fixed relation R, recursion variable X, valuation $\mathbb{V}$, chop formula $\Psi \in CF_{(R,X)}$, let*

$$c_\sigma := \overbrace{R \frown \ldots \frown R}^{l-times}$$

be a primitive chop formula of length l. Then the following two statements must hold at the same time:

1. *There exists a trace $\sigma \in [\![\mu X.\Psi]\!]_\mathbb{V}$ of length $l \geq 1$ with $[\![c_\sigma]\!]_\mathbb{V} = \{\sigma\}$.*
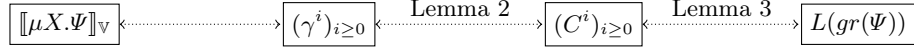2. *$grammarize(c_\sigma) \in L(gr(\Psi))$.*

Fig. 17: Visualization of established proof connections

Proof. Let us assume relation R, recursion variable X, valuation $\mathbb{V}$ and chop formula $\Psi = \bigvee_{1 \leq j \leq n} \varphi_j \in CF_{(R,X)}$ are arbitrary, but fixed. Let c_σ be a primitive chop formula of length l. We now prove the lemma by establishing the forward- and backward-direction, which both follow the outline visualized in Figure 17.

$\Rightarrow$: Let us assume trace $\sigma \in [\![\mu X.\Psi]\!]_{\mathbb{V}}$ of length $l \geq 1$ is arbitrary, but fixed, such that $[\![c_\sigma]\!]_{\mathbb{V}} = \{\sigma\}$. We now consider the following γ-sequence:

$$(\gamma^i)_{i \geq 0} \text{ with } \gamma^0 = \varnothing \wedge \gamma^{i+1} = [\![\Psi]\!]_{\mathbb{V}[X \mapsto \gamma^i]}$$

This sequence must (after possibly infinitely many steps) have reached its least fixed point $[\![\mu X.\Psi]\!]_{\mathbb{V}}$. Since $\sigma \in [\![\mu X.\Psi]\!]_{\mathbb{V}}$ is a finite trace by default, there exists a $k \geq 0$ such that $\sigma \in \gamma^k$, i.e. σ has been generated after k iterations. Using Lemma 2, we know that for the sequence of sets of primitive chop formulas $(C^i)_{i \geq 0}$ with

$$C^0 = \varnothing \text{ and } C^{i+1} = \bigcup_{1 \leq j \leq n} \{\varphi_j[c^1/X^{(1)}] \cdots [c^z/X^{(z)}]) \mid c^1, \ldots, c^z \in C^i\}$$

it holds that $\gamma^i = [\![C^i]\!]_{\mathbb{V}}$ for all $i \geq 0$. Since $\sigma \in \gamma^k$, we thus know that $\sigma \in [\![C^k]\!]_{\mathbb{V}}$. Any primitive chop formula included in C^k can only consist of relation R as its atoms. This is trivial, as C^0 is the empty set, while C^{i+1} replaces all occurrences of recursion variable X with primitive chop formulas of C^i. Since σ is of length l, $c_\sigma \in C^k$ must hold as well.

We can now construct a derivation for $grammarize(c_\sigma)$ in $gr(\Psi)$. Since $c_\sigma \in C^k$, $grammarize(c_\sigma) \in grammarize(C^k)$ must also hold. Using Lemma 3, this implies $grammarize(c_\sigma) \in L(gr(\Psi))$, which needed to be proven.

