Verification by Abstract Interpretation

Patrick Cousot

École normale supérieure, Département d'informatique
45 rue d'Ulm, 75230 Paris cedex 05, France
cousot@ens.fr, www.di.ens.fr/~cousot

Dedicated to Zohar Manna, for his 2^6th birthday.

Abstract. Abstract interpretation theory formalizes the idea of *abstraction* of mathematical structures, in particular those involved in the specification of properties and proof methods of computer systems. Verification by abstract interpretation is illustrated on the particular cases of *predicate abstraction*, which is *revisited* to handle infinitary abstractions, and on the new *parametric predicate abstraction*.

1 Introduction

Abstract interpretation theory [7,8,9,11,13] formalizes the idea of *abstraction* of mathematical structures, in particular those involved in the specification of properties and proof methods of computer systems.

Verification by abstract interpretation is illustrated on the particular cases of *predicate abstraction* [4,15,19] (where the finitary program-specific ground atomic propositional components of inductive invariants have to be provided) which is *revisited* (in that it is derived by systematic approximation of the concrete semantics of a programming language using an infinitary abstraction) and on the new *parametric predicate abstraction*, a program-independent generalization (where parameterized infinitary predicates are automatically combined by reduction and instantiated to particular programs by approximation).

2 Elements of Abstract Interpretation

Let us first recall a few elements of abstract interpretation from [7,9,11].

2.1 Properties and Their Abstraction. Given a set Σ of objects (such as program states, execution traces, etc.), we represent properties P of objects $s \in \Sigma$ as sets of objects $P \in \wp(\Sigma)$ (which have the considered property). Consequently, the set of properties of objects in Σ is a complete Boolean lattice $\langle \wp(\Sigma), \subseteq, \emptyset, \Sigma, \cup, \cap, \neg \rangle$. More generally, and up to an encoding, the object properties are assumed to belong to a complete lattice $\langle \mathcal{A}, \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle$.

By "*abstraction*", we understand a reasoning or (mechanical) computation on objects such that only some of the properties of these objects can be used. Let us call *concrete* the general properties in $\mathcal{A}$. Let $\overline{\mathcal{A}} \subseteq \mathcal{A}$ be the set of *abstract*

properties that can be used in the reasoning or computation. So, abstraction consists in *approximating* the concrete properties by the abstract ones. There are two possible directions of approximation. In the *approximation from above*, $P \in \mathcal{A}$ is over-approximated by $\overline{P} \in \overline{\mathcal{A}}$ such that $P \sqsubseteq \overline{P}$. In the *approximation from below*, $P \in \mathcal{A}$ is under-approximated by $\underline{P} \in \overline{\mathcal{A}}$ such that $\underline{P} \sqsubseteq P$. Obviously these notions are dual since an approximation from above/below for $\sqsubseteq/\sqsupseteq$ is an inverse approximation from below/above for $\sqsupseteq/\sqsubseteq$. Moreover, the complement dual of an approximation from above/below for P is an approximation from below/above for $\neg P$. Therefore, from a purely mathematical point of view, only approximations from above need to be presented.

We require $\top \in \overline{\mathcal{A}}$ to avoid that some concrete properties may have no abstraction (e.g. when $\overline{\mathcal{A}} = \emptyset$). Hence any concrete property $P \in \mathcal{A}$ can always be approximated from above (by $\top$ i.e. Σ, "true" or "*I don't know*"). For best precision we want to use *minimal abstractions* $\overline{P} \in \overline{\mathcal{A}}$, if any, such that $P \sqsubseteq \overline{P}$ and $\nexists \overline{P'} \in \overline{\mathcal{A}} : P \sqsubseteq \overline{P'} \sqsubsetneq \overline{P}$. If, for economy, we would like to avoid trying all possible minimal approximations, we can require that any concrete property $P \in \mathcal{A}$ has a *best abstraction* $\overline{P} \in \overline{\mathcal{A}}$, that is $P \sqsubseteq \overline{P}$ and $\forall \overline{P}' \in \overline{\mathcal{A}} : P \sqsubseteq \overline{P}' \Rightarrow \overline{P} \sqsubseteq \overline{P}'$ (otherwise see alternatives in [13]). By definition of the meet $\sqcap$, this hypothesis is equivalent to the fact that the meet of abstract properties should be abstract $\overline{P} = \bigsqcap\{\overline{P'} \in \overline{\mathcal{A}} \mid P \sqsubseteq \overline{P'}\} \in \overline{\mathcal{A}}$ (since otherwise $\bigsqcap\{\overline{P'} \in \overline{\mathcal{A}} \mid P \sqsubseteq \overline{P'}\}$ would have no best abstraction).

2.2 Moore Family-Based Abstraction. The hypothesis that any concrete property $P \in \mathcal{A}$ has a *best abstraction* $\overline{P} \in \overline{\mathcal{A}}$ implies that the set $\overline{\mathcal{A}}$ of abstract properties is a *Moore family* [11, Sec. 5.1] that is, by definition, $\overline{\mathcal{A}}$ is closed under meet i.e. $\mathbf{Mc}(\overline{\mathcal{A}}) = \overline{\mathcal{A}}$ where the *Moore-closure* $\mathbf{Mc}(X) \triangleq \{\bigsqcap S \mid S \subseteq X\}$ is the $\subseteq$-least Moore family containing $\overline{\mathcal{A}}$ and the set image $f(X)$ of a set $X \subseteq D$ by a map $f \in D \mapsto E$ is $f(X) \triangleq \{f(x) \mid x \in X\}$. In particular $\bigsqcap \emptyset = \top \in \overline{\mathcal{A}}$ so that any Moore family has a supremum. If the abstract domain $\overline{\mathcal{A}}$ is a Moore family of a complete lattice $\langle \mathcal{A}, \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle$ then it is a complete meet sub-semilattice $\langle \overline{\mathcal{A}}, \sqsubseteq, \sqcap\overline{\mathcal{A}}, \top, \lambda S \cdot \sqcap\{\overline{P} \in \overline{\mathcal{A}} \mid \sqcup S \subseteq \overline{P}\}, \sqcap \rangle$ of $\mathcal{A}$. The complete *lattice of abstractions* $\langle \mathbf{Mc}(\wp(\mathcal{A})), \supseteq, \mathcal{A}, \{\top\}, \lambda S \cdot \mathbf{Mc}(\cup S), \cap \rangle$ is the set of all abstractions i.e. of Moore families on the set $\mathcal{A}$ of concrete properties. $\bigcap_{i \in I} \overline{\mathcal{A}}_i$ is the most concrete among the abstract domains of $\mathbf{Mc}(\wp(\mathcal{A}))$ which are abstractions of all the abstract domains $\{\overline{\mathcal{A}}_i \mid i \in I\} \subseteq \mathbf{Mc}(\wp(\mathcal{A}))$. $\mathbf{Mc}(\bigcup_{i \in I} \overline{\mathcal{A}}_i)$ is the most abstract among the abstract domains of $\mathbf{Mc}(\wp(\mathcal{A}))$ which are more concrete than all the $\overline{\mathcal{A}}_i$'s and therefore isomorphic to the *reduced product* [11, Sec. 10.1]. The *disjunctive completion* of an abstract domain $\overline{\mathcal{A}}$ is the most abstract domain $\mathbf{Dc}(\overline{\mathcal{A}}) = \mathbf{Mc}(\{\sqcup X \mid X \subseteq \overline{\mathcal{A}}\})$ containing all concrete disjunctions of the abstract properties of $\overline{\mathcal{A}}$ [11, Sec. 9.2]. $\overline{\mathcal{A}}$ is disjunctive if and only if $\mathbf{Dc}(\overline{\mathcal{A}}) = \overline{\mathcal{A}}$.

2.3 Closure Operator-Based Abstraction. The map $\rho_{\overline{\mathcal{A}}}$ mapping a concrete property $P \in \mathcal{A}$ to its best abstraction $\rho_{\overline{\mathcal{A}}}(P)$ in the Moore family $\overline{\mathcal{A}}$ is

$\rho_{\overline{\mathcal{A}}}(P) \stackrel{\Delta}{=} \bigsqcap\{\overline{P} \in \overline{\mathcal{A}} \mid P \subseteq \overline{P}\}$. It is a *closure operator* [11, Sec. 5.2] (which is extensive ($\forall P \in \mathcal{A} : P \sqsubseteq \rho(P)$), idempotent ($\forall P \in \mathcal{A} : \rho(\rho(P)) = \rho(P)$) and monotone ($\forall P, P' \in \mathcal{A} : P \sqsubseteq P' \Rightarrow \rho(P) \sqsubseteq \rho(P')$)) such that $P \in \overline{\mathcal{A}} \Leftrightarrow P = \rho_{\overline{\mathcal{A}}}(P)$ hence $\overline{\mathcal{A}} = \rho_{\overline{\mathcal{A}}}(\mathcal{A})$. $\rho_{\overline{\mathcal{A}}}$ is called the *closure operator induced by the abstraction* $\overline{\mathcal{A}}$. Closure operators are isomorphic to their fixpoints hence to the Moore families. Therefore, any closure operator ρ on the set of properties $\mathcal{A}$ induces an abstraction $\rho(\mathcal{A})$. The abstract domain $\overline{\mathcal{A}} = \rho(\mathcal{A})$ defined by a closure operator ρ on a complete lattice of concrete properties $\langle \mathcal{A}, \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle$ is a complete lattice $\langle \rho(\mathcal{A}), \sqsubseteq, \rho(\bot), \top, \lambda S \bullet \rho(\sqcup S), \sqcap \rangle$. The set $\mathbf{uclo}(\mathcal{A} \mapsto \mathcal{A})$ of all abstractions, i.e. isomorphically, closure operators ρ on the set $\mathcal{A}$ of concrete properties is the complete *lattice of abstractions* for pointwise inclusion $\langle \mathbf{uclo}(\mathcal{A} \mapsto \mathcal{A}), \dot{\sqsubseteq}, \lambda P \bullet P, \lambda P \bullet \top, \lambda S \bullet \mathbf{ide}(\dot{\sqcup} S), \dot{\sqcap} \rangle$ where for all $\rho, \eta \in \mathbf{uclo}(\mathcal{A} \mapsto \mathcal{A})$, $\{\rho_i \mid i \in I\} \subseteq \mathbf{uclo}(\mathcal{A} \mapsto \mathcal{A})$ and $x \in \mathcal{A}$, $\rho \dot{\sqsubseteq} \eta \stackrel{\Delta}{=} \forall x \in \mathcal{A} : \rho(x) \subseteq \eta(x) \Leftrightarrow \eta(\mathcal{A}) \subseteq \rho(\mathcal{A})$, the glb $(\dot{\bigsqcap}_{i \in I} \rho_i)(x) \stackrel{\Delta}{=} \bigsqcap_{i \in I} \rho_i(x)$ is the *reduced product* and the lub $\mathbf{ide}(\dot{\bigsqcup}_{i \in I} \rho_i)$ where $\mathbf{ide}(\rho) = \mathrm{lfp}_{\rho}^{\dot{\sqsubseteq}} \lambda f \bullet f \circ f$ is the $\dot{\sqsubseteq}$-least idempotent operator on $\mathcal{A}$ $\dot{\sqsubseteq}$-greater than ρ satisfies $\mathbf{ide}(\dot{\bigsqcup}_{i \in I} \rho_i)(x) = x \Leftrightarrow \forall i \in I : \rho_i(x) = x$. The *disjunctive completion* of a closure operator $\rho \in \mathbf{uclo}(\mathcal{A} \mapsto \mathcal{A})$ is the most abstract closure $\mathbf{Dc}(\rho) = \dot{\sqcup}\{\eta \in \mathbf{uclo}(\mathcal{A} \mapsto \mathcal{A}) \cap \mathcal{A} \stackrel{\sqcup}{\longmapsto} \mathcal{A} \mid \eta \dot{\sqsubseteq} \rho\}$ which is more precise than ρ and is a *complete join morphism* (i.e. $f \in \mathcal{A} \stackrel{\sqcup}{\longmapsto} \mathcal{A}$ if and only if $\forall X \subseteq \mathcal{A} : f(\sqcup X) = \sqcup f(X)$) [11, Sec. 9.2].

2.4 Galois Connection-Based Abstraction. For closure operators ρ, we have $\rho(P) \sqsubseteq \rho(P') \Leftrightarrow P \sqsubseteq \rho(P')$ stating that $\rho(P')$ is an abstraction of a property P if and only if it $\sqsubseteq$-approximates its best abstraction $\rho(P)$. This can be written $\langle \mathcal{A}, \sqsubseteq \rangle \underset{\rho}{\overset{1}{\leftrightarrows}} \langle \rho(\mathcal{A}), \sqsubseteq \rangle$ where 1 is the identity and $\langle \mathcal{A}, \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \overline{\mathcal{D}}, \overline{\sqsubseteq} \rangle$ means that $\langle \alpha, \gamma \rangle$ is a *Galois surjection*, that is: $\forall P \in \mathcal{A}, \overline{P} \in \overline{\mathcal{D}} : \alpha(P) \mathrel{\overline{\sqsubseteq}} \overline{P} \Leftrightarrow P \sqsubseteq \gamma(\overline{P})$ and α is onto (equivalently $\alpha \circ \gamma = 1$ or γ is one-to-one). Reciprocally if $\langle \mathcal{A}, \sqsubseteq \rangle \underset{\rho}{\overset{1}{\leftrightarrows}} \langle \rho(\mathcal{A}), \sqsubseteq \rangle$ holds then ρ is a closure operator so that this can be taken as an equivalent definition of closure operators.

We can define an *abstract domain* as an isomorphic representation $\overline{\mathcal{D}}$ of the set $\overline{\mathcal{A}} \subseteq \mathcal{A} = \rho(\mathcal{A})$ of abstract properties (up to some order-isomorphism ι). Then, with such an encoding ι, we have the Galois surjection[1] $\langle \mathcal{A}, \sqsubseteq \rangle \underset{\iota \circ \rho}{\overset{\iota^{-1}}{\leftrightarrows}} \langle \overline{\mathcal{D}}, \overline{\sqsubseteq} \rangle$ More generally, the correspondence between concrete and abstract properties can be established by an arbitrary Galois surjection [11, Sec. 5.3] $\langle \mathcal{A}, \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \overline{\mathcal{D}}, \overline{\sqsubseteq} \rangle$. This is equivalent to the definition of the abstract domain $\overline{\mathcal{A}} \stackrel{\Delta}{=} \alpha \circ \gamma(\overline{\mathcal{D}})$ by the closure operator $\alpha \circ \gamma$ so the use of a Galois surjection is equivalent to that of a closure operator or a Moore family, up to the isomorphic representation $\alpha \circ \gamma$ of the abstract domain $\overline{\mathcal{A}} \cong \overline{\mathcal{D}}$.

Relaxing the condition that α is onto means that the same abstract property can have different representations. Then $\langle \mathcal{A}, \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \overline{\mathcal{D}}, \overline{\sqsubseteq} \rangle$ that is to say

[1] Also called Galois insertion since γ is injective.

$\forall P \in \mathcal{A}, \overline{P} \in \overline{\mathcal{D}} : \alpha(P) \mathrel{\overline{\sqsubseteq}} \overline{P} \Leftrightarrow P \sqsubseteq \gamma(\overline{P})$[2] or equivalently α is monotone ($\forall x, x' \in L : x \sqsubseteq x' \Rightarrow \alpha(x) \mathrel{\overline{\sqsubseteq}} \alpha(x')$, thus α preserves concrete implication $\sqsubseteq$ in the abstract), γ is monotone ($\forall y, y' \in M : y \mathrel{\overline{\sqsubseteq}} y' \Rightarrow \gamma(y) \sqsubseteq \gamma(y')$, thus γ preserves abstract implication $\mathrel{\overline{\sqsubseteq}}$ in the concrete), $\gamma \circ \alpha$ is extensive ($\forall x \in L : x \sqsubseteq \gamma(\alpha(x))$, so $\rho \triangleq \gamma \circ \alpha$ and the approximation is from above) and $\alpha \circ \gamma$ is reductive ($\forall y \in M : \alpha(\gamma(y)) \mathrel{\overline{\sqsubseteq}} y$, so concretization can loose no information). The composition $\alpha \circ \gamma$ is the identity if and only if α is onto or equivalently γ is one-to-one. If α is not onto, then the *reduction* of the abstract domain [12, Prop. 10] consists in considering the quotient $\overline{\mathcal{D}}/\equiv_\gamma$ by the equivalence $Q \equiv_\gamma Q' \Leftrightarrow \gamma(Q) = \gamma(Q')$, so that $\langle \mathcal{A}, \sqsubseteq \rangle \underset{\alpha_\equiv}{\overset{\gamma_\equiv}{\leftrightarrows\!\!\!\!\!\twoheadrightarrow}} \langle \overline{\mathcal{D}}/\equiv_\gamma, \overline{\sqsubseteq}_{\equiv_\gamma} \rangle$ is a Galois surjection where $\alpha_\equiv(P) \triangleq [\alpha(P)]_{\equiv_\gamma}$, $[Q]_{\equiv_\gamma} \triangleq \{Q' \in \overline{\mathcal{D}} \mid Q \equiv_\gamma Q'\}$, $\gamma_\equiv([Q]_{\equiv_\gamma}) \triangleq \gamma(Q)$ and $[P]_{\equiv_\gamma} \mathrel{\overline{\sqsubseteq}}_{\equiv_\gamma} [Q]_{\equiv_\gamma} \Leftrightarrow \exists P' \in [P]_{\equiv_\gamma} : \exists Q' \in [Q]_{\equiv_\gamma} : P' \mathrel{\overline{\sqsubseteq}} Q'$.

Observe that the inverse dual of $\langle \mathcal{A}, \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \overline{\mathcal{D}}, \overline{\sqsubseteq} \rangle$ is $\langle \overline{\mathcal{D}}, \overline{\sqsupseteq} \rangle \underset{\gamma}{\overset{\alpha}{\leftrightarrows}} \langle \mathcal{A}, \sqsupseteq \rangle$. The composition of Galois connections $\langle \mathcal{A}, \sqsubseteq \rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle \overline{\mathcal{D}}, \overline{\sqsubseteq} \rangle$ and $\langle \overline{\mathcal{D}}, \overline{\sqsubseteq} \rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle \overline{\overline{\mathcal{D}}}, \overline{\overline{\sqsubseteq}} \rangle$ is a Galois connection $\langle \mathcal{A}, \sqsubseteq \rangle \underset{\alpha_2 \circ \alpha_1}{\overset{\gamma_1 \circ \gamma_2}{\leftrightarrows}} \langle \overline{\overline{\mathcal{D}}}, \overline{\overline{\sqsubseteq}} \rangle$.

