Calculational Design of Hyperlogics by Abstract Interpretation

PATRICK COUSOT and JEFFERY WANG, CS, Courant Institute of Mathematical Studies, NYU, USA

We design various logics for proving hyper properties of iterative programs by application of abstract interpretation principles.

In part I, we design a generic, structural, fixpoint abstract interpreter parameterized by an algebraic abstract domain describing finite and infinite computations that can be instantiated for various operational, denotational, or relational program semantics. Considering semantics as program properties, we define a post algebraic transformer for execution properties (e.g. sets of traces) and a Post algebraic transformer for semantic (hyper) properties (e.g. sets of sets of traces), we provide corresponding calculuses as instances of the generic abstract interpreter, and we derive under and over approximation hyperlogics.

In part II, we define exact and approximate semantic abstractions, and show that they preserve the mathematical structure of the algebraic semantics, the collecting semantics post, the hyper collecting semantics Post, and the hyperlogics.

Since proofs by sound and complete hyperlogics require an exact characterization of the program semantics within the proof, we consider in part III abstractions of the (hyper) semantic properties that yield simplified proof rules. These abstractions include the join, the homomorphic, the elimination, the principal ideal, the order ideal, the frontier order ideal, and the chain limit algebraic abstractions, as well as their combinations, that lead to new algebraic generalizations of hyperlogics, including the $\forall\exists^*$, $\forall\forall^*$, and $\exists\forall^*$ hyperlogics.

CCS Concepts: • **Theory of computation → Logic and verification**.

Additional Key Words and Phrases: abstract interpretation, calculational design, completeness, correctness, hyperlogic, hyperproperty, incorrectness, nontermination, semantics, soundness, termination.

ACM Reference Format:
Patrick Cousot and Jeffery Wang. 2024. Calculational Design of Hyperlogics by Abstract Interpretation. 1, 1 (November 2024), 75 pages. https://doi.org/https://doi.org/10.1145/3704852

1 Introduction

Program (hyper) logics provide methods for reasoning about (sets of) program executions as defined by a semantics. For example, hyperproperties were defined by Michael Clarkson and Fred Schneider on execution traces [14] but more recent proposals consider relational logics. We aim at designing program (hyper) logics independently of a specific program semantics, and, more precisely, independently of the formal representation of program executions used by these semantics.

In part I, we recall elements of set and order theories (sect. 2) and then define a structural fixpoint *algebraic program semantics* (sect. 3.4) which is an abstract interpreter parameterized by an *algebraic abstract domain* (sect. 3.3) defined axiomatically. The abstract domain includes terminating and nonterminating executions and can be instantiated to various data and execution models such as the classic relational semantics (sect. 4) or the trace semantics corresponding to the original

Authors' Contact Information: Patrick Cousot, pcousot@cims.nyu.edu; Jeffery Wang, cw3736@nyu.edu, CS, Courant Institute of Mathematical Studies, NYU, New York, NY, USA.

ACM XXXX-XXXX/2024/11-ART
https://doi.org/https://doi.org/10.1145/3704852

definition of hyperproperties [14] (sect. B in the appendix) . Then in sect. 5, we define an *execution collecting semantics* (e.g. sets of traces i.e. trace properties) and introduce a sound and complete calculus post of execution properties. In sect. 6, we define a *semantic collecting semantics* (e.g. sets of sets of traces i.e. hyperproperties) and introduce a structural, fixpoint, sound, and complete calculus Post of semantics properties. In sect. 7, we define *upper and lower semantic logics* (e.g. a logic for trace hyperproperties) and derive over and under *sound and complete proof systems* by calculational design.

In part II, we define the abstraction of the structural algebraic program semantics (sect. 8) and show that it induces an abstraction of the algebraic execution collecting semantics (sect. 9), the algebraic semantic collecting semantics (sect. 10), and the algebraic upper and lower logics (sect. 11). Such abstractions preserve the mathematical structure of the algebraic semantic, collecting semantics, and logics in the abstract. This shows that the algebraic semantics, collecting semantics, and logics can be instantiated to any one in the *hierarchies of semantics* considered e.g. in [4, 18, 41].

Hyperlogics are under or over approximations of semantic properties that is sets of semantics. A program semantics satisfies a hyperproperty if and only if it appears *exactly* in the hyperproperty. It follows that proofs by semantic logics (for hyperproperties) require, for completeness, to describe the program semantics exactly in the proof. By analogy with Hoare logic, this would require the loop invariants to be the strongest, which is an extreme requirement.

This is why, in part III, we consider abstractions of semantic properties, which are less general, but otherwise offer adequate representations of semantic properties and/or allow for much simplified proof rules, closer to the tradition of classic program execution logics, and complete for well identified classes of *abstract semantic properties*. The classic *join abstraction* (sect. 12), *homomorphic abstraction* (sect. 13), and *intersection abstraction* (sect. 14) yield simplified proof rules for hyperlogics. The *principal ideal* (sect. 15), *order ideal* (sect. 16), *frontiers* (sect. 17), *chain limit* (sect. 18), *chain limit order ideal* (sect. 19) abstractions are more specific to hyperproperties. They are compared in sect. 22. These abstraction generalize known hyperlogics for the algebraic semantics and allow us to provide new sound and complete proof rules, including for $\forall\exists$ (sect. 18.2), $\forall\forall$ (sect. 19.2), and $\exists\forall$ (sect. 21) (hyper)properties.. This last case is based on conjunctive abstractions (i.e. conjunctions in logics or reduced products in static analysis) studied in sect. S.1 of the appendix).

We finally briefly refer to the related works (already cited extensively in the text) in sect. 23 and summarize our contributions in the conclusion which also proposes future work (sect. 24). When clickable, the symbol Ⓐ links to proofs and additional developments in the appendix. The paper together with its appendix is available in the auxiliary material.

Part I: Algebraic Semantics, Execution Properties, Semantic (Hyper) Properties, Calculi, and Logics

2 Elements of Set and Order Theories

2.1 Partially Ordered Sets

Definition 2.1 (Properties of posets). Let $\langle \mathbb{L}, \sqsubseteq \rangle$ be a poset with partially defined least upper bound (lub or join) $\sqcup$, greatest lower bound (glb or meet) $\sqcap$, infimum $\perp$, and supremum $\top$, if any. [31].

i. $\langle L, \sqsubseteq, \sqcup \rangle$ is a *join semilattice* when the least upper bound (lub, join) $\sqcup S$ exists for any non-empty finite subset $S \in \wp(L) \setminus \{\varnothing\}$ of L. If it exists, the infimum is $\perp = \sqcup\varnothing$. The dual is a *meet semilattice* with greatest lower bound (glb, meet) $\sqcap$ and supremum $\top = \sqcup L$, if it exists. A *lattice* is both a join and meet semilattice. By *limit* we mean either the join or the meet.

ii. A poset is *increasing chain complete* if and only if every nonempty increasing chain of L has a lub. It is *decreasing chain complete* if and only if every nonempty decreasing chain of L has a glb[1]. It is *chain complete* if both increasing and decreasing chain complete.

iii. A poset is a *complete lattice* if and only if any subset, including the empty set, has a lub (hence a glb and the infimum and supremum do exist).

Observe that (2.1.i) and (2.1.ii) are independent (i.e. none implies the other). We often use them simultaneously. For example, in a *increasing chain-complete join semilattice*, lubs exist for non-empty finite sets and non-empty increasing chains.

2.2 Ordinals

We let $\mathbb{O} = \{0, 1, 2, \ldots, \omega, \omega+1, \omega+2, \ldots, \omega\times 2, \omega\times 2+1, \omega\times 2+2, \ldots, \omega\times 3, \ldots, \omega\times\omega = \omega^2, \ldots, \omega^\omega, \ldots, \omega^{\omega^\omega}, \ldots, \omega^{\omega^{\cdot^{\cdot^{\cdot^{\omega}}}}}\}^{\omega \text{ times}}, \ldots\}$ be the class of ordinals where ω is the first infinite limit ordinal [72]. $\langle\mathbb{O}, \leqslant\rangle$ extends the order on the naturals $\langle\mathbb{N}, \leqslant\rangle$ into the infinite. Ordinals yield typical examples of well-orderings (such that any two elements are comparable and any $<$-strictly decreasing chain is finite). Any well-ordering is order-isomorphic to an ordinal (called its rank e.g. ω for $\mathbb{N}$), [72, th. 13.10 & 13.11]. We use Von Neumann definition of ordinals [72, ch. 2] with $0 = \varnothing$, the successor is $\delta+1 = \delta\cup\{\delta\}$, $<$ is $\in$, $\lambda = \bigcup_{\beta<\lambda}\beta$ for infinite limit ordinals λ (which are not a successor ordinal such as ω, ω^2, etc), and the corresponding transfinite induction [72, Sec. 10], $P(0)$, $\forall\delta\in\mathbb{O}\ .\ P(\delta) \Rightarrow P(\delta+1)$, and for all limit ordinals $\lambda\in\mathbb{O}$, $(\forall\beta<\lambda\ .\ P(\beta)) \Rightarrow P(\lambda)$ implies $\forall\delta\in\mathbb{O}\ .\ P(\delta)$.

2.3 Functions on Partially Ordered Sets

Definition 2.2 (Properties of functions on posets). Let $\langle L, \sqsubseteq\rangle$ be a poset and $f \in L \to L$.

i. f is *increasing* (sometimes referred to as *monotone* or *isotone*) means that $\forall x, y \in L\ .\ (x \sqsubseteq y) \Rightarrow (f(x) \sqsubseteq f(y))$. "Increasing" is order self-dual. *Decreasing* (or *antitone*) is $\forall x, y \in L\ .\ (x \sqsubseteq y) \Rightarrow (f(y) \sqsubseteq f(x))$;

For example, a sequence $\langle X^\delta \in L, \delta < \lambda\rangle$ for ordinals $\delta, \lambda \in \mathbb{O}$ is an increasing chain means that $\forall\delta\leqslant\delta'<\lambda\ .\ X^\delta \sqsubseteq X^{\delta'}$. A decreasing chain has $\forall\delta\leqslant\delta'<\lambda\ .\ X^{\delta'} \sqsubseteq X^\delta$;

ii. Function f is *existing finite join-preserving* (also written *existing finite* $\sqcup$*-preserving*) if and only if for any non-empty finite set $S \in \wp_f(L) \setminus \{\varnothing\}$ such that $\sqcup S$ exists in L then $\sqcup f(S)$ exists in L and $f(\sqcup S) = \sqcup f(S)$ with $f(S) = \{f(x) \mid x \in S\}$, and dually for meets. f is *existing finite limit-preserving* if and only if it is both existing finite join and meet preserving. "Existing" can be omitted in a lattice;

iii. f is *upper-continuous* (or existing increasing chain join-preserving) if and only if for any non-empty increasing chain $S \in \wp_f(L)$ such that $\sqcup S$ exists in L, then $\sqcup f(S)$ exists in L such that $f(\sqcup S) = \sqcup f(S)$. The dual is *lower-continuous* for existing decreasing chain meet-preserving, and *continuous* means both lower and upper continuous. By Scott-Kleene theorem, continuity ensures that functions reach fixpoints iteratively at ω [20, th. 15.36]. This condition for *convergence at* ω is sufficient but not necessary e.g. [20, th. 15.21];

iv. f is *existing join-preserving* (also written *existing* $\sqcup$*-preserving*) if and only if for any non-empty set $S \in \wp(L) \setminus \{\varnothing\}$ such that $\sqcup S$ exists in L, then $\sqcup f(S)$ exists in L such that $f(\sqcup S) = \sqcup f(S)$, and dually for meets. f is *existing limit-preserving* if and only if it is both existing join and meet preserving. "Existing" can be omitted in a complete lattice;

v. The definitions 2.2.ii to 2.2.iv are extended to $f \in (L\times L) \to L$ by f has *left limit property* if and only if $\forall y \in L\ .\ \boldsymbol{\lambda} x \bullet f(x, y)$ has that limit property and f has *right limit property* whenever

[1]We do not respectively use the classic *CPO* and *dual CPO* for which chains are usually restricted to be of length ω.

$\forall x \in L \,.\, \boldsymbol{\lambda} y \bullet f(x, y)$ has that limit property. f has that the limit property *in both parameters* if and only if f has both of the left and right limit properties;

vi. When extending the definitions 2.2.ii to 2.2.v to empty sets or chains, the function f is then said to be *lower strict*, dually *upper strict*, and *strict* for both cases.

Observe that 2.2.i $\Leftarrow$ 2.2.ii $\Leftarrow$ 2.2.iii $\Leftarrow$ 2.2.iv.

2.4 Fixpoints

Let $f \in \mathbb{L} \xrightarrow{\nearrow} \mathbb{L}$ be an increasing function on a poset $\langle \mathbb{L}, \sqsubseteq \rangle$. There are essentially two classic characterizations of the least fixpoint $\mathsf{lfp}^{\sqsubseteq} f$ of f (we also use their order duals).

Proposition 2.3 (Fixpoint). *$\mathsf{lfp}^{\sqsubseteq} f = \bigsqcap\{x \mid f(x) \sqsubseteq x\}$ by* [81] *on complete lattices which also holds on increasing chain complete posets* [38].

Proposition 2.4 (Iteration to fixpoint). *If $\langle \mathbb{L}, \sqsubseteq, \bot, \sqcup \rangle$ is a poset with infimum $\bot$ and partially defined join $\sqcup$ then the iterates $\langle X^{\delta}, \delta \in \mathbb{O} \rangle$ of f are partially defined as $X^{\delta+1} \triangleq f(X^{\delta})$, and $X^{\lambda} \triangleq \bigsqcup_{\beta<\lambda} X^{\beta}$ for limit ordinals λ (hence $X^0 = \bigsqcup \varnothing = \bot$ for limit ordinal 0). They are well defined when f is increasing (hence when it is finite join preserving, upper-continuous or existing join-preserving) and $\langle \mathbb{L}, \sqsubseteq, \bot, \sqcup \rangle$ is an increasing chain complete poset (hence when it is a complete lattice) in which case they form an increasing chain (i.e. $\forall \beta < \delta \in \mathbb{O} \,.\, X^{\beta} \sqsubseteq X^{\delta}$) ultimately stationary at the limit $\exists \epsilon \,.\, \forall \beta \geqslant \epsilon \,.\, X^{\beta} = \mathsf{lfp}^{\sqsubseteq} f$* [23]. *In case f is upper-continuous (hence when preserving existing joins), the iterates are stationary at $\epsilon = \omega$ so that the iterates may be restricted to $\mathbb{N}$ and* $\mathsf{lfp}^{\sqsubseteq} f = \bigsqcup_{n \in \mathbb{N}} X^n$ [81, page 305].

2.5 Galois Connections, Retractions, and Isomorphisms

Galois connections are used throughout the paper either to formalize correspondances between transformers or to formalize exact or approximate abstractions. Formally, a Galois connection $\langle C, \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \preceq \rangle$ is a pair $\langle \alpha, \gamma \rangle$ of functions between posets $\langle C, \sqsubseteq \rangle$ and $\langle A, \preceq \rangle$ satisfying $\forall x \in C \,.\, \forall y \in A \,.\, \alpha(x) \preceq y \Leftrightarrow x \sqsubseteq \gamma(y)$. We use a double headed arrow $\twoheadrightarrow$ to indicate surjection in Galois retractions and $\leftrightarrows$ for bijections. We use classic properties of Galois connections which proofs are found in [34].

2.6 Closures

We let $\mathbb{1}$ be the identity function. An upper closure operator ρ on $\mathbb{L}$ is increasing, extensive and idempotent so $\langle \mathbb{L}, \sqsubseteq \rangle \underset{\rho}{\overset{\mathbb{1}}{\leftrightarrows}} \langle \rho(\mathbb{L}), \sqsubseteq \rangle$ where $\rho(X) \triangleq \{\rho(x) \mid x \in X\}$ is the post image (dually, a lower closure operator is reductive). It follows that ρ preserves existing arbitrary joins so if $\langle \mathbb{L}, \sqsubseteq, \bot, \sqcup \rangle$ is an increasing chain complete poset (respectively complete lattice $\langle \mathbb{L}, \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle$) then $\langle \rho(\mathbb{L}), \sqsubseteq \rangle$ has the same structure with infimum $\rho(\bot)$, join $\boldsymbol{\lambda} X \bullet \rho(\sqcup X)$, meet $\sqcap$ and top $\top$, if any. In case of a complete lattice this is Morgan Ward's [83, th. 4.1]. If ρ_1 and ρ_2 are upper closures on $\mathbb{L}$ then $\rho_1 \circ \rho_2$ and $\rho_2 \circ \rho_1$ are upper closure operators on $\mathbb{L}$ if and only if ρ_1 and ρ_2 are commuting (i.e. $\rho_1 \circ \rho_2 = \rho_2 \circ \rho_1$) in which case $\rho_1 \circ \rho_2(\mathbb{L}) = \rho_2 \circ \rho_1(\mathbb{L}) = \rho_1(\mathbb{L}) \cap \rho_2(\mathbb{L})$ [76, p. 525].

3 Algebraic Semantics

We introduce the syntax and algebraic semantics of a simple iterative language based on an abstract domain that generalizes [20, Ch. 21] to include infinite program behaviors. The algebraic semantics is reminiscent of [12, 17, 37, 49, 50, 56, 57, 59, 60, 74] and others. Such algebraic semantics are a basis for studying a hierarchy of program properties independently of the data manipulated by programs.

3.1 Syntax

We consider an imperative language $\mathbb{S}$ with assignments, sequential composition, conditionals, and conditional iteration with breaks. The syntax is $\texttt{S} \in \mathbb{S} ::= \texttt{x = A} \mid \texttt{x = [}a,b\texttt{]} \mid \texttt{skip} \mid \texttt{S;S} \mid \texttt{if (B) S else S} \mid \texttt{while (B) S} \mid \texttt{break}$. $\texttt{A}$ is an arithmetic expression. The nondeterministic assignment $\texttt{x = [}a,\ b\texttt{]}$ with $a \in \mathbb{Z} \cup \{-\infty\}$ and $b \in \mathbb{Z} \cup \{\infty\}$, $-\infty - 1 = -\infty$, $\infty + 1 = \infty$ (or any, possibly unbounded, order isomorphic set). The Boolean expressions $\texttt{B}$ include the negation $\neg\texttt{B}$. A $\texttt{break}$ exits the closest enclosing loop (which existence is to be checked syntactically).

3.2 Structural Definitions

Let $\lhd$ be the "immediate strict syntactic component" well-founded partial order on statements $\mathbb{S}$ such that $\texttt{S}_1 \lhd \texttt{S}_1\texttt{;S}_2$, $\texttt{S}_2 \lhd \texttt{S}_1\texttt{;S}_2$, $\texttt{S}_1 \lhd \texttt{if (B) S}_1 \texttt{ else S}_2$, $\texttt{S}_2 \lhd \texttt{if (B) S}_1 \texttt{ else S}_2$, $\texttt{S} \lhd \texttt{while (B) S}$, and is otherwise false.

Given a nonempty set $\mathcal{V}$, the function $f \in \mathbb{S} \to \mathcal{V}$ has a structural definition if and only if $f(\texttt{S}) \in \mathcal{V}$ for basic commands (defined as minimal elements of $\lhd$) and, otherwise, is of the form $f(\texttt{S}) = F_{\texttt{S}}(\{\langle \texttt{S}', f(\texttt{S}')\rangle \mid \texttt{S}' \lhd \texttt{S}\})$ where $F_{\texttt{S}} \in \{\langle \texttt{S}',\ v'\rangle \mid \texttt{S}' \lhd \texttt{S} \wedge v' \in \mathcal{V}\} \to \mathcal{V}$ is a total function. Denotational semantics, Hoare logic, predicate transformers, and the abstract semantics of sect. 3.4 all have structural definitions (called "compositional" in denotational semantics).

3.3 Algebraic Computational Domain

We consider computational domains $\mathbb{D}^\sharp_+$ and $\mathbb{D}^\sharp_\infty$ to be abstract domains respectively abstracting the finite and infinite computations of statements and partially ordered by the respective computational orderings $\sqsubseteq^\sharp_+$ and $\sqsubseteq^\sharp_\infty$, as follows ($\fatsemi^\sharp$ is polymorphic).

$$\mathbb{D}^\sharp_+ \triangleq \langle \mathbb{L}^\sharp_+,\ \sqsubseteq^\sharp_+,\ \bot^\sharp_+,\ \sqcup^\sharp_+,\ \mathsf{init}^\sharp,\ \mathsf{assign}^\sharp[\![\mathsf{x},\mathsf{A}]\!],\ \mathsf{rassign}^\sharp[\![\mathsf{x},a,b]\!],\ \mathsf{test}^\sharp[\![\mathsf{B}]\!],\ \mathsf{break}^\sharp,\ \mathsf{skip}^\sharp,\ \fatsemi^\sharp\rangle \tag{1}$$

$$\mathbb{D}^\sharp_\infty \triangleq \langle \mathbb{L}^\sharp_\infty,\ \sqsubseteq^\sharp_\infty,\ \top^\sharp_\infty,\ \sqcap^\sharp_\infty,\ \fatsemi^\sharp\rangle \tag{2}$$

Example 3.1. Bi-inductive definitions [24] are used in [18] to define a trace semantics on states Σ which can be isomorphically decomposed into the domain of finite traces $\langle \mathbb{L}^\sharp_+, \sqsubseteq^\sharp_+, \bot^\sharp_+, \sqcup^\sharp_+\rangle = \langle \wp(\Sigma^*), \subseteq, \emptyset, \cup\rangle$ (where $\cup$ is the lub of increasing chains starting form $\emptyset$ for least fixpoints) and the domain of infinite traces $\langle \mathbb{L}^\sharp_\infty, \sqsubseteq^\sharp_\infty, \top^\sharp_\infty, \sqcap^\sharp_\infty\rangle = \langle \wp(\Sigma^\omega), \subseteq, \Sigma^\omega, \cap\rangle$ (where $\cap$ is the glb of decreasing chains starting form Σ^ω for greatest fixpoints), which abstractions yield a hierarchy of classic semantics, including Hoare logic.

Our objective in part I is to study hyperlogics abstracting away from a particular semantics thus allowing for multiple instantiations (such as traces in sect. B) and, in part II, for multiple abstractions (which include Hoare logic).

A single domain $\mathbb{D}^\sharp \triangleq \mathbb{D}^\sharp_+ \cup \mathbb{D}^\sharp_\infty$ is used in denotational semantics [78, 80] but this is not always possible e.g. when $\mathbb{D}^\sharp_+ \cap \mathbb{D}^\sharp_\infty \neq \emptyset$. Moreover the separation into two different domains for finite and infinite executions allows e.g. for the use of input-output relations for finite behaviors and traces for infinite behaviors. (see also the discussion in remark B.5 in the appendix.) ■

Definition 3.2 (Abstract domain well-definedness). We say that $\mathbb{D}^\sharp \triangleq \langle \mathbb{D}^\sharp_+,\ \mathbb{D}^\sharp_\infty\rangle$ is a well-defined chain-complete lattice (respectively complete lattice) with increasing (respectively finite limit-preserving, continuous, and existing limit-preserving) composition, if and only if

A. The finitary calculational domain $\langle \mathbb{L}^\sharp_+,\ \sqsubseteq^\sharp_+,\ \bot^\sharp_+,\ \sqcup^\sharp_+\rangle$ is an increasing chain-complete join semilattice with infimum, (respectively $\langle \mathbb{L}^\sharp_+,\ \sqsubseteq^\sharp_+,\ \bot^\sharp_+,\ \top^\sharp_+,\ \sqcup^\sharp_+,\ \sqcap^\sharp_+\rangle$ is a complete lattice);

B. $\mathsf{init}^\sharp$, $\mathsf{break}^\sharp$, $\mathsf{skip}^\sharp \in \mathbb{L}^\sharp_+$, $\mathsf{assign}^\sharp[\![\mathsf{x},\mathsf{A}]\!]$, $\mathsf{rassign}^\sharp[\![\mathsf{x},a,b]\!]$, $\mathsf{test}^\sharp[\![\mathsf{B}]\!] \in \mathbb{L}^\sharp_+$ are well-defined in $\mathbb{L}^\sharp_+$;

C. The infinitary calculational domain $\langle \mathbb{L}^\sharp_\infty,\ \sqsubseteq^\sharp_\infty,\ \top^\sharp_\infty,\ \sqcup^\sharp_\infty,\ \sqcap^\sharp_\infty\rangle$ is a decreasing chain-complete join lattice with supremum (respectively $\langle \mathbb{L}^\sharp_\infty,\ \sqsubseteq^\sharp_\infty,\ \bot^\sharp_\infty,\ \top^\sharp_\infty,\ \sqcup^\sharp_\infty,\ \sqcap^\sharp_\infty\rangle$ is a complete lattice);

D. The sequential composition $\mathbin{⨾}^\sharp \in (\mathbb{L}^\sharp_+ \times \mathbb{L}^\sharp_+ \to \mathbb{L}^\sharp_+) \cup (((\mathbb{L}^\sharp_+ \times \mathbb{L}^\sharp_\infty) \cup (\mathbb{L}^\sharp_\infty \times \mathbb{L}^\sharp_+) \cup (\mathbb{L}^\sharp_\infty \times \mathbb{L}^\sharp_\infty)) \to \mathbb{L}^\sharp_\infty)$ is associative and satisfies the following conditions (where $\langle \mathbb{L}^\sharp_x, \sqsubseteq^\sharp_x, \bot^\sharp_x, \top^\sharp_x, \sqcup^\sharp_x, \sqcap^\sharp_x \rangle$, $x \in \{+, \infty\}$ designates $\langle \mathbb{L}^\sharp_+, \sqsubseteq^\sharp_+, \bot^\sharp_+, \top^\sharp_+, \sqcup^\sharp_+, \sqcap^\sharp_+ \rangle$ when $x = +$ and $\langle \mathbb{L}^\sharp_\infty, \sqsubseteq^\sharp_\infty, \bot^\sharp_\infty, \top^\sharp_\infty, \sqcup^\sharp_\infty, \sqcap^\sharp_\infty \rangle$ when $x = \infty$).

   a. $\forall S \in \mathbb{L}^\sharp_+ \,.\, S \mathbin{⨾}^\sharp \mathsf{init}^\sharp = \mathsf{init}^\sharp \mathbin{⨾}^\sharp S = S$;
   b. $\forall S \in \mathbb{L}^\sharp_+ \,.\, S \mathbin{⨾}^\sharp \bot^\sharp_+ = \bot^\sharp_+$ and $\forall S \in \mathbb{L}^\sharp_x \,.\, \bot^\sharp_+ \mathbin{⨾}^\sharp S = \bot^\sharp_+$ (same for $\mathbb{L}^\sharp_\infty$ when $\bot^\sharp_\infty$ exists);
   c. $\forall S \in \mathbb{L}^\sharp_\infty \,.\, \forall S' \in \mathbb{L}^\sharp_x \,.\, S \mathbin{⨾}^\sharp S' = S$;
   d. In its left, right, or both parameters, the sequential composition $\mathbin{⨾}^\sharp$ is either
      i. increasing for $\sqsubseteq^\sharp_+$ and/or $\sqsubseteq^\sharp_\infty$;
      ii. finite join preserving for $\sqcup^\sharp_+$ and/or $\sqcup^\sharp_\infty$;
      iii. in addition to 3.2.D.d.ii, is lower continuous for $\sqcap^\sharp_+$ and/or $\sqcap^\sharp_\infty$ and/or upper continuous for $\sqcup^\sharp_+$ and/or $\sqcup^\sharp_\infty$;
      iv. existing arbitrary $\sqcup^\sharp_+$-preserving and/or existing arbitrary $\sqcap^\sharp_\infty$-preserving.

