Completeness in Abstract Interpretation: A Domain Perspective

Roberto Giacobazzi[⋆] Francesco Ranzato[⋆⋆]

[⋆] *Dipartimento di Informatica, Università di Pisa*
Corso Italia 40, 56125 Pisa, Italy
giaco@di.unipi.it

[⋆⋆] *Dipartimento di Matematica Pura ed Applicata, Università di Padova*
Via Belzoni 7, 35131 Padova, Italy
franz@math.unipd.it

Abstract. Completeness in abstract interpretation is an ideal and rare situation where the abstract semantics is able to take full advantage of the power of representation of the underlying abstract domain. In this paper, we develop an algebraic theory of completeness in abstract interpretation. We show that completeness is an abstract domain property and we prove that there always exist both the greatest complete restriction and the least complete extension of any abstract domain, with respect to continuous semantic functions. Under certain hypotheses, a constructive procedure for computing these complete domains is given. These methodologies provide advanced algebraic tools for manipulating abstract interpretations, which can be fruitfully used both in program analysis and in semantics design.

1 Introduction

Abstract interpretation [8, 9] is a widely established methodology for programming language semantics approximation, which is primarily used for specifying and then validating static program analyses. Given a so-called concrete semantics defined by a concrete domain C (a complete lattice) and a semantic function $[\![\cdot]\!] : \mathit{Program} \rightarrow C$, an abstract interpretation is specified by an abstract domain A (a complete lattice) and an abstract semantic function $[\![\cdot]\!]^\sharp : \mathit{Program} \rightarrow A$, where the relationship between concrete and abstract objects is formalized by a pair of adjoint maps $\alpha : C \rightarrow A$ and $\gamma : A \rightarrow C$ such that $\alpha(c) \leq_A a$ means that a is a correct approximation of c. Then, a typical *soundness* theorem for an abstract interpretation goes as follows: For all programs P, $\alpha([\![P]\!]) \leq_A [\![P]\!]^\sharp$. It is well-known [8] that for the non-restrictive case of least fixpoint based semantics, i.e. where $[\![P]\!] = \mathit{lfp}(T_P)$ and $[\![P]\!]^\sharp = \mathit{lfp}(T_P^\sharp)$ for some monotone operators $T_P : C \rightarrow C$ and $T_P^\sharp : A \rightarrow A$ indexed over *Program*, soundness is implied by the following stronger, but nevertheless much easier to check, condition: $\alpha \circ T_P \leq_A T_P^\sharp \circ \alpha$.

While soundness is the basic requirement for any abstract interpretation, the dual notion of *completeness* is instead an ideal and quite rare situation. Completeness arises when no loss of precision occurs by approximating $\alpha([\![P]\!])$ with $[\![P]\!]^\sharp$, i.e. when $\alpha([\![P]\!]) = [\![P]\!]^\sharp$. Roughly speaking, this means that the abstract semantics is able to take *full* advantage of the power of representation of the abstract domain A. In this sense, complete abstract interpretations can be rightfully considered as optimal. As before, for least fixpoint based semantics, completeness is implied by the following stronger condition called *full completeness*: $\alpha \circ T_P = T_P^\sharp \circ \alpha$ (cf. [9]). For instance, the classical naïve "rule of signs" abstract interpretation is fully complete. In fact, the sign of a concrete integer multiplication can be exactly retrieved by the rule of signs applied to its arguments, i.e., by leaving out the details, $\mathit{sign}(n \cdot m) = \mathit{sign}(n) \cdot^\sharp \mathit{sign}(m)$, where $\cdot^\sharp$ is the obvious abstract multiplication between signs.

The problem of achieving the completeness for an abstract interpretation, by enhancing either the abstract domain or the abstract semantic operators, has been investigated by a number of authors (see Section 9). While this has been successfully solved for some specific abstract interpretations and analyses, the more general problem of making a generic abstract interpretation complete in the best possible way (i.e. involving the most simple abstract domains and operators), is still, to the best of our knowledge, open.

We attack this problem from a domain perspective, since we show that, fixed a concrete semantics, both completeness and fully completeness for an abstract interpretation only depend on the underlying abstract domain. Thus, we develop an algebraic theory of domain completeness within the classical abstract interpretation framework. We concentrate on the set of all the domains, in the lattice $\mathcal{L}_C$ of abstract interpretations of the fixed concrete domain C, which are complete and fully complete for a given family of semantic operators F, denoted resp. by $\Delta(C,F)$ and $\Gamma(C,F)$. In Section 4, we prove that both $\Delta(C,F)$ and $\Gamma(C,F)$ are always complete meet subsemilattices of $\mathcal{L}_C$. Moreover, while we show that, in general, $\Delta(C,F)$ is not a join subsemilattice of $\mathcal{L}_C$, even under very restrictive hypotheses on C and F, by contrast we prove that, when the functions in F are (Scott-)continuous, $\Gamma(C,F)$ is a complete join subsemilattice of $\mathcal{L}_C$, and therefore a complete sublattice. It should be remarked that this latter result is far from being trivial.

Based on these results, in Section 5, we introduce a family of operators acting on abstract domains, which transform non-complete domains into complete or fully complete ones. There are two possibilities for doing this: Either by refining domains, i.e. by enhancing their precision by adding new elements, or by simplifying them by taking out some information which may cause incompleteness. Thus, following the ideas on systematic abstract domain refinements and simplifications introduced in [14, 18], we define the *complete* and *fully complete kernel* operators $\mathbb{K}$ and $\mathcal{K}$, and the *least fully complete extension* operator $\mathcal{E}$. The first two are abstract domain simplifications which, given a set of concrete monotone functions F and an input abstract domain A, give as output the most concrete domains $\mathbb{K}(A)$ and $\mathcal{K}(A)$ which are more abstract than A and complete, resp. fully complete, for any $f \in F$. $\mathcal{E}$ is instead an abstract domain refinement which, given a set of concrete continuous functions F and A, returns the most abstract domain $\mathcal{E}(A)$ which is an extension (i.e. more precise) of A and fully complete for any $f \in F$. By the aforementioned negative findings on the structure of $\Delta(C,F)$, an analogous least complete extension operator is not generally definable. These operators satisfy a number of relevant algebraic properties; in particular, we show that the least fully complete extension of a domain can be always achieved by decomposing the input domain into simpler factors and then by refining these simpler domains. In Section 6, we present a constructive method for designing least fully complete extensions and fully complete kernels of abstract domains, under the hypotheses that the concrete semantic functions in F are additive.