2.5 Function Abstraction. Given a complete lattice $\langle \mathcal{A}, \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle$ and an abstraction ρ on $\mathcal{A}$, the best abstraction of a monotone operator $f \in \mathcal{A} \overset{\text{mon}}{\longmapsto} \mathcal{A}$ on the complete lattice $\mathcal{A}$ is $\rho \circ f \in \rho(\mathcal{A}) \overset{\text{mon}}{\longmapsto} \rho(\mathcal{A})$ [11, Sec. 7.2]. Indeed given any other $\overline{f} \in \rho(\mathcal{A}) \overset{\text{mon}}{\longmapsto} \rho(\mathcal{A})$ and $\overline{x} \in \rho(\mathcal{A})$ the soundness requirement $\overline{f}(\overline{x}) \sqsupseteq f(\overline{x})$ implies $f(\overline{x}) \sqsubseteq \rho \circ \overline{f}(\overline{x})$ since ρ is idempotent whence $\rho \circ f(\overline{x}) \sqsubseteq \rho \circ \overline{f}(\overline{x})$ proving $\rho \circ f \mathrel{\dot{\sqsubseteq}} \overline{f}$ so that $\rho \circ f$ is more precise than any other sound abstraction $\overline{f} \in \rho(\mathcal{A}) \overset{\text{mon}}{\longmapsto} \rho(\mathcal{A})$ of $f \in \mathcal{A} \overset{\text{mon}}{\longmapsto} \mathcal{A}$. In terms of Galois connections, $\langle \mathcal{A}, \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \overline{\mathcal{A}}, \overline{\sqsubseteq} \rangle$ implies $\langle \mathcal{A} \overset{\text{mon}}{\longmapsto} \mathcal{A}, \mathrel{\dot{\sqsubseteq}} \rangle \underset{\lambda F \bullet \alpha \circ F \circ \gamma}{\overset{\lambda \overline{F} \bullet \gamma \circ \overline{F} \circ \alpha}{\leftrightarrows}} \langle \overline{\mathcal{A}} \overset{\text{mon}}{\longmapsto} \overline{\mathcal{A}}, \mathrel{\dot{\overline{\sqsubseteq}}} \rangle$.

2.6 Fixpoint Abstraction. Given a complete lattice $\langle \mathcal{A}, \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle$ and a monotone operator $f \in \mathcal{A} \overset{\text{mon}}{\longmapsto} \mathcal{A}$ on the complete lattice $\mathcal{A}$, its least fixpoint $\mathrm{lfp}_a^{\sqsubseteq} f$ greater than $a \in \mathcal{A}$ is defined, if it exists, as $a \sqsubseteq \mathrm{lfp}_a^{\sqsubseteq} f = f(\mathrm{lfp}_a^{\sqsubseteq} f)$ and $\forall x \in \mathcal{A} : a \sqsubseteq x = f(x) \Rightarrow \mathrm{lfp}_a^{\sqsubseteq} f \sqsubseteq x$. If $a \sqsubseteq f(a)$ then $\mathrm{lfp}_a^{\subseteq} f$ does exist and is the limit of the ultimately stationary transfinite sequence $f^\eta, \eta \in \mathbb{O}$ defined by $f^0 \triangleq a$, $f^{\eta+1} \triangleq f(f^\eta)$, for successor ordinals $\eta + 1$ and $f^\lambda \triangleq \bigcup_{\eta < \lambda} f^\eta$, for limit ordinals λ [7]. In particular the least fixpoint of f is $\mathrm{lfp}^{\sqsubseteq} f \triangleq \mathrm{lfp}_\bot^{\sqsubseteq} f$.

Given $f \in \mathcal{A} \overset{\text{mon}}{\longmapsto} \mathcal{A}$ on the complete lattice $\mathcal{A}$ and an abstraction ρ, we would like to approximate $\mathrm{lfp}_a^{\sqsubseteq} f$ in $\rho(\mathcal{A})$. The best abstraction $\rho(\mathrm{lfp}_a^{\sqsubseteq} f)$ is in general not computable since neither $\mathrm{lfp}_a^{\sqsubseteq} f$ nor ρ are. However, following [11, Sec. 7.1], a computable pointwise over approximation $\overline{f}$ is sufficient to check for an over approximation of $\mathrm{lfp}_a^{\sqsubseteq} f$ in $\rho(\mathcal{A})$ $\forall y \in \rho(\mathcal{A}) : \big(\rho \circ f \mathrel{\dot{\sqsubseteq}} \overline{f} \wedge \rho(a) \sqsubseteq y \wedge \overline{f}(y) \sqsubseteq y\big) \Rightarrow$

[2] In absence of best approximation, one can use a semi-connection, requiring only $\forall P \in \mathcal{A}, \overline{P} \in \overline{\mathcal{D}} : \alpha(P) \sqsubseteq \overline{P} \Rightarrow P \subseteq \gamma(\overline{P})$, see [13].

$\left(\mathrm{lfp}_a^{\sqsubseteq} f \sqsubseteq y\right)$. Moreover the $\sqsubseteq$-least such y is $\mathrm{lfp}_{\rho(a)}^{\sqsubseteq} \rho \circ f$ (which does exist since $a \sqsubseteq f(a)$ implies $\rho(a) \sqsubseteq \rho \circ f(\rho(a))$) such that $\rho(y) = y$. This means that we can abstract fixpoints by fixpoints. In terms of Galois connections, if $\langle \mathcal{A}, \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \overline{\mathcal{A}}, \overline{\sqsubseteq} \rangle$, $f \in \mathcal{A} \overset{\text{mon}}{\longmapsto} \mathcal{A}$ and $\overline{f} \in \overline{\mathcal{A}} \overset{\text{mon}}{\longmapsto} \overline{\mathcal{A}}$ then $\overline{f}$ is a *sound upper approximation* of f if and only if $\alpha \circ f \mathbin{\dot{\sqsubseteq}} \overline{f} \circ \alpha$ or equivalently $\alpha \circ f \circ \gamma \mathbin{\dot{\sqsubseteq}} \overline{f}$ where $\alpha \circ f \circ \gamma$ is the *best sound upper approximation* of f. This implies that $\alpha(\mathrm{lfp}_a^{\sqsubseteq} f) \sqsubseteq \mathrm{lfp}_{\alpha(a)}^{\overline{\sqsubseteq}} \overline{f}$. The approximation is said to be *complete* if and only if $\alpha \circ f = \overline{f} \circ \alpha$ in which case $\alpha(\mathrm{lfp}_a^{\sqsubseteq} f) = \mathrm{lfp}_{\alpha(a)}^{\overline{\sqsubseteq}} \overline{f}$.

An *iteration* for an operator $f \in \mathcal{A} \overset{\text{mon}}{\longmapsto} \mathcal{A}$ on the complete lattice $\mathcal{A}$ from $a \in \mathcal{A}$ *with widening* ∇ is $X^0 \triangleq a$, $X^{n+1} \triangleq \llbracket f(X^n) \sqsubseteq X^n \mathbin{?} X^n \mathbin{:} X^n \nabla f(X^n) \rrbracket$ [9] where the conditional notation is $\llbracket \mathtt{tt} \mathbin{?} t \mathbin{:} f \rrbracket = t$, $\llbracket \mathtt{ff} \mathbin{?} t \mathbin{:} f \rrbracket = f$ and $\llbracket b_1 \mathbin{?} t_1 \parallel b_2 \mathbin{?} t_2 \mathbin{:} f \rrbracket = \llbracket b_1 \mathbin{?} t_1 \mathbin{:} \llbracket b_2 \mathbin{?} t_2 \mathbin{:} f \rrbracket \rrbracket$. If the widening ensures that all such iterations are stationary at a finite rank $\ell < \omega$ and $a \sqsubseteq X^\ell$ then $f(X^\ell) \sqsubseteq X^\ell$ and so $\mathrm{lfp}_a^{\sqsubseteq} f \sqsubseteq X^\ell$. Widenings are also useful in absence of lubs for ascending chains in which case one requires in addition that $X \sqsubseteq X \nabla Y$ and $Y \sqsubseteq X \nabla Y$.

3 Application to Ground Predicate Abstraction

Predicate abstraction was introduced by [15] for computing invariants by abstract interpretation, following among others [4,10,19] where the general idea is that the atomic elements of the abstract domain are abstract predicates over program variables which interpretation is a set of program states (maybe memory states attached to program points). Predicate abstraction is restricted to a finitary abstraction [1,3,15,19]. For all such finite abstract domains, the transfer functions can be computed using a theorem prover ($\alpha \circ f \circ \gamma(P)$ is overapproximated by $\overline{f}(P) = \sqcap\{Q \in \overline{\mathcal{A}} \mid f \circ \gamma(P) \sqsubseteq \gamma(Q)\}$ where the $Q \in \overline{\mathcal{A}}$ are enumerated and the prover is asked for the proof $f \circ \gamma(P) \sqsubseteq \gamma(Q)$. Q is skipped if the prover fails which yields an overapproximation). All such finite abstract domains can be encoded with booleans so as to reuse existing model-checkers for fixpoint computations. We revisit predicate abstraction by systematic approximation of the concrete semantics of a programming language using an infinitary abstraction to *ground* (as opposed to parametric) predicates.

3.1 Syntax and Concrete Reachability Semantics of the Programming Language.

We consider a simple imperative programming language with programming variables $\mathtt{X} \in \mathbb{X}$, arithmetic expressions $E \in \mathbb{E}$ (? is the random choice), Boolean expressions $B \in \mathbb{B}$ and commands $C \in \mathbb{C}$:

$$
\begin{array}{l}
E ::= \mathtt{1} \mid \mathtt{X} \mid ? \mid E_1 - E_2, \\
B ::= E_1 < E_2 \mid \neg B_1 \mid B_1 \wedge B_2, \\
C ::= \mathtt{skip} \mid \mathtt{X} := E \mid C_1 \,;\, C_2 \mid \mathtt{if}\ B\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2 \mid \mathtt{while}\ B\ \mathtt{do}\ C_1\ .
\end{array}
$$

$\mathsf{var}[\![E]\!]$ (respectively $\mathsf{var}[\![B]\!]$, $\mathsf{var}[\![C]\!]$) is the set of programming variables appearing in the (Boolean) expression $E \in \mathbb{E}$ ($B \in \mathbb{B}$) or command $C \in \mathbb{C}$. Each command $C \in \mathbb{C}$ has unique labels $\mathsf{lab}[\![C]\!]$ to denote its execution points. These labels include an entry label $\mathsf{at}[\![C]\!]$, an exit label $\mathsf{after}[\![C]\!]$ and possibly labels of sub-commands $\mathsf{in}[\![C]\!]$. We assume that $\mathsf{lab}[\![C]\!] = \{\mathsf{at}[\![C]\!]\} \cup \mathsf{in}[\![C]\!] \cup \{\mathsf{after}[\![C]\!]\}$, $\mathsf{at}[\![C]\!] \neq \mathsf{after}[\![C]\!]$, $\{\mathsf{at}[\![C]\!], \mathsf{after}[\![C]\!]\} \cap \mathsf{in}[\![C]\!] = \emptyset$ and for $C_1 \,;\, C_2$, we have $\mathsf{lab}[\![C_1]\!] \cap \mathsf{lab}[\![C_2]\!] = \{\mathsf{after}[\![C_1]\!]\} = \{\mathsf{at}[\![C_2]\!]\}$.

We let $\mathcal{D}$ be the domain of value of the programming variables $\mathtt{X}$. The semantics $\mathcal{E}[\![E]\!] \in \mathcal{M} \mapsto \wp(\mathcal{D})$ of expressions $E \in \mathbb{E}$ is defined on all memory states $\mathcal{M} \stackrel{\triangle}{=} \mathcal{V} \mapsto \mathcal{D}$ where $\mathcal{V}$ is any set of programming variables including all such variables appearing in E, that is $\mathsf{var}[\![E]\!] \subseteq \mathcal{V}$. We define $\mathcal{E}[\![\mathtt{1}]\!]m \stackrel{\triangle}{=} \{1\}$, $\mathcal{E}[\![?]\!]m \stackrel{\triangle}{=} \mathcal{D}$, $\mathcal{E}[\![\mathtt{X}]\!]m \stackrel{\triangle}{=} \{m(\mathtt{X})\}$ and $\mathcal{E}[\![E_1 - E_2]\!]m \stackrel{\triangle}{=} \{v_1 - v_2 \mid v_1 \in \mathcal{E}[\![E_1]\!]m \wedge v_2 \in \mathcal{E}[\![E_2]\!]m\}$.

The semantics $\mathcal{B}[\![B]\!] \in \wp(\mathcal{M})$ and $\overline{\mathcal{B}}[\![B]\!] \in \wp(\mathcal{M})$ of Boolean expressions $B \in \mathbb{B}$ is defined for memory states in $\mathcal{M} \stackrel{\triangle}{=} \mathcal{V} \mapsto \mathcal{D}$ where $\mathsf{var}[\![B]\!] \subseteq \mathcal{V}$. $\mathcal{B}[\![B]\!]$ defines the set of memory states in which B may be true while $\overline{\mathcal{B}}[\![B]\!]$ defines the set of memory states in which B may be false. We have $\mathcal{B}[\![B]\!] \cup \overline{\mathcal{B}}[\![B]\!] = \mathcal{M}$ but maybe $\mathcal{B}[\![B]\!] \cap \overline{\mathcal{B}}[\![B]\!] \neq \emptyset$ because of non-determinism as in $? < \mathtt{1}$. We define $\mathcal{B}[\![E_1 < E_2]\!] \stackrel{\triangle}{=} \{m \in \mathcal{M} \mid \exists x_1, x_2 \in \mathcal{D} : x_1 \in \mathcal{E}[\![E_1]\!]m \wedge x_2 \in \mathcal{E}[\![E_2]\!]m \wedge x_1 < x_2\}$, $\mathcal{B}[\![\neg B]\!] \stackrel{\triangle}{=} \overline{\mathcal{B}}[\![B]\!]$, $\mathcal{B}[\![B_1 \wedge B_2]\!] \stackrel{\triangle}{=} \mathcal{B}[\![B_1]\!] \cap \mathcal{B}[\![B_2]\!]$, $\overline{\mathcal{B}}[\![E_1 < E_2]\!] \stackrel{\triangle}{=} \{m \in \mathcal{M} \mid \exists x_1, x_2 \in \mathcal{D} : x_1 \in \mathcal{E}[\![E_1]\!]m \wedge x_2 \in \mathcal{E}[\![E_2]\!]m \wedge x_1 \geq x_2\}$ $\overline{\mathcal{B}}[\![\neg B_1]\!] \stackrel{\triangle}{=} \mathcal{B}[\![B]\!]$ and $\overline{\mathcal{B}}[\![B_1 \wedge B_2]\!] \stackrel{\triangle}{=} \overline{\mathcal{B}}[\![B_1]\!] \cup \overline{\mathcal{B}}[\![B_2]\!]$.

The reachability semantics $\mathcal{R}[\![C]\!]$ of a command $C \in \mathbb{C}$ is defined on any set of states $\Sigma = \langle \mathcal{L}, \mathcal{M} \rangle$ such that $\mathcal{L}$ is a set of labels and $\mathcal{M} \stackrel{\triangle}{=} \mathcal{V} \mapsto \mathcal{D}$ is a set of memory states on variables $\mathcal{V}$ chosen such that $\mathsf{lab}[\![C]\!] \subseteq \mathcal{L}$ and $\mathsf{var}[\![C]\!] \subseteq \mathcal{V}$. The reachability semantics for the command C is then $\langle \Sigma, \mathcal{R}[\![C]\!] \rangle$. For a program C, we can choose $\Sigma = \langle \mathsf{lab}[\![C]\!], \mathsf{var}[\![C]\!] \mapsto \mathcal{D} \rangle$.

$$
\begin{array}{l}
\mathcal{R}[\![C]\!]P \stackrel{\triangle}{=} \text{match } C \text{ with} \\
\mid \quad \mathtt{skip} \;\rightarrow\; P \cup \{\langle \mathsf{after}[\![C]\!],\, m\rangle \mid \langle \mathsf{at}[\![C]\!],\, m\rangle \in P\} \\
\mid \quad \mathtt{X} := E \;\rightarrow\; P \cup \{\langle \mathsf{after}[\![C]\!],\, m[\mathtt{X} := v]\rangle \mid \langle \mathsf{at}[\![C]\!],\, m\rangle \in P \wedge v \in \mathcal{E}[\![E]\!]m\} \\
\mid \quad C_1 \,;\, C_2 \;\rightarrow\; \mathcal{R}[\![C_1]\!]P \cup \mathcal{R}[\![C_2]\!]\{\langle \mathsf{at}[\![C_2]\!],\, m\rangle \mid \langle \mathsf{after}[\![C_1]\!],\, m\rangle \in \mathcal{R}[\![C_1]\!]P\} \\
\mid \quad C = \mathtt{if}\; B\; \mathtt{then}\; C_1\; \mathtt{else}\; C_2 \;\rightarrow\; P \cup P_1 \cup P_2 \cup P_e \\
\qquad \text{where} \quad P_1 \stackrel{\triangle}{=} \mathcal{R}[\![C_1]\!]\{\langle \mathsf{at}[\![C_1]\!],\, m\rangle \mid \langle \mathsf{at}[\![C]\!],\, m\rangle \in P \wedge m \in \mathcal{B}[\![B]\!]\} \\
\qquad \text{and} \quad P_2 \stackrel{\triangle}{=} \mathcal{R}[\![C_2]\!]\{\langle \mathsf{at}[\![C_2]\!],\, m\rangle \mid \langle \mathsf{at}[\![C]\!],\, m\rangle \in P \wedge m \in \overline{\mathcal{B}}[\![B]\!]\} \\
\qquad \text{and} \quad P_e \stackrel{\triangle}{=} \{\langle \mathsf{after}[\![C]\!],\, m\rangle \mid \langle \mathsf{after}[\![C_1]\!],\, m\rangle \in P_1 \vee \langle \mathsf{after}[\![C_2]\!],\, m\rangle \in P_2\} \\
\mid \quad C = \mathtt{while}\; B\; \mathtt{do}\; C_1 \;\rightarrow\; \text{let} \quad P_1 = \mathrm{lfp}^{\subseteq}\, \lambda X \bullet \mathcal{R}[\![C_1]\!](\{\langle \mathsf{at}[\![C_1]\!],\, m\rangle \mid \qquad (1) \\
\qquad\qquad (\langle \mathsf{at}[\![C]\!],\, m\rangle \in P \vee \langle \mathsf{after}[\![C_1]\!],\, m\rangle \in X) \wedge m \in \mathcal{B}[\![B]\!]\}) \quad \text{in} \\
P \cup P_1 \cup \{\langle \mathsf{after}[\![C]\!],\, m\rangle \mid (\langle \mathsf{at}[\![C]\!],\, m\rangle \in P \vee \langle \mathsf{after}[\![C_1]\!],\, m\rangle \in P_1) \wedge m \in \overline{\mathcal{B}}[\![B]\!]\}
\end{array}
$$

3.2 Ground Abstract Predicates. Predicate abstraction is defined by a set $\mathfrak{P}$ of syntactic predicates that, for simplicity, we choose to be Boolean ex-

pressions $\mathfrak{P} \subseteq \mathbb{B}$. We define $\mathsf{var}[\![\mathfrak{P}]\!] \triangleq \bigcup_{\mathtt{p} \in \mathfrak{P}} \mathsf{var}[\![\mathtt{p}]\!]$. The set $\mathfrak{P}$ may be infinite as e.g. in the case of Kildall's constant propagation [16] for which $\mathfrak{P} \triangleq \{\mathtt{tt}, \mathtt{ff}\} \cup \bigcup_{\mathtt{X} \in \mathsf{var}[\![C]\!]} \bigcup_{v \in \mathcal{D}} \{\mathtt{X} = v\}$.