REMARK 3.3. In case $\mathbb{L}^\sharp_+ \cap \mathbb{L}^\sharp_\infty = \varnothing$, we can define $\mathbb{L}^\sharp \triangleq \mathbb{L}^\sharp_+ \cup \mathbb{L}^\sharp_\infty$ with $X^+ \triangleq X \cap \mathbb{L}^\sharp_+$, $X^\infty \triangleq X \cap \mathbb{L}^\sharp_\infty$, and $X \sqsubseteq^\sharp Y \triangleq X^+ \sqsubseteq^\sharp_+ Y^+ \wedge X^\infty \sqsubseteq^\sharp_\infty Y^\infty$ which corresponds to the bi-inductive definitions [24] mentioned in example 3.1. ■

REMARK 3.4. Hypotheses 3.2.B, 3.2.D.d.i and 3.2.D.d.ii determine the precision of the semantic of basic commands, composition, choices, conditionals, and iteration in the algebraic semantics. These hypotheses as well as 3.2.D.d.iii and 3.2.D.d.iv determine whether fixpoint iterations should be infinite or transfinite (see proposition 2.4). ■

3.4 Definition of the Algebraic Semantics

The algebraic semantics of statements $\mathtt{S} \in \mathbb{S}$ is an abstract property of executions. The basic commands $\mathtt{S}$ are assignment, random assignment, `break` out of the immediately enclosing loop, and `skip`, with the following $[\![\mathtt{S}]\!]^\sharp_e$ and break $[\![\mathtt{S}]\!]^\sharp_b$ finite/ending/terminating semantics in $\mathbb{L}^\sharp_+$ as well as infinite/nonterminating $[\![\mathtt{S}]\!]^\sharp_\bot$ abstract semantics in $\mathbb{L}^\sharp_\infty$.

3.4.1 Basic Statements.

$$\begin{array}{llllllll}
[\![\mathtt{x = A}]\!]^\sharp_e & \triangleq \mathsf{assign}^\sharp[\![\mathtt{x},\mathtt{A}]\!] & [\![\mathtt{x = A}]\!]^\sharp_b & \triangleq \bot^\sharp_+ & [\![\mathtt{x = A}]\!]^\sharp_\bot & \triangleq \bot^\sharp_\infty \\
[\![\mathtt{x = [}a\mathtt{,}\ b\mathtt{]}]\!]^\sharp_e & \triangleq \mathsf{rassign}^\sharp[\![\mathtt{x}, a, b]\!] & [\![\mathtt{x = [}a\mathtt{,}\ b\mathtt{]}]\!]^\sharp_b & \triangleq \bot^\sharp_+ & [\![\mathtt{x = [}a\mathtt{,}\ b\mathtt{]}]\!]^\sharp_\bot & \triangleq \bot^\sharp_\infty \\
[\![\mathtt{break}]\!]^\sharp_e & \triangleq \bot^\sharp_+ & [\![\mathtt{break}]\!]^\sharp_b & \triangleq \mathsf{break}^\sharp & [\![\mathtt{break}]\!]^\sharp_\bot & \triangleq \bot^\sharp_\infty & (3) \\
[\![\mathtt{skip}]\!]^\sharp_e & \triangleq \mathsf{skip}^\sharp & [\![\mathtt{skip}]\!]^\sharp_b & \triangleq \bot^\sharp_+ & [\![\mathtt{skip}]\!]^\sharp_\bot & \triangleq \bot^\sharp_\infty \\
[\![\mathtt{B}]\!]^\sharp_e & \triangleq \mathsf{test}^\sharp[\![\mathtt{B}]\!] & [\![\mathtt{B}]\!]^\sharp_b & \triangleq \bot^\sharp_+ & [\![\mathtt{B}]\!]^\sharp_\bot & \triangleq \bot^\sharp_\infty
\end{array}$$

For the assignment `x = A`, the abstract semantics $\mathsf{assign}^\sharp[\![\mathtt{x},\mathtt{A}]\!]$ is specified by the abstract domain, and so, is well-defined by 3.2.B. $[\![\mathtt{x = A}]\!]^\sharp_b = \bot^\sharp_+$ because the assignment cannot break. $[\![\mathtt{x = A}]\!]^\sharp_\bot = \bot^\sharp_\infty$ since the assignment always terminates. The algebraic semantics of the other primitives is similar, except for the `break` statement. $[\![\mathtt{break}]\!]^\sharp_e = \bot^\sharp_+$ since the `break` cannot continue in sequence. The semantics $[\![\mathtt{break}]\!]^\sharp_b$ of the `break` is given by the abstract domain primitive $\mathsf{break}^\sharp$ which is finite and well-defined. $[\![\mathtt{break}]\!]^\sharp_\bot = \bot^\sharp_\infty$ since a `break` always terminates.

3.4.2 Structural Statements. For the sequential composition and the conditional where $[\![\mathtt{B;S}]\!]^\sharp_x \triangleq \mathsf{test}^\sharp[\![\mathtt{B}]\!] \mathbin{⨾}^\sharp [\![\mathtt{S}]\!]^\sharp_x$, $x \in \{e, b, \bot\}$, we define

$$\begin{array}{llllll}
[\![\mathtt{S}_1\mathtt{;S}_2]\!]^\sharp_e & \triangleq & [\![\mathtt{S}_1]\!]^\sharp_e \mathbin{⨾}^\sharp [\![\mathtt{S}_2]\!]^\sharp_e & [\![\mathtt{if\ (B)\ S}_1\mathtt{\ else\ S}_2]\!]^\sharp_e & \triangleq & [\![\mathtt{B;S}_1]\!]^\sharp_e \sqcup^\sharp_+ [\![\neg\mathtt{B;S}_2]\!]^\sharp_e \\
[\![\mathtt{S}_1\mathtt{;S}_2]\!]^\sharp_b & \triangleq & [\![\mathtt{S}_1]\!]^\sharp_b \sqcup^\sharp_+ ([\![\mathtt{S}_1]\!]^\sharp_e \mathbin{⨾}^\sharp [\![\mathtt{S}_2]\!]^\sharp_b) & [\![\mathtt{if\ (B)\ S}_1\mathtt{\ else\ S}_2]\!]^\sharp_b & \triangleq & [\![\mathtt{B;S}_1]\!]^\sharp_b \sqcup^\sharp_+ [\![\neg\mathtt{B;S}_2]\!]^\sharp_b \quad (4) \\
[\![\mathtt{S}_1\mathtt{;S}_2]\!]^\sharp_\bot & \triangleq & [\![\mathtt{S}_1]\!]^\sharp_\bot \sqcup^\sharp_\infty ([\![\mathtt{S}_1]\!]^\sharp_e \mathbin{⨾}^\sharp [\![\mathtt{S}_2]\!]^\sharp_\bot) & [\![\mathtt{if\ (B)\ S}_1\mathtt{\ else\ S}_2]\!]^\sharp_\bot & \triangleq & [\![\mathtt{B;S}_1]\!]^\sharp_\bot \sqcup^\sharp_\infty [\![\neg\mathtt{B;S}_2]\!]^\sharp_\bot
\end{array}$$

The semantics of the composition and conditional are well-defined by 3.2.D for $\fatsemi^\sharp$ and 3.2.A and 3.2.C which ensure the existence of the finite and infinite joins.

$\mathtt{S}_1\,;\mathtt{S}_2$ terminates if $\mathtt{S}_1$ terminates and is followed by $\mathtt{S}_2$ that terminates. $\mathtt{S}_1\,;\mathtt{S}_2$ breaks (resp. nonterminates) if either $\mathtt{S}_1$ breaks (resp. nonterminates) or $\mathtt{S}_1$ terminates and is followed by $\mathtt{S}_2$ that breaks (resp. nonterminates).

For a given execution of the conditional `if (B) S`$_1$ `else S`$_2$ only one branch is taken, so the semantics of the other one will be empty by definition (3) of $[\![\mathtt{B}]\!]^\sharp_e$ that should return $\bot^\sharp_+$[2] and 3.2.D.b.

Example 3.5. Assume that $\mathtt{S}_1$ never terminates in that $[\![\mathtt{S}_1]\!]^\sharp_\bot = \top^\sharp_\infty$ (sometimes named "chaos" modelling all possible nonterminating behaviors). Then, by (4), $[\![\mathtt{S}_1\,;\mathtt{S}_2]\!]^\sharp_\bot \triangleq [\![\mathtt{S}_1]\!]^\sharp_\bot \sqcup^\sharp_\infty ([\![\mathtt{S}_1]\!]^\sharp_e \fatsemi^\sharp [\![\mathtt{S}_2]\!]^\sharp_\bot)$ $= \top^\sharp_\infty \sqcup^\sharp_\infty ([\![\mathtt{S}_1]\!]^\sharp_e \fatsemi^\sharp [\![\mathtt{S}_2]\!]^\sharp_\bot) = \top^\sharp_\infty$ meaning that $\mathtt{S}_1\,;\mathtt{S}_2$ never terminates either in chaos.

For the conditional, assume $\mathtt{B}$ is always true and $\mathtt{S}_1$ never terminates in that $[\![\mathtt{S}_1]\!]^\sharp_\bot = \top^\sharp_\infty$. Then the false branch is never taken so that $[\![\neg\mathtt{B}\,;\mathtt{S}_2]\!]^\sharp_\bot = \bot^\sharp_\infty$. It follows, by (4), that $[\![\mathtt{if\ (B)\ S_1\ else\ S_2}]\!]^\sharp_\bot$ $\triangleq [\![\mathtt{B}\,;\mathtt{S}_1]\!]^\sharp_\bot \sqcup^\sharp_\infty [\![\neg\mathtt{B}\,;\mathtt{S}_2]\!]^\sharp_\bot = \top^\sharp_\infty \sqcup^\sharp_\infty \bot^\sharp_\infty = \top^\sharp_\infty$ so that the conditional `if (B) S`$_1$ `else S`$_2$ never terminates. ■

3.4.3 Iteration. For iteration `while (B) S`, we define the transformers

$$\begin{array}{llll} \text{backward} & \overleftarrow{F}^\sharp_e & \triangleq & \boldsymbol{\lambda} X \in \mathbb{L}^\sharp_+ \bullet \mathsf{init}^\sharp \sqcup^\sharp_+ ([\![\mathtt{B}\,;\mathtt{S}]\!]^\sharp_e \fatsemi^\sharp X) \quad (5)\\ \text{forward} & \vec{F}^\sharp_e & \triangleq & \boldsymbol{\lambda} X \in \mathbb{L}^\sharp_+ \bullet \mathsf{init}^\sharp \sqcup^\sharp_+ (X \fatsemi^\sharp [\![\mathtt{B}\,;\mathtt{S}]\!]^\sharp_e) \quad (6)\\ \text{infinite} & F^\sharp_\bot & \triangleq & \boldsymbol{\lambda} X \in \mathbb{L}^\sharp_\infty \bullet [\![\mathtt{B}\,;\mathtt{S}]\!]^\sharp_e \fatsemi^\sharp X \quad (7) \end{array}$$

Lemma 3.6 (Finite fixpoints well-definedness). Ⓐ *If $\mathbb{D}^\sharp_+$ is a well-defined increasing chain complete join semilattice and $\fatsemi^\sharp$ left satisfies any one of the 3.2.D.d.i, 3.2.D.d.ii, 3.2.D.d.iii, or 3.2.D.d.iv properties for $\mathbb{D}^\sharp_+$ then $\overleftarrow{F}^\sharp_e$ satisfy the same property and its least fixpoint deso exist (and similarly for $\vec{F}^\sharp_e$ when $\fatsemi^\sharp$ right satisfies any one of the properties listed in 3.2.D.d).*

Let us show that $\mathsf{lfp}^{\sqsubseteq^\sharp_+} \overleftarrow{F}^\sharp_e = \mathsf{lfp}^{\sqsubseteq^\sharp_+} \vec{F}^\sharp_e$ inductively defines the set of finite executions reaching the entry of the iteration `while(B) S` after zero or more terminating body iterations. To see that, we define

the powers $\langle X^\delta,\ \delta \in \mathbb{O}\rangle$ of $X \in \mathbb{L}^\sharp_+$ are $X^0 \triangleq \mathsf{init}^\sharp$, $X^{\delta+1} \triangleq X \fatsemi^\sharp X^\delta$ for successor ordinals, (8)
and $X^\lambda \triangleq \bigsqcup^\sharp_{+\beta<\lambda} X^\beta$ for limit ordinals.

We now characterize the executions of iterations in terms of the fixpoints of the execution transformers 5—6. We show that $\mathsf{lfp}^{\sqsubseteq^\sharp_+} \overleftarrow{F}^\sharp_e = \mathsf{lfp}^{\sqsubseteq^\sharp_+} \vec{F}^\sharp_e$ inductively characterize 0 or more finite iterations of the loop body for which the loop condition holds and the loop body terminates.

Lemma 3.7 (Commutativity). Ⓐ *If $\mathbb{D}^\sharp_+$ is a well-defined complete lattice (resp. increasing chain-complete poset) with right existing $\sqcup^\sharp_+$-preserving (resp. right upper continuous) composition $\fatsemi^\sharp$ and $X \in \mathbb{L}^\sharp_+$ then $\forall \delta \in \mathbb{O}\ .\ X \fatsemi^\sharp X^\delta = X^\delta \fatsemi^\sharp X$ (resp. if $\langle X^\delta,\ \delta \in \mathbb{O}\rangle$ is an increasing chain).*

Lemma 3.8 (Finite body iterations). Ⓐ *If $\mathbb{D}^\sharp_+$ is a well-defined increasing chain-complete join semilattice with right upper continuous composition $\fatsemi^\sharp$ then $\mathsf{lfp}^{\sqsubseteq^\sharp_+} \overleftarrow{F}^\sharp_e = \bigsqcup^\sharp_{+\,\delta\in\mathbb{O}} ([\![\mathtt{B}\,;\mathtt{S}]\!]^\sharp_e)^\delta$.*

Lemma 3.9 (Forward versus backward). Ⓐ *If $\mathbb{D}^\sharp$ is a well-defined increasing chain-complete join semilattice with right upper continuous sequential composition $\fatsemi^\sharp$ then $\mathsf{lfp}^{\sqsubseteq^\sharp_+} \overleftarrow{F}^\sharp_e = \mathsf{lfp}^{\sqsubseteq^\sharp_+} \vec{F}^\sharp_e$.*

Example 3.10. Assume that the test $\mathtt{B}$ of the iteration `while (B) S` is always false, that is $\mathsf{test}^\sharp[\![\mathtt{B}]\!] = \bot^\sharp_\infty$. Then, by (5), (6), (3.2.D.b), and def. lub, $\overleftarrow{F}^\sharp_e = \vec{F}^\sharp_e = \boldsymbol{\lambda} X \in \mathbb{L}^\sharp_+ \bullet \mathsf{init}^\sharp$. It follows that $\mathsf{lfp}^{\sqsubseteq^\sharp_+} \overleftarrow{F}^\sharp_e =$

[2]unless the semantics of Boolean expressions is to be very exotic.

$\mathsf{lfp}^{\sqsubseteq^\sharp_+}\, \vec{F}^\sharp_e = \mathsf{init}^\sharp$ meaning that the loop is never entered. The semantics of the loop after 0 or more iterations is therefore that after 0 iterations. ∎

Lemma 3.11 (Infinite fixpoint well-definedness). Ⓐ *If $\mathbb{D}^\sharp_\infty$ is a well-defined decreasing chain complete poset and $\fatsemi^\sharp$ right satisfies any one of the 3.2.D.d.i, 3.2.D.d.ii, 3.2.D.d.iii, or 3.2.D.d.iv properties for $\mathbb{D}^\sharp_\infty$ then $F^\sharp_\perp$ satisfies the same property and $\mathsf{gfp}^{\sqsubseteq^\sharp_\infty}\, F^\sharp_\perp$ does exist.*

We now show that $\mathsf{gfp}^{\sqsubseteq^\sharp_\infty}\, F^\sharp_\perp$ coinductively characterizes the infinite executions of the iteration `while (B) S` after infinitely many terminating iterations of the body `S` with condition `B` always true.

Lemma 3.12 (Infinite body iterations). Ⓐ *If $\mathbb{D}^\sharp$ is a well-defined decreasing chain-complete poset and $\fatsemi^\sharp$ is right increasing for $\sqsubseteq^\sharp_\infty$ in 3.2.D.d.i then* $\mathsf{gfp}^{\sqsubseteq^\sharp_\infty}\, F^\sharp_\perp = \bigsqcap^\sharp_{\infty\,\delta\in\mathbb{O}}\big((\llbracket\mathtt{B;S}\rrbracket^\sharp_e)^\delta \fatsemi^\sharp \top^\sharp_\infty\big)$.

The abstract semantics of iteration is defined as

$$\llbracket\mathtt{while\ (B)\ S}\rrbracket^\sharp_e \triangleq (\mathsf{lfp}^{\sqsubseteq^\sharp_+}\, \vec{F}^\sharp_e) \fatsemi^\sharp (\llbracket\neg\mathtt{B}\rrbracket^\sharp_e \sqcup^\sharp_e \llbracket\mathtt{B;S}\rrbracket^\sharp_b) \qquad \llbracket\mathtt{while\ (B)\ S}\rrbracket^\sharp_b \triangleq \bot^\sharp_+ \tag{9}$$

$$\llbracket\mathtt{while\ (B)\ S}\rrbracket^\sharp_{bi} \triangleq (\mathsf{lfp}^{\sqsubseteq^\sharp_+}\, \vec{F}^\sharp_e) \fatsemi^\sharp \llbracket\mathtt{B;S}\rrbracket^\sharp_\perp \qquad \llbracket\mathtt{while\ (B)\ S}\rrbracket^\sharp_{li} \triangleq \mathsf{gfp}^{\sqsubseteq^\sharp_\infty}\, F^\sharp_\perp \tag{10}$$

$$\llbracket\mathtt{while\ (B)\ S}\rrbracket^\sharp_\perp \triangleq \llbracket\mathtt{while\ (B)\ S}\rrbracket^\sharp_{bi} \sqcup^\sharp_\infty \llbracket\mathtt{while\ (B)\ S}\rrbracket^\sharp_{li} \tag{11}$$

The least fixpoint $\mathsf{lfp}^{\sqsubseteq^\sharp_+}\, \vec{F}^\sharp_e$ defines executions reaching the loop entry point after zero or finitely many iterations. Then (9) defines the finite executions of the loop when, after 0 or more iterations, the iteration condition `B` is false, or a break is executed in the body which exists the loop. By (9) the `break` is from the closest enclosing loop (which existence must be checked syntactically). The loop nontermination in (11) can happen either because, after zero or finitely many iterations, the next execution of the iteration body never terminates (10), or results in (10) from infinitely many finite iterations, as defined by the greatest fixpoint $\mathsf{gfp}^{\sqsubseteq^\sharp_\infty}\, F^\sharp_\perp$, and obtained as the limit of iterations of $F^\sharp_\perp$ from $\top^\sharp_\infty$. These fixpoints in (9) and (10) do exist by lemmas 3.6 and 3.11.

Theorem 3.13. Ⓐ *If $\mathbb{D}^\sharp$ is well-defined then for all $\mathtt{S}\in\mathbb{S}$, $\llbracket\mathtt{S}\rrbracket^\sharp_e$, $\llbracket\mathtt{S}\rrbracket^\sharp_b$, and $\llbracket\mathtt{S}\rrbracket^\sharp_\perp$ are well-defined.*

3.5 Algebraic Abstract Semantic Domain and Abstract Semantics

The components of the abstract semantics can be recorded in a triple with named components, ordered componentwise by $\sqsubseteq^\sharp$, as follows

$$\begin{aligned}\mathbb{L}^\sharp &\triangleq (e:\mathbb{L}^\sharp_+ \times \perp:\mathbb{L}^\sharp_\infty \times br:\mathbb{L}^\sharp_+)\\ \llbracket\mathtt{S}\rrbracket^\sharp &\triangleq \langle e:\llbracket\mathtt{S}\rrbracket^\sharp_e,\ \perp:\llbracket\mathtt{S}\rrbracket^\sharp_\perp,\ br:\llbracket\mathtt{S}\rrbracket^\sharp_b\rangle\end{aligned} \tag{12}$$

If $T = \langle e:F,\ \perp:I,\ br:B\rangle \in \mathbb{L}^\sharp$, then we select the individual components of the Cartesian product T using the field selectors e, br, and $\perp$, as follows

$$T_+ = F, \quad T_\infty = I, \quad \text{and} \quad T_{br} = B. \tag{13}$$

By convention,

The shorthand F denotes $\langle e:F,\ \perp:\bot^\sharp_\infty,\ br:\bot^\sharp_+\rangle$ and similarly for other unique nonempty components. (14)

The abstract semantics $\llbracket\mathtt{S}\rrbracket^\sharp \in \mathbb{L}^\sharp$ records three components $\llbracket\mathtt{S}\rrbracket^\sharp_e$, $\llbracket\mathtt{S}\rrbracket^\sharp_\perp$, and $\llbracket\mathtt{S}\rrbracket^\sharp_b$ of the definition of the algebraic semantics of statements `S` in sect. 3.4.

Lemma 3.14. Ⓐ *If $\mathbb{D}^\sharp$ is a well-defined chain-complete join semilattice (respectively complete lattice) with sequential composition $\fatsemi^\sharp$ satisfying any one of the hypotheses 3.2.D.d then $\langle\mathbb{L}^\sharp, \sqsubseteq^\sharp\rangle$ has the same structure, componentwise.*

All semantic definitions are extended componentwise. For $\mathbin{⨾}^\sharp \in \mathbb{L}^\sharp \times \mathbb{L}^\sharp \to \mathbb{L}^\sharp$, we define

$$\langle ok{:}\langle e{:}F_1, \bot{:}I_1\rangle, b{:}B_1\rangle \mathbin{⨾}^\sharp \langle ok{:}\langle e{:}F_2, \bot{:}I_2\rangle, b{:}B_2\rangle \triangleq \langle ok{:}\langle e{:}F_1 \mathbin{⨾}^\sharp F_2, \bot{:}I_1 \sqcup^\sharp_\infty (F_1 \mathbin{⨾}^\sharp I_2)\rangle, b{:}B_1 \sqcup^\sharp_+ (F_1 \mathbin{⨾}^\sharp B_2)\rangle$$

so that, by (4), $[\![\mathsf{S}_1 \,;\mathsf{S}_2]\!]^\sharp = [\![\mathsf{S}_1]\!]^\sharp \mathbin{⨾}^\sharp [\![\mathsf{S}_2]\!]^\sharp$. (15)

Remark 3.15. The semantic domain of our algebraic semantics is much more refined than traditional ones such as [57] where, the computational and logical ordering are subset inclusion and, following the denotational semantics [80] approach, "Nontermination has to be represented by a fictitious "state at infinity" that can be "reached" only by a non-terminating program. Also, if the fictitious state is in the image of a state, then that image is universal." [56]. This can be achieved by instantiation e.g. to a trace semantics followed by an abstraction (mapping infinite traces to the "fictitious "state at infinity"").

Moreover, we do not specify the algebraic semantics by "laws" (or axioms) but in structural fixpoint form, which is known to be equivalent, according to the generalization [25] of Peter Aczel correspondance [2] between deductive/proof systems and fixpoint definitions. The "laws" for basic statements are the definitions (3). The other "laws" for structured statements and iteration are theorems following from the definition 3.2 of an abstract domain and fixpoint induction principles [19] following from propositions 2.3 and 2.4. ■

All semantics in [4, 18, 41] can be instantiated to the algebraic abstract semantics of sect. 3.5. There are obviously others, such as symbolic execution [61] (extended to infinite behaviors). For semantics defined by transformations such as compilation, the transformation is an instance of the algebraic abstract semantics, but the semantics of the transformed program is not, because of a different syntax, although it can certainly be also defined in an algebraic style.

The original definition of hyperproperties [14] was relative to a trace (or path) semantics $[\![\mathsf{S}]\!]^\pi$ which, as shown in the appendix Ⓐ, is an instance of the algebraic abstract semantics $[\![\mathsf{S}]\!]^\sharp$ where the domain $\mathbb{D}^\sharp_+$ is the complete lattice $\mathbb{D}^\pi_+$ of sets of finite traces and the domain $\mathbb{D}^\sharp_\infty$ is the complete lattice $\mathbb{D}^\pi_\infty$ of sets of infinite traces where traces account for the successive values taken by variables during execution, as recorded in states. All operators preserve arbitrary joins. For lower continuity, see counterexample B.1 for infinite traces and the following lower continuity proof for finite traces.

Notice that the algebraic semantics can be instantiated to semantics of probabilistic and quantum programs. In this cases the hyperlogics developed in this paper, which differentiate between computational and approximation orders, apply to probabilistic programs [33, 79] and to quantum programs [39, 84, 85]

4 Structural Fixpoint Natural Relational Semantics

The structural fixpoint natural relational semantics of [21, sect. II.1] is an instance of the algebraic semantics of sect. 3. Given states Σ, $\bot \notin \Sigma$ denoting nontermination, and $\Sigma_\bot \triangleq \Sigma \cup \{\bot\}$, the finitary domain $\mathbb{L}^\varrho_+ \triangleq \langle \wp(\Sigma\times\Sigma), \subseteq\rangle$ in 3.2.A and the infinitary domain $\mathbb{L}^\varrho_\infty \triangleq \langle \wp(\Sigma\times\{\bot\}), \subseteq\rangle$ in 3.2.C are both complete lattices for set inclusion $\subseteq$ so $\bot^\varrho_+ = \emptyset$. We let $\mathbb{1}$ be the identity function. The primitives 3.2.B are well-defined.

$$\begin{array}{rcl@{\qquad}rcl}
\mathsf{assign}^\varrho[\![\mathsf{x}, \mathsf{A}]\!] & \triangleq & \{\langle \sigma,\ \sigma[\mathsf{x} \leftarrow \mathcal{A}[\![\mathsf{A}]\!]\sigma]\rangle \mid \sigma \in \Sigma\} & \mathsf{init}^\varrho & \triangleq & \mathbb{1}\\
\mathsf{rassign}^\varrho[\![\mathsf{x}, a, b]\!] & \triangleq & \{\langle \sigma,\ \sigma[\mathsf{x} \leftarrow i]\rangle \mid \sigma \in \Sigma \wedge a-1 < i < b+1\} & \mathsf{break}^\varrho & \triangleq & \mathbb{1}\\
\mathsf{test}^\varrho[\![\mathsf{B}]\!] & \triangleq & \{\langle \sigma,\ \sigma\rangle \mid \sigma \in \mathcal{B}[\![\mathsf{B}]\!]\} & \mathsf{skip}^\varrho & \triangleq & \mathbb{1}\\
r \mathbin{⨾}^\varrho r' & \triangleq & \multicolumn{4}{l}{\{\langle x,\ \bot\rangle \mid \langle x,\ \bot\rangle \in r\} \cup \{\langle x,\ y\rangle \mid \exists z \in \Sigma\,.\,\langle x,\ z\rangle \in r \wedge \langle z,\ y\rangle \in r'\}}
\end{array} \tag{16}$$

$\mathbin{⨾}^\varrho$ left preserves arbitrary joins $\cup$ on $\wp(\Sigma \times \Sigma_\bot)$. $\mathbin{⨾}^\varrho$ right preserves non empty joins $\cup$ on $\wp(\Sigma \times \Sigma_\bot)$. $\mathbin{⨾}^\varrho$ is right increasing (but not necessarily lower continuous for the finitary and infinitary domains) Ⓐ. (17)

Example 4.1. Define $\mathtt{S}_1 \triangleq$ `while (y!=0) y=y-1;` with relational semantics

$$[\![\mathtt{S}_1]\!]^{\varrho} \quad = \quad \langle e : \{\langle \sigma,\ \sigma[\mathtt{y} \leftarrow 0]\rangle \mid \sigma(\mathtt{y}) \geqslant 0\},\ \bot : \{\langle \sigma,\ \bot\rangle \mid \sigma(\mathtt{y}) < 0\},\ br : \varnothing\rangle$$

meaning that $\mathtt{S}_1$ terminates with $\mathtt{y} = 0$ when $\mathtt{y}$ is initially positive and otherwise does not terminate.
Define $\mathtt{S}_2 \triangleq$ `y=[-oo,oo]; S1` with relational semantics

$$[\![\mathtt{S}_2]\!]^{\varrho} \quad = \quad \langle e : \{\langle \sigma,\ \sigma[\mathtt{y} \leftarrow 0]\rangle \mid \sigma \in \Sigma\},\ \bot : \{\langle \sigma,\ \bot\rangle \mid \sigma \in \Sigma\},\ br : \varnothing\rangle$$

meaning that either $\mathtt{S}_2$ terminates with y=0 or does not terminate Ⓐ. ■

Example 4.2. Define $\mathtt{S}_3 \triangleq$ `while (x!=0) {` $\mathtt{S}_2$ `x=x-1; }` with relational semantics

$$[\![\mathtt{S}_3]\!]^{\sharp} = \langle e : \{\langle \sigma,\ \sigma\rangle \mid \sigma(\mathtt{x}) = 0\} \cup \{\langle \sigma,\ \sigma[\mathtt{y} \leftarrow 0][\mathtt{x} \leftarrow 0]\rangle \mid \sigma(\mathtt{x}) > 0\},\ \bot : \{\langle \sigma,\ \bot\rangle \mid \sigma(\mathtt{x}) \neq 0\},\ br : \varnothing\rangle$$

meaning that $\mathtt{S}_3$ terminates because either the loop is not entered or it is entered with $\mathtt{x} > 0$ and $\mathtt{S}_2$ terminates at each iteration setting $\mathtt{y}$ to 0. $\mathtt{S}_3$ does not terminate when the loop is entered and either its body does not terminate or $\mathtt{x} < 0$.
Define $\mathtt{S}_4 \triangleq$ `x=[-oo,oo]; S3` with relational semantics

$$[\![\mathtt{S}_4]\!]^{\sharp} \ = \ \langle e : \{\langle \sigma,\ \sigma[\mathtt{x} \leftarrow 0]\rangle \mid \sigma \in \Sigma\} \cup \{\langle \sigma,\ \sigma[\mathtt{y} \leftarrow 0][\mathtt{x} \leftarrow 0]\rangle \mid \sigma \in \Sigma\},\ \bot : \{\langle \sigma,\ \bot\rangle \mid \sigma \in \Sigma\},\ br : \varnothing\rangle$$

meaning either termination with x=0 (when x is randomly assigned 0) or with x=0 and y=0 (when x is randomly assigned a positive number while x is randomly assigned a positive number or zero) or nontermination (when x is randomly assigned a negative number or x is randomly assigned a positive number and y are randomly assigned a negative number). Ⓐ. In this example, the fixpoint iterations are infinite but would be transfinite for a transition semantics (corresponding to the lexicographic ordering for the nested loops) [18]. ■

5 Algebraic Program Execution Properties

5.1 Algebraic Execution Properties

Traditionally, logics involve two formal languages, one to express programs and another one to express properties of the program executions. The syntax and semantics of these programming and logic languages are considered to be different. Therefore, in addition to the program syntax and semantics, this traditional approach requires to define the syntax and semantics of the logic expressing program properties.