As a relevant example, we reconstruct the Cousot and Cousot [8] abstract domain of integer intervals as the least fully complete extension for integer addition of the rule of signs domain. Clearly, to be an abstract domain is a relative notion. Thus, our systematic operators can be also applied to refine or simplify domains for analysis relatively to other more precise – but still approximated – ones. In Section 8, we show how to apply our operators to devise an intelligent strategy for improving the precision of abstract domains, which takes into account the efficiency/precision trade-off in a systematic refinement step. We apply this idea to compare the expressive power of some well-known abstract domains for ground-dependency analysis of logic programs.

2 Basic Notions

The structure $\langle uco(C), \sqsubseteq, \sqcup, \sqcap, \lambda x.\top, \lambda x.x\rangle$ denotes the complete lattice of all *upper closure operators* (shortly closures) on a complete lattice $\langle C, \leq, \vee, \wedge, \top, \bot\rangle$ (i.e., monotone, idempotent and extensive operators on C), where (i) $\rho \sqsubseteq \eta$ iff $\forall x \in C.\ \rho(x) \leq \eta(x)$, (ii) $(\sqcup_{i\in I}\rho_i)(x) = x \Leftrightarrow \forall i \in I.\ \rho_i(x) = x$; (iii) $(\sqcap_{i\in I}\rho_i)(x) = \wedge_{i\in I}\rho_i(x)$; (iv) $\lambda x.\top$ and $\lambda x.x$ are, respectively, the top and bottom. The complete lattice of all *lower closure operators* on C is denoted by $lco(C)$ and is dually isomorphic to $uco(C)$. Recall that each closure $\rho \in uco(C)$ is uniquely determined by the set of its fixpoints, which is its image, i.e. $\rho(C) = \{x \in C \mid \rho(x) = x\}$, that $\rho \sqsubseteq \eta$ iff $\eta(C) \subseteq \rho(C)$, and that a subset $X \subseteq C$ is the set of fixpoints of a closure iff X is meet-closed, i.e. $X = \mathcal{M}(X) = \{\wedge Y \mid Y \subseteq X\}$ (note that $\top \in X$). $\langle \rho(C), \leq\rangle$ is a complete meet subsemilattice of C, while it is a complete sublattice iff ρ is completely additive. Let us also recall that $uco(C)$ is dual-atomic, i.e., for any $\rho \in uco(C)$, $\rho = \sqcap_{x\in\rho(C)\setminus\{\top\}}\varphi_x$, where each closure $\varphi_x = \{\top, x\}$, for $x \in C \setminus \{\top\}$, is a dual-atom in $uco(C)$.

In the standard Cousot and Cousot abstract interpretation theory, abstract domains can be equivalently specified either by Galois connections (GCs) or by closure operators (see [9]). In the first case, the concrete domain C and the abstract domain A (both assumed to be complete lattices) are related by a pair of adjoint functions of a GC (α, C, A, γ). If (α, C, A, γ) is a Galois insertion (GI), each element in A is useful to represent the concrete domain C, being α onto. Any GC (α, C, A, γ) may be lifted to a GI by reduction of the abstract domain A, i.e. by identifying in an equivalence class those elements in A having the same concrete meaning. In the second case instead, an abstract domain is specified as (the set of fixpoints of) an upper closure on the concrete domain. These two approaches are completely equivalent: If $\rho \in uco(C)$ and $A \cong \rho(C)$ (with $\iota : \rho(C) \to A$ and $\iota^{-1} : A \to \rho(C)$ being the isomorphism) then $(\iota \circ \rho, C, A, \iota^{-1})$ is a GI; if (α, C, A, γ) is a GI then $\rho_A = \gamma \circ \alpha \in uco(C)$ is the closure associated with A such that $\rho_A(C) \cong A$; moreover, these two constructions are one the inverse of the other. Hence, we will identify $uco(C)$ with the so-called *lattice of abstract interpretations* of C, viz. the complete lattice of all abstract domains of the concrete domain C. Often, we will find convenient to identify closures with their sets of fixpoints, denoted as sets by capital Latin letters; instead, when viewing closures as functions, they will be denoted by Greek letters. We keep this soft ambiguity, since one can distinguish their use as functions or sets, according to the context. The ordering on $uco(C)$ corresponds precisely to the standard order used in abstract interpretation to compare abstract domains with regard to their precision: A_1 is *more precise* than A_2 iff $A_1 \sqsubseteq A_2$ in $uco(C)$. The *lub* and *glb* on $uco(C)$ have therefore the following meaning as operators on domains. Suppose $\{A_i\}_{i\in I} \subseteq uco(C)$: (i) $\sqcup_{i\in I}A_i$ is the most concrete among the domains which are abstractions of all the A_i's, i.e. it is their least common abstraction; (ii) $\sqcap_{i\in I}A_i$ is (isomorphic to) the well-known reduced product of all the A_i's, and, equivalently, it is the most abstract among the domains (abstracting C) which are more concrete than every A_i. Whenever C is a meet-continuous complete lattice (i.e., for any chain $Y \subseteq C$ and $x \in C$, $x \wedge (\vee Y) = \vee_{y\in Y}(x \wedge y)$), $uco(C)$ enjoys the lattice-theoretic property of *pseudocomplementedness* (cf. [17]). This property allowed to define the operation of *complementation* of abstract domains (cf. [6]), namely an operation which, starting from any two domains $D \sqsubseteq A$, where D is meet-continuous, gives as result the most abstract domain $D \sim A$, such that $(D \sim A) \sqcap A = D$. A (*conjunctive*) *decomposition* of an abstract domain $A \in uco(C)$ is any tuple of domains $\langle D_i\rangle_{i\in I} \subseteq uco(C)$ such that $A = \sqcap_{i\in I}D_i$. Complementation is important for decomposing abstract domains: If $D \sqsubseteq A$ then $\langle D \sim A, A\rangle$ is a (binary) decomposition for C, and more general decompositions can be obtained by complementation (see [6]).