Predicate abstraction uses a prover to prove theorems $t \in \mathbb{T}$ with interpretation $\mathcal{I} \in \mathbb{T} \mapsto \wp(\mathcal{M})$ assigning an interpretation $\mathcal{I}[\![t]\!]$ to all syntactic predicates $t \in \mathbb{T}$ with syntax ($\mathtt{p} \in \mathfrak{P}$):

$$t ::= \mathtt{p} \mid \mathtt{tt} \mid \mathtt{ff} \mid \mathtt{X} \mid \neg t_1 \mid t_1 \Rightarrow t_2 \mid \bigwedge_{i \in \Delta} t_i \mid \forall \mathtt{X} : t_1$$

with free variables $\mathsf{var}[\![t]\!]$ and semantics $\mathcal{I}[\![t]\!] \in \wp(\mathcal{M})$ defined by $\mathcal{I}[\![\mathtt{p}]\!] \triangleq \mathcal{B}[\![\mathtt{p}]\!]$, $\mathcal{I}[\![\mathtt{tt}]\!] \triangleq \mathcal{M}$, $\mathcal{I}[\![\mathtt{ff}]\!] \triangleq \emptyset$, $\mathcal{I}[\![\mathtt{X} \in E]\!] \triangleq \{m \in \mathcal{M} \mid m(\mathtt{X}) \in \mathcal{E}[\![E]\!]m\}$, $\mathcal{I}[\![\neg t]\!] \triangleq \{m \in \mathcal{M} \mid m \notin \mathcal{I}[\![t]\!]\}$, $\mathcal{I}[\![\bigwedge_{i \in \Delta} t_i]\!] \triangleq \bigcap_{i \in \Delta} \mathcal{I}[\![t_i]\!]$, $\mathcal{I}[\![t_1 \Rightarrow t_2]\!] \triangleq \{m \in \mathcal{M} \mid m \notin \mathcal{I}[\![t_1]\!] \vee m \in \mathcal{I}[\![t_2]\!]\}$ and $\mathcal{I}[\![\forall \mathtt{X} : t]\!] \triangleq \{m \in \mathcal{M} \mid \forall v \in \mathcal{D} : m[\mathtt{X} := v] \in \mathcal{I}[\![t]\!]\}$. Variable substitution $t[\mathtt{X}/\mathtt{X}']$ is defined as usual with renaming of conflicting dummy variables such that:

$$\begin{aligned}
\mathtt{X} \notin \mathsf{var}[\![t]\!] &\Rightarrow t[\mathtt{X}/\mathtt{X}'] = t, \\
\mathcal{E}[\![E[\mathtt{X}/\mathtt{X}']]\!]m &= \mathcal{E}[\![E]\!]m[\mathtt{X} := m(\mathtt{X}')], \qquad (2) \\
\mathtt{X} \neq \mathtt{Y} \wedge \mathtt{Y} \notin \mathsf{var}[\![t]\!] &\Rightarrow \mathcal{I}[\![\forall \mathtt{X} : t]\!] = \mathcal{I}[\![\forall \mathtt{Y} : (t[\mathtt{X}/\mathtt{Y}])]\!], \\
\mathtt{Z} \notin \mathsf{var}[\![t]\!] \cup \{\mathtt{X}, \mathtt{Y}\} &\Rightarrow \mathcal{I}[\![(\forall \mathtt{X} : t)[\mathtt{Y}/\mathtt{X}]]\!] = \mathcal{I}[\![\forall \mathtt{Z} : (t[\mathtt{X}/\mathtt{Z}][\mathtt{Y}/\mathtt{X})]]\!], \\
\mathcal{I}[\![t[\mathtt{X}/\mathtt{X}']]\!] &= \{m \mid m[\mathtt{X} := m(\mathtt{X}')] \in \mathcal{I}[\![t]\!]\} \ . \qquad (3)
\end{aligned}$$

The prover is assumed to be sound in that $\forall t \in \mathbb{T} : \mathrm{prover}[\![t]\!] \ \Rightarrow \ (\mathcal{I}[\![t]\!] = \mathcal{M})$. (The inverse is not valid since provers are incomplete.)

3.3 Ground Predicate Abstraction. Given a set of states in $\mathcal{A}$ where $\Sigma = \langle \mathcal{L}, \mathcal{M} \rangle$, we can use an isomorphic representation associating sets of memory states to labels thank to the following correspondence:

$$\langle \wp(\mathcal{L} \times \mathcal{M}), \subseteq \rangle \underset{\alpha_\downarrow}{\overset{\gamma_\downarrow}{\leftrightarrows}} \langle \mathcal{L} \mapsto \wp(\mathcal{M}), \dot{\subseteq} \rangle$$

where $\alpha_\downarrow(P) \triangleq \lambda \ell \cdot \{m \mid \langle \ell, m \rangle \in P\}$, $\gamma_\downarrow(Q) \triangleq \{\langle \ell, m \rangle \mid \ell \in \mathcal{L} \wedge m \in Q_\ell\}$ and $\dot{\subseteq}$ is the pointwise ordering $Q \dot{\subseteq} Q'$ if and only if $\forall \ell \in \mathcal{L} : Q_\ell \subseteq Q'_\ell$.

A memory state property $Q \in \wp(\mathcal{M})$ is approximated by the subset of predicates p of $\mathfrak{P}$ which holds when Q holds (formally $Q \subseteq \mathcal{B}[\![p]\!]$). This defines a Galois connection:

$$\langle \wp(\mathcal{M}), \subseteq \rangle \underset{\alpha_{\mathfrak{P}}}{\overset{\gamma_{\mathfrak{P}}}{\leftrightarrows}} \langle \wp(\mathfrak{P}), \supseteq \rangle$$

where $\alpha_{\mathfrak{P}}(Q) \triangleq \{\mathtt{p} \in \mathfrak{P} \mid Q \subseteq \mathcal{B}[\![\mathtt{p}]\!]\}$ and $\gamma_{\mathfrak{P}}(P) \triangleq \bigcap \{\mathcal{B}[\![\mathtt{p}]\!] \mid \mathtt{p} \in P\} = \mathcal{I}[\![\bigwedge P]\!]$. Observe that in general $\gamma_{\mathfrak{P}}$ is not one-to-one (e.g. $\gamma_{\mathfrak{P}}(\{\mathtt{X} = 1, \mathtt{X} \geq 1\}) = \gamma_{\mathfrak{P}}(\{\mathtt{X} = 1\})$ so $\alpha_{\mathfrak{P}}$ is not onto[3]. By pointwise extension, we have:

[3] In a Galois connection $\langle L, \leq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \sqsubseteq \rangle$, α is onto if and only if γ is one-to-one if and only if $\gamma \circ \alpha$ is the identity so, by duality, γ is onto if and only if α is one-to-one if and only if $\alpha \circ \gamma$ is the identity.

$$\langle \mathcal{L} \mapsto \wp(\mathcal{M}),\ \dot{\subseteq} \rangle \underset{\dot{\alpha}_{\mathfrak{P}}}{\overset{\dot{\gamma}_{\mathfrak{P}}}{\leftrightarrows}} \langle \mathcal{L} \mapsto \wp(\mathfrak{P}),\ \dot{\supseteq} \rangle$$

where $\dot{\alpha}_{\mathfrak{P}}(Q) \stackrel{\Delta}{=} \lambda \ell \bullet \alpha_{\mathfrak{P}}(Q_\ell)$, $\dot{\gamma}_{\mathfrak{P}}(P) \stackrel{\Delta}{=} \lambda \ell \bullet \gamma_{\mathfrak{P}}(P_\ell)$ and $P \dot{\supseteq} P' \stackrel{\Delta}{=} \forall \ell \in \mathcal{L} : P_\ell \supseteq P'_\ell$. By composition, we get:

$$\langle \wp(\mathcal{L} \times \mathcal{M}),\ \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{L} \mapsto \wp(\mathfrak{P}),\ \dot{\supseteq} \rangle \tag{4}$$

where $\alpha(P) \stackrel{\Delta}{=} \dot{\alpha}_{\mathfrak{P}} \circ \alpha_\downarrow(P) = \lambda \ell \bullet \{\mathfrak{p} \in \mathfrak{P} \mid \{m \mid \langle \ell,\ m \rangle \in P\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\}$ and $\gamma(Q) \stackrel{\Delta}{=} \gamma_\downarrow \circ \dot{\gamma}_{\mathfrak{P}} = \{\langle \ell, m \rangle \mid \ell \in \mathcal{L} \wedge m \in \gamma_{\mathfrak{P}}(Q_\ell)\} = \{\langle \ell,\ m \rangle \mid \ell \in \mathcal{L} \wedge \forall \mathfrak{p} \in Q_\ell : m \in \mathcal{I}[\![\mathfrak{p}]\!]\}$.

If $\mathfrak{P}$ is assumed to be finite then characteristic functions of subsets of $\mathfrak{P}$ can be encoded as Boolean vectors thus later allowing for the reuse of model-checker to solve fixpoint equations. However, this encoding of sets cannot be used to encode infinite sets such as Kildall's constant propagation [16].

3.4 Abstract Reachability Semantics of Commands $C \in \mathbb{C}$. Given a set of abstract predicates $\mathfrak{P}$, the abstract reachability semantics $\overline{\mathcal{R}}[\![C]\!] \in \wp(\mathfrak{P}) \mapsto (\mathsf{lab}[\![C]\!] \mapsto \wp(\mathfrak{P}))$ of command $C \in \mathbb{C}$ is

$$\overline{\mathcal{R}}[\![C]\!]\overline{P} = \alpha(\mathcal{R}[\![C]\!](\{\mathsf{at}[\![C]\!]\} \times \gamma_{\mathfrak{P}}(\overline{P})))$$

that is the abstraction of the reachable states of C from its entry point $\mathsf{at}[\![C]\!]$ in initial memory states $m \in \gamma_{\mathfrak{P}}(\overline{P})$.

Because of undecidability, whence theorem prover incompleteness, we look for a $\dot{\supseteq}$-over-approximation $\overline{\mathcal{R}}[\![C]\!]\overline{P}$ such that:

$$\alpha(\mathcal{R}[\![C]\!](\{\mathsf{at}[\![C]\!]\} \times \gamma_{\mathfrak{P}}(\overline{P}))) \dot{\supseteq} \overline{\mathcal{R}}[\![C]\!]\overline{P} \tag{5}$$

$$\Leftrightarrow \quad \mathcal{R}[\![C]\!](\{\mathsf{at}[\![C]\!]\} \times \gamma_{\mathfrak{P}}(\overline{P})) \subseteq \gamma(\overline{\mathcal{R}}[\![C]\!]\overline{P}) \tag{6}$$

We proceed by structural induction on the syntax of commands $C \in \mathbb{C}$. For short, we only consider the case of assignment and iteration.

— For the assignment $\mathtt{X} := E$, we have

$$\begin{aligned}
&\overline{\mathcal{R}}[\![\mathtt{X} := E]\!]\overline{P} \\
=\ &\alpha(\mathcal{R}[\![\mathtt{X} := E]\!](\{\mathsf{at}[\![\mathtt{X} := E]\!]\} \times \gamma_{\mathfrak{P}}(\overline{P}))) && \text{〈def. } \overline{\mathcal{R}}\text{〉} \\
=\ &\text{let } P = \{\mathsf{at}[\![\mathtt{X} := E]\!]\} \times \gamma_{\mathfrak{P}}(\overline{P}) \text{ in} \\
&\quad \alpha(P \cup \{\langle \mathsf{after}[\![C]\!],\ m[\mathtt{X} := v] \rangle \mid \langle \mathsf{at}[\![C]\!],\ m \rangle \in P \wedge v \in \mathcal{E}[\![E]\!]m\}) && \text{〈def. } \mathcal{R}\text{〉} \\
=\ &\text{let } P = \{\mathsf{at}[\![\mathtt{X} := E]\!]\} \times \gamma_{\mathfrak{P}}(\overline{P}) \text{ in} \\
&\quad \alpha(P) \dot{\cap} \alpha(\{\langle \mathsf{after}[\![C]\!],\ m[\mathtt{X} := v] \rangle \mid \langle \mathsf{at}[\![C]\!],\ m \rangle \in P \wedge v \in \mathcal{E}[\![E]\!]m\}) \\
&\qquad\qquad \text{〈by (4), so that } \alpha \text{ is a complete join morphism〉}
\end{aligned}$$

Let us go on with the first term $\alpha(P)$ of the form $P = \{\mathsf{loc}[\![C]\!]\} \times \gamma_{\mathfrak{P}}(\overline{P})$ with $\mathsf{loc} \in \{\mathsf{at}, \mathsf{after}\}$.

$$
\begin{array}{ll}
& \alpha(\{\mathsf{loc}[\![C]\!]\} \times \gamma_{\mathfrak{P}}(\overline{P})) \\
= & \lambda\ell\bullet\{\mathfrak{p} \in \mathfrak{P} \mid \{m \mid \langle \ell, m\rangle \in \{\mathsf{loc}[\![C]\!]\} \times \gamma_{\mathfrak{P}}(\overline{P})\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} & \text{ʅdef. } \alpha\text{ʃ} \\
= & \lambda\ell\bullet(\!|\ell = \mathsf{loc}[\![C]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \gamma_{\mathfrak{P}}(\overline{P}) \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \{\mathfrak{p} \in \mathfrak{P} \mid \emptyset \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\}\ |\!) & \text{ʅconditionalʃ} \\
= & \lambda\ell\bullet(\!|\ell = \mathsf{loc}[\![C]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \bigcap\{\mathcal{B}[\![\mathfrak{p}]\!] \mid \mathfrak{p} \in \overline{P}\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}\ |\!) & \text{ʅdef. } \gamma_{\mathfrak{P}} \text{ and } \subseteq\text{ʃ} \\
= & \lambda\ell\bullet(\!|\ell = \mathsf{loc}[\![C]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \mathcal{I}[\![\bigwedge \overline{P} \Rightarrow \mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}\ |\!) & \text{ʅdef. } \mathcal{I}\text{ʃ} \\
\dot{\supseteq} & \lambda\ell\bullet(\!|\ell = \mathsf{loc}[\![C]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \mathrm{prover}[\![\bigwedge \overline{P} \Rightarrow \mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}\ |\!) & \text{ʅprover soundnessʃ} \\
\dot{\supseteq} & \lambda\ell\bullet(\!|\ell = \mathsf{loc}[\![C]\!] \mathbin{?} \overline{P} \mathbin{:} \mathfrak{P}\ |\!) \quad \text{ʅless precise but avoids a call to the proverʃ} & (7)
\end{array}
$$