A semantics $[\![\mathtt{S}]\!]^{\sharp} \in \mathbb{L}^{\sharp}$ in (12) is an abstraction of a property of the executions of the statement S. Therefore $\mathbb{L}^{\sharp}$ will be the domain of execution properties whether used to describe the semantics or logic properties of programs executions. This will avoid us the necessary traditional distinction between programs semantics and program properties.

This idea follows [52–54]'s slogan that "Programs are predicates" and define properties of program executions as programs (which semantics is already defined). It is also found in Dexter Kozen's Kleene algebra with tests [62, 63, 82]. Therefore, from an abstract point of view, program execution specification and verification need nothing more than programs and an associated calculus $\mathsf{post}^{\sharp}$ on programs.

5.2 The Algebraic Program Execution Property Transformer

Let us define the transformer $\mathsf{post}^{\sharp} \in \mathbb{L}^{\sharp} \xrightarrow{\nearrow} \mathbb{L}^{\sharp} \xrightarrow{\nearrow} \mathbb{L}^{\sharp}$ such that

$$\mathsf{post}^{\sharp}(S)P \quad \triangleq \quad P \mathbin{⨾}^{\sharp} S \tag{18}$$

where S is a semantics in $\mathbb{L}^{\sharp}$ as defined by (12) and $⨾^{\sharp}$ is defined by (15). If P is a precondition when at S then $\mathsf{post}^{\sharp}[\![\mathtt{S}]\!]^{\sharp}P$ is the postcondition after S (including when breaking out of S).

For example, using the shorthand (14), $\mathsf{post}^{\sharp}(S)\mathsf{init}^{\sharp} = S$ by 3.2.D.a and $\mathsf{post}^{\sharp}(S)P = P$ for all $P \in \mathbb{L}^{\sharp}_{\infty}$ by 3.2.D.c.

In definition (18) of "predicate transformers" the meaning of "predicates" about programs executions is abstracted away as programs specifying executions. Further abstractions will yield the classic understanding of "predicates", "abstract property", etc. The classic Galois connections post–$\widetilde{\mathsf{pre}}$ [20, (12.22)] and post–post^{-1} [20, (12.6)] are still valid with this different definition of post.

The following lemmas show that the post transformer inherits the properties of sequential composition. It applies e.g. to $\langle \mathbb{L}^\sharp_+, \sqsubseteq^\sharp_+\rangle$ in 3.2.A, $\langle \mathbb{L}^\sharp_\infty, \sqsubseteq^\sharp_\infty\rangle$ in 3.2.C, or $\langle \mathbb{L}^\sharp, \sqsubseteq^\sharp\rangle$ in (12).

Lemma 5.1. Ⓐ *Let $\langle L, \sqsubseteq, \sqcup\rangle$ be a poset with partially defined join $\sqcup$. Let $\fatsemi$ be the sequential composition on L. If $\fatsemi$ left-satisfies any one of the properties of definition 2.2 or their dual then for all $S \in \mathbb{L}$, $\mathsf{post}(S)$ satisfies the same property.*

The following Galois connection shows the equivalence of forward/deductive and backward/abductive reasonings on the program semantics.

Lemma 5.2. Ⓐ *If $\langle L, \sqsubseteq, \sqcup\rangle$ is a poset and the sequential composition $\fatsemi$ is existing $\sqcup$ left preserving then we have the Galois connection*

$$\forall S \in \mathbb{L}\,.\,\langle \mathbb{L}, \sqsubseteq\rangle \underset{\mathsf{post}(S)}{\overset{\widetilde{\mathsf{pre}}(S)}{\leftrightarrows}} \langle \mathbb{L}, \sqsubseteq\rangle \quad \textit{where} \quad \widetilde{\mathsf{pre}}(S)Q \triangleq \bigsqcup\{P \in \mathbb{L} \mid \mathsf{post}(S)P \sqsubseteq Q\}\big). \tag{19}$$

Lemma 5.3. Ⓐ *Let $\langle L, \sqsubseteq, \sqcup\rangle$ be a poset with partially defined join $\sqcup$. Let $\fatsemi$ be the sequential composition on L. If $\fatsemi$ right-satisfies any one of the properties of definition 2.2 or their dual then* post *satisfies the same property.*

The following Galois connection formalizes Dijkstra's program inversion [36].

Lemma 5.4. Ⓐ *If $\langle L, \sqsubseteq, \sqcup\rangle$ is a poset and the sequential composition $\fatsemi$ is existing $\sqcup$ right preserving then we have the following Galois connection ($\mathbb{L} \xrightarrow{\sqcup} \mathbb{L}$ is the set of existing join preserving operators on $\mathbb{L}$ and $\dot\sqsubseteq$ is the pointwise extension of $\sqsubseteq$)*

$$\langle \mathbb{L}, \sqsubseteq\rangle \underset{\mathsf{post}}{\overset{\mathsf{post}^{-1}}{\leftrightarrows}} \langle \mathbb{L} \xrightarrow{\sqcup} \mathbb{L}, \dot\sqsubseteq\rangle \quad \textit{where} \quad \mathsf{post}^{-1}(T) = \bigsqcup\{S \in \mathbb{L} \mid \mathsf{post}(S) \dot\sqsubseteq T\}. \tag{20}$$

5.3 A Calculus of Algebraic Program Execution Properties

We derive the sound and complete post$^\sharp$ calculus by calculational design, as follows.

Theorem 5.5 (Program execution property calculus). Ⓐ *If $\mathbb{D}^\sharp$ is a well-defined increasing and decreasing chain-complete join semilattice with right upper continuous sequential composition $\fatsemi^\sharp$ then*

$$\mathsf{post}^\sharp[\![\mathtt{x = A}]\!]^\sharp P = \langle e : P_+ \mathbin{\fatsemi^\sharp} \mathsf{assign}^\sharp[\![\mathtt{x}, \mathtt{A}]\!],\ \bot : P_\infty,\ br : P_{br}\rangle \tag{21}$$

$$\mathsf{post}^\sharp[\![\mathtt{x = [}a\mathtt{,}\ b\mathtt{]}]\!]^\sharp P = \langle e : P_+ \mathbin{\fatsemi^\sharp} \mathsf{rassign}^\sharp[\![\mathtt{x}, a, b]\!],\ \bot : P_\infty,\ br : P_{br}\rangle \tag{22}$$

$$\mathsf{post}^\sharp[\![\mathtt{skip}]\!]^\sharp P = \langle e : P_+ \mathbin{\fatsemi^\sharp} \mathsf{skip}^\sharp,\ \bot : P_\infty,\ br : P_{br}\rangle \tag{23}$$

$$\mathsf{post}^\sharp[\![\mathtt{B}]\!]^\sharp P = \langle e : P_+ \mathbin{\fatsemi^\sharp} \mathsf{test}^\sharp[\![\mathtt{B}]\!],\ \bot : P_\infty,\ br : P_{br}\rangle \tag{24}$$

$$\mathsf{post}^\sharp[\![\mathtt{break}]\!]^\sharp P = \langle e : \bot_+^\sharp,\ \bot : P_\infty,\ br : P_{br} \sqcup_+^\sharp (P_e \mathbin{\fatsemi^\sharp} \mathsf{break}^\sharp)\rangle \tag{25}$$

$$\mathsf{post}^\sharp[\![\mathtt{S}_1\mathtt{;S}_2]\!]^\sharp P = \mathsf{post}^\sharp[\![\mathtt{S}_2]\!]^\sharp(\mathsf{post}^\sharp[\![\mathtt{S}_1]\!]^\sharp P) \tag{26}$$

$$\mathsf{post}^\sharp[\![\mathtt{if\ (B)\ S}_1\ \mathtt{else\ S}_2]\!]^\sharp P = \mathsf{post}^\sharp[\![\mathtt{B;S}_1]\!]^\sharp P \sqcup^\sharp \mathsf{post}^\sharp[\![\neg\mathtt{B;S}_2]\!]^\sharp P \tag{27}$$

$$\vec{F}_{pe}^\sharp \triangleq \boldsymbol{\lambda} P \bullet \boldsymbol{\lambda} X \bullet \mathsf{post}^\sharp(\mathsf{init}^\sharp)P \sqcup_+^\sharp \mathsf{post}^\sharp([\![\mathtt{B;S}]\!]_e^\sharp)(X) \tag{28}$$

$$F_{p\bot}^\sharp \triangleq \boldsymbol{\lambda} X \bullet \mathsf{post}^\sharp(X)([\![\mathtt{B;S}]\!]_e^\sharp) \tag{29}$$

$$\begin{aligned}\mathsf{post}^\sharp[\![\mathtt{while\ (B)\ S}]\!]^\sharp P = \langle ok : \langle & e : \mathsf{post}^\sharp([\![\neg\mathtt{B}]\!]_e^\sharp \sqcup_e^\sharp [\![\mathtt{B;S}]\!]_b^\sharp)(\mathsf{lfp}^{\sqsubseteq_+^\sharp} (\vec{F}_{pe}^\sharp(P))),\\ & \bot : \mathsf{post}^\sharp([\![\mathtt{B;S}]\!]_\bot^\sharp)(\mathsf{lfp}^{\sqsubseteq_+^\sharp} (\vec{F}_{pe}^\sharp(P))) \sqcup_\infty^\sharp\\ & \qquad \mathsf{post}^\sharp(\mathsf{gfp}^{\sqsubseteq_\infty^\sharp} F_{p\bot}^\sharp)P\ \rangle,\\ br : P_{br}\rangle &\end{aligned} \tag{30}$$

is sound and complete.

Remark 5.6. By defining the appropriate primitives, the post program execution calculus (21) — (30) of theorem 5.5 is an instance of the generic abstract semantics (12). ■

Example 5.7 (Finitary powerset deterministic calculational domain). In [5], the while language is deterministic and has no breaks so the random assignment and breaks have to be eliminated in (3). The denotational semantics is $[\![\mathtt{S}]\!] \in (\Sigma \times \Sigma)_\bot \to (\Sigma \times \Sigma)_\bot$ where $(\Sigma \times \Sigma)_\bot$ is the domain of relations between states extended by $\bot$ to denote nontermination with Scott flat ordering $\sqsubseteq$.

Anticipating on the abstractions of part II, this is an abstraction [18, sect. 8.2] of the trace semantics of sect. B. Then a semantic abstraction 8.1 gets rid of nontermination [18, sect. 8.1.6] and another one [18, sect. 9.1] abstracts relations to transformers to yield the collecting semantics [5, p. 876].

Skipping these abstractions of the trace semantics, we can directly instantiate the generic abstract semantics of sect. 3 to a finitary relational semantics such as $[\![S]\!]^e$ in [21]. Then $\mathsf{post}^\sharp$ in (18) becomes $\mathsf{post}^\sharp(S)P = \{\langle\sigma,\ \sigma''\rangle \mid \exists \sigma' \in \Sigma\ .\ \langle\sigma,\ \sigma'\rangle \in P \wedge \langle\sigma',\ \sigma''\rangle \in S\}$, which is a specification of the collecting semantics postulated in [5, p. 876]. $\mathsf{post}^\sharp(S)$ preserves arbitrary unions so, in absence of breaks and ignoring nontermination, together with $[\![\mathtt{B}]\!]_e^\sharp \circ [\![\mathtt{B}]\!]_e^\sharp = [\![\mathtt{B}]\!]_e^\sharp$, $[\![\mathtt{B}]\!]_e^\sharp \circ [\![\neg\mathtt{B}]\!]_e^\sharp = \emptyset$, and$[\![\mathtt{skip}]\!]_e^\sharp = \mathsf{init}^\sharp$ by 3.2.D.a, (30) in theorem 5.5 simplifies to

$$\mathsf{post}^\sharp[\![\mathtt{while\ (B)\ S}]\!]^\sharp P = \mathsf{post}^\sharp([\![\neg\mathtt{B}]\!]_e^\sharp)(\mathsf{lfp}^{\subseteq} \boldsymbol{\lambda} X \bullet P \cup \mathsf{post}^\sharp([\![\mathtt{if\ (B)\ S\ else\ skip}]\!]_e^\sharp)(X)$$

which is precisely the data-independent abstraction of the collecting semantics of [5, p. 876]. ■

5.4 Algebraic Logics of Program Execution Properties

By defining $\overline{\{}\,P\,\overline{\}}\,\mathtt{S}\,\overline{\{}\,Q\,\overline{\}} \triangleq (\langle P,\ Q\rangle \in \hat{\alpha}([\![\mathtt{S}]\!]^\sharp))$ with $\hat{\alpha}(S) \triangleq \{\langle P,\ Q\rangle \mid \mathsf{post}^\sharp(S)P \sqsubseteq^\sharp Q\}$ and dually $\underline{\{}\,P\,\underline{\}}\,\mathtt{S}\,\underline{\{}\,Q\,\underline{\}} \triangleq (\langle P,\ Q\rangle \in \check{\alpha}([\![\mathtt{S}]\!]^\sharp))$ with $\check{\alpha}(S) \triangleq \{\langle P,\ Q\rangle \mid Q \sqsubseteq^\sharp \mathsf{post}^\sharp(S)P\}$, we respectively get the abstract version [20, chapter 26] of Hoare logic [55] and that of reverse/incorrectness logic [32, 75] (extended to loops breaks and nontermination [21, 65]). This is now classic and will be used but not be further detailled.

6 A Calculus of Algebraic Program Semantic (Hyper) Properties

We now study proof methods for semantic properties, that is properties of the semantics, that we define in extension. This is called hyperproperties when the semantics is a set of traces [13, 14], and by extension, for their abstractions, in particular to relational semantics.

6.1 Algebraic Semantic (Hyper) Properties

Defined in extension, program semantic properties are in $\wp(\mathbb{L}^\sharp)$.

Example 6.1 (Algebraic noninterference). Noninterference [46], can be generalized to semantic (hyper) properties of algebraic semantics, as follows. The precondition $R_i \in \wp(\mathbb{L}^\sharp_+ \times \mathbb{L}^\sharp_+)$ is a relation between prelude executions extended to $\mathbb{L}^\sharp$ by (14). The postcondition $R_f \in \wp(\mathbb{L}^\sharp \times \mathbb{L}^\sharp)$ is a relation between terminated or infinite executions. Then algebraic noninterference is $\mathsf{ANI} \triangleq \{\mathcal{P} \in \wp(\mathbb{L}^\sharp) \mid \forall S_1, S_2 \in \mathcal{P} \,.\, \forall P_1, P_2 \in \mathbb{L}^\sharp_+ \,.\, \langle P_1,\ P_2\rangle \in R_i \implies \langle \mathsf{post}^\sharp(S_1)P_1,\ \mathsf{post}^\sharp(S_2)P_2\rangle \in R_f\}$. An instance is algebraic abstract noninterference $\mathsf{AANI} \triangleq \{\mathcal{P} \in \wp(\mathbb{L}^\sharp) \mid \forall S_1, S_2 \in \mathcal{P} \,.\, \forall P_1, P_2 \in \mathbb{L}^\sharp_+ \,.\, \alpha_1(P_1) = \alpha_1(P_2) \implies \alpha_2(\mathsf{post}^\sharp(S_1)P_1) = \alpha_2(\mathsf{post}^\sharp(S_2)P_2)\}$ for abstractions $\alpha_1 \in \mathbb{L}^\sharp \to A_1$ and $\alpha_2 \in \mathbb{L}^\sharp \to A_2$ with special case $\alpha_1 = \alpha_2$ to characterize abstract domain completeness in abstract interpretation [42, 43, 68]. After [14], the generalized algebraic noninterference is $\mathsf{GANI} \triangleq \{\mathcal{P} \in \wp(\mathbb{L}^\sharp) \mid \forall S_1, S_2 \in \mathcal{P} \,.\, \exists \bar{S} \in \mathcal{P} \,.\, \forall P_1, P_2 \in \mathbb{L}^\sharp_+ \,.\, \forall \bar{P} \in \bar{S} \,.\, \langle \bar{P},\ P_1\rangle \in R_i \implies \langle \mathsf{post}^\sharp(S_1)\bar{P},\ \mathsf{post}^\sharp(S_2)P_2\rangle \in R_f\}$. ■

6.2 The Algebraic Program Semantic (Hyper) Properties Transformer

When considering semantic properties in extension, the traditional view of transformers is that they now belong to $\wp(\mathbb{L}^\sharp) \to \wp(\mathbb{L}^\sharp)$ with

$$\begin{array}{rcl} \mathsf{Post}^\sharp & \in & \mathbb{L}^\sharp \to \wp(\mathbb{L}^\sharp) \xrightarrow{\nearrow} \wp(\mathbb{L}^\sharp) \\ \mathsf{Post}^\sharp(S)\mathcal{P} & \triangleq & \{\mathsf{post}^\sharp(S)P \mid P \in \mathcal{P}\} \end{array} \tag{31}$$

[5, 29, 30, 67] are all instances of this definition. The advantage is that logical implication is the traditional $\subseteq$. But the classic structural definition (see sect. 3.2) of the transformer $\mathsf{Post}^\sharp$ fails (unless restrictions are placed on the considered hyperproperties). For the conditional

$$\begin{array}{lll}
 & \mathsf{Post}^\sharp[\![\texttt{if (B) S}_1\texttt{ else S}_2]\!]^\sharp\mathcal{P} & \\
= & \{\mathsf{post}^\sharp[\![\texttt{if (B) S}_1\texttt{ else S}_2]\!]^\sharp P \mid P \in \mathcal{P}\} & \langle\!\!\!\langle\text{def. (31) of } \mathsf{Post}^\sharp(S)\rangle\!\!\!\rangle \\
= & \{\mathsf{post}^\sharp[\![\texttt{B;S}_1]\!]^\sharp P \sqcup^\sharp \mathsf{post}^\sharp[\![\neg\texttt{B;S}_2]\!]^\sharp P \mid P \in \mathcal{P}\} & \langle\!\!\!\langle(27)\rangle\!\!\!\rangle \quad (32) \\
\subseteq & \{\mathsf{post}^\sharp[\![\texttt{B;S}_1]\!]^\sharp P_1 \sqcup^\sharp \mathsf{post}^\sharp[\![\neg\texttt{B;S}_2]\!]^\sharp P_2 \mid P_1 \in \mathcal{P} \wedge P_2 \in \mathcal{P}\} & \langle\!\!\!\langle\text{def. } \subseteq\rangle\!\!\!\rangle \quad (33) \\
= & \{Q_1 \sqcup^\sharp Q_2 \mid Q_1 \in \{\mathsf{post}^\sharp[\![\texttt{B;S}_1]\!]^\sharp P_1 \mid P_1 \in \mathcal{P}\} \wedge Q_2 \in \{\mathsf{post}^\sharp[\![\neg\texttt{B;S}_2]\!]^\sharp P_2 \mid P_2 \in \mathcal{P}\}\} & \langle\!\!\!\langle\text{def. } \in\rangle\!\!\!\rangle \\
= & \{Q_1 \sqcup^\sharp Q_2 \mid Q_1 \in \mathsf{Post}^\sharp[\![\texttt{B;S}_1]\!]^\sharp\mathcal{P} \wedge Q_2 \in \mathsf{Post}^\sharp[\![\neg\texttt{B;S}_2]\!]^\sharp\mathcal{P}\} & \langle\!\!\!\langle\text{def. (31) of } \mathsf{Post}^\sharp(S)\rangle\!\!\!\rangle
\end{array}$$

The problem is that in (32) the two possible executions of the conditional are tight together, whereas, by necessity of traditional independent structural induction on both branches of the conditional, this link is lost in (33). So the hypercollecting semantics of [5, p. 877] is incomplete (the inclusion (33) may be strict). A solution to preserve structurality is to observe that

$$\{\mathsf{post}^\sharp(S)P\} \quad = \quad \mathsf{Post}^\sharp(S)\{P\} \tag{34}$$

so that the calculation goes on at (32)

$$\begin{array}{lll}
= & \{Q_1 \sqcup^\sharp Q_2 \mid Q_1 \in \{\mathsf{post}^\sharp[\![\texttt{B;S}_1]\!]^\sharp P\} \wedge Q_2 \in \{\mathsf{post}^\sharp[\![\neg\texttt{B;S}_2]\!]^\sharp P\} \wedge P \in \mathcal{P}\} & \langle\!\!\!\langle\text{def. singleton and } \in\rangle\!\!\!\rangle \\
= & \{Q_1 \sqcup^\sharp Q_2 \mid Q_1 \in \mathsf{Post}^\sharp[\![\texttt{B;S}_1]\!]^\sharp\{P\} \wedge Q_2 \in \mathsf{Post}^\sharp[\![\neg\texttt{B;S}_2]\!]^\sharp\{P\} \wedge P \in \mathcal{P}\} & \langle\!\!\!\langle\text{def. (31) of } \mathsf{Post}^\sharp(S)\rangle\!\!\!\rangle
\end{array}$$

so that $\mathsf{Post}^\sharp[\![\texttt{if (B) S}_1\texttt{ else S}_2]\!]^\sharp$ is exactly defined structurally as a function of the components $\mathsf{Post}^\sharp[\![\texttt{B;S}_1]\!]^\sharp$ and $\mathsf{Post}^\sharp[\![\neg\texttt{B;S}_2]\!]^\sharp$.

Of course, this element wise reasoning may be considered inelegant. Its necessity becomes more clear when considering the trace semantics of sect. B. When reasoning on paths e.g. in an iteration statement, the same paths must be considered consistently at each iteration. This requirement may be lifted after abstraction, for example with invariants which forget about computation history. For backward reasonings, we define Pre such that for all $S \in \mathbb{L}^\sharp$, we have Ⓐ

$$\mathsf{Pre}(S)\mathcal{Q} \triangleq \{P \mid \mathsf{post}^\sharp(S)P \in \mathcal{Q}\} \quad (35) \qquad \langle\wp(\mathbb{L}^\sharp), \subseteq\rangle \underset{\mathsf{Post}^\sharp(S)}{\overset{\mathsf{Pre}(S)}{\leftrightarrows}} \langle\wp(\mathbb{L}^\sharp), \subseteq\rangle \quad (36)$$

If $\mathbb{D}^\sharp$ is a well-defined chain-complete lattice with right finite $\boxed{x}$ preservation composition $\fatsemi^\sharp$ then we have ($\boxed{x}$, $x \in \{+, \infty\}$, stands for $\sqcup^\sharp_+$ in definition 3.2.A when $x = +$ and for $\sqcup^\sharp_\infty$ in definition 3.2.C when $x = \infty$) Ⓐ

$$\begin{array}{lrcl} & \mathsf{Post}^\sharp(S_1 \boxed{x} S_2)\mathcal{P} & = & (\mathsf{Post}^\sharp(S_1) \boxed{x} \mathsf{Post}^\sharp(S_2))\mathcal{P} \\ \text{where} & (S_1 \boxed{x} S_2)\mathcal{P} & \triangleq & \{Q_1 \boxed{x} Q_2 \mid Q_1 \in S_1\{P\} \wedge Q_2 \in S_2\{P\} \wedge P \in \mathcal{P}\} \end{array} \quad (37)$$

Remark 6.2. Contrary to join preservation lemma 5.1 for post, Post may not preserve existing joins and meets so that, in general, $\bigsqcup_{i\in\Delta} \mathsf{Post}^\sharp(S_i) \neq \mathsf{Post}^\sharp(\bigsqcup_{i\in\Delta} S_i)$ and dually. For example, let $\mathcal{P}$ be a semantic property. By (31), $\bigsqcup^\sharp_{+\,n\in\mathbb{N}} \mathsf{Post}^\sharp((\llbracket \mathsf{B} \fatsemi \mathsf{S}\rrbracket^\sharp)^n)\mathcal{P} = \bigsqcup^\sharp_{+\,n\in\mathbb{N}}\{\mathsf{post}^\sharp((\llbracket \mathsf{B} \fatsemi \mathsf{S}\rrbracket^\sharp)^n)P \mid P \in \mathcal{P}\}$ is the set of finite executions, for every precondition $P \in \mathcal{P}$, reaching the entry of the iteration `while(B) S` after exactly n terminating body iterations, for all $n \in \mathbb{N}$. On the contrary $\mathsf{Post}^\sharp(\bigsqcup^\sharp_{+\,n\in\mathbb{N}}(\llbracket \mathsf{B} \fatsemi \mathsf{S}\rrbracket^\sharp)^n)\mathcal{P} = \{\mathsf{post}^\sharp(\bigsqcup^\sharp_{+\,n\in\mathbb{N}}(\llbracket \mathsf{B} \fatsemi \mathsf{S}\rrbracket^\sharp)^n)P \mid P \in \mathcal{P}\} = \{\bigsqcup^\sharp_{+\,n\in\mathbb{N}} \mathsf{post}^\sharp((\llbracket \mathsf{B} \fatsemi \mathsf{S}\rrbracket^\sharp)^n)P \mid P \in \mathcal{P}\}$ is the set of finite executions, for every precondition $P \in \mathcal{P}$, reaching the entry of the iteration `while(B) S` after any number of terminating body iterations. ∎

6.3 A Calculus of Algebraic Semantic (Hyper) Properties

In the calculational design of the $\mathsf{Post}^\sharp$, we will need the following trivial proposition.

Proposition 6.3 (Singleton fixpoint). *There is an obvious isomorphism between a poset $\langle L, \sqsubseteq, \bot, \sqcup\rangle$ and its singletons $\langle \breve{L}, \breve{\sqsubseteq}, \breve{\bot}, \breve{\sqcup}\rangle$ with $\breve{L} \triangleq \{\{x\} \mid x \in L\}$, $\{x\}\breve{\sqsubseteq}\{y\} \triangleq x \sqsubseteq y$, $\breve{\bot} \triangleq \{\bot\}$, $\{x\}\breve{\sqcup}\{y\} \triangleq \{x \sqcup y\}$, so that, for a increasing chain complete poset we have $\{\mathsf{lfp}^{\sqsubseteq} F\} = \{\bigsqcup_{\delta\in\mathbb{O}} F^\delta\} = \breve{\bigsqcup}_{\delta\in\mathbb{O}}\{F^\delta\} = \mathsf{lfp}^{\breve{\sqsubseteq}} \breve{F}$ where $\langle F^\delta, \delta \in \mathbb{O}\rangle$ are the transfinite iterates of F from $\bot$ and $\breve{F}(\{x\}) \triangleq \{F(x)\}$. Dually for greatest fixpoints.*

We derive the sound and complete $\mathsf{Post}^\sharp$ calculus by calculational design, as follows.