3 Completeness in Abstract Interpretation

Let *Program* denote the set of (syntactically well-formed) programs. The concrete standard semantics is in general specified by a semantic function $[\![\cdot]\!] : \mathit{Program} \rightarrow C$, where C is a concrete semantic domain of denotations, which we assume to be a complete lattice. If an abstract interpretation is specified by a GI (α, C, A, γ) and by an abstract semantic function $[\![\cdot]\!]^\sharp : \mathit{Program} \rightarrow A$, then $[\![\cdot]\!]^\sharp$ is a *sound* abstract semantics, or (*correctly*) *approximates* $[\![\cdot]\!]$, if, for any program P, $\alpha([\![P]\!]) \leq_A [\![P]\!]^\sharp$, or, equivalently, $[\![P]\!] \leq_C \gamma([\![P]\!]^\sharp)$. The pattern of definition of $[\![\cdot]\!]$ obviously depends on the considered programming language and on the semantics style adopted. We follow here a customary *least fixpoint* semantic approach, which is general enough to subsume and include most kinds of semantic specifications (see [12]). In the following, for two complete lattices C and D, we denote by $C \xrightarrow{\mathrm{m}} D$, $C \xrightarrow{\mathrm{c}} D$, and $C \xrightarrow{\mathrm{a}} D$, respectively, the set of all monotone, (Scott-)continuous and (completely) additive (i.e. preserving all *lub*'s) functions from C to D. A concrete semantics is therefore specified by a pair $\langle C, T \rangle$, where C is a complete lattice and $T : \mathit{Program} \rightarrow (C \xrightarrow{\mathrm{m}} C)$. For $P \in \mathit{Program}$, we use T_P to denote more compactly $T(P)$. The least fixpoint semantics of any program P is then given by $[\![P]\!] = \mathit{lfp}(T_P) \in C$. On the abstract side, for some $T^\sharp : \mathit{Program} \rightarrow (A \xrightarrow{\mathrm{m}} A)$, the abstract least fixpoint semantics is analogously defined by $[\![P]\!]^\sharp = \mathit{lfp}(T^\sharp_P)$.

Given a concrete semantics $\mathcal{S} = \langle C, T \rangle$ and an abstract semantics $\mathcal{S}^\sharp = \langle A, T^\sharp \rangle$, related by a GI (α, C, A, γ), $\mathcal{S}^\sharp$ is called a *sound* abstraction of $\mathcal{S}$ if for all $P \in \mathit{Program}$, $\alpha(\mathit{lfp}(T_P)) \leq_A \mathit{lfp}(T^\sharp_P)$. This soundness condition can be more easily verified by checking whether for all $P \in \mathit{Program}$, $\alpha \circ T_P \leq_A T^\sharp_P \circ \alpha$, or, equivalently, $\alpha \circ T_P \circ \gamma \leq_C T^\sharp_P$. We distinguish between these two forms of soundness and we say that $\mathcal{S}^\sharp$ is a *fully sound* abstraction of $\mathcal{S}$ if for all $P \in \mathit{Program}$, $\alpha \circ T_P \leq_A T^\sharp_P \circ \alpha$.

In abstract interpretation, the term completeness is used dually to the above notion of soundness [9, 11, 22]. Again, one distinguishes between a weaker form of completeness, involving least fixpoints only, and a stronger one (but easier to verify) involving semantic functions. We say that $\mathcal{S}^\sharp$ is a (*fully*) *complete* abstraction of $\mathcal{S}$ if for all $P \in \mathit{Program}$, $(T^\sharp_P \circ \alpha \leq_A \alpha \circ T_P)$ $\mathit{lfp}(T^\sharp_P) \leq_A \alpha(\mathit{lfp}(T_P))$. Because soundness is always required in abstract interpretation, in the following we abuse terminology and say that $\mathcal{S}^\sharp$ is (fully) complete for $\mathcal{S}$ if for all $P \in \mathit{Program}$, $(\alpha \circ T_P = T^\sharp_P \circ \alpha)$ $\alpha(\mathit{lfp}(T_P)) = \mathit{lfp}(T^\sharp_P)$. We also use such notions of completeness and full completeness locally for a given pair of semantic functions $T^\sharp_P$ and T_P.

Completeness as a Domain Property. For a pair of semantic functions $T_P : C \rightarrow C$ and $T^\sharp_P : A \rightarrow A$, when $\alpha \circ T_P \circ \gamma \leq_C T^\sharp_P$ holds, $T^\sharp_P$ is traditionally called a correct approximation of T_P [9]. It is also well-known since [9, Corollary 7.2.0.4], that the abstract domain A induces a *best* correct approximation of T_P given by $T^A_P = \alpha \circ T_P \circ \gamma$. Consequently, A always induces an (automatically) fully sound abstract semantics $\langle A, \lambda P.\, T^A_P \rangle$. By contrast, this is not true for completeness, i.e., for a given abstract domain A it may well happen that it is not possible to define a fully complete or merely complete abstract semantics based on A – on the contrary, this is the most frequent situation. Furthermore, if A admits a fully complete abstract semantic operator $T^\sharp_P$, then T^A_P is fully complete as well, and $T^A_P = T^\sharp_P$: $T^\sharp_P = T^\sharp_P \circ \alpha \circ \gamma = \alpha \circ T_P \circ \gamma = T^A_P$. Likewise, if $T^\sharp_P$ is complete then T^A_P is complete: $\alpha(\mathit{lfp}(T_P)) \leq_A \mathit{lfp}(T^A_P) \leq_A \mathit{lfp}(T^\sharp_P) = \alpha(\mathit{lfp}(T_P))$. Thus, we get the following important characterization of completeness as a domain property:

> *It is possible to define a (fully) complete abstract semantic operator on an abstract domain A if and only if the best correct approximation induced by A is (fully) complete.*

4 The Lattice of Complete Abstract Interpretations

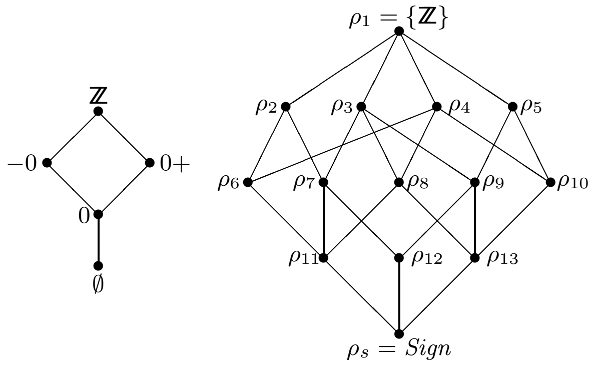
We have seen that one can consider, without loss of generality, completeness and full completeness for best correct approximations only. Moreover, by the equivalence between the GI and closure operator approaches to abstract domain design, completeness and full completeness can be equivalently specified for closure operators: In fact, it turns out that for a GI (α, C, A, γ) and $f : C \xrightarrow{\mathrm{m}} C$, the best correct approximation f^A is (fully) complete iff $\gamma(\alpha(\mathit{lfp}(f))) = \mathit{lfp}(\gamma \circ \alpha \circ f)$ $((\gamma \circ \alpha) \circ f = (\gamma \circ \alpha) \circ f \circ (\gamma \circ \alpha))$. Thus, in the following, we will study completeness and full completeness relatively to closure operators and generic (monotone) functions from a purely algebraic point of view, and say that A is (fully) complete for f if f^A is (fully) complete. We generalize full completeness to cope with generic (possibly nonmonotone) n-ary functions. If $\vec{o}$ denotes a generic tuple of objects, then $\vec{o}_i$ denotes its i-th component.