Given $P = \{\mathsf{at}[\![\mathtt{X} := E]\!]\} \times \gamma_{\mathfrak{P}}(\overline{P})$, the second term is

$$
\begin{array}{ll}
& \alpha(\{\langle \mathsf{after}[\![\mathtt{X} := E]\!],\ m[\mathtt{X} := v]\rangle \mid \langle \mathsf{at}[\![\mathtt{X} := E]\!],\ m\rangle \in P \wedge v \in \mathcal{E}[\![E]\!]m\}) \\
= & \alpha(\{\langle \mathsf{after}[\![\mathtt{X} := E]\!],\ m[\mathtt{X} := v]\rangle \mid m \in \gamma_{\mathfrak{P}}(\overline{P}) \wedge v \in \mathcal{E}[\![E]\!]m\}) & \text{ʅdef. } P\text{ʃ} \\
= & \lambda\ell\bullet\{\mathfrak{p} \in \mathfrak{P} \mid \{m' \mid \langle \ell, m'\rangle \in \{\langle \mathsf{after}[\![\mathtt{X} := E]\!],\ m[\mathtt{X} := v]\rangle \mid m \in \gamma_{\mathfrak{P}}(\overline{P}) \wedge v \in \mathcal{E}[\![E]\!]m\}\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} & \text{ʅdef. } \alpha\text{ʃ} \\
= & \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m[\mathtt{X} := v] \mid m \in \gamma_{\mathfrak{P}}(\overline{P}) \wedge v \in \mathcal{E}[\![E]\!]m\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}\ |\!) & \text{ʅdef. conditional and } \subseteq\text{ʃ} \\
= & \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m[\mathtt{X} := v] \mid m \in \bigcap\{\mathcal{B}[\![\mathfrak{p}]\!] \mid \mathfrak{p} \in \overline{P}\} \wedge v \in \mathcal{E}[\![E]\!]m\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}\ |\!) & \text{ʅdef. } \gamma_{\mathfrak{P}}\text{ʃ} \\
= & \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m[\mathtt{X} := v] \mid m \in \mathcal{I}[\![\bigwedge \overline{P}]\!] \wedge v \in \mathcal{E}[\![E]\!]m\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}\ |\!) & \text{ʅdef. } \mathcal{I}\text{ʃ} \\
= & \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m[\mathtt{X} := v'][\mathtt{X} := v] \mid m[\mathtt{X} := v'] \in \mathcal{I}[\![\bigwedge \overline{P}]\!] \wedge v \in \mathcal{E}[\![E]\!]m[\mathtt{X} := v']\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}\ |\!) & \text{ʅby letting } v' = m(\mathtt{X}) \text{ so that } m = m[\mathtt{X} := v']\text{ʃ} \\
= & \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m[\mathtt{X} := v] \mid \exists v' \in \mathcal{D} : m[\mathtt{X} := v'] \in \mathcal{I}[\![\bigwedge \overline{P}]\!] \wedge v \in \mathcal{E}[\![E]\!]m[\mathtt{X} := v']\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}\ |\!) & \text{ʅsince } m[\mathtt{X} := v'][\mathtt{X} := v] = m[\mathtt{X} := v]\text{ʃ} \\
= & \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m[\mathtt{X} := v] \mid m[\mathtt{X} := m(\mathtt{X}')] \in \mathcal{I}[\![\bigwedge \overline{P}]\!] \wedge v \in \mathcal{E}[\![E]\!]m[\mathtt{X} := m(\mathtt{X}')]\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}\ |\!) & \text{ʅby letting } v' = m(\mathtt{X}') \text{ where } \mathtt{X}' \text{ is a fresh variable such that } \mathtt{X}' \notin \{\mathtt{X}\} \cup \mathsf{var}[\![E]\!] \cup \mathsf{var}[\![\mathfrak{P}]\!] \text{ so that } \mathtt{X}' \notin \mathsf{var}[\![\overline{P}]\!] \text{ and neither } \mathcal{I}[\![\bigwedge \overline{P}]\!],\ \mathcal{E}[\![E]\!] \text{ nor } \mathcal{I}[\![\mathfrak{p}]\!] \text{ depend upon the value of } \mathtt{X}'\text{ʃ} \\
= & \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m' \mid m'[\mathtt{X} := m'(\mathtt{X}')] \in \mathcal{I}[\![\bigwedge \overline{P}]\!] \wedge m'(\mathtt{X}) \in \mathcal{E}[\![E]\!]m'[\mathtt{X} := m'(\mathtt{X}')]\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}\ |\!) & \text{ʅby letting } m' = m[\mathtt{X} := v] \text{ so that } v = m'(\mathtt{X}),\ m(\mathtt{X}') = m'(\mathtt{X}') \text{ and } m[\mathtt{X} := m(\mathtt{X}')] = m'[\mathtt{X} := m'(\mathtt{X}')]\text{ʃ} \\
= & \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m \in \mathcal{M} \mid \exists v \in \mathcal{D} : m[\mathtt{X} := m(\mathtt{X}')] \in \mathcal{I}[\![(\bigwedge \overline{P})]\!] \wedge m(\mathtt{X}) \in \mathcal{E}[\![E]\!]m[\mathtt{X} := v]\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}\ |\!) & \text{ʅby letting } m = m' \text{ and } v = m'(\mathtt{X}')\text{ʃ} \\
= & \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m \in \mathcal{M} \mid \exists v \in \mathcal{D} : m[\mathtt{X}' := v][\mathtt{X} := m(\mathtt{X}')] \in \mathcal{I}[\![(\bigwedge \overline{P})]\!] \wedge m(\mathtt{X}) \in \mathcal{E}[\![E]\!]m[\mathtt{X} := v]\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}\ |\!) & \text{ʅsince } \mathtt{X}' \notin \mathsf{var}[\![\overline{P}]\!]\text{ʃ}
\end{array}
$$

$= \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m \in \mathcal{M} \mid \exists v \in \mathcal{D} : m[\mathtt{X}' := v][\mathtt{X} := m[\mathtt{X} := v](\mathtt{X}')] \in \mathcal{I}[\![(\bigwedge \overline{P})]\!] \wedge m(\mathtt{X}) \in \mathcal{E}[\![E]\!]m[\mathtt{X} := v]\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}|\!) \quad$ ⟅since $\mathtt{X} \neq \mathtt{X}'$ so $m(\mathtt{X}') = m[\mathtt{X} := v](\mathtt{X}')$⟆

$= \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m \in \mathcal{M} \mid \exists v \in \mathcal{D} : m[\mathtt{X}' := v] \in \{m' \mid m'[\mathtt{X} := m'(\mathtt{X}')] \in \mathcal{I}[\![(\bigwedge \overline{P})]\!]\} \wedge m(\mathtt{X}) \in \mathcal{E}[\![E]\!]m[\mathtt{X} := v]\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}|\!)$ ⟅by def. $\in$ while letting $m' = m[\mathtt{X} := v]$⟆

$= \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m \in \mathcal{M} \mid \exists v \in \mathcal{D} : m[\mathtt{X}' := v] \in \mathcal{I}[\![(\bigwedge \overline{P})[\mathtt{X}/\mathtt{X}']]\!] \wedge m(\mathtt{X}) \in \mathcal{E}[\![E]\!]m[\mathtt{X} := v]\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}|\!)$ ⟅by (3)⟆

$= \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m \in \mathcal{M} \mid \exists v \in \mathcal{D} : m[\mathtt{X}' := v] \in \mathcal{I}[\![(\bigwedge \overline{P})[\mathtt{X}/\mathtt{X}']]\!] \wedge m[\mathtt{X}' := v](\mathtt{X}) \in \mathcal{E}[\![E]\!]m[\mathtt{X} := m[\mathtt{X}' := v](\mathtt{X}')]\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}|\!)$ ⟅def. $m[\mathtt{X} := v]$ and $\mathtt{X} \neq \mathtt{X}'$⟆

$= \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m \in \mathcal{M} \mid \exists v \in \mathcal{D} : m[\mathtt{X}' := v] \in \mathcal{I}[\![(\bigwedge \overline{P})[\mathtt{X}/\mathtt{X}']]\!] \wedge m[\mathtt{X}' := v](\mathtt{X}) \in \mathcal{E}[\![E]\!]m[\mathtt{X}' := v][\mathtt{X} := m[\mathtt{X}' := v](\mathtt{X}')]\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}|\!)$ ⟅since $\mathtt{X}' \notin \mathsf{var}[\![E]\!]$⟆

$= \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m \in \mathcal{M} \mid \exists v \in \mathcal{D} : m[\mathtt{X}' := v] \in \mathcal{I}[\![(\bigwedge \overline{P})[\mathtt{X}/\mathtt{X}']]\!] \wedge m[\mathtt{X}' := v] \in \{m' \in \mathcal{M} \mid m'(\mathtt{X}) \in \mathcal{E}[\![E]\!]m'[\mathtt{X} := m'(\mathtt{X}')]\}\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}|\!)$ ⟅def. $\in$ while letting $m' = m[\mathtt{X}' := v]$⟆

$= \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m \in \mathcal{M} \mid \exists v \in \mathcal{D} : m[\mathtt{X}' := v] \in \mathcal{I}[\![(\bigwedge \overline{P})[\mathtt{X}/\mathtt{X}']]\!] \cap \{m' \in \mathcal{M} \mid m'(\mathtt{X}) \in \mathcal{E}[\![E]\!]m'[\mathtt{X} := m'(\mathtt{X}')]\}\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}|\!)$ ⟅def. $\cap$⟆

$= \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m \in \mathcal{M} \mid \exists v \in \mathcal{D} : m[\mathtt{X}' := v] \in \mathcal{I}[\![(\bigwedge \overline{P})[\mathtt{X}/\mathtt{X}']]\!] \cap \{m' \in \mathcal{M} \mid m'(\mathtt{X}) \in \mathcal{E}[\![E[\mathtt{X}/\mathtt{X}']]\!]m'\}\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}|\!)$ ⟅by (2)⟆

$= \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m \in \mathcal{M} \mid \exists v \in \mathcal{D} : m[\mathtt{X}' := v] \in \mathcal{I}[\![(\bigwedge \overline{P})[\mathtt{X}/\mathtt{X}']]\!] \cap \mathcal{I}[\![\mathtt{X} \in E[\mathtt{X}/\mathtt{X}']]\!]\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}|\!)$ ⟅def. $\mathcal{I}$⟆

$= \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m \in \mathcal{M} \mid \exists v \in \mathcal{D} : m[\mathtt{X}' := v] \in \mathcal{I}[\![\bigwedge(\overline{P}[\mathtt{X}/\mathtt{X}'])]\!] \cap \mathcal{I}[\![\mathtt{X} \in E[\mathtt{X}/\mathtt{X}']]\!]\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}|\!)$ ⟅def. substitution⟆

$= \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \{m \in \mathcal{M} \mid \exists v \in \mathcal{D} : m[\mathtt{X}' := v] \in \mathcal{I}[\![\bigwedge \overline{P}[\mathtt{X}/\mathtt{X}'] \wedge \mathtt{X} \in E[\mathtt{X}/\mathtt{X}']]\!]\} \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}|\!)$ ⟅def. $\mathcal{I}$⟆

$= \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \mathcal{I}[\![\exists \mathtt{X}' : \bigwedge \overline{P}[\mathtt{X}/\mathtt{X}'] \wedge \mathtt{X} \in E[\mathtt{X}/\mathtt{X}']]\!] \subseteq \mathcal{I}[\![\mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}|\!)$ ⟅def. $\mathcal{I}$⟆

$= \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \mathcal{I}[\![(\exists \mathtt{X}' : \bigwedge \overline{P}[\mathtt{X}/\mathtt{X}'] \wedge \mathtt{X} \in E[\mathtt{X}/\mathtt{X}']) \Rightarrow \mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}|\!)$ ⟅def. $\mathcal{I}$⟆

$\dot{\supseteq}\ \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \mathrm{prover}[\![(\exists \mathtt{X}' : \bigwedge \overline{P}[\mathtt{X}/\mathtt{X}'] \wedge \mathtt{X} \in E[\mathtt{X}/\mathtt{X}']) \Rightarrow \mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}|\!)$ ⟅def. $\mathcal{I}$⟆

Grouping both cases together, we have

$\overline{\mathcal{R}}[\![\mathtt{X} := E]\!]\overline{P}$

$$
\begin{array}{ll}
= & \lambda\ell\bullet(\!|\ell = \mathsf{at}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \mathrm{prover}[\![\bigwedge \overline{P} \Rightarrow \mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}|\!) \mathbin{\dot\cap} \lambda\ell\bullet(\!|\ell = \mathsf{after}[\![\mathtt{X} := \\
& E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \mathrm{prover}[\![(\exists \mathtt{X}' : \bigwedge \overline{P}[\mathtt{X}/\mathtt{X}'] \wedge \mathtt{X} \in E[\mathtt{X}/\mathtt{X}'])] \Rightarrow \mathfrak{p}]\!]\} \mathbin{:} \mathfrak{P}|\!) \\
= & \lambda\ell\bullet (\!|\ell = \mathsf{at}[\![C]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \mathrm{prover}[\![\bigwedge \overline{P} \Rightarrow \mathfrak{p}]\!]\} \\
& \quad [\!] \ell = \mathsf{after}[\![\mathtt{X} := E]\!] \mathbin{?} \{\mathfrak{p} \in \mathfrak{P} \mid \mathrm{prover}[\![(\exists \mathtt{X}' : \bigwedge \overline{P}[\mathtt{X}/\mathtt{X}'] \wedge \mathtt{X} \in E[\mathtt{X}/\mathtt{X}'])] \Rightarrow \mathfrak{p}]\!]\} \\
& \quad \mathbin{:} \mathfrak{P}|\!) \\
& \quad \text{(def. } \dot\cap\text{, conditional and } \mathsf{at}[\![C]\!] \neq \mathsf{after}[\![C]\!]\text{)}
\end{array}
$$

— For tests B, we have:

$$
\begin{array}{ll}
& \gamma_{\mathfrak{P}}(\overline{P}) \cap \mathcal{B}[\![B]\!] \\
= & \mathcal{I}[\![\bigwedge \overline{P}]\!] \cap \mathcal{B}[\![B]\!] & \text{(def. } \gamma_{\mathfrak{P}}\text{)} \\
\subseteq & \bigcap\{\mathcal{B}[\![\mathfrak{p}]\!] \mid (\mathcal{I}[\![\bigwedge \overline{P}]\!] \cap \mathcal{B}[\![B]\!]) \subseteq \mathcal{B}[\![\mathfrak{p}]\!]\} & \text{(def. upper bounds)} \\
= & \bigcap\{\mathcal{B}[\![\mathfrak{p}]\!] \mid \mathcal{I}[\![(\bigwedge \overline{P} \wedge B) \Rightarrow \mathfrak{p}]\!] = \mathcal{M}\} & \text{(def. } \mathcal{I}\text{)} \\
\subseteq & \bigcap\{\mathcal{B}[\![\mathfrak{p}]\!] \mid \mathrm{prover}[\![(\bigwedge \overline{P} \wedge B) \Rightarrow \mathfrak{p}]\!]\} & \text{(prover soundness)} \\
\subseteq & \gamma_{\mathfrak{P}}(\{\mathfrak{p} \mid \mathrm{prover}[\![(\bigwedge \overline{P} \wedge B) \Rightarrow \mathfrak{p}]\!]\}) & \text{(def. } \gamma_{\mathfrak{P}}\text{)}
\end{array}
$$

Therefore we define:

$$\overline{\mathcal{R}}[\![B]\!]\overline{P} \triangleq \{\mathfrak{p} \mid \mathrm{prover}[\![(\bigwedge P \wedge B) \Rightarrow \mathfrak{p}]\!]\}$$

such that:

$$\gamma_{\mathfrak{P}}(\overline{\mathcal{R}}[\![B]\!]\overline{P}) \supseteq \gamma_{\mathfrak{P}}(\overline{P}) \cap \mathcal{B}[\![B]\!] \quad \Leftrightarrow \quad \overline{\mathcal{R}}[\![B]\!]\overline{P} \subseteq \alpha_{\mathfrak{P}}(\gamma_{\mathfrak{P}}(\overline{P}) \cap \mathcal{B}[\![B]\!]) \qquad (8)$$

— For the iteration $C = \mathtt{while}\, B\, \mathtt{do}\, C_1$, given $\overline{P} \in \mathcal{L} \mapsto \wp(\mathfrak{P})$ and $X \subseteq \Sigma = \mathcal{L} \times \mathcal{M}$, we define:

$$
\begin{array}{l}
F(X) \triangleq \mathcal{R}[\![C_1]\!](\{\langle \mathsf{at}[\![C_1]\!],\, m\rangle \mid (\langle \mathsf{at}[\![C]\!],\, m\rangle \in (\{\mathsf{at}[\![C]\!]\} \times \gamma_{\mathfrak{P}}(\overline{P})) \vee \\
\qquad\qquad\qquad\qquad\qquad\qquad \langle \mathsf{after}[\![C_1]\!],\, m\rangle \in X) \wedge m \in \mathcal{B}[\![B]\!]\})
\end{array}
$$

We can now design the abstraction $\overline{F}$ of F such that $\alpha \circ F \mathrel{\dot\supseteq} \overline{F} \circ \alpha$:

$$
\begin{array}{ll}
& \alpha(F(X)) \\
= & \alpha(\mathcal{R}[\![C_1]\!](\{\mathsf{at}[\![C_1]\!]\} \times \{m \mid (\langle \mathsf{at}[\![C]\!],\, m\rangle \in (\{\mathsf{at}[\![C]\!]\} \times \gamma_{\mathfrak{P}}(\overline{P})) \vee \langle \mathsf{after}[\![C_1]\!], \\
& m\rangle \in X) \wedge m \in \mathcal{B}[\![B]\!]\})) & \text{(def. } F\text{)} \\
\dot\supseteq & \alpha(\mathcal{R}[\![C_1]\!](\{\mathsf{at}[\![C_1]\!]\} \times \{m \mid (m \in \gamma_{\mathfrak{P}}(\overline{P}) \vee \langle \mathsf{after}[\![C_1]\!],\, m\rangle \in \gamma \circ \alpha(X)) \wedge m \in \\
& \mathcal{B}[\![B]\!]\})) & \text{(by (4) so that } \gamma \circ \alpha \text{ is extensive and monotony of } \mathcal{R}[\![C_1]\!] \text{ and } \alpha\text{)} \\
\dot\supseteq & \alpha(\mathcal{R}[\![C_1]\!](\{\mathsf{at}[\![C_1]\!]\} \times \gamma_{\mathfrak{P}} \circ \alpha_{\mathfrak{P}}(\{m \mid (m \in \gamma_{\mathfrak{P}}(\overline{P}) \vee \langle \mathsf{after}[\![C_1]\!],\, m\rangle \in \gamma \circ \\
& \alpha(X)) \wedge m \in \mathcal{B}[\![B]\!]\}))) & (\gamma_{\mathfrak{P}} \circ \alpha_{\mathfrak{P}} \text{ is extensive, monotony of } \mathcal{R}[\![C_1]\!] \text{ and (4) so that } \alpha \text{ is monotone}) \\
\dot\supseteq & \alpha(\mathcal{R}[\![C_1]\!](\{\mathsf{at}[\![C_1]\!]\} \times \gamma_{\mathfrak{P}}(\alpha_{\mathfrak{P}}(\gamma_{\mathfrak{P}}(\overline{P}) \cap \mathcal{B}[\![B]\!]) \cap \alpha_{\mathfrak{P}}(\{m \mid \langle \mathsf{after}[\![C_1]\!],\, m\rangle \in \gamma \circ \\
& \alpha(X) \wedge m \in \mathcal{B}[\![B]\!]\})))) & (\alpha_{\mathfrak{P}} \text{ is a complete join morphism}) \\
\dot\supseteq & \alpha(\mathcal{R}[\![C_1]\!](\{\mathsf{at}[\![C_1]\!]\} \times \gamma_{\mathfrak{P}}(\alpha_{\mathfrak{P}}(\gamma_{\mathfrak{P}}(\overline{P}) \cap \mathcal{B}[\![B]\!]) \cap \alpha_{\mathfrak{P}}(\gamma_{\mathfrak{P}}(\alpha(X)_{\mathsf{after}[\![C_1]\!]}) \cap \mathcal{B}[\![B]\!])))) \\
& \text{(def. of } \gamma\text{)} \\
\dot\supseteq & \alpha(\mathcal{R}[\![C_1]\!](\{\mathsf{at}[\![C_1]\!]\} \times \gamma_{\mathfrak{P}}(\overline{\mathcal{R}}[\![B]\!]\overline{P} \cap \overline{\mathcal{R}}[\![B]\!](\alpha(X)_{\mathsf{after}[\![C_1]\!]})))) & \text{(by (8) and monotony)} \\
\dot\supseteq & \overline{\mathcal{R}}[\![C_1]\!](\overline{\mathcal{R}}[\![B]\!]\overline{P} \cap \overline{\mathcal{R}}[\![B]\!](\alpha(X)_{\mathsf{after}[\![C_1]\!]})) & \text{(by induction hypothesis (5))}
\end{array}
$$