Theorem 6.4 (Program semantic (hyper) property calculus). Ⓐ *If $\mathbb{D}^\sharp$ is a well-defined increasing and decreasing chain-complete join semilattice with right upper continuous sequential composition $\fatsemi^\sharp$ then*

$$
\begin{aligned}
\mathsf{Post}^\sharp[\![\mathtt{x\ =\ A}]\!]^\sharp\mathcal{P} &= \{\langle e: P_+ \,\mathbin{⨟}^\sharp\, \mathsf{assign}^\sharp[\![\mathsf{x},\mathsf{A}]\!],\ \bot: P_\infty,\ br: P_{br}\rangle \mid P \in \mathcal{P}\} && (38)\\
\mathsf{Post}^\sharp[\![\mathtt{x\ =\ [}a\mathtt{,}\ b\mathtt{]}]\!]^\sharp\mathcal{P} &= \{\langle e: P_+ \,\mathbin{⨟}^\sharp\, \mathsf{rassign}^\sharp[\![\mathsf{x},a,b]\!],\ \bot: P_\infty,\ br: P_{br}\rangle \mid P \in \mathcal{P}\} && (39)\\
\mathsf{Post}^\sharp[\![\mathtt{skip}]\!]\mathcal{P} &= \{\langle e: P_+ \,\mathbin{⨟}^\sharp\, \mathsf{skip}^\sharp,\ \bot: P_\infty,\ br: P_{br}\rangle \mid P \in \mathcal{P}\} && (40)\\
\mathsf{Post}^\sharp[\![\mathtt{B}]\!]^\sharp\mathcal{P} &= \{\langle e: P_+ \,\mathbin{⨟}^\sharp\, \mathsf{test}^\sharp[\![\mathsf{B}]\!],\ \bot: P_\infty,\ br: P_{br}\rangle \mid P \in \mathcal{P}\} && (41)\\
\mathsf{Post}^\sharp[\![\mathtt{break}]\!]^\sharp\mathcal{P} &= \{\langle e: \bot_+^\sharp,\ \bot: P_\infty,\ br: P_{br} \sqcup_+^\sharp (P_e \,\mathbin{⨟}^\sharp\, \mathsf{break}^\sharp)\rangle \mid P \in \mathcal{P}\} && (42)\\
\mathsf{Post}^\sharp[\![\mathtt{S}_1\mathtt{;S}_2]\!]^\sharp\mathcal{P} &= \mathsf{Post}^\sharp[\![\mathtt{S}_2]\!]^\sharp(\mathsf{Post}^\sharp[\![\mathtt{S}_1]\!]^\sharp\mathcal{P}) && (43)\\
\mathsf{Post}^\sharp[\![\mathtt{if(B)\ S_1\ else\ S_2}]\!]^\sharp\mathcal{P} &= (\mathsf{Post}^\sharp[\![\mathtt{B;S}_1]\!]^\sharp \sqcup^\sharp \mathsf{Post}^\sharp[\![\neg\mathtt{B;S}_2]\!]^\sharp)\mathcal{P} && (44)\\
\breve{\vec{F}}{}^\sharp_{pe} &\triangleq \boldsymbol{\lambda} P\bullet\boldsymbol{\lambda} X\bullet \mathsf{Post}^\sharp(\mathsf{init}^\sharp)\{P\} \mathbin{\breve{\sqcup}}_+^\sharp \mathsf{Post}^\sharp([\![\mathtt{B;S}]\!]^\sharp_e)(X) && (45)\\
\breve{F}^\sharp_{p\bot} &\triangleq \boldsymbol{\lambda} X\bullet \bigcup\{\mathsf{Post}^\sharp(S)([\![\mathtt{B;S}]\!]^\sharp_e) \mid S \in X\} && (46)\\
\mathsf{Post}^\sharp[\![\mathtt{while(B)\ S}]\!]^\sharp\mathcal{P} &= \{\langle e: Q_e,\ \bot: Q_{\bot\ell} \sqcup_\infty^\sharp Q_{\bot b},\ br: P_{br}\rangle \mid && (47)\\
&\qquad Q_e \in \mathsf{Post}^\sharp([\![\neg\mathtt{B}]\!]^\sharp_e \sqcup^\sharp_e [\![\mathtt{B;S}]\!]^\sharp_b)(\mathsf{lfp}^{\mathrel{\breve{\sqsubseteq}}^\sharp_+}\, \breve{\vec{F}}{}^\sharp_{pe}(P)) \wedge\\
&\qquad Q_{\bot\ell} \in \mathsf{Post}^\sharp([\![\mathtt{B;S}]\!]^\sharp_\bot)(\mathsf{lfp}^{\mathrel{\breve{\sqsubseteq}}^\sharp_+}\, (\breve{\vec{F}}{}^\sharp_{pe}(P))) \wedge\\
&\qquad \exists Q_{\bot b}\ .\ Q_{\bot b} \in \mathsf{Post}^\sharp(Q_{p\bot})\{P\} \wedge Q_{p\bot} \in \mathsf{gfp}^{\mathrel{\breve{\sqsubseteq}}^\sharp_\infty}\, \breve{F}^\sharp_{p\bot} \wedge P \in \mathcal{P}\}
\end{aligned}
$$

(*where* $S_1 ⨝ S_2$ *is defined in* (37)) *is sound and complete.*

Example 6.5 (Finitary powerset calculational domain). Continuing example 5.7 ignoring `breaks` and nontermination, the hypercollecting semantics of [5, p. 877] is

$$
\begin{aligned}
&\mathsf{Post}^\sharp([\![\neg\mathtt{B}]\!]^\sharp_e)(\mathsf{lfp}^\subseteq \boldsymbol{\lambda} X\bullet \mathcal{P} \cup \mathsf{Post}^\sharp([\![\mathtt{if\ (B)\ S\ else\ skip}]\!]^\sharp_e)(X)) && (48)\\
=\ &\{\mathsf{Post}^\sharp([\![\neg\mathtt{B}]\!]^\sharp_e)(\mathsf{Post}^\sharp([\![\mathtt{if\ (B)\ S\ else\ skip}]\!]^\sharp_e)^n\mathcal{P}) \mid n \in \mathbb{N}\}\\
=\ &\{\mathsf{Post}^\sharp([\![\neg\mathtt{B}]\!]^\sharp_e)(\mathsf{Post}^\sharp([\![\mathtt{if\ (B)\ S\ else\ skip}]\!]^\sharp_e)^n\{P\}) \mid n \in \mathbb{N} \wedge P \in \mathcal{P}\}\\
\neq\ &\bigcup\{\mathsf{Post}^\sharp([\![\neg\mathtt{B}]\!]^\sharp_e)(\mathsf{lfp}^\subseteq \breve{\vec{F}}{}^\sharp_{pe}(P)) \mid P \in \mathcal{P}\}
\end{aligned}
$$

By remark 6.2, this is different from (47) (even when ignoring nontermination and `breaks`) so that [5, p. 877] is incomplete and cannot be used as a hypercollecting semantics for general hyperproperties, as further discussed in sect. 20. Moreover (48) is unsound, invalidating [5, th. 1]. This will be fixed by the weak hypercollecting semantics defined in (91). ■

7 Abstract Logic of Semantic (Hyper) Properties

7.1 Definition of the Upper and Lower Abstract Logics

The upper (respectively lower) logic $\overline{\mathsf{L}}^\sharp$ (resp. $\underline{\mathsf{L}}^\sharp$) maps the semantics S of a statement into a pair of a precondition and postcondition that is $\overline{\mathsf{L}}^\sharp, \underline{\mathsf{L}}^\sharp \in \mathbb{L}^\sharp \to (\wp(\mathbb{L}^\sharp) \times \wp(\mathbb{L}^\sharp))$ ordered pointwise by $\subseteq$ (the larger the precondition, the larger is the postcondition). We have

$$\overline{\mathsf{L}}^\sharp(S) \quad\triangleq\quad \{\langle\mathcal{P},\ \mathcal{Q}\rangle \mid \mathsf{Post}^\sharp(S)\mathcal{P} \subseteq \mathcal{Q}\} \tag{49}$$

where $\langle\mathcal{P},\ \mathcal{Q}\rangle \in \overline{\mathsf{L}}^\sharp[\![\mathtt{S}]\!]^\sharp$ is traditionally written $\overline{\{\!|}\,\mathcal{P}\,\overline{|\!\}}\ \mathtt{S}\ \overline{\{\!|}\,\mathcal{Q}\,\overline{|\!\}}$. The $\subseteq$-dual holds for the lower abstract logic. As was the case in sect. 5.4 for execution properties, this is an abstraction $\overset{\blacktriangle}{\alpha}(\mathsf{P}) \triangleq \boldsymbol{\lambda} S\bullet\{\langle\mathcal{P},\ \mathcal{Q}\rangle \mid \mathsf{P}(S)\mathcal{P} \subseteq \mathcal{Q}\}$

$$\langle\mathbb{L}^\sharp \to \wp(\mathbb{L}^\sharp) \xrightarrow{\nearrow} \wp(\mathbb{L}^\sharp),\ \subseteq\rangle \underset{\overset{\blacktriangle}{\alpha}}{\overset{\overset{\blacktriangle}{\gamma}}{\leftrightarrows}} \langle\mathbb{L}^\sharp \to (\wp(\mathbb{L}^\sharp) \times \wp(\mathbb{L}^\sharp)),\ \subseteq\rangle \tag{50}$$

where $\overline{\mathsf{L}}^{\sharp}(S) = \mathring{\alpha}(\mathsf{Post}^{\sharp})S$.

Defining the upper and lower logic triples

$$\begin{array}{lllllll}
\overline{\{\!\!|}\,\mathcal{P}\,\overline{|\!\!\}}\;\mathsf{S}\;\overline{\{\!\!|}\,\mathcal{Q}\,\overline{|\!\!\}} & \triangleq & \langle\mathcal{P},\ \mathcal{Q}\rangle \in \overline{\mathsf{L}}^{\sharp}[\![\mathsf{S}]\!]^{\sharp} & = & \mathsf{Post}^{\sharp}[\![\mathsf{S}]\!]^{\sharp}\mathcal{P} \subseteq \mathcal{Q} & = & \forall P \in \mathcal{P}\,.\ \mathsf{post}^{\sharp}[\![\mathsf{S}]\!]^{\sharp}P \in \mathcal{Q} \\
\underline{\{\!\!|}\,\mathcal{P}\,\underline{|\!\!\}}\;\mathsf{S}\;\underline{\{\!\!|}\,\mathcal{Q}\,\underline{|\!\!\}} & \triangleq & \langle\mathcal{P},\ \mathcal{Q}\rangle \in \underline{\mathsf{L}}^{\sharp}[\![\mathsf{S}]\!]^{\sharp} & = & \mathcal{Q} \subseteq \mathsf{Post}^{\sharp}[\![\mathsf{S}]\!]^{\sharp}\mathcal{P} & = & \forall Q \in \mathcal{Q}\,.\ \exists P \in \mathcal{P}\,.\ \mathsf{post}^{\sharp}[\![\mathsf{S}]\!]^{\sharp}P = Q
\end{array} \tag{51}$$

(where for symmetry, we can write $\overline{\{\!\!|}\,\mathcal{P}\,\overline{|\!\!\}}\;\mathsf{S}\;\overline{\{\!\!|}\,\mathcal{Q}\,\overline{|\!\!\}} \triangleq \forall P \in \mathcal{P}\ .\ \exists Q \in \mathcal{Q}\ .\ \mathsf{post}^{\sharp}(S)P = Q$.) We get generalizations of Hoare logic [55] and incorrectness logic [32, 75] from execution to semantic properties.

Example 7.1 (Finitary powerset nondeterministic calculational domain). In [29, 30], the relational semantics is identical to that of [5] in example 5.7 but for a nondeterministic language. Nontermination is abstracted away. The extended semantics [29, 30, Definition 4] is $\mathsf{post}^{\sharp}(S)P = \{\langle\sigma,\ \sigma''\rangle \mid \exists\sigma' \in \Sigma\ .\ \langle\sigma,\ \sigma'\rangle \in P \wedge \langle\sigma',\ \sigma''\rangle \in S\}$, the same as in example 5.7. Hyper-triples $\overline{\{\!\!|}\,\mathcal{P}\,\overline{|\!\!\}}\;\mathsf{S}\;\overline{\{\!\!|}\,\mathcal{Q}\,\overline{|\!\!\}}$ are defined in [29, 30, Definition 5] to be the powerset instance of (51), the same instance used in example 5.7. ■

The upper and lower abstract logics can always be expressed in terms of singleton (although the equivalent formula is not part of the logic).

Lemma 7.2. Ⓐ

$$\begin{array}{lll}
\overline{\{\!\!|}\,\mathcal{P}\,\overline{|\!\!\}}\;\mathsf{S}\;\overline{\{\!\!|}\,\mathcal{Q}\,\overline{|\!\!\}} & \Leftrightarrow & \forall P \in \mathcal{P}\,.\ \exists Q \in \mathcal{Q}\,.\ \overline{\{\!\!|}\,\{P\}\,\overline{|\!\!\}}\;\mathsf{S}\;\overline{\{\!\!|}\,\{Q\}\,\overline{|\!\!\}} \qquad \text{(a)} \\
\underline{\{\!\!|}\,\mathcal{P}\,\underline{|\!\!\}}\;\mathsf{S}\;\underline{\{\!\!|}\,\mathcal{Q}\,\underline{|\!\!\}} & \Leftrightarrow & \forall Q \in \mathcal{Q}\,.\ \exists P \in \mathcal{P}\,.\ \underline{\{\!\!|}\,\{P\}\,\underline{|\!\!\}}\;\mathsf{S}\;\underline{\{\!\!|}\,\{Q\}\,\underline{|\!\!\}} \qquad \text{(b)}
\end{array}$$

Corollary 7.3. Ⓐ $(\exists P \in \mathcal{P}\,.\ \underline{\{\!\!|}\,\{P\}\,\underline{|\!\!\}}\;\mathsf{S}\;\underline{\{\!\!|}\,\{Q\}\,\underline{|\!\!\}}) \Leftrightarrow \underline{\{\!\!|}\,\mathcal{P}\,\underline{|\!\!\}}\;\mathsf{S}\;\underline{\{\!\!|}\,\{Q\}\,\underline{|\!\!\}}$.

For singletons, the two logics are equivalent.

Lemma 7.4. Ⓐ *For all $P, Q \in \mathbb{L}^{\sharp}$, $\overline{\{\!\!|}\,\{P\}\,\overline{|\!\!\}}\;\mathsf{S}\;\overline{\{\!\!|}\,\{Q\}\,\overline{|\!\!\}} = \underline{\{\!\!|}\,\{P\}\,\underline{|\!\!\}}\;\mathsf{S}\;\underline{\{\!\!|}\,\{Q\}\,\underline{|\!\!\}}$.*

7.2 The Proof Systems of the Upper and Lower Abstract Logics

Since the definition (38)—(47) of $\mathsf{Post}^{\sharp}[\![\mathsf{S}]\!]^{\sharp}$ by a Hilbert proof system is structural, it is the same for the logics. Following [21], this is obtained by Aczel correspondance between set-based fixpoints and proof rules [2]. For iteration fixpoint, over-approximation is provided by [21, th. II.3.4] generalizing Park fixpoint induction [77], whereas under-approximation can be handled by [21, th. II.3.6] generalizing Scott's induction or [21, th. II.3.8] generalizing Turing/Floyd variant functions.

Therefore the sound and complete Hilbert deductive system can be designed calculationally to be the following (where $\mathcal{P}, \mathcal{Q} \in \wp(\mathbb{L}^{\sharp})$, $\bowtie$ and $\{\!\!|\,\mathcal{P}\,|\!\!\}\;\mathsf{S}\;\{\!\!|\,\mathcal{Q}\,|\!\!\}$ are respectively $\subseteq$ and $\overline{\{\!\!|}\,\mathcal{P}\,\overline{|\!\!\}}\;\mathsf{S}\;\overline{\{\!\!|}\,\mathcal{Q}\,\overline{|\!\!\}}$ for the Upper Abstract Logic and $\supseteq$ and $\underline{\{\!\!|}\,\mathcal{P}\,\underline{|\!\!\}}\;\mathsf{S}\;\underline{\{\!\!|}\,\mathcal{Q}\,\underline{|\!\!\}}$ for the Lower Abstract Logic and the calculational design proving theorem 7.5 follows in sect. 7.3).

Theorem 7.5 (Upper abstract logic proof system). *If $\mathbb{D}^{\sharp}$ is a well-defined increasing and decreasing chain-complete join semilattice with right upper continuous sequential composition $\mathbin{ⲻ}^{\sharp}$ then*

$$\frac{\{\langle e : P_+ \mathbin{\text{⨾}^\sharp} \mathsf{assign}^\sharp[\![\mathsf{x},\mathsf{A}]\!],\ \bot : P_\infty,\ br : P_{br}\rangle \mid P \in \mathcal{P}\} \bowtie \mathcal{Q}}{\{\!\!|\,\mathcal{P}\,|\!\!\}\ \mathtt{x\ =\ A}\ \{\!\!|\,\mathcal{Q}\,|\!\!\}} \tag{52}$$

$$\frac{\{\langle e : P_+ \mathbin{\text{⨾}^\sharp} \mathsf{rassign}^\sharp[\![\mathsf{x},a,b]\!],\ \bot : P_\infty,\ br : P_{br}\rangle \mid P \in \mathcal{P}\} \bowtie \mathcal{Q}}{\{\!\!|\,\mathcal{P}\,|\!\!\}\ \mathtt{x\ =\ [}a\mathtt{,}\ b\mathtt{]}\ \{\!\!|\,\mathcal{Q}\,|\!\!\}} \tag{53}$$

$$\frac{\{\langle e : P_+ \mathbin{\text{⨾}^\sharp} \mathsf{skip}^\sharp,\ \bot : P_\infty,\ br : P_{br}\rangle \mid P \in \mathcal{P}\} \bowtie \mathcal{Q}}{\{\!\!|\,\mathcal{P}\,|\!\!\}\ \mathtt{skip}\ \{\!\!|\,\mathcal{Q}\,|\!\!\}} \tag{54}$$

$$\frac{\{\langle e : P_+ \mathbin{\text{⨾}^\sharp} \mathsf{test}^\sharp[\![\mathsf{B}]\!],\ \bot : P_\infty,\ br : P_{br}\rangle \mid P \in \mathcal{P}\} \bowtie \mathcal{Q}}{\{\!\!|\,\mathcal{P}\,|\!\!\}\ \mathtt{B}\ \{\!\!|\,\mathcal{Q}\,|\!\!\}} \tag{55}$$

$$\frac{\{\langle e : \bot_+^\sharp,\ \bot : P_\infty,\ br : P_{br} \sqcup_+^\sharp (P_e \mathbin{\text{⨾}^\sharp} \mathsf{break}^\sharp)\rangle \mid P \in \mathcal{P}\} \bowtie \mathcal{Q}}{\{\!\!|\,\mathcal{P}\,|\!\!\}\ \mathtt{break}\ \{\!\!|\,\mathcal{Q}\,|\!\!\}} \tag{56}$$

$$\frac{\{\!\!|\,\mathcal{P}\,|\!\!\}\ \mathtt{S}_1\ \{\!\!|\,\mathcal{Q}\,|\!\!\},\quad \{\!\!|\,\mathcal{Q}\,|\!\!\}\ \mathtt{S}_2\ \{\!\!|\,\mathcal{R}\,|\!\!\}}{\{\!\!|\,\mathcal{P}\,|\!\!\}\ \mathtt{S}_1\mathtt{;}\mathtt{S}_2\ \{\!\!|\,\mathcal{R}\,|\!\!\}} \tag{57}$$

$$\frac{\forall P \in \mathcal{P},\quad \big(\overline{\{\!\!|}\,\{P\}\,\overline{|\!\!\}}\ \mathtt{B;S}_1\ \overline{\{\!\!|}\,\{Q_1\}\,\overline{|\!\!\}} \wedge \overline{\{\!\!|}\,\{P\}\,\overline{|\!\!\}}\ \neg\mathtt{B;S}_2\ \overline{\{\!\!|}\,\{Q_2\}\,\overline{|\!\!\}}\big) \Rightarrow (Q_1 \sqcup^\sharp Q_2 \in \mathcal{Q})}{\overline{\{\!\!|}\,\mathcal{P}\,\overline{|\!\!\}}\ \mathtt{if\ (B)\ S_1\ else\ S_2}\ \overline{\{\!\!|}\,\mathcal{Q}\,\overline{|\!\!\}}} \tag{58}$$

$$\frac{\begin{array}{c}\big(P_e = \mathsf{lfp}^{\sqsubseteq_+^\sharp}\,\vec{F}_{pe}^\sharp(P') \wedge\ \overline{\{\!\!|}\,\{P_e\}\,\overline{|\!\!\}}\ \neg\mathtt{B}\ \overline{\{\!\!|}\,\{Q_e\}\,\overline{|\!\!\}} \wedge\ \overline{\{\!\!|}\,\{P_e\}\,\overline{|\!\!\}}\ \mathtt{B;S}\ \overline{\{\!\!|}\,\{Q_b\}\,\overline{|\!\!\}} \wedge \\ \overline{\{\!\!|}\,\{P_e\}\,\overline{|\!\!\}}\ \mathtt{B;S}\ \overline{\{\!\!|}\,\{Q_{\bot\ell}\}\,\overline{|\!\!\}} \wedge\ Q_{\bot b} = \mathsf{gfp}^{\sqsubseteq_\infty^\sharp}\,F_{p\bot}^\sharp \wedge\ P' \in \mathcal{P}\big)\ \Rightarrow \\ \big(\langle e : Q_e \sqcup_e^\sharp Q_b,\ \bot : Q_{\bot\ell} \sqcup_\infty^\sharp Q_{\bot b},\ br : P_{br}\rangle \in \mathcal{Q}\big)\end{array}}{\overline{\{\!\!|}\,\mathcal{I}\,\overline{|\!\!\}}\ \mathtt{while\ (B)\ S}\ \overline{\{\!\!|}\,\mathcal{Q}\,\overline{|\!\!\}}} \tag{59}$$

is sound and complete.

Remarkably in (58) and (59), we have to consider all possible over approximations, and in (59) P_e and $Q_{\bot b}$ must be exact fixpoints. This is because, for completeness and in full generality, hyperlogics cannot make any approximation of the program semantics defined by $\mathsf{post}^\sharp$ in (31) hence prohibiting approximations in (51).

Notice that no consequence rule is required for completeness, although they are sound Ⓐ.

$$\frac{\mathcal{P} \subseteq \mathcal{P}',\quad \overline{\{\!\!|}\,\mathcal{P}'\,\overline{|\!\!\}}\ \mathtt{S}\ \overline{\{\!\!|}\,\mathcal{Q}'\,\overline{|\!\!\}},\quad \mathcal{Q}' \subseteq \mathcal{Q}}{\overline{\{\!\!|}\,\mathcal{P}\,\overline{|\!\!\}}\ \mathtt{S}\ \overline{\{\!\!|}\,\mathcal{Q}\,\overline{|\!\!\}}} \qquad \frac{\mathcal{P}' \subseteq \mathcal{P},\quad \underline{\{\!\!|}\,\mathcal{P}'\,\underline{|\!\!\}}\ \mathtt{S}\ \underline{\{\!\!|}\,\mathcal{Q}'\,\underline{|\!\!\}},\quad \mathcal{Q} \subseteq \mathcal{Q}'}{\underline{\{\!\!|}\,\mathcal{P}\,\underline{|\!\!\}}\ \mathtt{S}\ \underline{\{\!\!|}\,\mathcal{Q}\,\underline{|\!\!\}}} \tag{60}$$

Example 7.6 (Choice). Let us define the choice $\mathtt{S}_1 + \mathtt{S}_2 \triangleq$ `c = [0,1]; if (c)` $\mathtt{S}_1$ `else` $\mathtt{S}_2$ where auxiliary variable c does not appear in $\mathtt{S}_1$ nor in $\mathtt{S}_2$. The proof rule can be derived as follows

$$\begin{array}{ll} & \overline{\{\!\!|}\,\mathcal{P}\,\overline{|\!\!\}}\ \mathtt{S}_1 + \mathtt{S}_2\ \overline{\{\!\!|}\,\mathcal{Q}\,\overline{|\!\!\}} \\ \Leftrightarrow & \overline{\{\!\!|}\,\mathcal{P}\,\overline{|\!\!\}}\ \mathtt{c\ =\ [0,1];\ if\ (c)\ S_1\ else\ S_2}\ \overline{\{\!\!|}\,\mathcal{Q}\,\overline{|\!\!\}} \hfill \text{(def. choice +)} \\ \Leftrightarrow & \exists\mathcal{R}\,.\,\overline{\{\!\!|}\,\mathcal{P}\,\overline{|\!\!\}}\ \mathtt{c\ =\ [0,1]}\ \overline{\{\!\!|}\,\mathcal{R}\,\overline{|\!\!\}} \wedge \overline{\{\!\!|}\,\mathcal{R}\,\overline{|\!\!\}}\ \mathtt{if\ (c)\ S_1\ else\ S_2}\ \overline{\{\!\!|}\,\mathcal{Q}\,\overline{|\!\!\}} \hfill \text{(sequential composition (57))} \\ \Leftrightarrow & \exists\mathcal{R}\,.\,\{P \mathbin{\text{⨾}^\sharp} \mathsf{rassign}^\sharp[\![\mathsf{c},0,1]\!] \mid P \in \mathcal{P}\} \subseteq \mathcal{R} \wedge \overline{\{\!\!|}\,\mathcal{R}\,\overline{|\!\!\}}\ \mathtt{if\ (c)\ S_1\ else\ S_2}\ \overline{\{\!\!|}\,\mathcal{Q}\,\overline{|\!\!\}} \hfill \text{((53))} \\ \Leftrightarrow & \overline{\{\!\!|}\,\{P \mathbin{\text{⨾}^\sharp} \mathsf{rassign}^\sharp[\![\mathsf{c},0,1]\!] \mid P \in \mathcal{P}\}\,\overline{|\!\!\}}\ \mathtt{if\ (c)\ S_1\ else\ S_2}\ \overline{\{\!\!|}\,\mathcal{Q}\,\overline{|\!\!\}} \\ & \hfill \text{(taking } \mathcal{R} = \{P \mathbin{\text{⨾}^\sharp} \mathsf{rassign}^\sharp[\![\mathsf{c},0,1]\!] \mid P \in \mathcal{P}\}\text{)} \\ \Leftrightarrow & \forall P \in \{P' \mathbin{\text{⨾}^\sharp} \mathsf{rassign}^\sharp[\![\mathsf{c},0,1]\!] \mid P' \in \mathcal{P}\}, Q_1, Q_2\,.\,\big(\underline{\{\!\!|}\,\{P\}\,\underline{|\!\!\}}\ \mathtt{B;S}_1\ \underline{\{\!\!|}\,\{Q_1\}\,\underline{|\!\!\}} \wedge \underline{\{\!\!|}\,\{P\}\,\underline{|\!\!\}}\ \neg\mathtt{B;S}_2\ \underline{\{\!\!|}\,\{Q_2\}\,\underline{|\!\!\}}\big) \Rightarrow \\ & (Q_1 \sqcup^\sharp Q_2 \in \mathcal{Q}) \hfill \text{((58))} \\ \Leftrightarrow & \forall P \in \mathcal{P}, Q_1, Q_2\,.\,\big(\underline{\{\!\!|}\,\{P\}\,\underline{|\!\!\}}\ \mathtt{S}_1\ \underline{\{\!\!|}\,\{Q_1\}\,\underline{|\!\!\}} \wedge \underline{\{\!\!|}\,\{P\}\,\underline{|\!\!\}}\ \mathtt{S}_2\ \underline{\{\!\!|}\,\{Q_2\}\,\underline{|\!\!\}}\big) \Rightarrow (Q_1 \sqcup^\sharp Q_2 \in \mathcal{Q}) \end{array} \tag{61}$$

$\wr$assuming states where c is assigned 0 or 1, B is true for 0 and $\neg$B is true for 1 (or conversely)$\int$

so that we get the sound and complete rule

$$\frac{\forall P \in \mathcal{P}, Q_1, Q_2 \,.\, (\underline{\{\!|}\,\{P\}\,\underline{|\!\}}\,\mathsf{S}_1\,\underline{\{\!|}\,\{Q_1\}\,\underline{|\!\}} \wedge \underline{\{\!|}\,\{P\}\,\underline{|\!\}}\,\mathsf{S}_2\,\underline{\{\!|}\,\{Q_2\}\,\underline{|\!\}}) \Rightarrow (Q_1 \sqcup^\sharp Q_2 \in \mathcal{Q})}{\overline{\{\!|}\,\mathcal{P}\,\overline{|\!\}}\,\mathsf{S}_1 + \mathsf{S}_2\,\overline{\{\!|}\,\mathcal{Q}\,\overline{|\!\}}} \tag{62}$$

Let us now consider the particular case $\mathsf{post}^\sharp(S)P = \{\langle\sigma, \sigma''\rangle \mid \exists\sigma' \in \Sigma \,.\, \langle\sigma, \sigma'\rangle \in P \wedge \langle\sigma', \sigma''\rangle \in S\}$ as in example 5.7 (but this time with unbounded nondeterminism) so that $\sqcup^\sharp$ is $\cup$ in (62). Then (62) is implied, but not conversely, by the proof rule

$$\frac{\overline{\{\!|}\,\mathcal{P}\,\overline{|\!\}}\,\mathsf{S}_1\,\overline{\{\!|}\,\mathcal{Q}_1\,\overline{|\!\}},\quad \overline{\{\!|}\,\mathcal{P}\,\overline{|\!\}}\,\mathsf{S}_2\,\overline{\{\!|}\,\mathcal{Q}_2\,\overline{|\!\}}}{\overline{\{\!|}\,\mathcal{P}\,\overline{|\!\}}\,\mathsf{S}_1 + \mathsf{S}_2\,\overline{\{\!|}\,\{Q_1 \cup Q_2 \mid Q_1 \in \mathcal{Q}_1 \wedge Q_2 \in \mathcal{Q}_2\}\,\overline{|\!\}}} \quad (\mathit{Choice})$$

of [29], which is sound but incomplete. For completeness, [29, p. 207:9] has to introduce an (*Exist*) proof rule which amounts to the case by case analysis of rule (62). ■

Example 7.7 (Finitary powerset nondeterministic calculational domain). Continuing example 7.1, the iteration rule postulated in [29, 30, Fig. 2] is (59), ignoring nontermination and breaks, and applying proposition 2.4 to reason on the fixpoint iterates. ■

7.3 Calculational Design of the Proof System of the Upper Abstract Logic

Proof of (52) — (59). The proof of soundness and completeness is by structural induction. We show the calculational design for the iteration (59). The other cases are in the appendix Ⓐ.