Definition 4.1 Let C be a complete lattice.
(i) Given $f : C^n \to C$ $(n \geq 1)$, $\rho \in \mathit{uco}(C)$ is *fully complete* for f if for any $\vec{x} \in C^n$, $\rho(f(\vec{x})) = \rho(f(\rho(\vec{x}_1), ..., \rho(\vec{x}_n)))$.
(ii) Given $f \in C \xrightarrow{\mathrm{m}} C$, $\rho \in \mathit{uco}(C)$ is *complete* for f if $\rho(\mathit{lfp}(f)) = \mathit{lfp}(\rho \circ f)$. □

We denote the condition in (i) simply by $\rho \circ f = \rho \circ f \circ \rho$. Note that (i) encompasses also functions of type $C \to (C \to \cdots (C \to C) \cdots)$, by "Curryfying" them; moreover, $\mathit{lfp}(\rho \circ f)$ in (ii) could be equivalently replaced by $\mathit{lfp}(\rho \circ f \circ \rho)$.

For $f \in C^n \to C$, we define $\Gamma(C,f) \subseteq \mathit{uco}(C)$ to be the set of fully complete closures on C for f: $\Gamma(C,f) = \{\rho \in \mathit{uco}(C) \mid \rho \circ f = \rho \circ f \circ \rho\}$. If $f : C \xrightarrow{\mathrm{m}} C$ then we define $\Delta(C,f) \subseteq \mathit{uco}(C)$ as the set of complete closures on C for f: $\Delta(C,f) = \{\rho \in \mathit{uco}(C) \mid \rho(\mathit{lfp}(f)) = \mathit{lfp}(\rho \circ f)\}$. Also, if $\eta \in \mathit{uco}(C)$ then $\Gamma^{\uparrow\eta}(C,f)$ and $\Delta^{\uparrow\eta}(C,f)$ are the set of closures on $\langle \eta(C), \leq_C \rangle$ that are, respectively, fully complete and complete (for f); since $\rho \in \mathit{uco}(\eta(C))$ iff $\rho \in \mathit{uco}(C)$ and $\rho \sqsubseteq \eta$, then, by denoting $\uparrow \eta$ the principal filter of $\mathit{uco}(C)$ generated by η, we have that $\Gamma^{\uparrow\eta}(C,f) = \Gamma(C,f) \cap \uparrow\eta$ and $\Delta^{\uparrow\eta}(C,f) = \Delta(C,f) \cap \uparrow\eta$. If (α, C, A, γ) is a GI such that $\gamma \circ \alpha = \eta$, then $\Gamma^{\uparrow A}(C,f)$ and $\Delta^{\uparrow A}(C,f)$ are alternative notations for $\Gamma^{\uparrow\eta}(C,f)$ and $\Delta^{\uparrow\eta}(C,f)$ respectively. We can also define completeness and full completeness relatively to any set of concrete functions: If $F \subseteq C^n \to C$ and $G \subseteq C \xrightarrow{\mathrm{m}} C$ then $\Gamma(C, F) = \cap_{f \in F}\Gamma(C,f)$ and $\Delta(C, G) = \cap_{g \in G}\Delta(C, g)$ (obviously, $\Gamma(C, \emptyset) = \Delta(C, \emptyset) = \mathit{uco}(C)$). We can then restate the basic Cousot and Cousot [9] result on completeness using our notation as follows: If $F \subseteq C \xrightarrow{\mathrm{m}} C$ then $\Gamma(C, F) \subseteq \Delta(C, F)$.

Example 4.2 Consider the classical "rule of signs" domain *Sign* in Fig. 1, which is an abstraction of $\langle \wp(\mathbb{Z}), \subseteq \rangle$ [8]. If ρ_s denotes the closure on $\langle \wp(\mathbb{Z}), \subseteq \rangle$ corresponding to *Sign*, i.e. $\mathit{Sign} \cong \rho_s(\wp(\mathbb{Z}))$, as noted by [22], it is easy to check that ρ_s is fully complete for the multiplication $* : \wp(\mathbb{Z})^2 \to \wp(\mathbb{Z})$ given by $X * Y = \{n \cdot m \mid n \in X, m \in Y\}$. Moreover, *Sign* (i.e. ρ_s) is not fully complete for integer addition $\oplus : \wp(\mathbb{Z})^2 \to \wp(\mathbb{Z})$: For instance, $\rho_s(\rho_s(\{-3, -1\} \oplus \rho_s(\{4, 7\})) = \rho_s(\mathbb{Z}) = \mathbb{Z}$, whereas $\rho_s(\{-3, -1\} \oplus \{4, 7\}) = \rho_s(\{1, 3, 4, 6\}) = 0+$. Also, consider the unary monotone function f that selects, e.g., even numbers, i.e. $f = \lambda X. X \cap \mathbb{Z}_{even}$. While *Sign* is not fully complete for f (e.g., $\rho_s(f(\{-1, 2\})) = 0+ \neq \mathbb{Z} = \rho_s(f(\rho_s(\{-1, 2\}))))$, it is instead complete for f: $\rho_s(\mathit{lfp}(f)) = \rho_s(\emptyset) = \emptyset = \mathit{lfp}(\rho_s \circ f)$. Consider now the lattice $\mathit{uco}(\mathit{Sign})$ in Fig. 1 of all possible abstractions of *Sign*, and the monotone unary square operation $sq = \lambda X. X * X$. It is a routine task to check that the sets of complete and fully complete abstractions of *Sign* for sq are as follows:
(i) $\Delta^{\uparrow \mathit{Sign}}(\wp(\mathbb{Z}), sq) = \mathit{uco}(\mathit{Sign}) \setminus \{\rho_5\}$: In fact, $\rho_5(\mathit{lfp}(sq)) = \rho_5(\emptyset) = -0$, whilst $\mathit{lfp}(\rho_5 \circ sq) = \mathbb{Z}$, and this holds for ρ_5 only;



$\rho_2 = \{\mathbb{Z}, 0+\}$
$\rho_3 = \{\mathbb{Z}, 0\}$
$\rho_4 = \{\mathbb{Z}, \emptyset\}$
$\rho_5 = \{\mathbb{Z}, -0\}$
$\rho_6 = \{\mathbb{Z}, 0+, \emptyset\}$
$\rho_7 = \{\mathbb{Z}, 0+, 0\}$
$\rho_8 = \{\mathbb{Z}, 0, \emptyset\}$
$\rho_9 = \{\mathbb{Z}, -0, 0\}$
$\rho_{10} = \{\mathbb{Z}, -0, \emptyset\}$
$\rho_{11} = \{\mathbb{Z}, 0+, 0, \emptyset\}$
$\rho_{12} = \{\mathbb{Z}, -0, 0+, 0\}$
$\rho_{13} = \{\mathbb{Z}, -0, 0, \emptyset\}$

Fig. 1. The lattices *Sign* and $uco(Sign)$.