$\Leftarrow$: Let us assume that $grammarize(c_\sigma) \in L(gr(\Psi))$. We consider the sequence of sets of primitive chop formulas $(C^i)_{i \geq 0}$ with

$$C^0 = \varnothing \text{ and } C^{i+1} = \bigcup_{1 \leq j \leq n} \{\varphi_j[c^1/X^{(1)}] \cdots [c^z/X^{(z)}]) \mid c^1, \ldots, c^z \in C^i\}$$

Applying Lemma 3, since $grammarize(c_\sigma) \in L(gr(\Psi))$, we know that there exists a corresponding set of primitive chop formulas C^k, such that necessarily $grammarize(c_\sigma) \in grammarize(C^k)$. This implies $c_\sigma \in C^k$ for some $k \geq 0$. Since c_σ is of length l, there exists some trace σ of length l with $[\![c_\sigma]\!]_{\mathbb{V}} = \{\sigma\}$. This implies that $\sigma \in [\![C_k]\!]_{\mathbb{V}}$. Let us consider the γ-sequence

$$(\gamma^i)_{i \geq 0} \text{ with } \gamma^0 = \varnothing \wedge \gamma^{i+1} = [\![\Psi]\!]_{\mathbb{V}[X \mapsto \gamma^i]}$$

generated by the fixed point operation $\mu X.\Psi$. Due to Lemma 2, we also know that $\sigma \in [\![C^k]\!]_{\mathbb{V}}$ implies $\sigma \in \gamma^k$. $\sigma \in \gamma^k$ again implies that $\sigma \in [\![\mu X.\Psi]\!]_{\mathbb{V}}$, which needed to be proven. $\square$

Lemma 5 (Application of Trace Synchronization). *For any fixed relation R, recursion variable X, valuation $\mathbb{V}$ and chop formulas $\Psi, \Psi' \in CF_{(R,X)}$, if we assume $L(gr(\Psi')) \subseteq L(gr(\Psi))$, then also $[\![\mu X.\Psi']\!]_{\mathbb{V}} \subseteq [\![\mu X.\Psi]\!]_{\mathbb{V}}$.*

Proof. Let us assume relation R, recursion variable X, valuation $\mathbb{V}$ and chop formulas $\Psi, \Psi' \in CF_{(R,X)}$ are arbitrary, but fixed, such that $L(gr(\Psi')) \subseteq L(gr(\Psi))$. Let us choose a trace $\sigma \in [\![\mu X.\Psi']\!]_{\mathbb{V}}$ of length $l \geq 1$ arbitrary, but fixed. Let us now consider the primitive chop formula c_σ with

$$c_\sigma = \overbrace{R \frown \ldots \frown R}^{l-times}$$

Considering that Ψ' is a chop formula, trace $\sigma \in [\![\mu X.\Psi']\!]_{\mathbb{V}}$ of length l must be a trace that has R applied $l-times$ as a chop-sequence, i.e. $[\![c_\sigma]\!]_{\mathbb{V}} = \{\sigma\}$. Using Lemma 4, we hence know that $grammarize(c_\sigma) \in L(gr(\Psi'))$. Using our premise, we can deduce that $grammarize(c_\sigma) \in L(gr(\Psi))$. Applying Lemma 4 again, we can infer that there also exists a trace $\sigma' \in [\![\mu X.\Psi]\!]_{\mathbb{V}}$ with $[\![c_\sigma]\!]_{\mathbb{V}} = \{\sigma'\}$. Since $\{\sigma\} = [\![c_\sigma]\!]_{\mathbb{V}} = \{\sigma'\}$, we conclude that $\sigma \in [\![\mu X.\Psi]\!]_{\mathbb{V}}$, which was to be proven. $\square$

D.2   Proof of Theorem 6

Proof. We prove that each new rule is *locally sound*.

(SYNC). Let us assume $[\![\bigwedge \Gamma]\!]_{\mathbb{V}} \subseteq [\![\mu X.\Psi' \vee \bigvee \Delta]\!]_{\mathbb{V}}$. Let us further assume that the side condition holds, i.e. $L(gr(\Psi')) \subseteq L(gr(\Psi))$. Using Lemma 5, we infer that

$$[\![\bigwedge \Gamma]\!]_{\mathbb{V}} \subseteq [\![\mu X.\Psi' \vee \bigvee \Delta]\!]_{\mathbb{V}} \subseteq [\![\mu X.\Psi \vee \bigvee \Delta]\!]_{\mathbb{V}}$$

$\square$