We have $\alpha(\emptyset) = \lambda\ell\bullet\mathfrak{P}$ so that it follows that:

$$\begin{array}{l}\alpha(\mathrm{lfp}^{\subseteq}\,\lambda X\bullet\mathcal{R}[\![C_1]\!](\{\langle \mathsf{at}[\![C_1]\!],\,m\rangle \mid (\langle \mathsf{at}[\![C]\!],\,m\rangle\in P\vee\langle \mathsf{after}[\![C_1]\!],\,m\rangle\in X)\\ \qquad\qquad\qquad\qquad\wedge\, m\in\mathcal{B}[\![B]\!]\}))\\ \dot{\supseteq}\ \mathrm{lfp}^{\dot{\supseteq}}_{\lambda\ell\bullet\,\mathfrak{P}}\,\lambda X\bullet\overline{\mathcal{R}}[\![C_1]\!](\overline{\mathcal{R}}[\![B]\!]\overline{P}\cap\overline{\mathcal{R}}[\![B]\!](X_{\mathsf{after}[\![C_1]\!]}))\stackrel{\Delta}{=}\overline{P}_1\end{array}\qquad(9)$$

We will need to evaluate:

$$\begin{array}{ll}&\alpha(\{\langle\ell_1,\,m\rangle\mid\langle\ell_2,\,m\rangle\in\gamma(\overline{P})\wedge m\in\mathcal{B}[\![B]\!]\})\\ =&\alpha(\{\langle\ell_1,\,m\rangle\mid m\in\gamma_{\mathfrak{P}}(\overline{P}_{\ell_2})\wedge m\in\mathcal{B}[\![B]\!]\})&\text{\{def. }\gamma\text{\}}\\ =&\alpha(\{\ell_1\}\times(\gamma_{\mathfrak{P}}(\overline{P}_{\ell_2})\cap\mathcal{B}[\![B]\!]))&\text{\{def. }\cap\text{ and }\times\text{\}}\\ \dot{\supseteq}&\alpha(\{\ell_1\}\times\gamma_{\mathfrak{P}}(\overline{\mathcal{R}}[\![B]\!]\overline{P}_{\ell_2}))&\text{\{by (8) and (4), so that }\alpha\text{ monotone\}}\\ =&\dot\alpha_{\mathfrak{P}}\circ\alpha_{\downarrow}(\{\ell_1\}\times\gamma_{\mathfrak{P}}(\overline{\mathcal{R}}[\![B]\!]\overline{P}_{\ell_2}))&\text{\{def. }\alpha\text{\}}\\ =&\lambda\ell\bullet\alpha_{\mathfrak{P}}(\{m\mid\langle\ell,\,m\rangle\in\{\ell_1\}\times\gamma_{\mathfrak{P}}(\overline{\mathcal{R}}[\![B]\!]\overline{P}_{\ell_2}))&\text{\{def. }\dot\alpha_{\mathfrak{P}}\text{ and def. }\alpha_{\downarrow}\text{\}}\\ \dot{\supseteq}&\lambda\ell\bullet(\!|\ell=\ell_1\,?\,\overline{\mathcal{R}}[\![B]\!]\overline{P}_{\ell_2}\,:\,\mathfrak{P}|\!)&\text{\{}\alpha_{\mathfrak{P}}\circ\gamma_{\mathfrak{P}}\text{ is reductive\}}\end{array}\qquad(10)$$

and therefore for $C = \texttt{while}\,B\,\texttt{do}\,C_1$:

$$\begin{array}{ll}&\alpha(\mathcal{R}[\![C]\!](\{\mathsf{at}[\![C]\!]\}\times\gamma_{\mathfrak{P}}(\overline{P})))\\ =&\alpha((\{\mathsf{at}[\![C]\!]\}\times\gamma_{\mathfrak{P}}(\overline{P}))\cup P_1\cup\{\langle\mathsf{after}[\![C]\!],\,m\rangle\mid(m\in\gamma_{\mathfrak{P}}(\overline{P})\vee\langle\mathsf{after}[\![C_1]\!],\,m\rangle\in\\ &P_1)\wedge m\in\overline{\mathcal{B}}[\![B]\!]\})&\text{\{by (1)\}}\\ \dot{\supseteq}&\alpha(\{\mathsf{at}[\![C]\!]\}\times\gamma_{\mathfrak{P}}(\overline{P}))\,\dot\cap\,\overline{P}_1\,\dot\cap\,\alpha(\{\langle\mathsf{after}[\![C]\!],\,m\rangle\mid m\in\gamma_{\mathfrak{P}}(\overline{P})\wedge m\in\\ &\overline{\mathcal{B}}[\![B]\!]\})\,\dot\cap\,\alpha(\{\langle\mathsf{after}[\![C]\!],\,m\rangle\mid\langle\mathsf{after}[\![C_1]\!],\,m\rangle\in P_1\wedge m\in\overline{\mathcal{B}}[\![B]\!]\})&\text{\{by (9) and}\\ &(4),\text{ so that }\alpha\text{ is a complete join morphism\}}\\ \dot{\supseteq}&\lambda\ell\bullet(\!|\ell=\mathsf{at}[\![C]\!]\,?\,\overline{P}\,:\,\mathfrak{P}|\!)\,\dot\cap\,\overline{P}_1\,\dot\cap\,\lambda\ell\bullet(\!|\ell=\mathsf{after}[\![C]\!]\,?\,\overline{\mathcal{R}}[\![\neg B]\!]\overline{P}\,:\,\mathfrak{P}|\!)\,\dot\cap\\ &\lambda\ell\bullet(\!|\ell=\mathsf{after}[\![C]\!]\,?\,\overline{\mathcal{R}}[\![\neg B]\!]\overline{P}_{\mathsf{after}[\![C_1]\!]}\,:\,\mathfrak{P}|\!)&\text{\{by (7), }\overline{\mathcal{B}}[\![B]\!]=\mathcal{B}[\![\neg B]\!]\text{ and (10)\}}\\ =&\overline{P}_1\,\dot\cap\,\lambda\ell\bullet(\!|\ell=\mathsf{at}[\![C]\!]\,?\,\overline{P}\parallel\ell=\mathsf{after}[\![C]\!]\,?\,\overline{\mathcal{R}}[\![\neg B]\!]\overline{P}\cap\overline{\mathcal{R}}[\![\neg B]\!]\overline{P}_{\mathsf{after}[\![C_1]\!]}\,:\,\mathfrak{P}|\!)\\ &\text{\{def }\dot\cap\text{ and conditional\}}\end{array}$$

— In conclusion of these calculi, we have proved that:

$$\begin{array}{l}\overline{\mathcal{R}}[\![C]\!]\overline{P}\ell=(\!|\ell\notin\mathsf{lab}[\![C]\!]\,?\,\mathfrak{P}\,:\,\text{match }C,\ \ell\text{ with}\\ \quad|\quad\_,\mathsf{at}[\![C]\!]\ \to\ \{\mathfrak{p}\in\mathfrak{P}\mid\mathrm{prover}[\![\bigwedge\overline{P}\Rightarrow\mathfrak{p}]\!]\}\\ \quad|\quad\texttt{skip},\mathsf{after}[\![C]\!]\ \to\ \{\mathfrak{p}\in\mathfrak{P}\mid\mathrm{prover}[\![\bigwedge\overline{P}\Rightarrow\mathfrak{p}]\!]\}\\ \quad|\quad\mathtt{X}:=E,\mathsf{after}[\![C]\!]\ \to\ \{\mathfrak{p}\in\mathfrak{P}\mid\mathrm{prover}[\![(\exists\mathtt{X}':\bigwedge\overline{P}[\mathtt{X}/\mathtt{X}']\wedge\mathtt{X}\in E[\mathtt{X}/\mathtt{X}'])\Rightarrow\mathfrak{p}]\!]\}\\ \quad|\quad C_1\,;C_2,\_\ \to\ \text{let }\overline{P}_1=\overline{\mathcal{R}}[\![C_1]\!]\overline{P}\text{ and }\overline{P}_2=\overline{\mathcal{R}}[\![C_2]\!]\overline{P}_1(\mathsf{after}[\![C_1]\!])\text{ in}\\ \qquad\qquad\qquad\overline{P}_1(\ell)\cap\overline{P}_2(\ell)\\ \quad|\quad\texttt{if}\,B\,\texttt{then}\,C_1\,\texttt{else}\,C_2,\_\ \to\\ \qquad\text{let }\overline{P}_t=\{\mathfrak{p}\in\mathfrak{P}\mid\mathrm{prover}[\![(\bigwedge\overline{P}\wedge B)\Rightarrow\mathfrak{p}]\!]\}\text{ and }\overline{P}_1=\overline{\mathcal{R}}[\![C_1]\!]\overline{P}_t\\ \qquad\text{and }\overline{P}_f=\{\mathfrak{p}\in\mathfrak{P}\mid\mathrm{prover}[\![(\bigwedge\overline{P}\wedge\neg B)\Rightarrow\mathfrak{p}]\!]\}\text{ and }\overline{P}_2=\overline{\mathcal{R}}[\![C_2]\!]\overline{P}_f\\ \qquad\quad\text{in }\overline{P}_1(\ell)\cap\overline{P}_2(\ell)\cap(\!|\ell=\mathsf{after}[\![C]\!]\,?\,\overline{P}_1(\mathsf{after}[\![C_1]\!])\cap\overline{P}_2(\mathsf{after}[\![C_2]\!])\,:\,\mathfrak{P}|\!)\end{array}$$

$$
\begin{array}{l}
\mid \quad \texttt{while}\, B\, \texttt{do}\, C_1, \_ \;\rightarrow \\
\quad \text{let } \overline{P}_1 = \mathrm{gfp}^{\dot{\subseteq}}_{\lambda\ell\cdot\mathfrak{P}}\, \lambda X \cdot \lambda\ell' \cdot (\!|\ell' = \mathsf{at}[\![C_1]\!] \;?\; \{\mathfrak{p} \in \mathfrak{P} \mid \mathrm{prover}[\![((\bigwedge \overline{P} \vee \\
\qquad\qquad\qquad\qquad X(\mathsf{after}[\![C_1]\!])) \wedge B) \Rightarrow \mathfrak{p}]\} \,;\, \mathfrak{P}|\!) \dot{\cap} \overline{\mathcal{R}}[\![C_1]\!] X(\mathsf{at}[\![C_1]\!])\ell' \\
\qquad \text{in } \overline{P}_1(\ell) \cap (\!|\ell = \mathsf{after}[\![C]\!] ? \{\mathfrak{p} \in \mathfrak{P} \mid \mathrm{prover}[\![((\bigwedge \overline{P} \vee \\
\qquad\qquad\qquad\qquad\qquad\qquad \overline{P}_1(\mathsf{after}[\![C_1]\!]) \wedge \neg B) \Rightarrow \mathfrak{p}]\} \,;\, \mathfrak{P}|\!) \\
|\!)
\end{array}
$$

3.5 Reduced Set of Ground Abstract Predicates. Observe that because of the normalization $\{\mathfrak{p} \in \mathfrak{P} \mid \mathrm{prover}[\![\bigwedge \overline{P} \Rightarrow \mathfrak{p}]\!]\}$ of $\overline{P} \in \wp(\mathfrak{P})$, the abstract domain $\langle \wp(\mathfrak{P}, \supseteq\rangle$ can be reduced [12, Prop. 10] to $\langle \{\{\mathfrak{p} \in \mathfrak{P} \mid \mathrm{prover}[\![\bigwedge \overline{P} \Rightarrow \mathfrak{p}]\!]\} \mid \overline{P} \in \wp(\mathfrak{P}\}, \supseteq\rangle$. The abstract domain $\langle \wp(\mathfrak{P}), \supseteq\rangle$ may not be Notherian while this reduced abstract domain is Notherian. For example, in Kildall's constant propagation [16] for a command C, the set of abstract predicates is $\mathfrak{P} = \{\mathtt{X} = v \mid \mathtt{X} \in \mathsf{var}[\![C]\!] \wedge v \in \mathcal{D}\}$. The reduction yields the abstract predicates $\emptyset$ ("I don't know"), $\mathfrak{P}$ ("false") and the $\{\mathtt{X}_1 = v_1, \ldots, \mathtt{X}_n = v_n\}$ (where $i \neq j$ implies $\mathtt{X}_i \neq \mathtt{X}_j$). This reduced set of abstract predicates is still infinite but Notherian. The abstract semantics $\overline{\mathcal{R}}[\![C]\!]$ can be computed by induction on the structure of C with a fixpoint iteration in $\overline{\mathcal{R}}[\![\texttt{while}\, B\, \texttt{do}\, C_1]\!]$ which is finite for Notherian abstract domains (as in [16] which corresponds to a particular *chaotic iteration* strategy [10,7]). Otherwise the iterative fixpoint computation may not terminate whence may require to be over approximated by a widening.

3.6 Local Ground Abstract Predicates. Instead of choosing the set $\mathfrak{P}$ of abstract predicates globally, it can be chosen locally, by choosing a particular set of abstract predicates $\mathfrak{P}_\ell$ attached to each command label $\ell \in \mathsf{lab}[\![C]\!]$. Then terms of the form $\{\mathfrak{p} \in \mathfrak{P} \mid \mathrm{prover}[\![P \Rightarrow \mathfrak{p}]\!]\}$ attached to program point ℓ in the definition of the abstract predicate transformer $\overline{\mathcal{R}}[\![C]\!]$ are to be simply replaced by $\{\mathfrak{p} \in \mathfrak{P}_\ell \mid \mathrm{prover}[\![P \Rightarrow \mathfrak{p}]\!]$.

3.7 Safety Verification of Commands $C \in \mathbb{C}$. The verification that a command satisfies a safety specification $S \in \mathsf{lab}[\![C]\!] \mapsto \wp(\mathfrak{P})$ consists in checking for each point $\ell \in \mathsf{lab}[\![C]\!]$ that:

$$\mathrm{prover}[\![(\bigwedge \overline{\mathcal{R}}[\![C]\!](S(\mathsf{at}[\![C]\!]))\ell) \Rightarrow \bigwedge S(\ell)]\!] \;.$$

This is sound since:

$$
\begin{array}{ll}
\quad \forall \ell \in \mathsf{lab}[\![C]\!] : \mathrm{prover}[\![(\bigwedge \overline{\mathcal{R}}[\![C]\!](S(\mathsf{at}[\![C]\!]))\ell) \Rightarrow \bigwedge S(\ell)]\!] & \\
\Rightarrow\; \forall \ell \in \mathsf{lab}[\![C]\!] : \mathcal{I}[\![(\bigwedge \overline{\mathcal{R}}[\![C]\!](S(\mathsf{at}[\![C]\!]))\ell) \Rightarrow \bigwedge S(\ell)]\!] = \mathcal{M} & \wr\text{prover soundness}\wr \\
\Rightarrow\; \forall \ell \in \mathsf{lab}[\![C]\!] : \mathcal{I}[\![(\bigwedge \overline{\mathcal{R}}[\![C]\!](S(\mathsf{at}[\![C]\!]))\ell)]\!] \subseteq \mathcal{I}[\![\bigwedge S(\ell)]\!] & \wr\text{def. of } \mathcal{I}\wr \\
\Rightarrow\; \forall \ell \in \mathsf{lab}[\![C]\!] : \gamma_{\mathfrak{P}}(\overline{\mathcal{R}}[\![C]\!](S(\mathsf{at}[\![C]\!]))\ell) \subseteq \gamma_{\mathfrak{P}}(S(\ell)) & \wr\text{def. } \gamma_{\mathfrak{P}}\wr \\
\Rightarrow\; \dot{\gamma}_{\mathfrak{P}}(\overline{\mathcal{R}}[\![C]\!](S(\mathsf{at}[\![C]\!]))) \,\dot{\subseteq}\, \dot{\gamma}_{\mathfrak{P}}(S) & \wr\text{def. } \dot{\gamma}_{\mathfrak{P}}\wr
\end{array}
$$

$$\Rightarrow \gamma_{\downarrow} \circ \dot{\gamma}_{\mathfrak{P}}(\overline{\mathcal{R}}[\![C]\!](S(\mathsf{at}[\![C]\!]))) \mathrel{\dot{\subseteq}} \gamma_{\downarrow} \circ \dot{\gamma}_{\mathfrak{P}}(S) \qquad \wr\text{by monotony}\int$$

$$\Rightarrow \gamma(\overline{\mathcal{R}}[\![C]\!](S(\mathsf{at}[\![C]\!]))) \mathrel{\dot{\subseteq}} \gamma(S) \qquad \wr\text{def. } \gamma\int$$

$$\Rightarrow \mathcal{R}[\![C]\!](\{\mathsf{at}[\![C]\!]\} \times \gamma_{\mathfrak{P}}(S(\mathsf{at}[\![C]\!]))) \subseteq \gamma(S) \qquad \wr\text{by (6)}\int$$

$$\Rightarrow \forall \ell \in \mathsf{lab}[\![C]\!] : \mathcal{R}[\![C]\!](\{\mathsf{at}[\![C]\!]\} \times \gamma_{\mathfrak{P}}(S(\mathsf{at}[\![C]\!])))\ell \subseteq \gamma_{\mathfrak{P}}(S(\ell)) \qquad \wr\text{def. } \gamma\int$$

so that, informally, $S(\ell)$ holds whenever program point ℓ is reached during any execution of command C starting at point $\mathsf{at}[\![C]\!]$ with an initial memory state satisfying $S(\mathsf{at}[\![C]\!])$.