(ii) $\Gamma^{\uparrow Sign}(\wp(\mathbb{Z}), sq) = uco(Sign) \setminus \{\rho_5, \rho_{10}\}$: For ρ_5 and ρ_{10}, just consider $X = \{0\}$. Observe that $\Delta(\wp(\mathbb{Z}), sq)$ is not a complete sublattice of $uco(\wp(\mathbb{Z}))$: In fact, $\rho_9, \rho_{10} \in \Delta(\wp(\mathbb{Z}), sq)$, whereas $\rho_9 \sqcup \rho_{10} = \rho_5 \notin \Delta(\wp(\mathbb{Z}), sq)$. Similarly, it is not difficult to check that $\Gamma^{\uparrow Sign}(\wp(\mathbb{Z}), *) = uco(Sign) \setminus \{\rho_2, \rho_5, \rho_6, \rho_{10}\}$. □

The following result summarizes some helpful basic properties of the set of complete and fully complete abstract domains.

Proposition 4.3 *Let* $f : C^n \to C$, $g : C \to C$, *and* $h : C \xrightarrow{m} C$.

(i) $\lambda x. \top_C, \lambda x. x \in \Gamma(C, f) \cap \Delta(C, h)$.
(ii) *For all* $c \in C$, $\Gamma(C, \lambda \vec{x}. c) = uco(C) = \Delta(C, \lambda x. c)$.
(iii) $\Gamma(C, \lambda \vec{x}. \vee_{i=1}^{n} \vec{x}_i) = uco(C)$.
(iv) $\rho \in \Gamma(C, \lambda \vec{x}. \wedge_{i=1}^{n} \vec{x}_i) \Leftrightarrow \forall X \subseteq C.\ |X| < \omega \Rightarrow \rho(\wedge X) = \wedge \rho(X)$.
(v) *If* $\rho \in \Gamma(C, \{f, g\})$ *then* $\rho \in \Gamma(C, g \circ f)$.
(vi) *If* $\rho, \eta \in \Gamma(C, f)$ *and* $\rho \circ \eta = \eta \circ \rho$ *then* $\rho \circ \eta \in \Gamma(C, f)$.
(vii) *For all* $i \in [1, n]$, $\Gamma(C, f) = \Gamma(C, \{\lambda \vec{x} \in C^{n-1}. f(\langle x_1, ..., x_{i-1}, c, x_i, ..., x_{n-1} \rangle)\}_{c \in C})$.

For $F \subseteq C^n \xrightarrow{m} C$ and $G \subseteq C \xrightarrow{m} C$, we now consider both $\Gamma(C, F)$ and $\Delta(C, G)$ equipped with the pointwise partial order $\sqsubseteq$ of relative precision of domains, inherited from the lattice of abstract interpretations $uco(C)$. Our first finding is that $\Gamma(C, F)$ and $\Delta(C, G)$ are always (the sets of fixpoints of) lower closures on $uco(C)$, i.e. complete meet subsemilattices of $uco(C)$. The fact that full completeness for monotone unary functions is preserved by *glb* was already observed in [6, Proposition 5.2.3].

Theorem 4.4 *If* $F \subseteq C^n \xrightarrow{m} C$, $G \subseteq C \xrightarrow{m} C$ *then* $\Gamma(C, F), \Delta(C, G) \in lco(uco(C))$.

As far as the *lub* is concerned, in Example 4.2 we observed that, in general, a subset of complete closures $\Delta(C, f)$ is not closed under *lub*'s.

Example 4.5 Let us consider the finite chain of five points $C = \{0 < 1 < 2 < 3 < 4\}$ and the function $f : C \to C$ defined as $f = \{0 \mapsto 0, 1 \mapsto 0, 2 \mapsto 0, 3 \mapsto 4, 4 \mapsto 4\}$. Note that f is monotone and hence it is both additive and co-additive, while $lfp(f) = 0$. Next, consider the closures $\rho_1, \rho_2 \in uco(C)$ given by $\rho_1 = \{1, 3, 4\}$ and $\rho_2 = \{2, 3, 4\}$. It is not difficult to verify that $\rho_1, \rho_2 \in \Delta(C, f)$: $\rho_k(lfp(f)) = k = lfp(\rho_k \circ f)$. It turns out that $\rho_1 \sqcup \rho_2 = \{3, 4\}$ does not belong to $\Delta(C, f)$. In fact, $(\rho_1 \sqcup \rho_2)(lfp(f)) = 3$, while $(\rho_1 \sqcup \rho_2) \circ f = \{0 \mapsto 3, 1 \mapsto 3, 2 \mapsto 3, 3 \mapsto 4, 4 \mapsto 5\}$, and hence $lfp((\rho_1 \sqcup \rho_2) \circ f) = 4$. □

In general, i.e. with no hypothesis on C and f, also $\Gamma(C,f)$ is not closed under *lub*'s, as the following example shows.

Example 4.6 Let the $\omega+1$ ordinal be C, and consider the upper closure $f \in \mathit{uco}(C)$ defined by $f(C) = \{x \mid x < \omega\} \cup \{\omega+1\}$ (thus, f is the identity on $C \setminus \{\omega\}$ whereas maps ω to the top $\omega+1$). Next, consider the closures $\rho_1, \rho_2 \in \mathit{uco}(C)$ defined by $\rho_1 = \{x < \omega \mid x \text{ is even}\} \cup \{\omega, \omega+1\}$ and $\rho_2 = \{x < \omega \mid x \text{ is odd}\} \cup \{\omega, \omega+1\}$. It is immediate to verify that $\rho_i \circ f = f \circ \rho_i$ $(i = 1, 2)$, and therefore, by Proposition 4.3 (vii), $\rho_1, \rho_2 \in \Gamma(C,f)$. Moreover, for the *lub* $\rho_1 \sqcup \rho_2 = \{\omega, \omega+1\}$, we have that $(\rho_1 \sqcup \rho_2) \circ f = \rho_1 \sqcup \rho_2$ whilst $(\rho_1 \sqcup \rho_2) \circ f \circ (\rho_1 \sqcup \rho_2) = \lambda x.\, \omega+1$, and therefore $\rho_1 \sqcup \rho_2 \notin \Gamma(C,f)$. □

Note that, in Example 4.6, f lacks of the continuity property. Indeed, the following key result shows that for continuous functions, *lub*'s of fully complete closures are still fully complete.[1] We already stated this fact, for continuous unary functions, in [18].