$\overline{\{\!|}\,\mathcal{P}\,\overline{|\!\}}\,\mathtt{while\ (B)\ S}\,\overline{\{\!|}\,\mathcal{R}\,\overline{|\!\}}$

$\Leftrightarrow$ $\mathsf{Post}^\sharp[\![\mathtt{while\ (B)\ S}]\!]^\sharp\mathcal{P} \subseteq \mathcal{R}$ $\wr$def. (51) of the logic triples$\int$

$\Leftrightarrow$ $\{\langle e : Q_e,\ \bot : Q_{\bot\ell} \sqcup^\sharp_\infty Q_{\bot b},\ br : P_{br}\rangle \mid Q_e \in \mathsf{Post}^\sharp([\![\neg\mathsf{B}]\!]^\sharp_e \sqcup^\sharp_e [\![\mathsf{B;S}]\!]^\sharp_b)(\mathsf{lfp}^{\dot{\sqsubseteq}^\sharp_+}\,\breve{\vec{F}}^\sharp_{pe}(P)) \wedge Q_{\bot\ell} \in \mathsf{Post}^\sharp([\![\mathsf{B;S}]\!]^\sharp_\bot)(\mathsf{lfp}^{\dot{\sqsubseteq}^\sharp_+}\,\breve{\vec{F}}^\sharp_{pe}(P)) \wedge \exists Q_{\bot b} \,.\, Q_{\bot b} \in \mathsf{Post}^\sharp(Q_{p\bot})\{P\} \wedge Q_{p\bot} \in \mathsf{gfp}^{\dot{\sqsubseteq}^\sharp_\infty}\,\breve{F}^\sharp_{p\bot} \wedge P \in \mathcal{P}\} \subseteq \mathcal{R}$ $\wr$(47)$\int$

$\Leftrightarrow$ $\{\langle e : Q_e, \bot : Q_{\bot\ell} \sqcup^\sharp_\infty Q_{\bot b}, br : P_{br}\rangle \mid \exists I_e \,.\, I_e \subseteq \mathsf{lfp}^{\dot{\sqsubseteq}^\sharp_+}\,\breve{\vec{F}}^\sharp_{pe}(P) \wedge Q_e \in \mathsf{Post}^\sharp([\![\neg\mathsf{B}]\!]^\sharp_e \sqcup^\sharp_e [\![\mathsf{B;S}]\!]^\sharp_b)I_e \wedge Q_{\bot\ell} \in \mathsf{Post}^\sharp([\![\mathsf{B;S}]\!]^\sharp_\bot)(I_e) \wedge \exists I_\bot \,.\, I_\bot \subseteq \mathsf{gfp}^{\dot{\sqsubseteq}^\sharp_\infty}\,\breve{F}^\sharp_{p\bot} \wedge Q_{\bot b} \in I_\bot \wedge P \in \mathcal{P}\} \subseteq \mathcal{R}$

$\wr$($\Rightarrow$) Take $I_e = \mathsf{lfp}^{\dot{\sqsubseteq}^\sharp_+}\,\breve{\vec{F}}^\sharp_{pe}(P)$, $I_\bot = \mathsf{gfp}^{\dot{\sqsubseteq}^\sharp_\infty}\,\breve{F}^\sharp_{p\bot}$, and $\subseteq$ reflexive
($\Leftarrow$) by (36), $\mathsf{Post}^\sharp(S)$ is $\subseteq$-increasing$\int$

$\Leftrightarrow$ $\{\langle e : Q_e, \bot : Q_{\bot\ell} \sqcup^\sharp_\infty Q_{\bot b}, br : P_{br}\rangle \mid \exists P_e \,.\, \{P_e\} = \mathsf{lfp}^{\dot{\sqsubseteq}^\sharp_+}\,\breve{\vec{F}}^\sharp_{pe}(P) \wedge Q_e \in \mathsf{Post}^\sharp([\![\neg\mathsf{B}]\!]^\sharp_e \sqcup^\sharp_e [\![\mathsf{B;S}]\!]^\sharp_b)\{P_e\} \wedge Q_{\bot\ell} \in \mathsf{Post}^\sharp([\![\mathsf{B;S}]\!]^\sharp_\bot)\{P_e\} \wedge \exists P_\bot \,.\, \{P_\bot\} = \mathsf{gfp}^{\dot{\sqsubseteq}^\sharp_\infty}\,\breve{F}^\sharp_{p\bot} \wedge Q_{\bot b} \in \{P_\bot\} \wedge P \in \mathcal{P}\} \subseteq \mathcal{R}$

$\wr$If I_e is empty then $\mathsf{Post}^\sharp([\![\neg\mathsf{B}]\!]^\sharp_e \sqcup^\sharp_e [\![\mathsf{B;S}]\!]^\sharp_b)I_e$ is empty by (31), contrary to $Q_e \in \mathsf{Post}^\sharp([\![\neg\mathsf{B}]\!]^\sharp_e \sqcup^\sharp_e [\![\mathsf{B;S}]\!]^\sharp_b)I_e$ proving that I_e cannot be empty. By (45), $\mathsf{lfp}^{\dot{\sqsubseteq}^\sharp_+}\,\breve{\vec{F}}^\sharp_{pe}(P)$ is a singleton, say $\{P_e\}$. For I_e to be non-empty and included in a singleton, it must be equal to that singleton so $I_e = \{P_e\}$. The reasoning is the same for $I_\bot = \{P_\bot\}$$\int$

$\Leftrightarrow$ $\{\langle e : Q_e, \bot : Q_{\bot\ell} \sqcup^\sharp_\infty Q_{\bot b}, br : P_{br}\rangle \mid \exists P_e \,.\, \{P_e\} = \mathsf{lfp}^{\dot{\sqsubseteq}^\sharp_+}\,\breve{\vec{F}}^\sharp_{pe}(P) \wedge Q_e \in \mathsf{Post}^\sharp([\![\neg\mathsf{B}]\!]^\sharp_e \sqcup^\sharp_e [\![\mathsf{B;S}]\!]^\sharp_b)\{P_e\} \wedge Q_{\bot\ell} \in \mathsf{Post}^\sharp([\![\mathsf{B;S}]\!]^\sharp_\bot)\{P_e\} \wedge \{Q_{\bot b}\} = \mathsf{gfp}^{\dot{\sqsubseteq}^\sharp_\infty}\,\breve{F}^\sharp_{p\bot} \wedge P \in \mathcal{P}\} \subseteq \mathcal{R}$

$\wr Q_{\bot b} \in \{P_\bot\}$ if and only if $Q_{\bot b} = P_{\bot b}\int$

$\Leftrightarrow$ $\{\langle e : Q_e,\ \bot : Q_{\bot\ell} \sqcup^\sharp_\infty Q_{\bot b},\ br : P_{br}\rangle \mid \exists P_e \,.\, \{P_e\} = \{\mathsf{lfp}^{\dot{\sqsubseteq}^\sharp_+}\,\vec{F}^\sharp_{pe}(P)\} \wedge Q_e \in \mathsf{Post}^\sharp([\![\neg\mathsf{B}]\!]^\sharp_e \sqcup^\sharp_e [\![\mathsf{B;S}]\!]^\sharp_b)\{P_e\} \wedge Q_{\bot\ell} \in \mathsf{Post}^\sharp([\![\mathsf{B;S}]\!]^\sharp_\bot)\{P_e\} \wedge \{Q_{\bot b}\} = \{\mathsf{gfp}^{\dot{\sqsubseteq}^\sharp_\infty}\,F^\sharp_{p\bot}\} \wedge P \in \mathcal{P}\} \subseteq \mathcal{R}$

$$\wr\text{since } \mathsf{lfp}^{\breve{\sqsubseteq}^\sharp_+}\, \breve{\vec{F}}{}^\sharp_{pe}(P) = \{\mathsf{lfp}^{\sqsubseteq^\sharp_+}\, \vec{F}^\sharp_{pe}(P)\} \text{ by (45), proposition 6.3, and } \mathsf{gfp}^{\breve{\sqsubseteq}^\sharp_\infty}(\breve{F}^\sharp_{p\bot}) = \{\mathsf{gfp}^{\sqsubseteq^\sharp_\infty}\, F^\sharp_{p\bot}\} \text{ by (29) and proposition 6.3}\wr$$

$$\Leftrightarrow \{\langle e : Q_e, \bot : Q_{\bot\ell} \sqcup^\sharp_\infty Q_{\bot b}, br : P_{br}\rangle \mid \exists P_e \,.\, P_e = \mathsf{lfp}^{\sqsubseteq^\sharp_+}\, \vec{F}^\sharp_{pe}(P) \wedge Q_e \in \mathsf{Post}^\sharp(\llbracket\neg\mathsf{B}\rrbracket^\sharp_e \sqcup^\sharp_e \llbracket\mathsf{B};\mathsf{S}\rrbracket^\sharp_b)\{P_e\} \wedge Q_{\bot\ell} \in \mathsf{Post}^\sharp(\llbracket\mathsf{B};\mathsf{S}\rrbracket^\sharp_\bot)\{P_e\} \wedge Q_{\bot b} = \mathsf{gfp}^{\sqsubseteq^\sharp_\infty}\, F^\sharp_{p\bot} \wedge P \in \mathcal{P}\} \subseteq \mathcal{R} \qquad \wr\text{def. set equality}\wr$$

$$\Leftrightarrow \{\langle e : Q_e, \bot : Q_{\bot\ell} \sqcup^\sharp_\infty Q_{\bot b}, br : P_{br}\rangle \mid \exists P_e \,.\, P_e = \mathsf{lfp}^{\sqsubseteq^\sharp_+}\, \vec{F}^\sharp_{pe}(P) \wedge \{Q_e\} \subseteq \mathsf{Post}^\sharp(\llbracket\neg\mathsf{B}\rrbracket^\sharp_e \sqcup^\sharp_e \llbracket\mathsf{B};\mathsf{S}\rrbracket^\sharp_b)\{P_e\} \wedge \{Q_{\bot\ell}\} \subseteq \mathsf{Post}^\sharp(\llbracket\mathsf{B};\mathsf{S}\rrbracket^\sharp_\bot)\{P_e\} \wedge Q_{\bot b} = \mathsf{gfp}^{\sqsubseteq^\sharp_\infty}\, F^\sharp_{p\bot} \wedge P \in \mathcal{P}\} \subseteq \mathcal{R} \qquad \wr\text{def. } \in \text{ and } \subseteq\wr$$

$$\Leftrightarrow \{\langle e : Q_e, \bot : Q_{\bot\ell} \sqcup^\sharp_\infty Q_{\bot b}, br : P_{br}\rangle \mid \exists P_e \,.\, P_e = \mathsf{lfp}^{\sqsubseteq^\sharp_+}\, \vec{F}^\sharp_{pe}(P) \wedge \{Q_e\} \subseteq \mathsf{Post}^\sharp(\llbracket\neg\mathsf{B}\rrbracket^\sharp_e \sqcup^\sharp_e \llbracket\mathsf{B};\mathsf{S}\rrbracket^\sharp_b)\{P_e\} \wedge \underline{\{\!\!\{}\,\{P_e\}\,\underline{\}\!\!\}}\,\mathsf{B};\mathsf{S}\,\underline{\{\!\!\{}\,\{Q_{\bot\ell}\}\,\underline{\}\!\!\}} \wedge Q_{\bot b} = \mathsf{gfp}^{\sqsubseteq^\sharp_\infty}\, F^\sharp_{p\bot} \wedge P \in \mathcal{P}\} \subseteq \mathcal{R}$$
$$\wr\text{def. (51) of } \underline{\{\!\!\{}\,\mathcal{P}\,\underline{\}\!\!\}}\,\mathsf{S}\,\underline{\{\!\!\{}\,\mathcal{Q}\,\underline{\}\!\!\}} \triangleq (\mathcal{Q} \subseteq \mathsf{Post}^\sharp\llbracket\mathsf{S}\rrbracket^\sharp\mathcal{P})\wr$$

$$\Leftrightarrow \{\langle e : Q_e, \bot : Q_{\bot\ell} \sqcup^\sharp_\infty Q_{\bot b}, br : P_{br}\rangle \mid \exists P_e \,.\, P_e = \mathsf{lfp}^{\sqsubseteq^\sharp_+}\, \vec{F}^\sharp_{pe}(P) \wedge \{Q_e\} \subseteq \mathsf{Post}^\sharp(\llbracket\neg\mathsf{B}\rrbracket^\sharp_e)\{P_e\} \sqcup^\sharp_e \mathsf{Post}^\sharp(\llbracket\mathsf{B};\mathsf{S}\rrbracket^\sharp_b)\{P_e\} \wedge \underline{\{\!\!\{}\,\{P_e\}\,\underline{\}\!\!\}}\,\mathsf{B};\mathsf{S}\,\underline{\{\!\!\{}\,\{Q_{\bot\ell}\}\,\underline{\}\!\!\}} \wedge Q_{\bot b} = \mathsf{gfp}^{\sqsubseteq^\sharp_\infty}\, F^\sharp_{p\bot} \wedge P \in \mathcal{P}\} \subseteq \mathcal{R} \qquad \wr(37)\wr$$

$$\Leftrightarrow \{\langle e : Q'_e, \bot : Q_{\bot\ell} \sqcup^\sharp_\infty Q_{\bot b}, br : P_{br}\rangle \mid \exists P_e \,.\, P_e = \mathsf{lfp}^{\sqsubseteq^\sharp_+}\, \vec{F}^\sharp_{pe}(P) \wedge \{Q'_e\} \subseteq \{Q_e \sqcup^\sharp_e Q_b \mid \{Q_e\} \subseteq \mathsf{Post}^\sharp(\llbracket\neg\mathsf{B}\rrbracket^\sharp_e)\{P\} \wedge \{Q_b\} \subseteq \mathsf{Post}^\sharp(\llbracket\mathsf{B};\mathsf{S}\rrbracket^\sharp_b)\{P\} \wedge P \in \{P_e\}\} \wedge \underline{\{\!\!\{}\,\{P_e\}\,\underline{\}\!\!\}}\,\mathsf{B};\mathsf{S}\,\underline{\{\!\!\{}\,\{Q_{\bot\ell}\}\,\underline{\}\!\!\}} \wedge Q_{\bot b} = \mathsf{gfp}^{\sqsubseteq^\sharp_\infty}\, F^\sharp_{p\bot} \wedge P \in \mathcal{P}\} \subseteq \mathcal{R} \qquad \wr\text{def. (37) of } \sqcup^\sharp_e\wr$$

$$\Leftrightarrow \{\langle e : Q'_e, \bot : Q_{\bot\ell} \sqcup^\sharp_\infty Q_{\bot b}, br : P_{br}\rangle \mid \exists P_e \,.\, P_e = \mathsf{lfp}^{\sqsubseteq^\sharp_+}\, \vec{F}^\sharp_{pe}(P') \wedge \exists Q_e, Q_b, P \,.\, Q'_e = Q_e \sqcup^\sharp_e Q_b \wedge \{Q_e\} \subseteq \mathsf{Post}^\sharp(\llbracket\neg\mathsf{B}\rrbracket^\sharp_e)\{P\} \wedge \{Q_b\} \subseteq \mathsf{Post}^\sharp(\llbracket\mathsf{B};\mathsf{S}\rrbracket^\sharp_b)\{P\} \wedge P \in \{P_e\} \wedge \underline{\{\!\!\{}\,\{P_e\}\,\underline{\}\!\!\}}\,\mathsf{B};\mathsf{S}\,\underline{\{\!\!\{}\,\{Q_{\bot\ell}\}\,\underline{\}\!\!\}} \wedge Q_{\bot b} = \mathsf{gfp}^{\sqsubseteq^\sharp_\infty}\, F^\sharp_{p\bot} \wedge P' \in \mathcal{P}\} \subseteq \mathcal{R} \qquad \wr\text{def. singleton and } \subseteq\text{, renaming}\wr$$

$$\Leftrightarrow \{\langle e : Q_e \sqcup^\sharp_e Q_b, \bot : Q_{\bot\ell} \sqcup^\sharp_\infty Q_{\bot b}, br : P_{br}\rangle \mid \exists P_e \,.\, P_e = \mathsf{lfp}^{\sqsubseteq^\sharp_+}\, \vec{F}^\sharp_{pe}(P') \wedge \exists P \,.\, \{Q_e\} \subseteq \mathsf{Post}^\sharp(\llbracket\neg\mathsf{B}\rrbracket^\sharp_e)\{P\} \wedge \{Q_b\} \subseteq \mathsf{Post}^\sharp(\llbracket\mathsf{B};\mathsf{S}\rrbracket^\sharp_b)\{P\} \wedge P \in \{P_e\} \wedge \underline{\{\!\!\{}\,\{P_e\}\,\underline{\}\!\!\}}\,\mathsf{B};\mathsf{S}\,\underline{\{\!\!\{}\,\{Q_{\bot\ell}\}\,\underline{\}\!\!\}} \wedge Q_{\bot b} = \mathsf{gfp}^{\sqsubseteq^\sharp_\infty}\, F^\sharp_{p\bot} \wedge P' \in \mathcal{P}\} \subseteq \mathcal{R} \qquad \wr\text{replacing } Q'_e \text{ by its value}\wr$$

$$\Leftrightarrow \{\langle e : Q_e \sqcup^\sharp_e Q_b, \bot : Q_{\bot\ell} \sqcup^\sharp_\infty Q_{\bot b}, br : P_{br}\rangle \mid \exists P_e \,.\, P_e = \mathsf{lfp}^{\sqsubseteq^\sharp_+}\, \vec{F}^\sharp_{pe}(P') \wedge \{Q_e\} \subseteq \mathsf{Post}^\sharp(\llbracket\neg\mathsf{B}\rrbracket^\sharp_e)\{P_e\} \wedge \{Q_b\} \subseteq \mathsf{Post}^\sharp(\llbracket\mathsf{B};\mathsf{S}\rrbracket^\sharp_b)\{P_e\} \wedge \underline{\{\!\!\{}\,\{P_e\}\,\underline{\}\!\!\}}\,\mathsf{B};\mathsf{S}\,\underline{\{\!\!\{}\,\{Q_{\bot\ell}\}\,\underline{\}\!\!\}} \wedge Q_{\bot b} = \mathsf{gfp}^{\sqsubseteq^\sharp_\infty}\, F^\sharp_{p\bot} \wedge P' \in \mathcal{P}\} \subseteq \mathcal{R}$$
$$\wr\text{corollary 7.3}\wr$$

$$\Leftrightarrow \{\langle e : Q_e \sqcup^\sharp_e Q_b, \bot : Q_{\bot\ell} \sqcup^\sharp_\infty Q_{\bot b}, br : P_{br}\rangle \mid \exists P_e \,.\, P_e = \mathsf{lfp}^{\sqsubseteq^\sharp_+}\, \vec{F}^\sharp_{pe}(P') \wedge \underline{\{\!\!\{}\,\{P_e\}\,\underline{\}\!\!\}}\,\neg\mathsf{B}\,\underline{\{\!\!\{}\,\{Q_e\}\,\underline{\}\!\!\}} \wedge \underline{\{\!\!\{}\,\{P_e\}\,\underline{\}\!\!\}}\,\mathsf{B};\mathsf{S}\,\underline{\{\!\!\{}\,\{Q_b\}\,\underline{\}\!\!\}} \wedge \underline{\{\!\!\{}\,\{P_e\}\,\underline{\}\!\!\}}\,\mathsf{B};\mathsf{S}\,\underline{\{\!\!\{}\,\{Q_{\bot\ell}\}\,\underline{\}\!\!\}} \wedge Q_{\bot b} = \mathsf{gfp}^{\sqsubseteq^\sharp_\infty}\, F^\sharp_{p\bot} \wedge P' \in \mathcal{P}\} \subseteq \mathcal{R}$$
$$\wr\text{def. (51) of } \underline{\{\!\!\{}\,\mathcal{P}\,\underline{\}\!\!\}}\,\mathsf{S}\,\underline{\{\!\!\{}\,\mathcal{Q}\,\underline{\}\!\!\}} \triangleq (\mathcal{Q} \subseteq \mathsf{Post}^\sharp\llbracket\mathsf{S}\rrbracket^\sharp\mathcal{P})\wr$$

$$\Leftrightarrow (P_e = \mathsf{lfp}^{\sqsubseteq^\sharp_+}\, \vec{F}^\sharp_{pe}(P') \wedge \underline{\{\!\!\{}\,\{P_e\}\,\underline{\}\!\!\}}\,\neg\mathsf{B}\,\underline{\{\!\!\{}\,\{Q_e\}\,\underline{\}\!\!\}} \wedge \underline{\{\!\!\{}\,\{P_e\}\,\underline{\}\!\!\}}\,\mathsf{B};\mathsf{S}\,\underline{\{\!\!\{}\,\{Q_b\}\,\underline{\}\!\!\}} \wedge \underline{\{\!\!\{}\,\{P_e\}\,\underline{\}\!\!\}}\,\mathsf{B};\mathsf{S}\,\underline{\{\!\!\{}\,\{Q_{\bot\ell}\}\,\underline{\}\!\!\}} \wedge Q_{\bot b} = \mathsf{gfp}^{\sqsubseteq^\sharp_\infty}\, F^\sharp_{p\bot} \wedge P' \in \mathcal{P}) \Rightarrow \langle e : Q_e \sqcup^\sharp_e Q_b, \bot : Q_{\bot\ell} \sqcup^\sharp_\infty Q_{\bot b}, br : P_{br}\rangle \in \mathcal{R} \qquad \wr\text{def. } \subseteq\wr$$

$$\Leftrightarrow (P_e = \mathsf{lfp}^{\sqsubseteq^\sharp_+}\, \vec{F}^\sharp_{pe}(P') \wedge \overline{\{\!\!\{}\,\{P_e\}\,\overline{\}\!\!\}}\,\neg\mathsf{B}\,\overline{\{\!\!\{}\,\{Q_e\}\,\overline{\}\!\!\}} \wedge \overline{\{\!\!\{}\,\{P_e\}\,\overline{\}\!\!\}}\,\mathsf{B};\mathsf{S}\,\overline{\{\!\!\{}\,\{Q_b\}\,\overline{\}\!\!\}} \wedge \overline{\{\!\!\{}\,\{P_e\}\,\overline{\}\!\!\}}\,\mathsf{B};\mathsf{S}\,\overline{\{\!\!\{}\,\{Q_{\bot\ell}\}\,\overline{\}\!\!\}} \wedge Q_{\bot b} = \mathsf{gfp}^{\sqsubseteq^\sharp_\infty}\, F^\sharp_{p\bot} \wedge P' \in \mathcal{P}) \Rightarrow \langle e : Q_e \sqcup^\sharp_e Q_b, \bot : Q_{\bot\ell} \sqcup^\sharp_\infty Q_{\bot b}, br : P_{br}\rangle \in \mathcal{R} \qquad \wr\text{lemma 7.4}\wr \qquad \square$$

Propositions 2.3 and 2.4 can be used to characterize the fixpoints of increasing functions in (59).

7.4 Calculational Design of the Proof System of the Lower Abstract Logic

Apart from (52)—(57), the sound and complete induction rules for the lower abstract logic are constructed by calculational design as follows.

Theorem 7.8 (Lower abstract logic proof system). Ⓐ *If $\mathbb{D}^\sharp$ is a well-defined increasing and decreasing chain-complete join semilattice with right upper continuous sequential composition $\mathbin{⨾}^\sharp$ then*

$$\frac{\forall Q \in \mathcal{Q}\,.\,\exists P \in \mathcal{P}, Q_1, Q_2\,.\,\underline{\{\!|}\,\{P\}\,\underline{|\!\}}\,\mathsf{B};\mathsf{S}_1\,\underline{\{\!|}\,\{Q_1\}\,\underline{|\!\}} \wedge \underline{\{\!|}\,\{P\}\,\underline{|\!\}}\,\neg\mathsf{B};\mathsf{S}_2\,\underline{\{\!|}\,\{Q_2\}\,\underline{|\!\}} \wedge Q = Q_1 \sqcup^\sharp Q_2}{\underline{\{\!|}\,\mathcal{P}\,\underline{|\!\}}\,\texttt{if (B) S}_1\texttt{ else S}_2\,\underline{\{\!|}\,\mathcal{Q}\,\underline{|\!\}}} \tag{63}$$

$$\frac{\begin{array}{c}\forall\langle e : Q'_e,\ \bot : Q'_\bot,\ br : Q'_{br}\rangle \in \mathcal{Q}\,.\,\exists Q_e, Q_b, Q_{\bot\ell}, Q_{\bot b}, P_e\,.\,Q'_e = Q_e \sqcup^\sharp_e Q_b \wedge Q'_\bot = \\ Q_{\bot\ell} \sqcup^\sharp_\infty Q_{\bot b} \wedge Q'_{br} = P'_{br} \wedge P_e = \mathsf{lfp}^{\sqsubseteq^\sharp_+}\,\vec{F}^\sharp_{pe}(P') \wedge \underline{\{\!|}\,\{P_e\}\,\underline{|\!\}}\,\neg\mathsf{B}\,\underline{\{\!|}\,\{Q_e\}\,\underline{|\!\}} \wedge \\ \underline{\{\!|}\,\{P_e\}\,\underline{|\!\}}\,\mathsf{B};\mathsf{S}\,\underline{\{\!|}\,\{Q_b\}\,\underline{|\!\}} \wedge \underline{\{\!|}\,\{P_e\}\,\underline{|\!\}}\,\mathsf{B};\mathsf{S}\,\underline{\{\!|}\,\{Q_{\bot\ell}\}\,\underline{|\!\}} \wedge Q_{\bot b} = \mathsf{gfp}^{\sqsubseteq^\sharp_\infty}\,F^\sharp_{p\bot} \wedge P' \in \mathcal{P}\end{array}}{\underline{\{\!|}\,\mathcal{P}\,\underline{|\!\}}\,\texttt{while (B) S}\,\underline{\{\!|}\,\mathcal{Q}\,\underline{|\!\}}} \tag{64}$$

Part II: Abstraction of Semantics, Execution Properties, Semantic (Hyper) Properties, Calculi, and Logics

Since hyperlogics deal with properties of semantics, there are four levels at which an abstraction can be applied.