4 Application to Parametric Predicate Abstraction

The inconvenience of ground predicate abstractions is that the ground predicates directly refer to the program states and control by explicitly naming program constants, variables and maybe control points. Consequently, the abstract domain, being program-specific, has to be redesigned for each new or modified program. This design can be partially automatized by refinement techniques, including convergence acceleration by widening [3], but this alternation of analyzes and refinements would be costly for precise analysis of large programs. An alternative is to provide program-independent predicates by designing *parametric abstract domains*. The presentation is made through a sorting example.

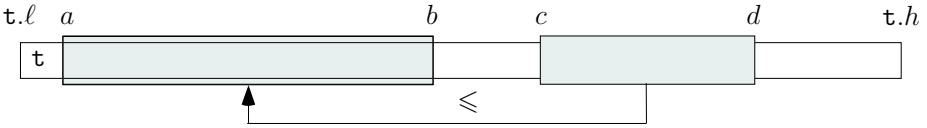
4.1 Parametric Abstract Domains. A parametric abstract domain is parameterized so that a particular abstract domain instantiation for a given program is obtained by binding the parameters to the constants, variables, control points, etc. of this specific program. For a simple example, Kildall's parametric abstract domain for constant propagation is $K(C, V) = \prod_{\ell \in C} \prod_{\mathtt{X} \in V(\ell)} L$ where L is Kildall's complete lattice $\bot \sqsubseteq v \sqsubseteq \top$ for all $v \in \mathcal{D}$. Given a command C, it is instantiated to $K(\mathsf{lab}[\![C]\!], \mathsf{var}[\![C]\!])$ where $\mathsf{lab}[\![C]\!]$ is the set of labels of command C and $\mathsf{var}[\![C]\!](\ell)$ is the set of program variables $\mathtt{X}$ which are visible at this program point ℓ of command C.

4.2 Parametric Comparison Abstract Domain. We let $\mathcal{D}_r(X)$ be a parametric relational integer abstract domain parameterized by a set X of program and auxiliary variables. This abstract domain is assumed to have abstract operations on $r, r_1, r_2 \in \mathcal{D}_r(X)$ such as the projection or variable elimination $\exists x \in X : r$, disjunction $r_1 \vee r_2$, conjunction $r_1 \wedge r_2$, abstract predicate transformers for assignments and tests, etc (see e.g. [18]).

Then we define the parametric comparison abstract domain:

$$\mathcal{D}_{\mathrm{lt}}(X) = \{\langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ r\rangle \mid \mathtt{t} \in X \wedge a, b, c, d \notin X \wedge r \in \mathcal{D}_r(X \cup \{a, b, c, d\})\}\ . \tag{11}$$

The meaning $\gamma(\langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ r\rangle)$ of an abstract predicate $\langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ r\rangle$ is informally that all elements of $\mathtt{t}$ between indices a and b are less than any element of $\mathtt{t}$ between indices c and d and moreover r holds:



More formally, there should be a declaration $\texttt{t} : \texttt{array}[\ell, h]\ \texttt{of int}$ so that $\gamma(\langle \mathrm{lt}(\texttt{t}, a, b, c, d), r\rangle)$ defines a set of environments ρ mapping program and auxiliary variables $\texttt{X}$ to their value $\rho(\texttt{X})$ for which the above concrete predicate holds:

$$\begin{aligned}\gamma(\langle \mathrm{lt}(t, a, b, c, d),\ r\rangle) = \{\rho \mid \exists a, b, c, d : \rho(\texttt{t}).\ell \le a \le b \le \rho(\texttt{t}).h \\ \wedge\ \rho(\texttt{t}).\ell \le c \le d \le \rho(\texttt{t}).h \\ \wedge\ \forall i \in [a, b] : \forall j \in [c, d] : \rho(\texttt{t})[i] \le \rho(\texttt{t})[j] \wedge \rho \in \gamma(r)\}\end{aligned}$$

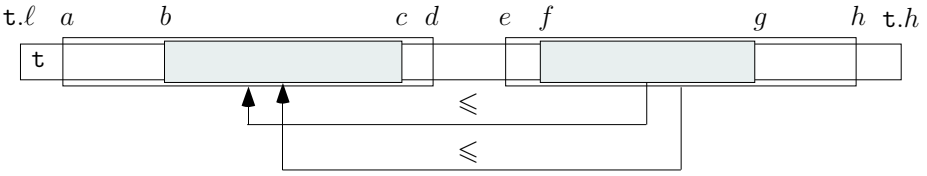
where the domain of the ρ is $X \cup \{a, b, c, d\}$ and $\gamma(r)$ is the concretization of the abstract predicate $r \in \mathcal{D}_r(X \cup \{a, b, c, d\})$ specifying the possible values of the variables in X and the auxiliary variables a, b, c, d.

4.3 Abstract Logical Operations of the Parametric Comparison Abstract Domain. Then the abstract domain must be equipped with abstract operations such as the implication $\Rightarrow$, conjunction $\wedge$, disjunction $\vee$, etc. We simply provided a few examples.

Abstract Implication. We have $\langle \mathrm{lt}(\texttt{t}, a, b, c, d),\ r\rangle \Rightarrow r$. If $r \Rightarrow r'$ and $a \le b \le c \le d$ and $e \le f \le g \le h$ then;

$$\langle \mathrm{lt}(\texttt{t}, a, d, e, h),\ r\rangle \Rightarrow \langle \mathrm{lt}(\texttt{t}, b, c, f, g),\ r'\rangle \tag{12}$$

as shown below:



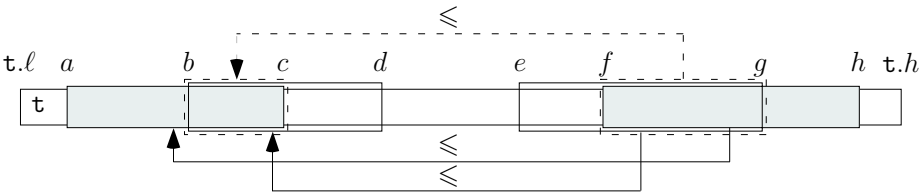
Abstract Conjunction. If $\texttt{t}, i, j, k, \ell \notin \mathsf{var}[\![r]\!]$, then:

$$r \wedge \langle \mathrm{lt}(\texttt{t}, a, c, f, h),\ r'\rangle = \langle \mathrm{lt}(\texttt{t}, a, c, f, h),\ r \wedge r'\rangle \tag{13}$$

If $a \le b \le c \le d$ and $e \le f \le g \le h$ then we have:

$$\langle \mathrm{lt}(\texttt{t}, a, c, f, h),\ r\rangle \wedge \langle \mathrm{lt}(\texttt{t}, b, d, e, g),\ r'\rangle, = \langle \mathrm{lt}(\texttt{t}, b, c, f, g),\ \exists a, d, e, h : r \wedge r'\rangle$$

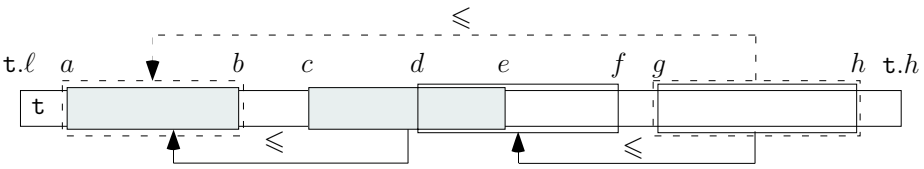
as shown below:



The same way, we have:

$$\langle \mathrm{lt}(\mathtt{t}, a, b, c, e),\ r\rangle \wedge \langle \mathrm{lt}(\mathtt{t}, d, f, g, h),\ r'\rangle = \langle \mathrm{lt}(\mathtt{t}, a, b, g, h),\ \exists c, e, d, f : r \wedge r'\rangle \quad (14)$$

when $(r \wedge r') \Rightarrow (c \leq d \leq e \leq f)$:



Abstract Disjunction. We have:

$$\langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ r\rangle \vee \langle \mathrm{lt}(\mathtt{t}, e, f, g, h),\ r'\rangle = \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ (\exists a, b, c, d : i = a \quad (15)$$
$$\wedge\ j = b \wedge k = c \wedge \ell = d \wedge r) \vee (\exists e, f, g, h : i = e \wedge j = f \wedge k = g \wedge \ell = h \wedge r')\rangle$$

In case one of the terms does not refer to the array ($\mathtt{t} \notin \mathsf{var}[\![r]\!]$), a criterion must be used to force the introduction of an identically true array term $\mathrm{lt}(\mathtt{t}, i, i, i, i)$. For example if the auxiliary variables d, f, g, h in r' depend upon one selectively chosen variable $\mathtt{I}$, then we have:

$$r \vee \langle \mathrm{lt}(\mathtt{t}, d, f, g, h),\ r'\rangle = \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ (i = j = k = \ell = \mathtt{I} \wedge r) \vee \quad (16)$$
$$(\exists d, f, g, h : i = d \wedge j = f \wedge k = g \wedge \ell = h \wedge r')\rangle$$

This case appears typically in loops, which can also be handled by unrolling, see Sec. 4.5.

4.4 Abstract Predicate Transformers for the Parametric Comparison Abstract Domain. Then the abstract domain must be equipped with abstract predicate transformers for tests, assignments, etc. We consider forward strongest postconditions (although weakest preconditions, which avoid an existential quantifier in assignments, may sometimes be simpler).

We depart from traditional predicate abstraction which uses a simplifier (or a theorem prover) to formally evaluate the abstract predicate transformer $\alpha \circ F \circ \gamma$ approximating the concrete predicate transformer F. The alternative proposed below is standard in abstract interpretation-based static program analysis and directly provides an over-approximation of the best abstract predicate transformer $\alpha \circ \ F \circ \gamma$ in the form of an algorithm (which correctness must be established formally). The simplifier/prover can be used to reduce the postcondition in the normal form (11) that is required for the abstract predicates and otherwise easily handled e.g. by pattern-matching.

Abstract Strongest Postconditions for Tests.

$\{\ P_1\ \}$
if $(\mathtt{t}[\mathtt{I}]\ >\ \mathtt{t}[\mathtt{I}+\mathtt{1}])$
then $\{\ P_1 \wedge \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{I} \wedge j = \mathtt{I}+1 \wedge k = \mathtt{I} \wedge \ell = \mathtt{I} \rangle\ \}\ \ldots\ \{\ P_2\ \}$ (17)
else $\{\ P_1 \wedge \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{I} \wedge j = k = \ell = \mathtt{I}+1 \rangle\ \}\ \ldots\ \{\ P_3\ \}$ (18)
$\{\ P_2 \vee P_3\ \}$ (19)

Abstract Strongest Postconditions for Assignments. For assignment, we have $\{\mathtt{t}[\mathtt{a}..\mathtt{I}] \leq \mathtt{t}[\mathtt{I}] \wedge \mathtt{t}[\mathtt{I}] > \mathtt{t}[\mathtt{I}+\mathtt{1}]\}\ \mathtt{t}[\mathtt{I}]\ :=:\ \mathtt{t}[\mathtt{I}+\mathtt{1}]\ \{\mathtt{t}[\mathtt{a}..\mathtt{I}+\mathtt{1}] \leq \mathtt{t}[\mathtt{I}+\mathtt{1}]\}$, so that in the parametric abstract domain, assuming $\mathtt{t} \notin \mathsf{var}[\![r]\!] \cup \mathsf{var}[\![r']\!]$, $r \Rightarrow (i \leq j = k = \ell = \mathtt{I})$ and $r' \Rightarrow (m = p = q = \mathtt{I} \wedge n = \mathtt{I}+1)$, we have:

$$\begin{array}{l}\{\ \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ r\rangle \wedge \langle \mathrm{lt}(\mathtt{t}, m, n, p, q),\ r'\rangle\} \\ \mathtt{t}[\mathtt{I}]\ :=:\ \mathtt{t}[\mathtt{I}+\mathtt{1}] \\ \{\ \langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ \exists i, j, k, \ell, m, n, p, q : r \wedge r' \wedge a = i \wedge b = c = d = \mathtt{I}+1\rangle\ \}\ .\end{array} \quad (20)$$

Similarly, if $\mathtt{t} \notin \mathsf{var}[\![r]\!]$ and $r \Rightarrow (\mathtt{I} \in [i, j] \wedge \mathtt{J} \in [i, j]) \vee (\mathtt{J} \in [k, \ell] \wedge \mathtt{I} \in [k, \ell])$ then:

$$\{\ \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ r\rangle\}\quad \mathtt{t}[\mathtt{I}]\ :=:\ \mathtt{t}[\mathtt{J}]\quad \{\ \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ r\rangle\} \quad (21)$$

since the swap of the array elements does not interfere with the assertions.

4.5 Widening for the Parametric Comparison Abstract Domain. Finally the abstract domain must be equipped with a widening to generate induction hypotheses (and optionally a narrowing to improve precision) to speed up the convergence of iterative fixpoint computations [9]. We choose to define the widening ∇ as:

$$\begin{array}{l}\langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ r\rangle \nabla \langle \mathrm{lt}(\mathtt{t}, m, n, p, q),\ r'\rangle \\ \quad = \text{let } \langle \mathrm{lt}(\mathtt{t}, r, s, t, u),\ r''\rangle = \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ r\rangle \vee \langle \mathrm{lt}(\mathtt{t}, m, n, p, q),\ r'\rangle \text{ in} \\ \qquad \langle \mathrm{lt}(\mathtt{t}, r, s, t, u),\ r \nabla r''\rangle\ .\end{array} \quad (22)$$

Typically, when handling loops entry condition r, one encounters widenings of the form $r \nabla \langle \mathrm{lt}(\mathtt{t}, m, n, p, q),\ r'\rangle$ where $\langle \mathrm{lt}(\mathtt{t}, m, n, p, q),\ r'\rangle$ appears during the analysis of the loop body. There are several ways to handle this situation:

1. incorporate the term $\mathrm{lt}(\mathtt{t}, i, j, k, \ell)$ in the form of a tautology, as already described in (16) for the abstract disjunction;
2. use disjunctive completion to preserve the disjunction within the loop (which may ultimately lead to infinite disjunctions) or, for a less precise but cheaper solution, allow only abstract predicates of a more restricted form, such as $r \vee \langle \mathrm{lt}(\mathtt{t}, m, n, p, q), r'\rangle$ (which definitively avoids the previous potential explosion and can be requested e.g. at widening points only);
3. *unroll loops semantically* (as in [5, Sec. 7.1.1]) so that the loop:

   $$\mathtt{while}\ B\ \mathtt{do}\ C\ \mathtt{od}$$

   is handled in the abstract semantics as if written in the form:

`if` B `then` C; `while` B `do` C `od` fi

which is equivalent in the concrete semantics. More generally, if several abstract terms of different kinds are considered (like $\mathrm{lt}(\mathtt{t}, i, j, k, \ell)$ and $\mathrm{s}(\mathtt{t}, m, n)$ in the forthcoming Sec. 4.10), a further semantic unrolling can be performed each time a term of a new kind does appear, while all terms of the same kind are merged by the widening.

4.6 Refined Parametric Comparison Abstract Domains. The parametric comparison abstract domain $\mathcal{D}_{\mathrm{lt}}(X)$ of Sec. 4.2 may be imprecise since it allows only for one term $\langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ r\rangle$. First we could consider several arrays, with one such term per array. Second, we could consider the conjunction of such terms for a given array, which is more precise but may potentially lead to infinite conjunctions within loops (e.g. for which termination cannot be established). So we will consider this alternative within tests only, then applying the above abstract domain operators term by term[4].

The same way we could consider the disjunctive completion of this domain, that is terms of the form $\bigvee_i \bigwedge_j \langle \mathrm{lt}(\mathtt{t}, a_{ij}, b_{ij}, c_{ij}, d_{ij}),\ r_{ij}\rangle$. This would introduce an exponential complexity factor, which we prefer to avoid. If necessary, we will use *local trace partitioning* [5, Sec. 7.1.5] instead.