Theorem 4.7 *If* $F \subseteq C^n \xrightarrow{c} C$ *then* $\Gamma(C,F) \in \mathit{uco}(\mathit{uco}(C))$.

Continuity is a well-known and sufficiently weak hypothesis, which makes the above result widely applicable in programming language semantics and analysis.

Corollary 4.8 *If* $F \subseteq C^n \xrightarrow{c} C$ *then* $\Gamma(C,F)$ *is a complete sublattice of* $\mathit{uco}(C)$. *Moreover, if* $\mathit{uco}(C)$ *is pseudocomplemented then* $\Gamma(C,F)$ *is pseudocomplemented.*

In general, $\Gamma(C,F)$ is not a sub-pseudocomplemented lattice of $\mathit{uco}(C)$: In Example 4.2, the pseudocomplement of ρ_7 in $\mathit{uco}(\mathit{Sign})$ is ρ_{10} while it is ρ_{13} in $\Gamma(\wp(\mathbb{Z}), \mathit{sq})$.

5 Completeness by Domain Transformers

The notion of *abstract domain refinement* has been studied in [14], and more recently in [18], as a formalization and generalization for most of the systematic operators enhancing the precision of abstract domains (e.g. reduced product and disjunctive completion). For a fixed concrete domain C, a (unary) abstract domain refinement is defined as a mapping $\Re : \mathit{uco}(C) \to \mathit{uco}(C)$, such that $\Re$ is monotone, and, for any $A \in \mathit{uco}(C)$, $\Re(A)$ is more precise than A (i.e., $\Re$ is reductive: $\Re(A) \sqsubseteq A$). Moreover, most of the times, refinements are idempotent, i.e. they upgrade domains all at once. This last condition evidently defines *idempotent refinements as lower closure operators* on $\mathit{uco}(C)$, i.e. mappings in $\mathit{lco}(\mathit{uco}(C))$. A dual theory holds for abstract domain *simplifications*, that are defined as upper closures in $\mathit{uco}(\mathit{uco}(C))$ (cf. [18]). Following this framework and exploiting the results of Section 4, we introduce the *complete* and *fully complete kernel* simplifications, for a set F of monotone functions, and the *least fully complete extension* refinement, for a set F of continuous functions. The (fully) complete kernel of an abstract domain A is defined as the most concrete domain abstracting A which is (fully) complete for any $f \in F$, while the least fully complete extension of A is the most abstract domain which is more precise than A and fully complete for any $f \in F$; by the negative observations of Section 4, a similar least complete extension refinement is not in general definable.

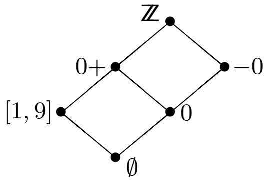
Complete Kernel Operators. Let us introduce the following basic definitions.

Definition 5.1 For $F \subseteq C^n \to C$, $G \subseteq C \xrightarrow{m} C$, define $\mathcal{K}^C_F, \mathbb{K}^C_G : \mathit{uco}(C) \to \mathit{uco}(C)$ such that $\mathcal{K}^C_F(X) = \sqcap\{Y \in \mathit{uco}(C) \mid X \sqsubseteq Y,\ Y \in \Gamma(C,F)\}$ and $\mathbb{K}^C_G(X) = \sqcap\{Y \in \mathit{uco}(C) \mid X \sqsubseteq Y,\ Y \in \Delta(C,G)\}$. □

[1] Amato and Levi [1] have independently made an observation similar to Example 4.5, and stated an analogous but weaker result to Theorem 4.7 for additive functions.

When $F = \{f\}$, we write $\mathcal{K}_f^C$ and $\mathbb{K}_f^C$. By definition, $\mathcal{K}_F^C, \mathbb{K}_G^C \in \mathit{uco}(\mathit{uco}(C))$, i.e. both $\mathcal{K}_F^C$ and $\mathbb{K}_G^C$ are idempotent abstract domain simplifications (cf. [18]). We say that $\mathcal{K}_F^C$ ($\mathbb{K}_G^C$) is *safely defined* when for any $X \in \mathit{uco}(C)$, $\mathcal{K}_F^C(X) \in \Gamma(C, F)$ ($\mathbb{K}_G^C(X) \in \Delta(C, G)$). In this case, we call $\mathcal{K}_F^C$ and $\mathbb{K}_G^C$, respectively, the *fully complete kernel* operator for F (in C) and the *complete kernel* operator for G (in C). By Theorem 4.4, $\mathbb{K}_G^C$ is always safely defined, while monotonicity on F is enough for $\mathcal{K}_F^C$.

Example 5.2 The domain $\{\mathbb{Z}, -0, \emptyset\}$ in Example 4.2 is not fully complete for *sq* and $\{\mathbb{Z}, 0-\}$ is not even complete for *sq*. By Theorem 4.4, there exist their (fully) complete kernels, and from the diagram of $\mathit{uco}(\mathit{Sign})$ in Fig. 1 we derive that $\mathcal{K}_{sq}^{\wp(\mathbb{Z})}(\{\mathbb{Z}, -0, \emptyset\}) = \{\mathbb{Z}, \emptyset\}$ and $\mathbb{K}_{sq}^{\wp(\mathbb{Z})}(\{\mathbb{Z}, -0\}) = \{\mathbb{Z}\}$. Let us also consider the following abstract domain Sign^+ depicted below, introduced by Mycroft in [22, Sect. 3.1]. With respect to *Sign* of Example 4.2, Sign^+ comprises a new element denoting the integer interval $[1, 9]$. ρ_s^+ denotes the closure associated with Sign^+ (i.e. such that $\rho_s^+(\wp(\mathbb{Z})) \cong \mathit{Sign}^+$). It is not difficult to check that Sign^+ is fully complete for $\lambda X. \{n\} * X$ iff $n \not\in [2, 9]$: In fact, for any $n \in [2, 9]$, there exists $X \subseteq [1, 9] \subset \mathbb{Z}$ such that $\rho_s^+(\{n\} * X) \subset \rho_s^+(\{n\} * \rho_s^+(X))$ (if $n = 2$ then $X = [1, 2]$, and if $n = 9$ then $X = [1, 1]$). Note that only by removing $[1, 9]$ from Sign^+ we get a fully complete domain for all $n \in \mathbb{Z}$, viz. *Sign*. This observation implies that *Sign* is the fully complete kernel of Sign^+ for $\lambda X. \{n\} * X$ when $n \in [2, 9]$. Obviously, by the above considerations, Sign^+ is not fully complete for $*$, and indeed $\mathcal{K}_*^{\wp(\mathbb{Z})}(\mathit{Sign}^+) = \mathit{Sign}$. □



The Least Fully Complete Extension Operator. Dually to what done above, we give the following definition.