(1) The first level is that of the program semantics considered in appendix sect. 8 and illustrated by the relational semantics in example 8.4 abstracting the trace semantics of sect. B. This abstraction is common in transformational logics [21] such as Hoare logic [55] but also in hyperlogics [29, 30];
(2) The second level is that of program properties of sect. 5.1;
(3) The third level is that of program hyperproperties of sect. 6;
(4) The fourth level is that of the abstract logics of sect. 7. (65)

Because logics are required to be sound and complete, abstractions should be exact so that any proof of abstract properties in the concrete should be doable in the abstract. This relies on Galois retractions in sect. 2.5. The main result is that the abstraction of a logic of semantic (hyper) properties of sect. 7 is a a logic of semantic (hyper) properties.

8 Abstraction of the Abstract Semantics

We show that the abstraction of an instance of the abstract semantics is itself an instance of the abstract semantics.

Definition 8.1 (Semantic abstraction). We say that $\bar{\mathbb{D}}^\sharp \triangleq \langle\bar{\mathbb{D}}^\sharp_+,\ \bar{\mathbb{D}}^\sharp_\infty\rangle$ is an exact (respectively approximate) abstraction of an abstract domain $\mathbb{D}^\sharp \triangleq \langle\mathbb{D}^\sharp_+,\ \mathbb{D}^\sharp_\infty\rangle$ if and only if

A. There exists a Galois retraction $\langle\mathbb{L}^\sharp_+,\ \sqsubseteq^\sharp_+\rangle \xrightleftharpoons[\alpha_+]{\gamma_+} \langle\bar{\mathbb{L}}^\sharp_+,\ \bar{\sqsubseteq}^\sharp_+\rangle$;

B. $\alpha_+(\mathsf{init}^\sharp) = \overline{\mathsf{init}}^\sharp$, $\alpha_+ \circ \mathsf{assign}^\sharp[\![\mathsf{x}, \mathsf{A}]\!] = \overline{\mathsf{assign}}^\sharp[\![\mathsf{x}, \mathsf{A}]\!] \circ \alpha_+$, $\alpha_+ \circ \mathsf{rassign}^\sharp[\![\mathsf{x}, a, b]\!] = \overline{\mathsf{rassign}}^\sharp[\![\mathsf{x}, a, b]\!] \circ \alpha_+$, $\alpha_+ \circ \mathsf{test}^\sharp[\![\mathsf{B}]\!] = \overline{\mathsf{test}}^\sharp[\![\mathsf{B}]\!] \circ \alpha_+$, $\alpha_+(\mathsf{break}^\sharp) = \overline{\mathsf{break}}^\sharp$, and $\alpha_+(\mathsf{skip}^\sharp) = \overline{\mathsf{skip}}^\sharp$;

C. There exists a Galois retraction $\langle\mathbb{L}^\sharp_\infty,\ \sqsupseteq^\sharp_\infty\rangle \xrightleftharpoons[\alpha_\infty]{\gamma_\infty} \langle\bar{\mathbb{L}}^\sharp_\infty,\ \bar{\sqsupseteq}^\sharp_\infty\rangle$ (i.e. α_∞ preserves existing $\sqcap^\sharp_\infty$);

D. For $S \in \mathbb{L}^\sharp_+$, $\alpha_+(S \fatsemi^\sharp S') = \alpha_+(S)\ \bar{\fatsemi}^\sharp\ \alpha_+(S')$ when $S' \in \mathbb{L}^\sharp_+$ and $\alpha_\infty(S \fatsemi^\sharp S') = \alpha_\infty(S)\ \bar{\fatsemi}^\sharp\ \alpha_\infty(S')$ when $S' \in \mathbb{L}^\sharp_\infty$.

(respectively "$\bar{\sqsubseteq}^\sharp_+$" or "$\bar{\sqsupseteq}^\sharp_\infty$" instead of "=" and $\leftrightarrows$ instead of $\xrightleftharpoons{}$ for approximate abstractions);

Following (12), the abstraction of the semantic domain and semantics are

$$\begin{array}{rcl}\bar{\mathbb{L}}^\sharp & \triangleq & (e : \bar{\mathbb{L}}^\sharp_+ \times \bot : \bar{\mathbb{L}}^\sharp_\infty \times br : \bar{\mathbb{L}}^\sharp_+) \\ \alpha(\langle e : S_+,\ \bot : S_\infty,\ br : S_{br}\rangle) & \triangleq & \langle e : \alpha_+(S_+),\ \bot : \alpha_\infty(S_\infty),\ br : \alpha_+(S_{br})\rangle\end{array} \tag{66}$$

are well-defined such that

$$\langle \mathbb{L}^\sharp,\ \sqsubseteq^\sharp\rangle \xleftrightarrow[\alpha]{\gamma} \langle \bar{\mathbb{L}}^\sharp,\ \bar{\sqsubseteq}^\sharp\rangle. \tag{67}$$

Lemma 8.2. Ⓐ *An exact abstraction $\bar{\mathbb{D}}^\sharp \triangleq \langle \bar{\mathbb{D}}^\sharp_+,\ \bar{\mathbb{D}}^\sharp_\infty\rangle$ of a well-defined concrete domain $\mathbb{D}^\sharp \triangleq \langle \mathbb{D}^\sharp_+,\ \mathbb{D}^\sharp_\infty\rangle$ satisfying any one of the hypotheses 3.2.D.a to 3.2.D.d.i to 3.2.D.d.iv of definition 3.2 is a well-defined abstract domain of the same nature.*

Theorem 8.3. Ⓐ *If $\bar{\mathbb{D}}^\sharp$ is an exact (respectively approximate) abstraction of $\mathbb{D}^\sharp$ then $\forall \mathtt{S} \in \mathbb{S}\ .\ \bar{[\![}\mathtt{S}\bar{]\!]}^\sharp = \alpha([\![\mathtt{S}]\!]^\sharp)$ (respectively "$\bar{\sqsubseteq}^\sharp$" instead of "$=$" for approximate abstractions).*

Example 8.4 (Relational semantics). The relational semantics $[\![\mathtt{S}]\!]^\varrho$ of [21] is the following abstraction of the trace semantics $[\![\mathtt{S}]\!]^\pi$.

$$\alpha_+(S) \quad \triangleq \quad \{\langle\sigma,\ \sigma'\rangle \mid \exists\pi\ .\ \sigma\pi\sigma' \in S \cap \Sigma^+\} \qquad\qquad \alpha_\infty(S) \quad \triangleq \quad \{\langle\sigma,\ \bot\rangle \mid \exists\pi\ .\ \sigma\pi \in S \cap \Sigma^\infty\}$$

It follows, by theorem 8.3, that $\forall \mathtt{S} \in \mathbb{S}\ .\ [\![\mathtt{S}]\!]^\varrho = \alpha([\![\mathtt{S}]\!]^\pi)$ and by a classic calculational design, we would get the relational semantics of [21, sect. I.1] (recalled in sect. 4 as a specific instance of the algebraic semantics of sect. 3). ■

9 Induced Abstraction of the Execution Transformer

We have defined properties of program executions as program semantics in $\mathbb{L}^\sharp$ (12). This formalizes the observation that program semantics specify exactly the properties of all possible executions of any program of the language. An abstraction (66) of the semantics in definition 8.1 induces an execution transformer $\overline{\mathsf{post}}^\sharp \in \bar{\mathbb{L}}^\sharp \xrightarrow{\nearrow} \bar{\mathbb{L}}^\sharp \xrightarrow{\nearrow} \bar{\mathbb{L}}^\sharp$ (18) for this abstract semantics Ⓐ

$$\begin{array}{rcl}\bar{\alpha}(\mathsf{p}) & \triangleq & \boldsymbol{\lambda}\bar{S}\bullet\boldsymbol{\lambda}\bar{P}\bullet\alpha(\mathsf{p}(\gamma(\bar{S}))\gamma(\bar{P})) \\ \overline{\mathsf{post}}^\sharp(\bar{S})\bar{P} & \triangleq & \bar{\alpha}(\mathsf{post}^\sharp)(\bar{S})\bar{P} \quad = \quad \alpha(\mathsf{post}^\sharp(\gamma(\bar{S}))\gamma(\bar{P})) \quad = \quad \bar{P}\ \bar{;}^\sharp\ \bar{S}\end{array} \tag{68}$$

Notice that defining $\bar{\gamma}(\bar{\mathsf{p}}) \triangleq \boldsymbol{\lambda}S\bullet\boldsymbol{\lambda}P\bullet\gamma(\bar{\mathsf{p}}(\alpha(S))\alpha(P))$, we have a Galois retraction Ⓐ

$$\langle \mathbb{L}^\sharp \xrightarrow{\nearrow} \mathbb{L}^\sharp \xrightarrow{\nearrow} \mathbb{L}^\sharp,\ \sqsubseteq^\sharp\rangle \xleftrightarrow[\bar{\alpha}]{\bar{\gamma}} \langle \bar{\mathbb{L}}^\sharp \xrightarrow{\nearrow} \bar{\mathbb{L}}^\sharp \xrightarrow{\nearrow} \bar{\mathbb{L}}^\sharp,\ \bar{\sqsubseteq}^\sharp\rangle \tag{69}$$

such that $\overline{\mathsf{post}}^\sharp = \bar{\alpha}(\mathsf{post})$ in (68). Observe that if an abstraction $\bar{\mathbb{D}}^\sharp \triangleq \langle \bar{\mathbb{D}}^\sharp_+,\ \bar{\mathbb{D}}^\sharp_\infty\rangle$ of an abstract domain $\mathbb{D}^\sharp \triangleq \langle \mathbb{D}^\sharp_+,\ \mathbb{D}^\sharp_\infty\rangle$ is commuting (71) then Ⓐ

$$\alpha(\mathsf{post}^\sharp(\gamma(\bar{S}))P) \quad = \quad \overline{\mathsf{post}}^\sharp(\bar{S})(\alpha(P)) \tag{70}$$

Lemma 9.1 (Commutation). Ⓐ *If the abstraction $\bar{\mathbb{D}}^\sharp \triangleq \langle \bar{\mathbb{D}}^\sharp_+,\ \bar{\mathbb{D}}^\sharp_\infty\rangle$ of an abstract domain $\mathbb{D}^\sharp \triangleq \langle \mathbb{D}^\sharp_+,\ \mathbb{D}^\sharp_\infty\rangle$ is exact then*

$$\alpha(P\ ;^\sharp\ \gamma(\bar{S})) \quad = \quad \alpha(P)\ \bar{;}^\sharp\ \bar{S} \qquad \textit{and} \qquad \alpha(\mathsf{post}(\gamma(\bar{S}))P) \quad = \quad \overline{\mathsf{post}}(\bar{S})(\alpha(P)) \tag{71}$$

Lemma 9.1 shows that doing the computation in the concrete and then abstracting is equivalent to doing the computation in the abstract. Relative to the abstraction, no information is lost. Moreover, instead of deriving the Galois connection (69) from that (67), we can start directly from an abstraction of post given by (69). The abstract semantics is then $\bar{S} = \overline{\mathsf{post}}^\sharp(\bar{S})$skip proving the equivalence of (65.1) and (65.2).

10 Induced Abstraction of the Semantic Transformer

The semantics transformer $\overline{\mathsf{Post}}^\sharp \in \bar{\mathbb{L}}^\sharp \to \wp(\bar{\mathbb{L}}^\sharp) \to \wp(\bar{\mathbb{L}}^\sharp)$ for this abstract semantics is Ⓐ

$$\bar{\bar{\alpha}}(\mathsf{P}) \quad \triangleq \quad \boldsymbol{\lambda}\bar{S}\bullet\boldsymbol{\lambda}\bar{\mathcal{P}}\bullet\{\alpha(R) \mid R \in \mathsf{P}(\gamma(\bar{S}))(\{\gamma(\bar{P}) \mid \bar{P} \in \bar{\mathcal{P}}\})\} \tag{72}$$

$$\overline{\mathsf{Post}}^\sharp(\bar{S})\bar{\mathcal{P}} \quad \triangleq \quad \bar{\bar{\alpha}}(\mathsf{Post}^\sharp)(\bar{S})\bar{\mathcal{P}} \quad = \quad \{\overline{\mathsf{post}}^\sharp(\bar{S})\bar{P} \mid \bar{P} \in \bar{\mathcal{P}}\} \tag{73}$$

Example 10.1 (Transformers for the relational semantics). For the relational semantics of example 8.4, the composition is $S \mathbin{\bar{\fatsemi}^{\varrho}} S' = (S \cap (\Sigma \times \{\bot\})) \cup (S \cap (\Sigma \times \Sigma) \circ S')$ (intuitively $\mathsf{S}_1\,;\mathsf{S}_2$ does not terminate if S_1 does not terminate or S_1 terminates but S_2 doesn't and terminates if both S_1 and S_2 terminate with the composition of their effects). Then $\overline{\mathsf{Post}}^{\varrho}[\![\mathsf{S}]\!]^{\varrho}\mathcal{P} = \{P \mathbin{\bar{\fatsemi}^{\varrho}} [\![\mathsf{S}]\!]^{\varrho} \mid P \in \mathcal{P}\}$ so that if $\mathcal{P}$ is a precondition relating the initial states of the command S to those of the program then $\overline{\mathsf{Post}}^{\varrho}[\![\mathsf{S}]\!]^{\varrho}$ relates the final states of the command S or nontermination to the initial states of the program. ■

We have the Galois retraction Ⓐ

$$\langle \mathbb{L}^{\sharp} \to \wp(\mathbb{L}^{\sharp}) \xrightarrow{\nearrow} \wp(\mathbb{L}^{\sharp}),\ \subseteq\rangle \underset{\bar{\bar{\alpha}}}{\overset{\bar{\bar{\gamma}}}{\leftrightarrows}} \langle \bar{\mathbb{L}}^{\sharp} \to \wp(\bar{\mathbb{L}}^{\sharp}) \xrightarrow{\nearrow} \wp(\bar{\mathbb{L}}^{\sharp}),\ \subseteq\rangle \tag{74}$$

Observe that instead of deriving (74) from (69), it is equivalent to start from a Galois retraction (74) since we can recover post from Post by (34).

11 Induced Abstraction of the Abstract Logics

Writing $f(X) \triangleq \{f(x) \mid x \in X\}$, the abstract logic $\overline{\mathsf{L}}^{\sharp} \in \bar{\mathbb{L}}^{\sharp} \to (\wp(\bar{\mathbb{L}}^{\sharp}) \times \wp(\bar{\mathbb{L}}^{\sharp}))$ is

$$\bar{\bar{\bar{\alpha}}}(\mathsf{L}) \triangleq \boldsymbol{\lambda}\bar{S}\bullet\{\langle\bar{\mathcal{P}},\ \bar{\mathcal{Q}}\rangle \mid \alpha(\bigcap\{\mathcal{Q} \mid \langle\gamma(\bar{\mathcal{P}}),\ \mathcal{Q}\rangle \in \mathsf{L}(\gamma(\bar{S}))\}) \subseteq \bar{\mathcal{Q}}\} \tag{75}$$

$$\overline{\overline{\mathsf{L}}}^{\sharp}(\bar{S}) \triangleq \bar{\bar{\bar{\alpha}}}(\overline{\mathsf{L}}^{\sharp})(\bar{S}) \qquad\qquad \underline{\overline{\mathsf{L}}}^{\sharp}(\bar{S}) \triangleq \bar{\bar{\bar{\alpha}}}(\underline{\mathsf{L}}^{\sharp})(\bar{S}) \tag{76}$$

Theorem 11.1. Ⓐ *If $\bar{\mathbb{D}}^{\sharp}$ is an exact abstraction of $\mathbb{D}^{\sharp}$ then $\overline{\overline{L}}^{\sharp}(\bar{S}) = \{\langle\bar{\mathcal{P}},\ \bar{\mathcal{Q}}\rangle \mid \overline{\mathsf{Post}}^{\sharp}(\bar{S})\bar{\mathcal{P}} \subseteq \bar{\mathcal{Q}}\}$ (and $\underline{\overline{L}}^{\sharp}(\bar{S}) = \{\langle\bar{\mathcal{P}},\ \bar{\mathcal{Q}}\rangle \mid \bar{\mathcal{Q}} \subseteq \overline{\mathsf{Post}}^{\sharp}(\bar{S})\bar{\mathcal{P}}\}$).*

It follows from theorem 11.1 that the logic proof system of theorem 7.5 is applicable to the upper abstract logic $\overline{\overline{\mathsf{L}}}^{\sharp}(\bar{S})$ (and dually theorem 7.8 for the lower abstract logic).

In conclusion of this part II, although the abstractions of the semantics, post, Post, and logics have been shown to be equally expressible for exact abstractions, they do not really solve the problem of the complexity of the resulting logic (although hyperproperties may be simpler). The logics still have to handle exactly the (abstract) semantics occurring in the (hyper) properties. So our proposed proof system has rules (52)—(59) plus simplified rules applicable to less general classes of properties defined by the abstractions studied in the following part III.

Part III: Abstractions for Semantic (Hyper) Logics

The problem with (hyper) logics studied in part I (and their abstractions in part II) is that for a program to satisfy a semantic (hyper) property, its semantics must exactly occur in this (hyper) property and therefore the proof must exactly characterize the program semantics. So, contrary to Hoare logic or its dual, (hyper) proof rules cannot make over or under approximations of the program semantics in semantic properties. In this part III, we study abstractions of semantic properties that yield simpler sound and complete proof rules for the less general semantic (hyper) properties defined by the abstraction. Such abstractions can also provide representations of abstract semantic (hyper) properties[3].

12 Semantic to Execution Property Abstraction

12.1 Join Abstraction

The join abstraction $\alpha_{\cup}(\mathcal{P}) \triangleq \bigcup\mathcal{P}$ is classic to abstract set-based semantics (hyper) properties $\mathcal{P}$ into execution properties $\alpha_{\cup}(\mathcal{P})$ [20, section 8.6]. It is relegated to the appendix Ⓐ.

[3]Another example is the possible representation of semantic properties satisfying the decreasing chain condition by join irreducibles [11, theorem 4.8].

13 Homomorphic Semantic Abstraction

The homomorphic abstraction $\alpha(S) \triangleq \{h(x) \mid x \in S\}$ is also well known [21, exercise 11.6] and can be used e.g. to define partial hypercorrectness, trace safety hyperproperties, etc. Ⓐ.

14 Execution Property Elimination

Given a set $\mathbb{I} \in \wp(\wp(\mathbb{L}^\sharp))$ of semantic properties of interest, the Galois retraction

$$\langle \wp(\mathbb{L}^\sharp),\ \subseteq\rangle \underset{\boldsymbol{\lambda}\mathcal{P}\bullet\mathcal{P}\cap\mathbb{I}}{\overset{\boldsymbol{\lambda}\mathcal{Q}\bullet\mathcal{Q}\cup\mathbb{I}}{\leftrightarrows}} \langle \mathbb{I},\ \subseteq\rangle$$

[20, exercise 11.5] eliminates the semantics of no interest. It can be used e.g. to handle k-semantic properties Ⓐ.

15 Principal Order Ideal Abstraction

15.1 Definition of the Principal Order Ideal Abstraction

Subject to the existence of the least upper bound, the principal ideal abstraction is

$$\alpha^{\curlywedge}(\mathcal{P}) \quad\triangleq\quad \{P \mid P \sqsubseteq \bigsqcup \mathcal{P}\} \tag{77}$$

Lemma 15.1. Ⓐ *$\alpha^{\curlywedge}$ is an upper closure operator and $\langle \alpha^{\curlywedge}(\wp(\mathbb{L})),\ \subseteq,\ \{\bot\},\ \mathbb{L},\ \boldsymbol{\lambda}X\bullet\alpha^{\curlywedge}(\cup X),\ \cap\rangle$ is a complete lattice.*

15.2 Proof Rule Simplification

If $\langle \mathbb{L},\ \sqsubseteq\rangle$ is a complete lattice and the composition preserves arbitrary existing limits in definition 3.2.D.d then proofs in the upper abstract semantic logic can be based on the classic upper abstract execution property logic of section 5.4 for principal ideal closed properties and their dual Ⓐ.

$$\frac{\overline{\{}\bigsqcup\mathcal{P}\overline{\}}\,\mathsf{S}\,\overline{\{}\bigsqcup\mathcal{Q}\overline{\}}}{\overline{\{\!|}\,\mathcal{P}\,\overline{|\!\}}\,\mathsf{S}\,\overline{\{\!|}\,\mathcal{Q}\,\overline{|\!\}}},\quad \alpha^{\curlywedge}(\mathcal{Q}) = \mathcal{Q} \qquad\qquad \frac{\forall P \in \mathcal{P}\ .\ \underline{\{}P\underline{\}}\,\mathsf{S}\,\underline{\{}\bigsqcap\mathcal{Q}\underline{\}}}{\overline{\{\!|}\,\mathcal{P}\,\overline{|\!\}}\,\mathsf{S}\,\overline{\{\!|}\,\mathcal{Q}\,\overline{|\!\}}},\quad \alpha^{\curlyvee}(\mathcal{Q}) = \mathcal{Q} \tag{78}$$

Example 15.2 (Proof reduction for principal ideal hyperproperties). Consider the instantiation for the natural relational semantics in section 4 with no break. Define the assertional execution postcondition $Q_1 \triangleq \{\sigma \in \Sigma \mid \sigma(x) \leq 10\}$ with relational equivalent $Q_2 \triangleq \Sigma \times Q_1$ and hyperproperty $\mathcal{Q} \triangleq \alpha^{\curlywedge}(Q_2) = \alpha^{\curlywedge}(\Sigma\times\{\sigma \in \Sigma \mid \sigma(x) \leq 10\})$ and similarly $\mathcal{P} \triangleq \{(\Sigma\times\{\sigma \in \Sigma \mid \sigma(x) = n\}) \mid n \in \mathbb{N} \wedge n > 10\}$. To prove the following hyperlogic triple $\overline{\{\!|}\,\mathcal{P}\,\overline{|\!\}}$`while(x>10) x=x-1`$\overline{\{\!|}\,\mathcal{Q}\,\overline{|\!\}}$, it is equivalent to prove the following.

$\quad\overline{\{\!|}\,\mathcal{P}\,\overline{|\!\}}$`while(x>10) x=x-1`$\overline{\{\!|}\,\mathcal{Q}\,\overline{|\!\}}$

$\Leftrightarrow \overline{\{}\bigcup\mathcal{P}\overline{\}}$`while(x>10) x=x-1`$\overline{\{}\bigcup\mathcal{Q}\overline{\}}$ $\qquad$ ≀By rule of (78)≀

$\Leftrightarrow \overline{\{}\Sigma\times\{\sigma \in \Sigma \mid \sigma(x) > 10\}\overline{\}}$`while(x>10) x=x-1`$\overline{\{}\Sigma\times\{\sigma \in \Sigma \mid \sigma(x) \leq 10\}\overline{\}}$

Then one can use the over-approximation logic with termination proof in [22]. ■

16 Order Ideal Abstraction

16.1 Definition of the Order Ideal Abstraction

The order ideal abstraction on $\langle \wp(\mathbb{L}),\ \subseteq\rangle$ is

$$\alpha^{\sqsubseteq}(\mathcal{P}) \quad\triangleq\quad \{P' \in \mathbb{L} \mid \exists P \in \mathcal{P}\ .\ P' \sqsubseteq P\} \qquad \langle \wp(\mathbb{L}),\ \subseteq\rangle \underset{\alpha^{\sqsubseteq}}{\overset{\mathbb{1}}{\leftrightarrows}} \langle \alpha^{\sqsubseteq}(\wp(\mathbb{L})),\ \subseteq\rangle \tag{79}$$

$\alpha^{\sqsubseteq}$ is an upper closure operator and $\langle \alpha^{\sqsubseteq}(\wp(\mathbb{L})),\ \subseteq,\ \varnothing,\ \mathbb{L},\ \boldsymbol{\lambda}X\bullet\alpha^{\sqsubseteq}(\cup X),\ \cap\rangle$ is a complete lattice [83, theorem 4.1]. The order filter abstraction $\alpha^{\sqsupseteq}$ is defined dually. Note that $\alpha^{\curlywedge}(\mathcal{P}) = \alpha^{\sqsubseteq}(\{\bigsqcup\mathcal{P}\})$. As observed by [66, page 239] for subset-closed hyperproperties, all execution properties are order-ideal closed for trace properties (where $\sqsubseteq$ is $\subseteq$), but not conversely, citing observational determinism [86] as a counterexample.

16.2 Proof Rule Simplification

The main interest of the order ideal/filter abstraction is the substantial simplification of the `while` rules (59) and (64). To show this consider properties in $\alpha^{\sqsupseteq^\sharp}(\wp(\mathbb{L}^\sharp)$ where $\sqsupseteq^\sharp$ is defined component wise on $\mathbb{L}^\sharp$ in (12) with $\sqsupseteq^\sharp_+$ on the exit and break components and $\sqsubseteq^\sharp_\infty$ on the infinite component. We abstract $\mathsf{Post}^\sharp$ in (31) to $\mathsf{Post}^{\sqsupseteq^\sharp} \in \mathbb{L}^\sharp \to \alpha^{\sqsupseteq^\sharp}(\wp(\mathbb{L}^\sharp)) \xrightarrow{\nearrow} \alpha^{\sqsupseteq^\sharp}(\wp(\mathbb{L}^\sharp))$ by $(\mathcal{P} \in \alpha^{\sqsupseteq^\sharp}(\wp(\mathbb{L}^\sharp)))$

$$\begin{aligned}
\mathsf{Post}^{\sqsupseteq^\sharp}(S)\mathcal{P} \triangleq \alpha^{\sqsupseteq^\sharp}(\mathsf{Post}^\sharp(S)\mathcal{P}) &= \{P' \in \mathbb{L}^\sharp \mid \exists P \in \mathsf{Post}^\sharp(S)\mathcal{P} \,.\, P' \sqsupseteq^\sharp P\} && \text{(def. (79) of } \alpha^{\sqsupseteq^\sharp}\text{)}\\
&= \{P' \in \mathbb{L}^\sharp \mid \exists P \in \{\mathsf{post}^\sharp(S)P \mid P \in \mathcal{P}\} \,.\, P \sqsubseteq^\sharp P'\} && \text{(def. (31) of Post and inversion of } \sqsupseteq^\sharp\text{)}\\
&= \{P' \in \mathbb{L}^\sharp \mid \exists P \in \mathcal{P} \,.\, \mathsf{post}^\sharp(S)P \in \mathcal{P} \sqsubseteq^\sharp P'\} && \text{(def. } \in\text{)}
\end{aligned}$$

The consequence is that the `while` loop verification condition (59) simplifies to $\mathsf{lfp}^{\sqsubseteq^\sharp_+} \vec{F}^\sharp_{pe}(P') \sqsubseteq^\sharp_+ P_e$ and $\mathsf{gfp}^{\sqsubseteq^\sharp_\infty} F^\sharp_{p\bot} \sqsubseteq^\sharp_\infty Q_{\bot b}$ which can respectively be handled by Park induction [21, theorem II.3.1] and greatest fixpoint over appproximation by transfinite iterates using the dual of [21, theorem II.3.6] as is the case, for classic execution properties, in Hoare logic and termination proofs. The reasoning is dual for (64).

Example 16.1 (Proof reduction for the order ideal abstraction: bounded nondeterminism). Let us consider proofs of programs with bounded nondeterminism, assuming that the value of variables could only be integers. Consider the instantiation of relational natural semantics in section 4 with no break and no nontermination where $\mathbb{V} = \mathbb{Z}$. Let $|S|$ be the cardinality of a set S and consider the semantic (hyper) property $\mathcal{F} \triangleq \wp_{\mathsf{fin}}(\mathbb{L}) \triangleq \{P \in \wp(\mathbb{L}) \mid |P| \in \mathbb{N}\}$ to be the set of finite execution semantics i.e. programs satisfying $\mathcal{F}$ cannot have infinitely many different executions although $\mathbb{L}$ has an infinite cardinality.