4.7 Parametric Comparison Static Program Analysis. Let us consider the opposite program (where $a \leq b$) which is similar to the inner loop of bubble sort [17]. We let P_p^i be the value of the local predicate attached to the program point $p = 1, ..., 8$ at the i^{th} iteration. Initially, $P_1^0 = (\mathtt{a} \leq \mathtt{b})$ while $P_p^0 = \text{false}$ for $p = 2, ..., 8$. We choose the octagonal abstract domain [18] as the parametric relational integer abstract domain $\mathcal{D}_r(X)$ parameterized by the set X of program variables `I`, `J`,... and auxiliary variables i, j, etc. The fixpoint iterates are as follows:

```
1 :  var t : array [a, b] of int;
2 :  I := a;
3 :  while (I < b) do
4 :      if (t[I] > t[I + 1]) then
5 :          t[I] :=: t[I + 1]
6 :      fi;
7 :      I := I + 1
8 :  od
```

$$
\begin{aligned}
P_1^1 &= (\mathtt{a} \leq \mathtt{b}) && \wr\text{initialization to } P_1^0\int\\
P_2^1 &= (\mathtt{I} = \mathtt{a} \leq \mathtt{b}) && \wr\text{assignment } (\mathtt{I} := \mathtt{a})\int\\
P_3^1 &= (\mathtt{I} = \mathtt{a} < \mathtt{b}) && \wr\text{loop condition } \mathtt{I} < \mathtt{b}\int\\
P_4^1 &= \langle \mathrm{lt}(\mathtt{t}, m, n, p, q),\ m = p = q = \mathtt{I} = \mathtt{a} < \mathtt{b} \wedge n = \mathtt{I} + 1\rangle && \wr\text{by (17) for test condition } (\mathtt{t}[\mathtt{I}] > \mathtt{t}[\mathtt{I} + 1])\int
\end{aligned}
$$

[4] For short we avoid to unroll loops semantically which is better adapted to automatization but would yield to lengthy handmade calculations in this section. This technique will be illustrated anyway in the forthcoming Sec. 4.10.

$$
\begin{aligned}
P_5^1 &= \langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ \exists i, j, k, \ell, m, n, p, q : i = j = k = \ell = \mathtt{I} \wedge m = p = q = \mathtt{I} = \\
&\quad \mathtt{a} < \mathtt{b} \wedge n = \mathtt{I} + 1 \wedge a = i \wedge b = c = d = \mathtt{I} + 1 \rangle \\
&\qquad \wr \text{by assignment (20) (with tautology } \mathrm{lt}(\mathtt{t}, \mathtt{I}, \mathtt{I}, \mathtt{I}, \mathtt{I})) \text{ which, by octagonal projection, simplifies into:} \int \\
&= \langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ a = \mathtt{I} = \mathtt{a} < \mathtt{b} \wedge b = c = d = \mathtt{I} + 1 \rangle \\
P_6^1 &= (P_3^1 \wedge \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{I} = \mathtt{a} < \mathtt{b} \wedge j = k = \ell = \mathtt{I} + 1 \rangle) \vee P_5^1 \\
&\qquad \wr \text{by (18) for test condition } (\mathtt{t}[\mathtt{I}] \ > \ \mathtt{t}[\mathtt{I} + \mathtt{1}]) \text{ and join (19)} \int \\
&= (\langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{I} = \mathtt{a} < \mathtt{b} \wedge j = k = \ell = \mathtt{I} + 1 \rangle) \vee (\langle \mathrm{lt}(\mathtt{t}, m, n, p, q), \\
&\quad m = \mathtt{I} = \mathtt{a} < \mathtt{b} \wedge n = p = q = \mathtt{I} + 1 \rangle) \\
&\qquad \wr \text{def. } P_3^1 \text{ and (13) as well as by def. of } P_5^1 \int \\
&= \langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ (\exists i, j, k, \ell\ :\ a = i \wedge b = j \wedge c = k \wedge d = \ell \wedge \\
&\quad i = \mathtt{I} = \mathtt{a} < \mathtt{b} \wedge j = k = \ell = \mathtt{I} + 1) \vee (\exists m, n, p, q : a = m \wedge b = n \wedge c = \\
&\quad p \wedge d = q \wedge m = \mathtt{I} = \mathtt{a} < \mathtt{b} \wedge n = p = q = \mathtt{I} + 1) \rangle \qquad \wr \text{def. (15) of the abstract union} \vee \int \\
&= \langle \mathrm{lt}(\mathtt{t}, a, b, c, d), (a = \mathtt{I} = \mathtt{a} < \mathtt{b} \wedge b = c = d = \mathtt{I} + 1) \vee (a = \mathtt{I} = \mathtt{a} < \mathtt{b} \wedge b = \\
&\quad c = d = \mathtt{I} + 1) \rangle) \qquad \wr \text{by octagonal projection} \int \\
&= \langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ a = \mathtt{I} = \mathtt{a} < \mathtt{b} \wedge b = c = d = \mathtt{I} + 1 \rangle \qquad \wr \text{by octagonal disjunction} \int \\
P_7^1 &= \langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ a = \mathtt{I} - 1 = \mathtt{a} < \mathtt{b} \wedge b = c = d = \mathtt{I} \rangle \qquad \wr \text{by invertible assignment } \mathtt{I} := \mathtt{I} + \mathtt{1} \int \\
&= \langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ \mathtt{I} = a + 1 = \mathtt{a} + 1 \leq \mathtt{b} \wedge b = c = d = \mathtt{I} \rangle \qquad \wr \text{octagonal simplification} \int \\
P_3^2 &= (P_2^1 \vee P_7^1) \wedge (\mathtt{I} < \mathtt{b}) \quad \wr \text{loop condition } \mathtt{I} < \mathtt{b} \text{ and absence of widening on first iterate} \int \\
&= ((\mathtt{I} = \mathtt{a} \leq \mathtt{b}) \vee (\langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ \mathtt{I} = a + 1 = \mathtt{a} + 1 \leq \mathtt{b} \wedge b = c = d = \\
&\quad \mathtt{I} \rangle)) \wedge (\mathtt{I} < \mathtt{b}) \qquad \wr \text{def. } P_2^1 \text{ and } P_7^1 \int \\
&= (\langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ (i = j = k = \ell = \mathtt{I} = \mathtt{a} \leq \mathtt{b}) \vee (\exists a, b, c, d : i = a \wedge b = \\
&\quad j \wedge c = k \wedge d = \ell \wedge \mathtt{I} = a + 1 = \mathtt{a} + 1 \leq \mathtt{b} \wedge b = c = d = \mathtt{I}) \rangle) \wedge (\mathtt{I} < \mathtt{b}) \\
&\qquad \wr \text{def. (16) of abstract disjunction, the octagonal predicate depending only on } \mathtt{I}, \mathtt{a} \text{ and } \mathtt{b} \text{ which leads to the selection of } \mathtt{I}, \\
&\qquad \text{the only of these variables which is modified within the loop body} \int \quad (23) \\
&= (\langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ (i = j = k = \ell = \mathtt{I} = \mathtt{a} \leq \mathtt{b}) \vee (\mathtt{I} = i + 1 = \mathtt{a} + 1 \leq \\
&\quad \mathtt{b} \wedge j = k = \ell = \mathtt{I}) \rangle) \wedge (\mathtt{I} < \mathtt{b}) \qquad \wr \text{by octagonal projection} \int \\
&= (\langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ (i = j = k = \ell = \mathtt{I} = \mathtt{a} < \mathtt{b}) \vee (\mathtt{I} = i + 1 = \mathtt{a} + 1 < \\
&\quad \mathtt{b} \wedge j = k = \ell = \mathtt{I}) \rangle) \qquad \wr \text{by octagonal conjunction} \int \\
&= \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{a} \leq j = k = \ell = \mathtt{I} \leq \mathtt{a} + 1 \leq \mathtt{b} \rangle \qquad \wr \text{by octagonal disjunction} \int \\
P_3^3 &= P_3^2 \nabla \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{a} \leq j = k = \ell = \mathtt{I} \leq \mathtt{a} + 2 \leq \mathtt{b} \rangle \qquad \wr \text{in absence of stabilization of the iterates, by a similar computation at the next iteration} \int \\
&= \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{a} \leq j = k = \ell = \mathtt{I} < \mathtt{b} \rangle \ \wr \text{def. (22) of the widening } \nabla \int \\
P_4^3 &= P_3^3 \wedge \langle \mathrm{lt}(\mathtt{t}, m, n, p, q),\ m = p = q = \mathtt{I} \wedge n = \mathtt{I} + 1 \rangle \qquad \wr \text{by (17) for test condition } (\mathtt{t}[\mathtt{I}] \ > \ \mathtt{t}[\mathtt{I} + \mathtt{1}]) \int
\end{aligned}
$$

$= \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{a} \le j = k = \ell = \mathtt{I} < \mathtt{b} \rangle \wedge \langle \mathrm{lt}(\mathtt{t}, m, n, p, q),\ m = p = q = \mathtt{I} \wedge n = \mathtt{I} + 1 \rangle$ ⟅def. P_3^3, the conjunction being left symbolic since it cannot be simplified, see Sec. 4.6⟆

$P_5^3 = \langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ \exists i, j, k, \ell, m, n, p, q : i = \mathtt{a} \le j = k = \ell = \mathtt{I} < \mathtt{b} \wedge m = p = q = \mathtt{I} \wedge n = \mathtt{I} + 1 \wedge a = i \wedge b = c = d = \mathtt{I} + 1 \rangle$ ⟅by (20) where $\mathtt{t} \notin \mathsf{var}[\![r]\!] \cup \mathsf{var}[\![r']\!]$ and $r = (i = \mathtt{a} \le j = k = \ell = \mathtt{I} < \mathtt{b}) \Rightarrow (i \le j = k = \ell = \mathtt{I})$ and $r' = (m = p = q = \mathtt{I} \wedge n = \mathtt{I} + 1) \Rightarrow (m = p = q = \mathtt{I} \wedge n = \mathtt{I} + 1)$⟆

$= \langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ a = \mathtt{a} \le b = c = d = \mathtt{I} + 1 \le \mathtt{b} \rangle$ ⟅by octagonal projection⟆

$P_6^3 = (\langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{a} \le j = k = \ell = \mathtt{I} < \mathtt{b} \rangle \wedge \langle \mathrm{lt}(\mathtt{t}, i', j', k', \ell'),\ i' = \mathtt{I} \wedge j' = k' = \ell' = \mathtt{I} + 1 \rangle) \vee \langle \mathrm{lt}(\mathtt{t}, i'', j'', k'', \ell''),\ i'' = \mathtt{a} \le j'' = k'' = \ell'' = \mathtt{I} + 1 \le \mathtt{b} \rangle$ ⟅by $P_6^3 = (P_3^3 \wedge (\mathtt{t}[\mathtt{I}] \le \mathtt{t}[\mathtt{I} + \mathtt{1}])) \vee P_5^3$ and (18)⟆

$= \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{a} \le j = k = \ell = \mathtt{I} + 1 \le \mathtt{b} \rangle \vee \langle \mathrm{lt}(\mathtt{t}, i'', j'', k'', \ell''),\ i'' = \mathtt{a} \le j'' = k'' = \ell'' = \mathtt{I} + 1 \le \mathtt{b} \rangle$ ⟅def. (14), of conjunction and octagonal projection⟆

$= \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{a} \le j = k = \ell = \mathtt{I} + 1 \le \mathtt{b} \rangle$ ⟅by $P \vee P = P$⟆

$P_7^3 = \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{a} \le j = k = \ell = \mathtt{I} \le \mathtt{b} \rangle$ ⟅by assignment $\mathtt{I} := \mathtt{I} + \mathtt{1}$⟆

Now the iterates have stabilized since:

$(P_2^3 \vee P_7^3) \wedge (\mathtt{I} < \mathtt{b}) = (P_2^1 \vee P_7^3) \wedge (\mathtt{I} < \mathtt{b})$ ⟅since $P_2^3 = P_2^1$ is stable⟆

$= ((\mathtt{I} = \mathtt{a} \le \mathtt{b}) \vee \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{a} \le j = k = \ell = \mathtt{I} \le \mathtt{b} \rangle) \wedge (\mathtt{I} < \mathtt{b})$ ⟅def. P_2^1 and P_7^3⟆

$= (\langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ (i = j = k = \ell = \mathtt{I} = \mathtt{a} \le \mathtt{b}) \vee (\exists a, b, c, d : i = a \wedge b = j \wedge c = k \wedge d = \ell \wedge \mathtt{I} = a + 1 = \mathtt{a} + 1 \le \mathtt{b} \wedge b = c = d = \mathtt{I}) \rangle) \wedge (\mathtt{I} < \mathtt{b})$ ⟅def. (16) of abstract disjunction with selection of $\mathtt{I}$ as in (23)⟆

$= (\langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ (i = j = k = \ell = \mathtt{I} = \mathtt{a} \le \mathtt{b}) \vee (j = k = \ell = \mathtt{I} = i + 1 = \mathtt{a} + 1 \le \mathtt{b}) \rangle) \wedge (\mathtt{I} < \mathtt{b})$ ⟅by octagonal projection⟆

$= (\langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{a} \le j = k = \ell = \mathtt{I} \le \mathtt{b} \wedge \mathtt{a} \le \mathtt{b} \rangle) \wedge (\mathtt{I} < \mathtt{b})$ ⟅by octagonal disjunction⟆

$= \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{a} \le j = k = \ell = \mathtt{I} < \mathtt{b} \rangle$ ⟅by abstract conjunction (13)⟆

$\Rightarrow\ P_3^3$ ⟅def. (12) of abstract implication⟆

It remains to compute the loop exit invariant:

$(P_2^3 \vee P_7^3) \wedge (\mathtt{I} \ge \mathtt{b})$

$= (\langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{a} \le j = k = \ell = \mathtt{I} \le \mathtt{b} \wedge \mathtt{a} \le \mathtt{b} \rangle) \wedge (\mathtt{I} \ge \mathtt{b})$ ⟅by octagonal disjunction⟆

$= \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{a} \le j = k = \ell = \mathtt{I} = \mathtt{b} \rangle$ ⟅by abstract conjunction (13)⟆

The static analysis has therefore discovered the following invariants:

```
var t : array [a, b] of int;
```

```
1 :   {a ≤ b}
      I := a;
2 :   {I = a ≤ b}
      while (I < b) do
3 :       {lt(t, a, I, I, I) ∧ I < b}
          if (t[I] > t[I + 1]) then
4 :           {lt(t, a, I, I, I) ∧ I < b ∧ lt(t, I, I + 1, I, I)}
              t[I] :=: t[I + 1]
5 :           {lt(t, a, I + 1, I + 1, I + 1) ∧ I + 1 ≤ b}
          fi;
6 :       {lt(t, a, I + 1, I + 1, I + 1) ∧ I + 1 ≤ b}
          I := I + 1
7 :       {lt(t, a, I, I, I) ∧ I ≤ b}
      od
8 :   {lt(t, a, I, I, I) ∧ I = b}
```

4.8 Parametric Sorting Abstract Domain. Then we define the parametric sorting abstract domain:

$$\mathcal{D}_s(X) = \{\langle \mathrm{s}(\mathtt{t}, a, b),\ r\rangle \mid \mathtt{t} \in X \wedge a, b \notin X \wedge r \in \mathcal{D}_r(X \cup \{a, b\})\}\ .$$

The meaning $\gamma(\langle \mathrm{s}(\mathtt{t}, a, b),\ r\rangle)$ of an abstract predicate $\langle \mathrm{s}(\mathtt{t}, a, b),\ r\rangle$ is, informally that the elements of $\mathtt{t}$ between indices a and b are sorted:

$$\begin{aligned}\gamma(\langle \mathrm{s}(\mathtt{t}, a, b),\ r\rangle) = \exists a, b : \mathtt{t}.\ell \leq a \leq b \leq \mathtt{t}.h \wedge \\ \forall i, j \in [a, b] : (i \leq j) \Rightarrow (\mathtt{t}[i] \leq \mathtt{t}[j]) \wedge r\ .\end{aligned}$$

4.9 Parametric Reduced Product of the Comparison and Sorting Abstract Domain. The analysis of sorting algorithms involves the reduced product [11] of the parametric comparison abstract domain of Sec. 4.2 and sorting abstract domain of Sec. 4.8, that is triples of the form:

$$\langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ \mathrm{s}(\mathtt{t}, e, f),\ r\rangle\ .$$

The reduction involves interactions between terms such as, e.g.:

$$\begin{aligned}&\mathrm{lt}(\mathtt{t}, a, b-1, b-1, b-1) \wedge \mathrm{lt}(\mathtt{t}, a, b, b, b) \\ &\qquad \Rightarrow \mathrm{s}(\mathtt{t}, b-1, b) \wedge \mathrm{lt}(\mathtt{t}, a, b-1, b-1, b) \qquad (24)\\ &\mathrm{s}(\mathtt{t}, b+1, c) \wedge \mathrm{lt}(\mathtt{t}, a, b+1, b+1, c) \wedge \mathrm{lt}(\mathtt{t}, a, b, b, b) \\ &\qquad \Rightarrow \mathrm{s}(\mathtt{t}, b, c) \wedge \mathrm{lt}(\mathtt{t}, a, b, b, c) \qquad (25)\\ &\mathrm{lt}(\mathtt{t}, a, a+1, a+1, b) \wedge \mathrm{s}(\mathtt{t}, a+1, b) \Rightarrow \mathrm{s}(\mathtt{t}, a, b) \qquad (26)\end{aligned}$$

The reduction [11] also involves the refinement of abstract predicate transformers that would be performed automatically e.g. if the abstract predicate transformers are obtained by automatic simplification of the formula $\alpha \circ F \circ \gamma$ (where F is the concrete semantics) by the simplifier of a theorem prover.

4.10 Parametric Comparison and Sorting Static Program Analysis.

Let us consider the opposite bubble sort [17]. The fixpoint approximation below starts form:

$P_1^0 = \quad (\mathtt{a} \leq \mathtt{b}) \quad$ ⟅initialization⟆
$P_i^0 = \quad$ false, $\qquad i = 2, \ldots, 8$

$P_p^{i,k}$ denotes the local assertion attached to program point p at the i^{th} iteration and k^{th} loop unrolling, $P_p^i = P_p^{i,0}$ where $k = 0$ means that the decision to semantically unroll the loop is not yet taken.

```
1 :  var t : array [a, b] of int;
2 :  J := b;
3 :  while (a < J) do
4 :      I := a;
5 :      while (I < J) do
6 :          if (t[I] > t[I + 1]) then
7 :              t[I] :=: t[I + 1]
8 :          fi;
9 :          I := I + 1
10 :     od;
11 :     J := J − 1
12 : od
```

$P_1^1 = \quad P_1^0 \quad = \quad (\mathtt{a} \leq \mathtt{b})$ ⟅def. P_1^0⟆

$P_2^1 = \quad (\mathtt{a} \leq \mathtt{b} = \mathtt{J})$ ⟅assignment $\mathtt{J} := \mathtt{b}$⟆

$P_3^{1,0} = \quad (\mathtt{a} < \mathtt{b} = \mathtt{J})$ ⟅test $(\mathtt{a} < \mathtt{J})$⟆

...