Definition 5.3 For $F \subseteq C^n \to C$, $G \subseteq C \xrightarrow{\mathrm{m}} C$, define $\mathcal{E}_F^C, \mathbb{E}_G^C : \mathit{uco}(C) \to \mathit{uco}(C)$ as $\mathcal{E}_F^C(X) = \sqcup\{Y \in \mathit{uco}(C) \mid Y \sqsubseteq X,\ Y \in \Gamma(C, F)\}$ and $\mathbb{E}_G^C(X) = \sqcup\{Y \in \mathit{uco}(C) \mid Y \sqsubseteq X,\ Y \in \Delta(C, G)\}$. □

As before, when $F = \{f\}$ we write $\mathcal{E}_f^C$ and $\mathbb{E}_f^C$. By definition, both $\mathcal{E}_F^C$ and $\mathbb{E}_G^C$ are abstract domain refinements in the sense of [14], i.e. $\mathcal{E}_F^C, \mathbb{E}_G^C \in \mathit{uco}(\mathit{uco}(C))$. Similarly to what done above, we define $\mathcal{E}_F^C$ and $\mathbb{E}_G^C$ to be *safely defined* when their ranges are subsets of $\Gamma(C, F)$ and $\Delta(C, G)$, respectively. In this case, $\mathcal{E}_F^C$ and $\mathbb{E}_G^C$ are called, respectively, the *least fully complete extension* operator (for F) and the *least complete extension* operator (for G).

Recall that, in Examples 4.2 and 4.5, we have shown that even when the concrete domain is either an atomic complete Boolean algebra $\wp(X)$, or a finite chain and the concrete function is both additive and co-additive, a *lub* of complete closures is not necessarily still complete. Hence, this key observation precludes us the possibility of finding some reasonable conditions on C and/or G in order that $\mathbb{E}_G^C$ is safely defined. For instance, from the diagram of $\mathit{uco}(\mathit{Sign})$ in Fig. 1 we derive that the least complete extension for *sq* of $\{\mathbb{Z}, -0\}$ does not exist: In fact, $\{\mathbb{Z}, -0\}$ just admits $\{\mathbb{Z}, -0, 0\}$ and $\{\mathbb{Z}, -0.\emptyset\}$ as minimal complete extensions, and therefore, wrongly, we would get $\mathbb{E}_{sq}^{\wp(\mathbb{Z})}(\{\mathbb{Z}, -0\}) = \{\mathbb{Z}, -0\}$. By contrast, when F consists of continuous functions, by Theorem 4.7, for any $X \in \mathit{uco}(C)$, $\mathcal{E}_F^C(X)$ is the least fully complete extension of X, i.e. in this case $\mathcal{E}_F^C$ is safely defined.

Example 5.4 Since *sq* in Example 4.2 is obviously continuous, by Theorem 4.7, every abstract domain of $\wp(\mathbb{Z})$ admits the least fully complete extension for *sq*. For instance, $\mathcal{E}_{sq}^{\wp(\mathbb{Z})}(\{\mathbb{Z}, -0\}) = \{\mathbb{Z}, -0, 0\}$ and $\mathcal{E}_*^{\wp(\mathbb{Z})}(\{\mathbb{Z}, 0+, \emptyset\}) = \{\mathbb{Z}, 0+, 0, \emptyset\}$. □

Algebraic Properties. We study now the basic algebraic properties of the above operators $\mathcal{K}$, $\mathcal{E}$, $\mathbb{K}$, and $\mathbb{E}$ with respect to the operations of reduced product and least common abstraction of domains. These results are particularly important in order to simplify the construction of complete domains.

Lemma 5.5 *Let $F \subseteq C^n \to C$ and $G \subseteq C \xrightarrow{\mathrm{m}} C$. Then, $(\mathcal{K}_F^C, uco(C), uco(C), \mathcal{E}_F^C)$ and $(\mathbb{K}_G^C, uco(C), uco(C), \mathbb{E}_G^C)$ are Galois connections.*

The following relevant algebraic properties are consequences of the above result.

Proposition 5.6 *Let $F \subseteq C^n \to C$, $G \subseteq C \xrightarrow{\mathrm{m}} C$, $X, Y \in uco(C)$ such that $X \sqsubseteq Y$.*
(i) *$\mathcal{K}_F^C$ and $\mathbb{K}_G^C$ are additive, and $\mathcal{E}_F^C$ and $\mathbb{E}_G^C$ are co-additive.*
(ii) *If C is meet-continuous then $\mathcal{E}_F^C(X) = \mathcal{E}_F^C(X \sim Y) \sqcap \mathcal{E}_F^C(Y)$ and $\mathbb{E}_G^C(X) = \mathbb{E}_G^C(X \sim Y) \sqcap \mathbb{E}_G^C(Y)$.*

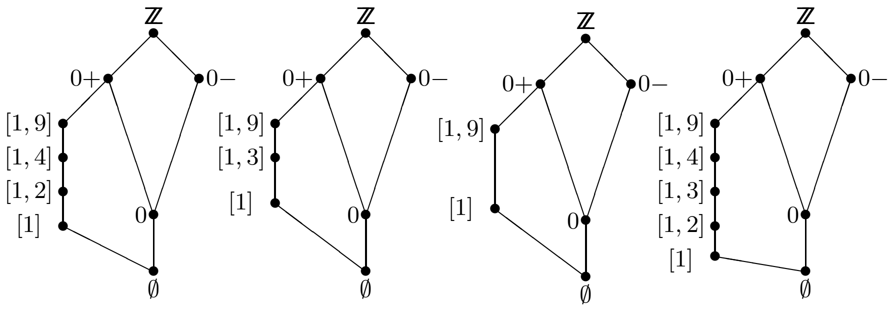
By point (i), the least (fully) complete extension of a domain A can always be obtained by computing separately the least (fully) complete extensions of the factors in a decomposition of A. This decomposition can be obtained by complementation. For instance, by (ii), if C is meet-continuous then $\mathcal{E}_F^C(X) = \mathcal{E}_F^C(X \sim \mathcal{K}_F^C(X)) \sqcap \mathcal{K}_F^C(X)$.