Now, suppose we want prove that $\overline{\{\!|}\mathcal{F}\overline{|\!\}}\,\mathtt{S}\,\overline{\{\!|}\mathcal{F}\overline{|\!\}}$, where `S` $\triangleq$ `x = [0, ∞]; while(x>0) x=x-1`. Since $\mathcal{F}$ is an order ideal abstraction (subset-closed), we need to find a function $\mathcal{I} \in \mathcal{F} \to \mathcal{F}$ such that for arbitrary $P \in \mathcal{P}$, we have $\mathsf{post}[\![\mathtt{S}]\!] \subseteq \mathcal{I}(P)$, and, at the same time, the image of $\mathcal{I}$ is a subset of $\mathcal{F}$. Let m and n to be any integer such that $m < 0 < n$, we can set this $\mathcal{I}$ to be

$$\mathcal{I} = \boldsymbol{\lambda} P \bullet \{\langle\sigma,\, \sigma'\rangle \in \Sigma \times \Sigma \mid m < \sigma'(x) \leq n \wedge \exists \langle\sigma_1,\, \sigma_1'\rangle \in P \,.\, (\sigma_1 = \sigma \wedge \forall v \in \mathbb{V} \,.\, v \neq x \Rightarrow \sigma_1'(x) = \sigma'(x))\}$$

We notice that this program component eventually assigns the value 0 to x while keeping the value of the other variables unchanged. As a result, for arbitrary $P \in \mathcal{P}$

$$\begin{aligned}
\mathsf{post}[\![\mathtt{S}]\!](P) \;=\; \{\langle\sigma,\, \sigma'\rangle \in \Sigma \times \Sigma \mid\; & \sigma'(x) = 0 \wedge \exists \langle\sigma_1,\, \sigma_1'\rangle \in P \,.\, (\sigma_1 = \sigma \wedge \forall v \in \mathbb{V} \,.\, v \neq x \Rightarrow \\
& \sigma_1'(x) = \sigma'(x))\} \subseteq \mathcal{I}(P)
\end{aligned}$$

For the cardinality of $\mathcal{I}(P)$, we let the sequence $\langle X^i,\ n < i \leq m\rangle$ such that $X^i = \{\langle\sigma,\, \sigma'\rangle \in \Sigma \times \Sigma \mid \sigma'(x) = i \wedge \exists \langle\sigma_1,\, \sigma_1'\rangle \in P \,.\, (\sigma_1 = \sigma \wedge \forall v \in \mathbb{V} \,.\, v \neq x \Rightarrow \sigma_1'(x) = \sigma'(x))\}$. The cardinality of X^i in this case will be smaller than that of P, meaning $|X^i| \in \mathbb{N}$. Thus, the finite union of X^i, $\bigcup\limits_{m<i\leq n} X^i$ also has finite cardinality. ■

17 Frontiers Abstractions

Another solution to represent order ideal abstractions as proposed by [66, proposition 1] is to consider the maximal elements of the order ideal closed semantic (hyper) property only. Unfortunately, this is not the same abstraction.

Counter example 17.1. Consider the hyperproperty $\mathcal{F} \triangleq \wp_{\mathsf{fin}}(\mathbb{L}) \triangleq \{P \in \wp(\mathbb{L}) \mid |P| \in \mathbb{N}\}$ in example 16.1 i.e. programs satisfying $\mathcal{F}$ cannot have infinitely many different executions although

$\mathbb{L}$ has an infinite cardinality. Then the order ideal abstraction is $\alpha^{\sqsubseteq}(\mathcal{F}) = \mathcal{F}$ which has no maximal elements so the maximal elements abstraction of this order ideal abstraction $\alpha^{\sqsubseteq}(\mathcal{F}) = \mathcal{F}$ is the empty set which is definitely different from this order ideal abstraction $\alpha^{\sqsubseteq}(\mathcal{F}) = \mathcal{F}$. ■

Let us study this abstraction in more detail.

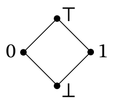
17.1 Lower Frontier Abstraction

The lower frontier abstraction abstracts a subset of a poset to its mimimal elements

$$\alpha^{\underline{F}}(\mathcal{P}) \quad \triangleq \quad \{P \in \mathcal{P} \mid \forall P' \in \mathcal{P} \,.\, P' \sqsubseteq P \Rightarrow P' = P\} \tag{80}$$

$\alpha^{\underline{F}}$ is reductive and idempotent by not necessarily increasing (and so does not necessarily preserve existing joins) hence may not be the lower adjoint of a Galois connection.

Counter example 17.2. Consider the complete lattice $\{\bot, 0, 1, \top\}$ with $\bot \sqsubseteq \bot \sqsubseteq 0 \sqsubseteq 0 \sqsubseteq \top \sqsubseteq \top$ and $\bot \sqsubseteq 1 \sqsubseteq 1 \sqsubseteq \top$. We have $\mathcal{P}_1 = \{\top\} \subseteq \{0, 1, \top\} = \mathcal{P}_2$ but $\alpha^{\underline{F}}(\mathcal{P}_1) = \{\top\} \nsubseteq \{0, 1\} = \alpha^{\underline{F}}(\mathcal{P}_2)$ proving that $\alpha^{\overline{F}}$ is not increasing hence does not preserve existing joins hence is not the lower adjoint of a Galois connection. By duality, neither is $\alpha^{\overline{F}}$. ■



17.2 Frontier Order Ideal Abstraction

The frontier order ideal abstraction

$$\alpha^{\sqsupseteq\underline{F}} \quad \triangleq \quad \alpha^{\sqsupseteq} \circ \alpha^{\underline{F}} \tag{81}$$

closes the frontier by its over approximations, as shown by the following

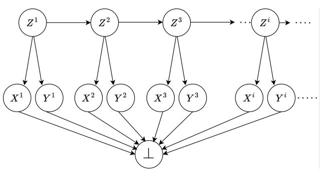
Lemma 17.3. Ⓐ $\alpha^{\sqsupseteq\underline{F}}(\mathcal{P}) = \{P \in \mathbb{L} \mid \exists F \in \alpha^{\underline{F}}(\mathcal{P}) \,.\, F \sqsubseteq P\} = \{P \in \mathbb{L} \mid \exists F \in \mathcal{P} \,.\, \forall P' \in \mathcal{P} \,.\, P' \sqsubseteq F \Rightarrow P' = F \wedge F \sqsubseteq P\}$.

Observe that $\alpha^{\sqsupseteq\underline{F}}$ is idempotent but not necessarily increasing or extensive.

Counter example 17.4. Consider $\mathbb{L} = \{\langle a,\ n\rangle \mid n \in \mathbb{N}\} \cup \{\langle b,\ m\rangle \mid m \in \mathbb{N}\}$ with $\langle x,\ n\rangle \sqsubseteq \langle y,\ m\rangle \triangleq x = y \wedge n \geqslant m$ be two incomparable infinite decreasing chains. $\mathbb{L} \notin \alpha^{\sqsupseteq\underline{F}}(\mathbb{L}) = \emptyset$. Take $\mathcal{P} = \{\langle a,\ n\rangle \mid n \in \mathbb{N}\} \cup \{\langle b,\ 0\rangle\}$ so that $\mathcal{P} \subseteq \mathbb{L}$ but $\alpha^{\sqsupseteq\underline{F}}(\mathcal{P}) = \{\langle b,\ m\rangle \mid m \in \mathbb{N}\} \nsubseteq \alpha^{\sqsupseteq\underline{F}}(\mathbb{L}) = \emptyset$. ■

$\alpha^{\sqsupseteq\underline{F}}(\wp(\mathbb{L}))$ is not closed by intersection.

Counter example 17.5. Consider the lattice on the right. Let $\mathcal{P}_1 = \{Z^i \mid i \in \mathbb{N}_*\} \cup \{X^i \mid i \in \mathbb{N}_*\}$ with frontier $\mathcal{F}_1 = \{X^i \mid i \in \mathbb{N}_*\}$ and $\mathcal{P}_2 = \{Z^i \mid i \in \mathbb{N}_*\} \cup \{Y^i \mid i \in \mathbb{N}_*\}$ with frontier $\mathcal{F}_2 = \{Y^i \mid i \in \mathbb{N}_*\}$. There is no largest set smaller than $\mathcal{P}_1$ and $\mathcal{P}_2$ with an existing frontier. ■



Lemma 17.6. Ⓐ $\langle \alpha^{\sqsupseteq\underline{F}}(\wp(\mathbb{L})),\ \subseteq,\ \emptyset,\ \mathbb{L},\ \cup\rangle$ *is a join semilattice.*

17.3 A Frontier Characterization of the Order Ideal Abstraction

Lemma 17.7. Ⓐ *There is a Galois isomorphism* $\langle \alpha^{\sqsubseteq\overline{F}}(\wp(\mathbb{L})),\ \subseteq\rangle \underset{\alpha^{\overline{F}}}{\overset{\alpha^{\sqsubseteq}}{\leftrightarrows}} \langle \alpha^{\overline{F}}(\wp(\mathbb{L})),\ \preceq^{\overline{F}}\rangle$ *and* $\langle \alpha^{\overline{F}}(\wp(\mathbb{L})),\ \preceq^{\overline{F}},\ \curlyvee^{\overline{F}}\rangle$ *is a join semi lattice with* $P \preceq^{\overline{F}} Q \triangleq (\alpha^{\sqsubseteq}(P) \subseteq \alpha^{\sqsubseteq}(Q))$ *and* $P \curlyvee^{\overline{F}} Q \triangleq \alpha^{\overline{F}}(\alpha^{\sqsubseteq}(P) \cup \alpha^{\sqsubseteq}(Q))$.

Define the principal ideal $\downarrow^{\sqsubseteq}(P) \triangleq \{P' \in \mathbb{L} \mid P' \sqsubseteq P\}$. The following lemma 17.8 is a characterization of $\alpha^{\sqsubseteq\overline{F}}(\wp(\mathbb{L}))$ that corrects and generalizes [66, Proposition 1].

Lemma 17.8. Ⓐ *If* $\mathcal{P} \in \alpha^{\sqsubseteq\overline{F}}(\wp(\mathbb{L}))$ *then* $\mathcal{P} = \bigcup_{P \in \alpha^{\overline{F}}(\mathcal{P})} \downarrow^{\sqsubseteq}(P)$.

18 Chain Limit Abstraction

18.1 Chain Limit Abstraction Definition and Properties

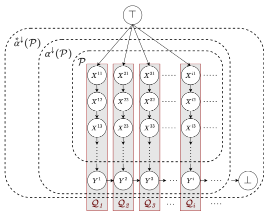
Another possible representation of order ideal abstractions would be by limits of chains. Define

$$\alpha^{\downarrow}(\mathcal{P}) \quad \triangleq \quad \{\bigsqcap_{i\in\mathbb{N}} P_i \mid \langle P_i,\ i \in \mathbb{N}\rangle \in \mathcal{P} \text{ is a decreasing chain with existing glb}\} \tag{82}$$

$\alpha^{\downarrow}$ is $\subseteq$ increasing and extensive but not necessarily idempotent as shown by counter example 18.1 below. The iteration of $\alpha^{\downarrow}$ (possibly transfinitely)

$$\overset{*}{\alpha}^{\downarrow}(\mathcal{P}) \quad \triangleq \quad \mathsf{lfp}^{\subseteq}\, \boldsymbol{\lambda} X \bullet \mathcal{P} \cup \alpha^{\downarrow}(X) \tag{83}$$

yields an upper closure operator [20, lemma 29.1].



Counter example 18.1. Consider the complete lattice $\mathbb{L}$ on the right. Let $\mathcal{P} = \{X^{ij} \mid i, j > 0\}$. We have $\alpha^{\downarrow}(\mathcal{P}) = \{X^{ij} \mid i, j > 0\} \cup \{Y^i \mid i > 0\}$. We have $\bigsqcap\{Y^i \mid i > 0\} = \perp$ so $\alpha^{\downarrow}(\alpha^{\downarrow}(\mathcal{P})) = \{X^{ij} \mid i, j > 0\} \cup \{Y^i \mid i > 0\} \cup \{\perp\} \neq \alpha^{\downarrow}(\mathcal{P})$.
Moreover $\overset{*}{\alpha}^{\downarrow}(\mathcal{Q}_i) \in \overset{*}{\alpha}^{\downarrow}(\wp(\mathbb{L}))$, $i > 0$ but $\bigcup_{i>0} \overset{*}{\alpha}^{\downarrow}(\mathcal{Q}_i) \notin \overset{*}{\alpha}^{\downarrow}(\wp(\mathbb{L}))$. ■

Lemma 18.2. Ⓐ $\langle\wp(\mathbb{L}), \subseteq\rangle \xrightleftharpoons[\overset{*}{\alpha}^{\downarrow}]{\mathbb{1}} \langle\overset{*}{\alpha}^{\downarrow}(\wp(\mathbb{L})), \subseteq\rangle$ *and* $\langle\overset{*}{\alpha}^{\downarrow}(\wp(\mathbb{L})), \subseteq, \varnothing, \mathbb{L}, \boldsymbol{\lambda} X \bullet \overset{*}{\alpha}^{\downarrow}(\bigcup X), \bigcap\rangle$ *is a complete lattice.*

Lemma 18.3. Ⓐ $\forall \mathcal{P} \in \wp(\mathbb{L})\ .\ \alpha^{\downarrow}(\overset{*}{\alpha}^{\downarrow}(\mathcal{P})) = \overset{*}{\alpha}^{\downarrow}(\mathcal{P})$.

Lemma 18.4. Ⓐ *For all* $\mathcal{P} \in \wp(\mathbb{L})$, $\alpha^{\downarrow}(\mathcal{P}) = \mathcal{P}$ *implies* $\overset{*}{\alpha}^{\downarrow}(\mathcal{P}) = \mathcal{P}$.

$\alpha^{\uparrow}$ is defined $\sqsubseteq$ dually, and $\overset{*}{\alpha}^{\uparrow}(\mathcal{P}) \triangleq \mathsf{lfp}^{\subseteq}\, \boldsymbol{\lambda} X \bullet \mathcal{P} \cup \alpha^{\uparrow}(X)$ is an upper closure operator.

18.2 Forall Exists Hyperproperties

Assuming that $\langle\mathbb{L}, \sqsubseteq\rangle = \langle\wp(\Pi), \subseteq\rangle$ (where e.g. $\Pi = \Sigma^{+\infty}$ is a set of traces) $\forall\exists$ hyperproperties have the form

$$\mathcal{AEH} \quad \triangleq \quad \{\{P \in \wp(\Pi) \mid \forall \pi_1 \in P\ .\ \exists \pi_2 \in P\ .\ \langle\pi_1,\ \pi_2\rangle \in A\} \mid A \in \wp(\Pi \times \Pi)\} \tag{84}$$

(this easily generalizes to $\forall \pi_1, \ldots, \pi_n \in P\ .\ \exists \pi'_1, \ldots, \pi'_m \in P\ .\ \langle\pi_1,\ \ldots,\ \pi_n,\ \pi'_1,\ \ldots,\ \pi'_m\rangle \in A$ [40]).

Example 18.5 (Generalized non-interference). A typical forall exists hyperproperty is generalized non interference [35, 69, 70] for the trace semantics of appendix B. Let $\mathtt{L} \in \mathbb{X}$ be a low variable and $\mathtt{H} \in \mathbb{X}$ be a high variable, we have

$$\begin{aligned} \mathit{GNI} \quad \triangleq \quad & \{P \in \wp(\Sigma^{+}) \mid \forall \sigma_1\pi_1\sigma'_1, \sigma_2\pi_2\sigma'_2 \in P\ .\ \exists \sigma_3\pi_3\sigma'_3 \in P\ .\ (\sigma_1(\mathtt{L}) = \sigma_2(\mathtt{L})) \Rightarrow \\ & (\sigma_3(\mathtt{L}) = \sigma_1(\mathtt{L}) \wedge \sigma_3(\mathtt{H}) = \sigma_2(\mathtt{H}) \wedge \sigma'_3(\mathtt{L}) = \sigma'_1(\mathtt{L}))\} \end{aligned} \tag{85}$$

■

Assuming chain-complete lattices in 3.2.A and 3.2.C, chain limit closed semantic properties in $\overset{*}{\alpha}^{\uparrow}(\wp(\wp(\Pi)))$ subsume $\forall\exists$ hyperproperties in $\mathcal{AEH}$ in that Ⓐ

$$\mathcal{AEH} \quad \subseteq \quad \overset{*}{\alpha}^{\uparrow}(\wp(\wp(\Pi))) \tag{86}$$

19 Chain Limit Order Ideal Abstraction

19.1 Chain Limit Order Ideal Abstraction Definition and Properties

Define

$$\alpha^{\sqsubseteq\uparrow} \quad \triangleq \quad \alpha^{\sqsubseteq} \circ \alpha^{\uparrow} \qquad\qquad \text{and} \qquad\qquad \overset{*}{\alpha}^{\sqsubseteq\uparrow}(\mathcal{P}) \quad \triangleq \quad \mathsf{lfp}^{\subseteq}\, \boldsymbol{\lambda} X \bullet \mathcal{P} \cup \alpha^{\sqsubseteq\uparrow}(X) \tag{87}$$

to get an upper closure operator (since $\alpha^{\sqsubseteq\uparrow}$ is increasing and expansive although not idempotent).

Counter example 19.1. Define $\langle\mathbb{L}, \sqsubseteq\rangle = \langle\wp(\mathbb{N}), \subseteq\rangle$ and $\mathcal{N} \triangleq \{\mathbb{N} \setminus \{n\} \mid n \in \mathbb{N}\} \in \wp(\mathbb{N})$ to be the set of all sets $\mathbb{N}$ with one missing element. Since any two different elements of $\mathcal{N}$ are $\subseteq$- incomparable,

$\mathcal{N}$ is both a lower and upper frontier so chains are reduced to one element. Therefore $\overset{*}{\alpha}^{\downarrow}(\mathcal{N}) = \overset{*}{\alpha}^{\uparrow}(\mathcal{N}) = \mathcal{N}$. By (87), it follows that $\alpha^{\sqsubseteq\uparrow}(\mathcal{N}) = \alpha^{\sqsubseteq}(\mathcal{N}) = \wp(\mathbb{N}) \setminus \{\mathbb{N}\}$. Consider the increasing chain $\mathcal{C} = \langle \{i \mid i < j\},\ j \in \mathbb{N}\rangle$ of elements of $\overset{*}{\alpha}^{\uparrow}(\mathcal{N})$. Its limit is $\bigcup_{j\in\mathbb{N}} \{i \mid i < j\} = \mathbb{N} \notin \alpha^{\sqsubseteq\uparrow}(\mathcal{N}) = \wp(\mathbb{N}) \setminus \{\mathbb{N}\}$ proving that $\alpha^{\sqsubseteq\uparrow}$ is not idempotent. ■

Lemma 19.2. Ⓐ $\langle \wp(\mathbb{L}), \subseteq\rangle \mathrel{\underset{\overset{*}{\alpha}^{\sqsubseteq\uparrow}}{\overset{\mathbb{1}}{\leftrightarrows}}} \langle \overset{*}{\alpha}^{\sqsubseteq\uparrow}(\wp(\mathbb{L})), \subseteq\rangle$ *and* $\langle \overset{*}{\alpha}^{\sqsubseteq\uparrow}(\wp(\mathbb{L})), \subseteq, \varnothing, \mathbb{L}, \boldsymbol{\lambda} X \boldsymbol{\cdot} \overset{*}{\alpha}^{\sqsubseteq\uparrow}(\bigcup X), \bigcap\rangle$ *is a complete lattice.*

19.2 Forall Hyperproperties

$\forall$ hyperproperties are usually defined in the context of trace semantics of section B, for which, in absence of `breaks`, $\langle \mathbb{L}, \sqsubseteq\rangle = \langle \wp(\Sigma^{+\infty}), \subseteq\rangle)$ as in section B.3. In this case, by definition of $\subseteq$, we get

$$\mathcal{AAH} \quad \triangleq \quad \{\{P \in \wp(\Sigma^{+\infty}) \mid \forall \pi_1, \pi_2 \in P \,.\, \langle \pi_1,\ \pi_2\rangle \in A\} \mid A \in \wp(\Sigma^{+\infty} \times \Sigma^{+\infty})\} \tag{88}$$

Example 19.3 (Non-interference). A typical forall hyperproperty is non interference $\mathit{NI} \in \mathcal{AAH}$ for the trace semantics of section B [16, 47, 48]. Let $\mathtt{L} \in \mathbb{X}$ be a low variable, we have

$$\mathit{NI} \quad \triangleq \quad \{P \in \wp(\Sigma^{+}) \mid \forall \sigma_1\pi_1\sigma_1', \sigma_2\pi_2\sigma_2' \in P \,.\, (\sigma_1(\mathtt{L}) = \sigma_2(\mathtt{L})) \Rightarrow (\sigma_1'(\mathtt{L}) = \sigma_2'(\mathtt{L}))\} \tag{89}$$

We have $\mathit{NI} \in \mathcal{AAH}$ by defining $A \triangleq \{\langle \sigma_1\pi_1\sigma_1',\ \sigma_2\pi_2\sigma_2'\rangle \mid (\sigma_1(\mathtt{L}) = \sigma_2(\mathtt{L})) \Rightarrow (\sigma_1'(\mathtt{L}) = \sigma_2'(\mathtt{L}))\}$. ■

20 Logic Rule for Chain Limit Order Ideal Abstract Semantic Properties

[30, sect. 5.3] have introduced a sound but incomplete logic for proving $\forall^*\exists^*$ hyperproperties. We generalize the rule in our algebraic lattice-theoretic framework for the chain limit abstract semantic properties in $\overset{*}{\alpha}^{\uparrow}(\wp(\mathbb{L}))$.

20.1 A Sound and Incomplete Rule

[30] does not consider `breaks` and nontermination so that the fields $\perp$ and b of $\langle e : F,\ \perp : I,\ b : B\rangle$ in (12) can be ignored and the tuple reduces to the value F of the field e. In this section, 3.2.A is a lattice which is increasing chain complete, 3.2.C and 3.2.D.c are omitted, and limits of increasing chains are assumed to be preserved in 3.2.D.d. We also assume that $[\![\neg\mathtt{B}]\!]^\sharp_e \mathbin{\text{;}^\sharp} [\![\neg\mathtt{B}]\!]^\sharp_e = [\![\neg\mathtt{B}]\!]^\sharp_e$, $[\![\neg\mathtt{B}]\!]^\sharp_e \mathbin{\text{;}^\sharp} [\![\mathtt{B}]\!]^\sharp_e = [\![\mathtt{B}]\!]^\sharp_e \mathbin{\text{;}^\sharp} [\![\neg\mathtt{B}]\!]^\sharp_e = \perp^\sharp_+$, and $[\![\mathtt{skip}]\!]^\sharp_e \triangleq \mathsf{skip}^\sharp = \mathsf{init}^\sharp$ in (3), which does not hold for traces but holds e.g. for a relational semantics.

The rule of [30] generalizes to

$$\frac{\mathcal{P} \subseteq \mathcal{I}, \quad \overline{\{\!|}\,\mathcal{I}\,\overline{|\!\}}\,\mathtt{if\ (B)\ else\ skip}\,\overline{\{\!|}\,\mathcal{I}\,\overline{|\!\}}, \quad \overline{\{\!|}\,\mathcal{I}\,\overline{|\!\}}\,\neg\mathtt{B}\,\overline{\{\!|}\,\mathcal{Q}\,\overline{|\!\}}}{\overline{\{\!|}\,\mathcal{P}\,\overline{|\!\}}\,\mathtt{while\ (B)\ S}\,\overline{\{\!|}\,\mathcal{Q}\,\overline{|\!\}}}, \quad \mathcal{Q} \in \overset{*}{\alpha}^{\uparrow}(\wp(\mathbb{L}^\sharp)) \tag{90}$$

The key idea to prove that for any $P \in \mathcal{P} \in \wp(\mathbb{L}^\sharp_+)$, the exact postcondition $Q = \mathsf{post}^\sharp[\![\mathtt{while\ (B)\ S}]\!]^\sharp_e P$ will be in $\mathcal{Q}$ is to exhibit an increasing chain in $\mathcal{Q}$ with least upper bound Q, also in $\mathcal{Q}$ by the hypothesis that $\mathcal{Q}$ is a chain limit order ideal abstract semantic property. Soundness follows from theorem R.4 in the appendix Ⓐ. A counter-example proving incompleteness is also provided by lemma R.5 in the appendix Ⓐ.

20.2 Completeness Relative to an Abstract Hypercollecting Semantics

Proof rule (90) is incomplete relative to the hypercollecting semantics (47) of section 6. We show that the rule is complete relative to the following abstraction of the hypercollecting semantics (47).

Definition 20.1 (Weak structural hypercollecting semantics for iteration).

$$\overline{\mathsf{Post}}^\sharp[\![\mathtt{while(B)\ S}]\!]^\sharp_e\mathcal{P} \triangleq \mathsf{Post}^\sharp[\![\neg B]\!]^\sharp_e(\mathsf{lfp}^{\subseteq}\, \boldsymbol{\lambda}\mathcal{X} \boldsymbol{\cdot} \mathcal{P} \cup \overline{\mathsf{Post}}^\sharp[\![\mathtt{if(B)\ S\ else\ skip}]\!]^\sharp_e(\mathcal{X})) \tag{91}$$

$\overline{\text{Post}}^\sharp[\![\texttt{while(B) S}]\!]^\sharp_e$ is an algebraic form of the hypercollecting semantics postulated by [5, p. 877]. We characterize by theorem R.6 in the appendix the executions satisfying (91) Ⓐ.

Therefore $\overline{\text{Post}}^\sharp[\![\texttt{while(B) S}]\!]^\sharp_e\mathcal{P}$ may contain chains $\text{post}^\sharp[\![\neg B]\!]^\sharp_e X^n(P_0) \sqsubseteq^\sharp_+ \text{post}^\sharp[\![\neg B]\!]^\sharp_e X^n(P_1) \sqsubseteq^\sharp_+ \ldots \sqsubseteq^\sharp_+ \text{post}^\sharp[\![\neg B]\!]^\sharp_e X^n(P_k) \sqsubseteq^\sharp_+ \ldots$ which limit will be in $\alpha^\uparrow(\overline{\text{Post}}^\sharp[\![\texttt{while(B) S}]\!]^\sharp_e\mathcal{P})$ but not necessarily in $\text{Post}^\sharp[\![\texttt{while(B) S}]\!]^\sharp_e\mathcal{P}$. It follows that $\overline{\text{Post}}^\sharp[\![\texttt{while(B) S}]\!]^\sharp_e$ may miss limits but also may introduce chains with irrelevant limits of infeasible executions (which invalidates [5, theorem 1] soundness claim).

The following theorem shows the soundness and completeness of rule (90) for the abstract hypercollecting semantics $\overline{\text{Post}}^\sharp[\![\texttt{while(B) S}]\!]^\sharp_e$ requires the consequent $\mathcal{Q}$ to contain the post condition of any number of iterations for any element P of the antecedent $\mathcal{P}$.

Theorem 20.2. Ⓐ *The proof rule* (90) *is sound and complete relative to* (91).

Theorem 20.2 illustrates the importance of the proper choice of the collecting semantics since proof rule (90) is unsound if $\mathcal{Q} \notin \mathring{\alpha}^\uparrow(\wp(\mathbb{L}^\sharp))$ and is complete for collecting semantics (91) but not with respect to collecting semantics (47) hence not for the algebraic semantics of section 3.

By deriving the collecting semantics post for execution properties and hypercollecting semantics Post for semantic properties by systematic abstraction of the algebraic semantics of section 3, we guarantee, by composition of successive abstractions satisfying definition 8.1, that the proof rules for these abstractions are sound with respect to any instance of the algebraic semantics satisfying definition 3.2. Moreover, the proof rules are guaranteed to be complete with respect to these abstract properties, by construction.