$P_{10}^{1,0} = \quad \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{I}, \mathtt{I}, \mathtt{I}) \wedge \mathtt{a} < \mathtt{b} = \mathtt{I} = \mathtt{J}^5$ ⟅as in Sec. 4.7 since the inner loop does not modify a, b or I⟆

$\Rightarrow \quad \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}, \mathtt{J}, \mathtt{b}) \wedge \mathtt{a} < \mathtt{b} = \mathtt{J}$ ⟅by elimination (octagonal projection) of program variable I which is no longer live at program point 10⟆

$P_{11}^{1,0} = \quad \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{a} < \mathtt{b} \wedge \mathtt{J} = \mathtt{b} - 1$ ⟅postcondition for assignment $\mathtt{J} := \mathtt{J} - 1$⟆

$P_3^{1,1} = \quad \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{a} < \mathtt{J} = \mathtt{b} - 1$ ⟅by semantical loop unrolling (since a new symbolic "lt" term has appeared, see Sec. 4.5,) and test $(\mathtt{a} < \mathtt{J})$⟆

...

$P_{10}^{1,1} = \quad \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{J}+1) \wedge \mathtt{a} < \mathtt{J} = \mathtt{b} - 1 \wedge \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{I}, \mathtt{I}, \mathtt{I}) \wedge \mathtt{I} = \mathtt{J}$

⟅as in Sec. 4.7 since the inner loop does not modify a, b or I and the swap $\mathtt{t}[\mathtt{I}] :=: \mathtt{t}[\mathtt{I}+1]$ does not interfere with $\mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{J}+1)$ according to $\mathtt{a} \leq \mathtt{I} < \mathtt{I}+1 \leq \mathtt{J} < \mathtt{J}+1$ so $\mathtt{I}, \mathtt{I}+1 \in [\mathtt{a}, \mathtt{J}+1]$ and (21)⟆ (27)

$\Rightarrow \quad \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{J}+1) \wedge \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}, \mathtt{J}, \mathtt{J}) \wedge \mathtt{a} < \mathtt{J} = \mathtt{b} - 1$ ⟅by elimination of I is dead at program point 10⟆

$\Rightarrow \quad \mathrm{s}(\mathtt{t}, \mathtt{J}, \mathtt{b}) \wedge \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}, \mathtt{J}, \mathtt{b}) \wedge \mathtt{a} < \mathtt{J} = \mathtt{b} - 1$ ⟅by reduction (24)⟆

$P_{11}^{1,1} = \quad \mathrm{s}(\mathtt{t}, \mathtt{J}+1, \mathtt{b}) \wedge \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{a} \leq \mathtt{J} = \mathtt{b} - 2$ ⟅by assignment $\mathtt{J} := \mathtt{J} - 1$⟆

$P_3^{1,2} = \quad \mathrm{s}(\mathtt{t}, \mathtt{J}+1, \mathtt{b}) \wedge \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{a} < \mathtt{J} = \mathtt{b} - 2$ ⟅by semantical loop unrolling (since a new symbolic "s" term has appeared, see Sec. 4.5,) and test $(\mathtt{a} < \mathtt{J})$⟆

...

$$
\begin{array}{rcl}
P_{10}^{1,2} & = & \mathrm{s}(\mathtt{t},\mathtt{J}{+}1,\mathtt{b}){\wedge}\mathrm{lt}(\mathtt{t},\mathtt{a},\mathtt{J}{+}1,\mathtt{J}{+}1,\mathtt{b}){\wedge}\mathtt{a}<\mathtt{J}=\mathtt{b}{-}2{\wedge}\mathrm{lt}(\mathtt{t},\mathtt{a},\mathtt{I},\mathtt{I},\mathtt{I}){\wedge}\mathtt{I}=\mathtt{J} \\
 & & \text{⟅by Sec. 4.7 and non interference, see (27)⟆} \\
 & \Rightarrow & \mathrm{s}(\mathtt{t},\mathtt{J}+1,\mathtt{b}) \wedge \mathrm{lt}(\mathtt{t},\mathtt{a},\mathtt{J}+1,\mathtt{J}+1,\mathtt{b}) \wedge \mathtt{a} < \mathtt{J} = \mathtt{b}-2 \wedge \mathrm{lt}(\mathtt{t},\mathtt{a},\mathtt{J},\mathtt{J},\mathtt{J}) \\
 & & \text{⟅since I is dead⟆} \\
 & \Rightarrow & \mathrm{s}(\mathtt{t},\mathtt{J},\mathtt{b}) \wedge \mathrm{lt}(\mathtt{t},\mathtt{a},\mathtt{J},\mathtt{J},\mathtt{b}) \wedge \mathtt{a} < \mathtt{J} = \mathtt{b}-2 \qquad \text{⟅by reduction (25)⟆} \\
P_{11}^{1,2} & = & \mathrm{s}(\mathtt{t},\mathtt{J}+1,\mathtt{b}) \wedge \mathrm{lt}(\mathtt{t},\mathtt{a},\mathtt{J}+1,\mathtt{J}+1,\mathtt{b}) \wedge \mathtt{a} \leq \mathtt{J} = \mathtt{b}-3 \quad \text{⟅by assignment } \mathtt{J} := \mathtt{J}-1\text{⟆} \\
P_{3}^{2,2} & = & (P_3^{1,2} \nabla (P_{11}^{1,2} \wedge (\mathtt{a} < \mathtt{J}))) \wedge (\mathtt{a} < \mathtt{J}) \quad \text{⟅loop unrolling stops in absence of new abstract term and widening speeds-up convergence⟆} \\
 & = & ((\mathrm{s}(\mathtt{t},\mathtt{J}+1,\mathtt{b}) \wedge \mathrm{lt}(\mathtt{t},\mathtt{a},\mathtt{J}+1,\mathtt{J}+1,\mathtt{b}) \wedge \mathtt{a} < \mathtt{J} = \mathtt{b}-2) \nabla (\mathrm{s}(\mathtt{t},\mathtt{J}+1,\mathtt{b}) \wedge \\
 & & \mathrm{lt}(\mathtt{t},\mathtt{a},\mathtt{J}+1,\mathtt{J}+1,\mathtt{b}) \wedge \mathtt{a} \leq \mathtt{J} = \mathtt{b}-3 \wedge (\mathtt{a} < \mathtt{J}))) \wedge (\mathtt{a} < \mathtt{J}) \quad \text{⟅def. } P_3^{1,2} \text{ and } P_{11}^{1,2}\text{⟆} \\
 & = & \mathrm{s}(\mathtt{t},\mathtt{J}+1,\mathtt{b}) \wedge \mathrm{lt}(\mathtt{t},\mathtt{a},\mathtt{J}+1,\mathtt{J}+1,\mathtt{b}) \wedge ((\mathtt{a} < \mathtt{J} = \mathtt{b}-2) \nabla (\mathtt{a} < \mathtt{J} = \\
 & & \mathtt{b}-3)) \wedge (\mathtt{a} < \mathtt{J}) \qquad \text{⟅def. widening⟆} \\
 & = & \mathrm{s}(\mathtt{t},\mathtt{J}+1,\mathtt{b}) \wedge \mathrm{lt}(\mathtt{t},\mathtt{a},\mathtt{J}+1,\mathtt{J}+1,\mathtt{b}) \wedge \mathtt{a} < \mathtt{J} \leq \mathtt{b}-2 \quad \text{⟅def. octagonal widening and conjunction⟆}
\end{array}
$$

...

$$
\begin{array}{rcl}
P_{10}^{2,2} & = & \mathrm{s}(\mathtt{t},\mathtt{J}{+}1,\mathtt{b}){\wedge}\mathrm{lt}(\mathtt{t},\mathtt{a},\mathtt{J}{+}1,\mathtt{J}{+}1,\mathtt{b}){\wedge}\mathtt{a}<\mathtt{J}\leq\mathtt{b}{-}2{\wedge}\mathrm{lt}(\mathtt{t},\mathtt{a},\mathtt{I},\mathtt{I},\mathtt{I}){\wedge}\mathtt{I}=\mathtt{J} \\
 & & \text{⟅by Sec. 4.7 and non interference, see (27)⟆} \\
 & = & \mathrm{s}(\mathtt{t},\mathtt{J}+1,\mathtt{b}) \wedge \mathrm{lt}(\mathtt{t},\mathtt{a},\mathtt{J}+1,\mathtt{J}+1,\mathtt{b}) \wedge \mathtt{a} < \mathtt{J} \leq \mathtt{b}-2 \wedge \mathrm{lt}(\mathtt{t},\mathtt{a},\mathtt{J},\mathtt{J},\mathtt{J}) \\
 & & \text{⟅by elimination of the dead variable I⟆} \\
 & \Rightarrow & \mathrm{s}(\mathtt{t},\mathtt{J},\mathtt{b}) \wedge \mathrm{lt}(\mathtt{t},\mathtt{a},\mathtt{J},\mathtt{J},\mathtt{b}) \wedge \mathtt{a} < \mathtt{J} \leq \mathtt{b}-2 \qquad \text{⟅by reduction (25)⟆} \\
P_{11}^{2,2} & = & \mathrm{s}(\mathtt{t},\mathtt{J}+1,\mathtt{b}) \wedge \mathrm{lt}(\mathtt{t},\mathtt{a},\mathtt{J}+1,\mathtt{J}+1,\mathtt{b}) \wedge \mathtt{a} \leq \mathtt{J} \leq \mathtt{b}-3 \quad \text{⟅by assignment } \mathtt{J} := \mathtt{J}-1\text{⟆}
\end{array}
$$

Now $(P_{11}^{2,2} \wedge \mathtt{a} < \mathtt{J}) \Rightarrow P_3^{1,2}$ so that the loop iterates stabilize to a post-fixpoint. On loop exit, we must collect all cases following from semantic unrolling:

$$
\begin{array}{rcll}
P_{12}^{2} & = & (P_2^1 \wedge \mathtt{a} \geq \mathtt{J}) & \text{⟅no entry in the loop⟆} \\
 & & \vee (P_{11}^{1,0} \wedge \mathtt{a} \geq \mathtt{J}) & \text{⟅loop exit after one iteration⟆} \\
 & & \vee (P_{11}^{1,1} \wedge \mathtt{a} \geq \mathtt{J}) & \text{⟅loop exit after two iterations⟆} \\
 & & \vee (P_{11}^{2,2} \wedge \mathtt{a} \geq \mathtt{J}) & \text{⟅loop exit after three iterations or more⟆} \\
 & = & (\mathtt{a} = \mathtt{J} = \mathtt{b}) \vee (\mathrm{s}(\mathtt{t},\mathtt{J}+1,\mathtt{b}) \wedge \mathrm{lt}(\mathtt{t},\mathtt{a},\mathtt{J}+1,\mathtt{J}+1,\mathtt{b}) \wedge \mathtt{a} = \mathtt{J} \leq \mathtt{b}-1) & \\
 & & \text{⟅def. abstract disjunction⟆} & \\
 & = & (\mathtt{a} = \mathtt{J} = \mathtt{b}) \vee (\mathrm{s}(\mathtt{t},\mathtt{a}+1,\mathtt{b}) \wedge \mathrm{lt}(\mathtt{t},\mathtt{a},\mathtt{a}+1,\mathtt{a}+1,\mathtt{b}) \wedge \mathtt{a} < \mathtt{b}) & \text{⟅elimination of dead variable J⟆} \\
 & = & (\mathtt{a} = \mathtt{b}) \vee (\mathrm{s}(\mathtt{t},\mathtt{a},\mathtt{b}) \wedge \mathtt{a} < \mathtt{b}) & \text{⟅by reduction (26)⟆} \\
 & = & \mathrm{s}(\mathtt{t},\mathtt{a},\mathtt{b}) \wedge \mathtt{a} \leq \mathtt{b} & \text{⟅definition of abstract disjunction similar to (16)⟆}
\end{array}
$$

[5] Notice that this notation is a shorthand for the more explicit notation $\exists i,j,k,\ell : \mathrm{lt}(\mathtt{t},i,j,k,\ell) \wedge i = \mathtt{a} \wedge j = \mathtt{I} \wedge k = \mathtt{I} \wedge \ell = \mathtt{I}) \wedge \mathtt{a} < \mathtt{b} \wedge \mathtt{b} = \mathtt{J} \wedge \mathtt{I} = \mathtt{J}$ as used in Sec. 4.7, so that, in particular, we freely replace i, j, k and ℓ in $\mathrm{lt}(\mathtt{t},i,j,k,\ell)$ by equivalent expressions.

The sorting proof would proceed in the same way by proving that the final array is a permutation of the original one.

Observe that *parametric predicate abstraction* is defined for a programming language as opposed to *ground predicate abstraction* which is specific to a program, a usual distinction between abstract interpretation-based static program analysis (a parametric abstraction for an infinite set of programs) and abstract model checking (an abstract model for a given program). Notice that the *polymorphic predicate abstraction* of [2] is an instance of symbolic relational separate procedural analysis [14, Sec. 7] for *ground* predicate abstraction. The generalization to parametric predicate abstraction is immediate since it only depends on the way concrete predicate transformers are defined (see [14, Sec. 7]).

5 Conclusion

In safety proofs by ground predicate abstraction, one has to provide (or compute by refinement) the ground atomic components of the inductive invariant which is to be discovered for the proof. Then the routine work of assembling the atomic components into a valid inductive invariant is mechanized which simplifies the proof. If the set of atomic components is finite then a Boolean encoding allows for the reuse of model-checkers for fixpoint computation. Otherwise a specific fixpoint engine has to be used, which is mandatory even in cases as simple as constant propagation if the constants are to be discovered automatically and not explicitly provided in the list of ground atomic predicates.

Parametric predicate abstraction provides a further abstraction step in that hints for the proof are provided in the form of parameterized atomic predicates (which will be instantiated automatically to program-specific ground predicates) and reduction rules (which are hints for inductive reasoning on these parametric predicates). This parametricity immediately leads to infinite abstract domains which means that fixpoint iterations need more sophisticated inferences, which we can provide in the simple form of widenings. Moreover, the presentation in the form of structured abstract domains, which can be systematically composed, reduces the need to appeal to theorem provers by reduction of the widening to well-studied and powerful basic relational abstract domains which can be viewed as undecidable theories with finitary extrapolation through widenings. Parametric predicate abstractions can handle large families of algorithms and data structures (as considered in [5]) and so is of much wider scope than ground predicate abstraction restricted to a single program and requiring a costly refinement process when ground predicates are missing.

Acknowledgements. I thank the participants to the informal meeting on predicate abstraction at New York University on Thursday Jan. 30^{th} 2003 (Radhia Cousot, Dennis Dams, Kedar Namjoshi, Amir Pnueli, Lenore Zuck), in particular Amir Pnueli who proposed the bubble sort as a challenge that is handled in Sec. 4. I thank Pavol Černý for providing a prototype implementation [6] and further examples (such as array initialization and Quicksort).

References

1. T. Ball, R. Majumdar, T.D. Millstein and S.K. Rajamani. Automatic Predicate Abstraction of C Programs. In *Proc. ACM SIGPLAN '2001 Conf. PLDI*, pp. 203–213, 2001. ACM Press.
2. T. Ball, T. Millstein, and S. Rajamani. Polymorphic predicate abstraction. Tech. rep. MSR-TR-2001-10, Microsoft Research, Redmond, 17 June 2002. 21 p.
3. T. Ball, A. Podelski, and S. Rajamani. Relative completeness of abstraction refinement for software model checking. In J.-P. Katoen and P. Stevens, eds., *Proc. 8^{th} Int. Conf. TACAS '2002*, LNCS 2280, pp. 158–172. Springer, 2002.
4. N. Bjørner, I.A. Browne and Z. Manna. Automatic Generation of Invariants and Intermediate Assertions. *Theor. Comp. Sci.*, 173(1):49–87, 1997.
5. B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. A static analyzer for large safety-critical software. In *Proc. ACM SIGPLAN '2003 Conf. PLDI*, pp. 196–207, 2003. ACM Press.
6. P. Černý. Vérification par intérprétation abstraite de prédicats paramétriques. D.E.A. report, Univ. Paris VII & École normale supérieure, Paris, 20 Sep. 2003.
7. P. Cousot. *Méthodes itératives de construction et d'approximation de points fixes d'opérateurs monotones sur un treillis, analyse sémantique de programmes.* Thèse d'État ès sciences mathématiques, Université scientifique et médicale de Grenoble, Grenoble, 21 Mar. 1978.
8. P. Cousot. Semantic foundations of program analysis. In S.S. Muchnick and N.D. Jones, editors, *Program Flow Analysis: Theory and Applications*, chapter 10, pages 303–342. Prentice-Hall, 1981.
9. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *4^{th} POPL*, pp. 238–252, 1977. ACM Press.
10. P. Cousot and R. Cousot. Automatic synthesis of optimal invariant assertions: mathematical foundations. In *ACM Symp. on Artificial Intelligence & Programming Languages*, ACM SIGPLAN Not. 12(8):1–12, 1977.
11. P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *6^{th} POPL*, pp. 269–282, 1979. ACM Press.
12. P. Cousot and R. Cousot. Abstract interpretation and application to logic programs. *J. Logic Programming*[6], 13(2–3):103–179, 1992.
13. P. Cousot and R. Cousot. Abstract interpretation frameworks. *J. Logic and Comp.*, 2(4):511–547, Aug. 1992.
14. P. Cousot and R. Cousot. Modular static program analysis, invited paper. In R. Horspool, ed., *Proc. 11^{th} Int. Conf. CC '2002*, pp. 159–178, 2002. LNCS 2304, Springer.
15. S. Graf and H. Saïdi. Construction of abstract state graphs with PVS. In O. Grumberg, ed., *Proc. 9^{th} Int. Conf. CAV '97*, LNCS 1254, pp. 72–83. Springer, 1997.
16. G. Kildall. A unified approach to global program optimization. In *1^{st} POPL*, pp. 194–206, 1973. ACMpress.
17. Z. Manna. *Mathematical theory of computation.* McGraw Hill, 1972.
18. A. Miné. A few graph-based relational numerical abstract domains. In M. Hermenegildo and G. Puebla, eds., *SAS'02*, LNCS 2477, pp. 117–132. Springer, 2002.
19. A. Mycroft. Completeness and predicate-based abstract interpretation. In *Proc. PEPM '93*, 1993, pp. 80–87. ACM Press, 1993.

[6] The editor of J. Logic Programming has mistakenly published the unreadable galley proof. For a correct version of this paper, see `http://www.di.ens.fr/~cousot`.