21 Sound and Complete Proof Rules for Generalized Exists Forall Hyperproperties

In section S.1 of the appendix Ⓐ, we furthermore introduce conjunctive abstractions (i.e. conjunctions in logics or reduced products in static analysis). Such conjunctive abstractions are used in section S.2 of the appendix to provide the following sound and complete proof rule for generalized $\exists\forall$-hyperproperties Ⓐ. Define $\varrho^{\sqsubseteq\underline{F}}(\mathcal{P}) \triangleq \bigcup_{F \in \alpha^{\underline{F}}(\mathcal{P})} \varphi^{\sqsubseteq}(F)\mathcal{P}$ and $\varphi^{\sqsubseteq}(F) \triangleq \boldsymbol{\lambda}\mathcal{X}\bullet\{P \in \mathcal{X} \mid F \sqsubseteq P \wedge \forall P' \in \mathbb{L}\,.\,F \sqsubseteq P' \sqsubseteq P \rightarrow P' \in \mathcal{X}\}$, $F \in \mathbb{L}$ then, for $\mathcal{Q} \in \varrho^{\sqsubseteq\underline{F}}(\wp(\mathbb{L}^\sharp))$,

$$\frac{\exists \mathcal{X} \in \alpha^{\underline{F}}(\mathcal{Q}) \rightarrow \wp(\mathbb{L}^\sharp)\,.\,\mathcal{P} \subseteq \bigcup\limits_{F \in \alpha^{\underline{F}}(\mathcal{Q})} \mathcal{X}_F, \quad (\forall F \in \alpha^{\underline{F}}(\mathcal{Q})\,.\,\forall P \in \mathcal{X}_F\,.\,\exists Q \in \varphi^{\sqsubseteq}(F)\mathcal{Q}\,.\,\overline{\{P\}}S\overline{\{Q\}} \wedge \underline{\{P\}}S\underline{\{F\}})}{\overline{\{\!|}\,\mathcal{P}\,\overline{|\!\}}\;\texttt{S}\;\overline{\{\!|}\,\mathcal{Q}\,\overline{|\!\}}}$$

An example Ⓐ is provided in the appendix.

22 Hierarchy of hyperproperties abstractions

To compare these abstractions, we first show that chain limit order ideal abstract properties have an equivalent frontier order ideal representation Ⓐ.

$$\langle \alpha^{\sqsubseteq\overline{F}}(\wp(\mathbb{L})),\ \subseteq\rangle \underset{\mathring{\alpha}^{\sqsubseteq\uparrow}}{\overset{\mathbb{1}}{\leftrightarrows}} \langle \mathring{\alpha}^{\sqsubseteq\uparrow}(\wp(\mathbb{L})),\ \subseteq\rangle \tag{92}$$

Figure 1 shows a lattice of hyperproperties derived by our abstractions as well as the related hyperproperties that they subsume.

23 Related Work

Algebraic semantics [45, 49, 58, 71] is rooted in the previous concept of program schemes [12, 37, 44, 46, 74]. The idea of handling logics algebraically using an abstract domain goes back to [28, section 5]. It requires a distinction between computational and logical orderings which first appeared in strictness analysis (using Scott partial order for computational ordering and inclusion for logical

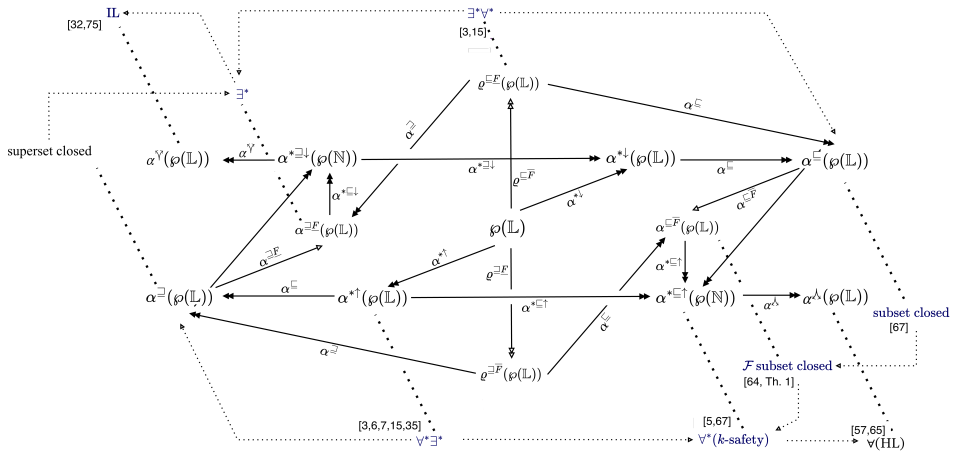


Fig. 1. The hierarchy of hyperproperties by abstraction. The arrow is interpreted as "more general than" where the double arrow represents Galois surjection. Dotted line indicated the hyperproperties subsumed by our abstract in the related works. Ⓐ

ordering [73]). It is not uncommon in abstract interpretation since then. The calculational methodology that we have used is based on [21]. Following the introduction of trace hyperproperties [14], most semantics [5, 66] and verification methods for semantic (hyper) properties have been on subclasses of hyperproperties [6–10, 13, 15, 29, 30, 67], further reviewed in extreme great detail in [30, section 6].

24 Conclusion and Future Work

Transformational (hyper) logics have traditionally been based on transformers themselves equivalent to an operational semantics. When considering nontermination, other semantics like denotational semantics are relevant, but the corresponding logics are in a separate world [1, 51].

In an attempt to design (hyper) logics valid for various (abstract) semantics, we have defined an algebraic semantics (which can be instantiated to operational, denotational, or relational semantics, and is also useful for deductive methods and static analysis).

We have designed, by calculus, a structural fixpoint collecting semantics post for execution properties (e.g. sets of execution traces), its hypercollecting semantics Post for semantic properties (e.g sets of sets of traces), and the various over or under approximation logics corresponding to these transformers for correctness and incorrectness (part III is for over approximation only, but the main reason to use the under approximation logic is to disprove over approximations which is expressible as $\neg\,\overline{\{\!|}\,\mathcal{P}\,\overline{|\!\}}\ \mathtt{S}\ \overline{\{\!|}\,\mathcal{Q}\,\overline{|\!\}} \Leftrightarrow \exists \varnothing \subsetneq \mathcal{P}' \subseteq \mathcal{P}\ .\ \overline{\{\!|}\,\mathcal{P}'\,\overline{|\!\}}\ \mathtt{S}\ \overline{\{\!|}\,\neg\mathcal{Q}\,\overline{|\!\}}$ Ⓐ).

Since, and contrary to classic logics, proofs of general semantic (hyper) properties relative to a program semantics requires the exact characterization of this semantics in the proof, an extreme complication, we have considered abstractions of the semantic properties for which this constraint can be relaxed. This has yielded to new sound and complete simplified proof rules, including for algebraic generalizations of forall-forall, forall-exists, and exists-forall semantic (hyper) properties.

The verification of semantic (hyper) properties is still in its infancy and far from reaching the simplicity observed in the verification of execution properties. Several compromises will be needed maybe by relaxing implication (e.g. using Egli-Milner order instead of inclusion), considering abstract properties (for classes of properties of practical interest), and possibly by preserving soundness but renouncing to completeness. However, in full generality, the sound and complete proof methods introduced in this paper, will ultimately be, up to equivalence, the only one applicable.

Data Availability Statement

The full version of this article is available with its appendix as auxiliary material and in a single file on Zenodo with clickable hyper references to the appendix https://doi.org/10.5281/zenodo.14173477.

References

[1] Samson Abramsky. 1991. Domain Theory in Logical Form. *Ann. Pure Appl. Log.* 51, 1-2 (1991), 1–77. https://doi.org/10.1016/0168-0072(91)90065-T

[2] Peter Aczel. 1977. An Introduction to Inductive Definitions. In *Handbook of Mathematical Logic*, John Barwise (Ed.). North–Holland, Amsterdam, Chapter 7, 739–782.

[3] Timos Antonopoulos, Eric Koskinen, Ton Chanh Le, Ramana Nagasamudram, David A. Naumann, and Minh Ngo. 2023. An Algebra of Alignment for Relational Verification. *Proc. ACM Program. Lang.* 7, POPL (2023), 573–603. https://doi.org/10.1145/3571213

[4] Krzysztof R. Apt and Gordon D. Plotkin. 1986. Countable Nondeterminism and Random Assignment. *J. ACM* 33, 4 (1986), 724–767. https://doi.org/10.1145/6490.6494

[5] Mounir Assaf, David A. Naumann, Julien Signoles, Eric Totel, and Frédéric Tronel. 2017. Hypercollecting semantics and its application to static analysis of information flow. In *POPL*. ACM, 874–887. https://doi.org/10.1145/3009837.3009889

[6] Raven Beutner. 2024. Automated Software Verification of Hyperliveness. In *TACAS (2) (Lecture Notes in Computer Science, Vol. 14571)*. Springer, 196–216. https://doi.org/10.1007/978-3-031-57249-4_10

[7] Raven Beutner and Bernd Finkbeiner. 2022. Software Verification of Hyperproperties Beyond k-Safety. In *CAV (1) (Lecture Notes in Computer Science, Vol. 13371)*. Springer, 341–362. https://doi.org/10.1007/978-3-031-13185-1_17

[8] Raven Beutner and Bernd Finkbeiner. 2023. HyperATL*: A Logic for Hyperproperties in Multi-Agent Systems. *Log. Methods Comput. Sci.* 19, 2 (2023), 13:1–13:44. https://doi.org/10.46298/LMCS-19(2:13)2023

[9] Raven Beutner, Bernd Finkbeiner, Hadar Frenkel, and Niklas Metzger. 2023. Second-Order Hyperproperties. In *CAV (2) (Lecture Notes in Computer Science, Vol. 13965)*. Springer, 309–332. https://doi.org/10.1007/978-3-031-37703-7_15

[10] Raven Beutner, Bernd Finkbeiner, Hadar Frenkel, and Niklas Metzger. 2024. Monitoring Second-Order Hyperproperties. In *AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 180–188. https://doi.org/10.5555/3635637.3662865

[11] Thomas S. Blyth. 2005. *Lattices and Ordered Algebraic Structures.* Springer. https://doi.org/10.1007/b139095

[12] Manfred Broy, Martin Wirsing, and Peter Pepper. 1987. On the Algebraic Definition of Programming Languages. *ACM Trans. Program. Lang. Syst.* 9, 1 (1987), 54–99. https://doi.org/10.1145/9758.10501

[13] Michael R. Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe, and César Sánchez. 2014. Temporal Logics for Hyperproperties. In *POST (Lecture Notes in Computer Science, Vol. 8414)*. Springer, 265–284. https://doi.org/10.1007/978-3-642-54792-8_15

[14] Michael R. Clarkson and Fred B. Schneider. 2010. Hyperproperties. *J. Comput. Secur.* 18, 6 (2010), 1157–1210. https://doi.org/10.3233/JCS-2009-0393

[15] Norine Coenen, Bernd Finkbeiner, César Sánchez, and Leander Tentrup. 2019. Verifying Hyperliveness. In *CAV (1) (Lecture Notes in Computer Science, Vol. 11561)*. Springer, 121–139. https://doi.org/10.1007/978-3-030-25540-4_7

[16] Ellis S. Cohen. 1977. Information Transmission in Computational Systems. In *SOSP*. ACM, 133–139. https://doi.org/10.1145/800214.806556

[17] Bruno Courcelle and Maurice Nivat. 1978. The Algebraic Semantics of Recursive Program Schemes. In *Mathematical Foundations of Computer Science 1978, Proceedings, 7th Symposium, Zakopane, Poland, September 4-8, 1978 (Lecture Notes in Computer Science, Vol. 64)*, Józef Winkowski (Ed.). Springer, 16–30. https://doi.org/10.1007/3-540-08921-7_53

[18] Patrick Cousot. 2002. Constructive Design of a Hierarchy of Semantics of a Transition System by Abstract Interpretation. *Theor. Comput. Sci.* 277, 1–2 (2002), 47–103. https://doi.org/10.1016/S0304-3975(00)00313-3

[19] Patrick Cousot. 2019. On Fixpoint/Iteration/Variant Induction Principles for Proving Total Correctness of Programs with Denotational Semantics. In *LOPSTR (Lecture Notes in Computer Science, Vol. 12042)*. Springer, 3–18. https://doi.org/10.1007/978-3-030-45260-5_1

[20] Patrick Cousot. 2021. *Principles of Abstract Interpretation* (1 ed.). MIT Press.

[21] Patrick Cousot. 2024. Calculational Design of [In]Correctness Transformational Program Logics by Abstract Interpretation. *Proc. ACM Program. Lang.* 8, POPL (2024), 175–208. https://doi.org/10.1145/3632849

[22] Patrick Cousot. 2024. Full version of “Calculational Design of [In]Correctness Transformational Program Logics by Abstract Interpretation”, Proc. ACM Program. Lang. 8, POPL (2024), 7:1–10:33, https://doi.org/10.1145/3632849. *Zenodo* (Dec. 2024), 66 pages. https://doi.org/10.5281/zenodo.10439108

[23] Patrick Cousot and Radhia Cousot. 1979. Constructive Versions of Tarski's Fixed Point Theorems. *Pacific J. of Math.* 82, 1 (1979), 43–57. https://doi.org/10.2140/pjm.1979.82.43

[24] Patrick Cousot and Radhia Cousot. 1992. Inductive Definitions, Semantics and Abstract Interpretation. In *POPL*. ACM Press, 83–94. https://doi.org/10.1145/143165.143184

[25] Patrick Cousot and Radhia Cousot. 1995. Compositional and Inductive Semantic Definitions in Fixpoint, Equational, Constraint, Closure-condition, Rule-based and Game-Theoretic Form. In *CAV (Lecture Notes in Computer Science, Vol. 939)*. Springer, 293–308. https://doi.org/10.1007/3-540-60045-0_58

[26] Patrick Cousot and Radhia Cousot. 2009. Bi-inductive structural semantics. *Inf. Comput.* 207, 2 (2009), 258–283. https://doi.org/10.1016/J.IC.2008.03.025

[27] Patrick Cousot and Radhia Cousot. 2012. An abstract interpretation framework for termination. In *POPL*. ACM, 245–258. https://doi.org/10.1145/2103656.2103687

[28] Patrick Cousot, Radhia Cousot, Francesco Logozzo, and Michael Barnett. 2012. An abstract interpretation framework for refactoring with application to extract methods with contracts. In *OOPSLA*. ACM, 213–232. https://doi.org/10.1145/2384616.2384633

[29] Thibault Dardinier. 2024. Formalization of Hyper Hoare Logic: A Logic to (Dis-)Prove Program Hyperproperties. Arch. Formal Proofs, 2023. https://www.isa-afp.org/entries/HyperHoareLogic.html

[30] Thibault Dardinier and Peter Müller. 2024. Hyper Hoare Logic: (Dis-)Proving Program Hyperproperties. *Proceedings of the ACM on Programming Languages (PACMPL)* 8, Issue PLDI, Article No.: 207 (June 2024), 1485–1509. https://doi.org/10.1145/3656437

[31] Brian A. Davey and Hilary A. Priestley. 2002. *Introduction to Lattices and Order, Second Edition.* Cambridge University Press. https://doi.org/10.1017/CBO9780511809088

[32] Edsko de Vries and Vasileios Koutavas. 2011. Reverse Hoare Logic. In *SEFM (Lecture Notes in Computer Science, Vol. 7041)*. Springer, 155–171. https://doi.org/10.1007/978-3-642-24690-6_12

[33] Jerry den Hartog and Erik P. de Vink. 2002. Verifying Probabilistic Programs Using a Hoare Like Logic. *Int. J. Found. Comput. Sci.* 13, 3 (2002), 315–340. https://doi.org/10.1142/S012905410200114X

[34] Klaus Denecke, Marcel Erné, and Shelly L. Wismath. 2003. *Galois Connections and Applications.* Kluwer Academic Publishers. https://doi.org/10.1007/978-1-4020-1898-5

[35] Robert Dickerson, Qianchuan Ye, Michael K. Zhang, and Benjamin Delaware. 2022. RHLE: Modular Deductive Verification of Relational ∀ ∃ Properties. In *APLAS (Lecture Notes in Computer Science, Vol. 13658)*. Springer, 67–87. https://doi.org/10.1007/978-3-031-21037-2_4

[36] Edsger W. Dijkstra. 1978. Program Inversion. In *Program Construction, International Summer School, July 26 - August 6, 1978, Marktoberdorf, Germany (Lecture Notes in Computer Science, Vol. 69)*, Friedrich L. Bauer and Manfred Broy (Eds.). Springer, 54–57. https://doi.org/10.1007/BFB0014657

[37] Andrei P. Ershov. 1979. Abstract computability on algebraic structures. In *Algorithms in Modern Mathematics and Computer Science (Lecture Notes in Computer Science, Vol. 122)*. Springer, 397–420. https://doi.org/10.1007/3-540-11157-3_38

[38] M. Escardó. 2003. Joins in the frame of nuclei. *Applied Categorical Structures* 11, 2 (April 2003), 117–124.

[39] Yuan Feng and Sanjiang Li. 2023. Abstract interpretation, Hoare logic, and incorrectness logic for quantum programs. *Inf. Comput.* 294 (2023), 105077. https://doi.org/10.1016/J.IC.2023.105077

[40] Bernd Finkbeiner and Christopher Hahn. 2016. Deciding Hyperproperties. In *CONCUR (LIPIcs, Vol. 59)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 13:1–13:14. https://doi.org/10.4230/LIPICS.CONCUR.2016.13

[41] Roberto Giacobazzi and Isabella Mastroeni. 2005. Transforming semantics by abstract interpretation. *Theor. Comput. Sci.* 337, 1-3 (2005), 1–50. https://doi.org/10.1016/J.TCS.2004.12.021

[42] Roberto Giacobazzi and Isabella Mastroeni. 2018. Abstract Non-Interference: A Unifying Framework for Weakening Information-flow. *ACM Trans. Priv. Secur.* 21, 2 (2018), 9:1–9:31. https://doi.org/10.1145/3175660

[43] Roberto Giacobazzi, Isabella Mastroeni, and Elia Perantoni. 2024. Adversities in Abstract Interpretation - Accommodating Robustness by Abstract Interpretation. *ACM Trans. Program. Lang. Syst.* 46, 2 (2024), 5. https://doi.org/10.1145/3649309

[44] Joseph A. Goguen. 1974. On Homomorphisms, Correctness, Termination, Unfoldments, and Equivalence of Flow Diagram Programs. *J. Comput. Syst. Sci.* 8, 3 (1974), 333–365. https://doi.org/10.1016/S0022-0000(74)80028-0

[45] Joseph A. Goguen and Grant Malcolm. 1996. *Algebraic semantics of imperative programs.* MIT Press.

[46] Joseph A. Goguen and José Meseguer. 1977. Correctness of Recursive Flow Diagram Programs. In *MFCS (Lecture Notes in Computer Science, Vol. 53)*. Springer, 580–595. https://doi.org/10.1007/3-540-08353-7_183

[47] Joseph A. Goguen and José Meseguer. 1982. Security Policies and Security Models. In *S&P*. IEEE Computer Society, 11–20. https://doi.org/10.1109/SP.1982.10014

[48] Joseph A. Goguen and José Meseguer. 1984. Unwinding and Inference Control. In *S&P*. IEEE Computer Society, 75–87. https://doi.org/10.1109/SP.1984.10019

[49] Joseph A. Goguen, James W. Thatcher, Eric G. Wagner, and Jesse B. Wright. 1977. Initial Algebra Semantics and Continuous Algebras. *J. ACM* 24, 1 (1977), 68–95. https://doi.org/10.1145/321992.321997

[50] Irène Guessarian. 1978. Some Applications of Algebraic Semantics. In *Mathematical Foundations of Computer Science 1978, Proceedings, 7th Symposium, Zakopane, Poland, September 4-8, 1978 (Lecture Notes in Computer Science, Vol. 64)*, Józef Winkowski (Ed.). Springer, 257–266. https://doi.org/10.1007/3-540-08921-7_73

[51] Reinhold Heckmann. 1993. Power Domains and Second-Order Predicates. *Theor. Comput. Sci.* 111, 1&2 (1993), 59–88. https://doi.org/10.1016/0304-3975(93)90182-S

[52] Eric C. R. Hehner. 1990. A Practical Theory of Programming. *Sci. Comput. Program.* 14, 2-3 (1990), 133–158. https://doi.org/10.1016/0167-6423(90)90018-9

[53] Eric C. R. Hehner. 1993. *A Practical Theory of Programming.* Springer. https://doi.org/10.1007/978-1-4419-8596-5

[54] Eric C. R. Hehner. 1999. Specifications, Programs, and Total Correctness. *Sci. Comput. Program.* 34, 3 (1999), 191–205. https://doi.org/10.1016/S0167-6423(98)00027-6

[55] Charles Antony Richard Hoare. 1969. An Axiomatic Basis for Computer Programming. *Commun. ACM* 12, 10 (1969), 576–580. https://doi.org/10.1145/363235.363259

[56] C. A. R. Hoare, Ian J. Hayes, Jifeng He, Carroll Morgan, A. W. Roscoe, Jeff W. Sanders, Ib Holm Sørensen, J. Michael Spivey, and Bernard Sufrin. 1987. Laws of Programming. *Commun. ACM* 30, 8 (1987), 672–686. https://doi.org/10.1145/27651.27653

[57] Tony Hoare. 2013. Generic Models of the Laws of Programming. In *Theories of Programming and Formal Methods (Lecture Notes in Computer Science, Vol. 8051).* Springer, 213–226. https://doi.org/10.1007/978-3-642-39698-4_13

[58] Tony Hoare. 2014. Laws of Programming: The Algebraic Unification of Theories of Concurrency. In *CONCUR (Lecture Notes in Computer Science, Vol. 8704).* Springer, 1–6. https://doi.org/10.1007/978-3-662-44584-6_1

[59] Tony Hoare and Stephan van Staden. 2014. The laws of programming unify process calculi. *Sci. Comput. Program.* 85 (2014), 102–114. https://doi.org/10.1016/J.SCICO.2013.08.012

[60] Iu. I. Ianov and M. D. Friedman. 1958. On The Equivalence and Transformation of Program Schemes. *Commun. ACM* 1, 10 (1958), 8–12. https://doi.org/10.1145/368924.368930

[61] James C. King. 1976. Symbolic Execution and Program Testing. *Commun. ACM* 19, 7 (1976), 385–394. https://doi.org/10.1145/360248.360252

[62] Dexter Kozen. 1997. Kleene Algebra with Tests. *ACM Trans. Program. Lang. Syst.* 19, 3 (1997), 427–443. https://doi.org/10.1145/256167.256195

[63] Dexter Kozen. 2000. On Hoare logic and Kleene algebra with tests. *ACM Trans. Comput. Log.* 1, 1 (2000), 60–76. https://doi.org/10.1145/343369.343378

[64] Xavier Leroy and Hervé Grall. 2009. Coinductive big-step operational semantics. *Inf. Comput.* 207, 2 (2009), 284–304. https://doi.org/10.1016/J.IC.2007.12.004

[65] Zohar Manna and Amir Pnueli. 1974. Axiomatic Approach to Total Correctness of Programs. *Acta Inf.* 3 (1974), 243–263. https://doi.org/10.1007/BF00288637

[66] Isabella Mastroeni and Michele Pasqua. 2017. Hyperhierarchy of Semantics - A Formal Framework for Hyperproperties Verification. In *SAS (Lecture Notes in Computer Science, Vol. 10422).* Springer, 232–252. https://doi.org/10.1007/978-3-319-66706-5_12

[67] Isabella Mastroeni and Michele Pasqua. 2018. Verifying Bounded Subset-Closed Hyperproperties. In *SAS (Lecture Notes in Computer Science, Vol. 11002).* Springer, 263–283. https://doi.org/10.1007/978-3-319-99725-4_17

[68] Isabella Mastroeni and Michele Pasqua. 2023. Domain Precision in Galois Connection-Less Abstract Interpretation. In *Static Analysis - 30th International Symposium, SAS 2023, Cascais, Portugal, October 22-24, 2023, Proceedings (Lecture Notes in Computer Science, Vol. 14284)*, Manuel V. Hermenegildo and José F. Morales (Eds.). Springer, 434–459. https://doi.org/10.1007/978-3-031-44245-2_19

[69] Daryl McCullough. 1987. Specifications for Multi-Level Security and a Hook-Up Property. In *S&P.* IEEE Computer Society, 161–166. https://doi.org/10.1109/SP.1987.10009

[70] John McLean. 1996. A General Theory of Composition for a Class of ”Possibilistic” Properties. *IEEE Trans. Software Eng.* 22, 1 (1996), 53–67. https://doi.org/10.1109/32.481534

[71] Bernhard Möller, Peter W. O’Hearn, and Tony Hoare. 2021. On Algebra of Program Correctness and Incorrectness. In *RAMiCS (Lecture Notes in Computer Science, Vol. 13027).* Springer, 325–343. https://doi.org/10.1007/978-3-030-88701-8_20

[72] James Donald Monk. 1969. *Introduction to Set Theory.* McGraw–Hill. http://euclid.colorado.edu/~monkd/monk11.pdf

[73] Alan Mycroft. 1982. *Abstract interpretation and optimising transformations for applicative programs.* Ph. D. Dissertation. University of Edinburgh, UK. https://hdl.handle.net/1842/6602

[74] Maurice Nivat. 1980. Non Deterministic Programs: An Algebraic Overview. In *IFIP Congress.* North-Holland/IFIP, 17–28.

[75] Peter W. O'Hearn. 2020. Incorrectness logic. *Proc. ACM Program. Lang.* 4, POPL (2020), 10:1–10:32. https://doi.org/10.1145/3371078

[76] Oystein Ore. 1943. Combinations of Closure Relations. *Annals of Mathematics* 44, 3 (July 1943), 514–533. https://doi.org/10.2307/1968978

[77] David Michael Ritchie Park. 1969. Fixpoint Induction and Proofs of Program Properties. In *Machine Intelligence Volume 5*, Donald Mitchie and Bernard Meltzer (Eds.). Edinburgh Univ. Press, Chapter 3, 59–78.

[78] Gordon D. Plotkin. 1976. A Powerdomain Construction. *SIAM J. Comput.* 5, 3 (1976), 452–487. https://doi.org/10.1137/0205035

[79] Robert Rand and Steve Zdancewic. 2015. VPHL: A Verified Partial-Correctness Logic for Probabilistic Programs. In *The 31st Conference on the Mathematical Foundations of Programming Semantics, MFPS 2015, Nijmegen, The Netherlands, June 22-25, 2015 (Electronic Notes in Theoretical Computer Science, Vol. 319)*, Dan R. Ghica (Ed.). Elsevier, 351–367. https://doi.org/10.1016/J.ENTCS.2015.12.021

[80] Dana S. Scott and Christopher Strachey. 1971. *Towards a Mathematical Semantics for Computer Languages.* Technical Report PRG-6. Oxford University Computer Laboratory. 49 pages. https://www.cs.ox.ac.uk/files/3228/PRG06.pdf

[81] Alfred Tarski. 1955. A Lattice Theoretical Fixpoint Theorem and Its Applications. *Pacific J. of Math.* 5 (1955), 285–310. https://doi.org/10.2140/pjm.1955.5.285

[82] Lena Verscht and Benjamin Lucien Kaminski. 2023. Hoare-Like Triples and Kleene Algebras with Top and Tests: Towards a Holistic Perspective on Hoare Logic, Incorrectness Logic, and Beyond. *CoRR* abs/2312.09662 (2023), 4 pages. https://doi.org/10.48550/ARXIV.2312.09662

[83] Morgan Ward. 1942. The Closure Operators of a Lattice. *Annals of Mathematics* 43, 2 (April 1942), 191–196. https://doi.org/10.2307/1968865

[84] Peng Yan, Hanru Jiang, and Nengkun Yu. 2022. On incorrectness logic for Quantum programs. *Proc. ACM Program. Lang.* 6, OOPSLA1 (2022), 1–28. https://doi.org/10.1145/3527316

[85] Mingsheng Ying. 2011. Floyd-Hoare logic for quantum programs. *ACM Trans. Program. Lang. Syst.* 33, 6 (2011), 19:1–19:49. https://doi.org/10.1145/2049706.2049708

[86] Steve Zdancewic and Andrew C. Myers. 2003. Observational Determinism for Concurrent Program Security. In *CSFW*. IEEE Computer Society, 29. https://doi.org/10.1109/CSFW.2003.1212703