The concrete domain C is not always the best place where completeness and full completeness for a semantic operator can be verified, as it might be far too concrete for this task. We show that one can always refine or simplify a domain A abstracting C by our operators, with the same result, on any (fully) complete abstraction D of C which is more concrete than A. We know that, for $A \in uco(C)$ and $f : C^n \to C$ (when needed, we assume $n = 1$), the best correct approximation of f for A is $f^A : A^n \to A$ defined as $f^A = \rho_A \circ f$, where ρ_A is the closure associated with A, and, on the right hand side, f is thought of being restricted to A^n. If F is a set of functions, then we denote by F^A the corresponding set of best correct approximations for A. The idea is that we can reason about completeness issues relatively to the concrete domain A and function f^A, instead of, resp., C and f. Note that when f is monotone then f^A is monotone as well. Thus, for $F \subseteq C^n \to C$ and $G \subseteq C \xrightarrow{\mathrm{m}} C$, and for any $A \in uco(C)$, we can define the operators of Definitions 5.1 and 5.3 relatively to $\Gamma(A, F^A)$ and $\Delta(A, G^A)$, denoted by $\mathcal{K}_F^A$, $\mathbb{K}_G^A$, $\mathcal{E}_F^A$, and $\mathbb{E}_G^A$, where the superscript is intended to mean that the best correct approximations are considered. By Theorem 4.4, $\mathbb{K}_G^A$ is always a complete kernel operator, and, for $F \subseteq C^n \xrightarrow{\mathrm{m}} C$, $\mathcal{K}_G^A$ is a fully complete kernel operator. Moreover, it is easy to prove that if $f : C^n \xrightarrow{\mathrm{c}} C$ and $A \in \Gamma(C, f)$, then $f^A : A^n \xrightarrow{\mathrm{c}} A$. Thus, if $F \subseteq C^n \xrightarrow{\mathrm{c}} C$ and $A \in \Gamma(C, f)$ then also $\mathcal{E}_F^A$ is a safely defined least fully complete extension operator.

Theorem 5.7 *Let $F \subseteq C^n \to C$, $G \subseteq C \xrightarrow{\mathrm{m}} C$, and $A \in uco(C)$.*
(i) $A \in \Gamma(C, F) \Leftrightarrow \forall X \in {\uparrow}A.\ \mathcal{K}_F^C(X) = \mathcal{K}_F^A(X) \Leftrightarrow \forall X \in {\uparrow}A.\ \mathcal{E}_F^C(X) = \mathcal{E}_F^A(X)$.
(ii) $A \in \Delta(C, G) \Leftrightarrow \forall X \in {\uparrow}A.\ \mathbb{K}_G^C(X) = \mathbb{K}_G^A(X) \Leftrightarrow \forall X \in {\uparrow}A.\ \mathbb{E}_G^C(X) = \mathbb{E}_G^A(X)$.

The following example shows how to practically exploit the properties in Proposition 5.6 and Theorem 5.7 to design the least fully complete extension of Mycroft's domain $Sign^+$ for integer multiplication.

Example 5.8 Consider the domain $Sign^+$ in Example 5.2. We constructively define the least fully complete extension of $Sign^+$ for integer multiplication. As we observed in Example 5.2, $Sign^+$ is fully complete for $f_n = \lambda X.\{n\} * X$ (which is obviously continuous) iff $n \notin [2, 9]$. Let $n \in [2, 9]$. As we have shown in Example 5.2, $\mathcal{K}_{f_n}^{\wp(\mathbb{Z})}(Sign^+) =$



$\mathcal{E}^{Int}_{f_2}(Sign^+)$ $\mathcal{E}^{Int}_{f_3}(Sign^+)$ $\{\mathcal{E}^{Int}_{fn}(Sign^+)\}_{n\in[5,9]}$ $\mathcal{E}^{\wp(\mathbb{Z})}_{*}(Sign^+)$

Fig. 2. Least fully complete extensions of $Sign^+$.

$Sign$ and $Sign^+ \sim Sign = \{\mathbb{Z}, [1, 9]\}$. By Proposition 5.6 (ii), we get $\mathcal{E}^{\wp(\mathbb{Z})}_{fn}(Sign^+) = \mathcal{E}^{\wp(\mathbb{Z})}_{fn}(Sign^+ \sim \mathcal{K}^{\wp(\mathbb{Z})}_{fn}(Sign^+)) \sqcap \mathcal{E}^{\wp(\mathbb{Z})}_{fn}(\mathcal{K}^{\wp(\mathbb{Z})}_{fn}(Sign^+)) = \mathcal{E}^{\wp(\mathbb{Z})}_{fn}(\{\mathbb{Z}, [1, 9]\}) \sqcap Sign$. We compute the least fully complete extension $\mathcal{E}^{\wp(\mathbb{Z})}_{fn}(\{\mathbb{Z}, [1, 9]\})$ of the atomic domain $\{\mathbb{Z}, [1, 9]\}$ by a recursive computation. Consider the Cousot and Cousot [8] domain of integer intervals $Int \in uco(\wp(\mathbb{Z}))$ defined by

$$Int = \{[a, b]\}_{a\leq b} \cup \{(-\infty, b]\}_{b\in\mathbb{Z}} \cup \{[a, +\infty)\}_{a\in\mathbb{Z}} \cup \{(-\infty, +\infty)\} \cup \{\bot\}.$$

It is easy to prove that $Int \in \Gamma(\wp(\mathbb{Z}), *)$ and hence, for any $n \in \mathbb{Z}$, $Int \in \Gamma(\wp(\mathbb{Z}), f_n)$. Thus, by Theorem 5.7, for any $n \in \mathbb{Z}$ and A such that $Int \sqsubseteq A$, $\mathcal{E}^{\wp(\mathbb{Z})}_{fn}(A) = \mathcal{E}^{Int}_{fn}(A)$, and therefore we can consider the more abstract (and simple) domain Int as concrete domain of reference. The following proposition characterizes the least fully complete extension in Int of any atomic domain of the form $\{\mathbb{Z}, [1, m]\}$. We assume that $m \geq 1$ and $n \in [2, m]$. An analogous result holds for $m \leq -1$, $\{\mathbb{Z}, [m, 1]\}$ and $n \in [m, 2]$.

Proposition 5.9 $\mathcal{E}^{Int}_{fn}(\{\mathbb{Z}, [1, m]\}) = \{\mathbb{Z}, [1, m]\} \sqcap \mathcal{E}^{Int}_{fn}(\{\mathbb{Z}, [1, \lfloor \frac{m}{n} \rfloor]\})$.

The above recursive definition has a solution which can be obtained finitely for any $n \leq m$. For $Sign^+$, i.e. $m = 9$, it provides the domains depicted in Fig. 2. In particular, if $F = \{f_n \mid n \in \mathbb{Z}\}$ then $\mathcal{E}^{\wp(\mathbb{Z})}_{*}(Sign^+) = \mathcal{E}^{\wp(\mathbb{Z})}_{F}(Sign^+) = \mathcal{E}^{Int}_{\{f_2,f_3\}}(Sign^+)$. □

6 Systematic Construction of Complete Domains

In Example 5.8 we have shown how to constructively design the least fully complete extension for integer multiplication of a simple domain for sign analysis. In this section, we show that under some hypotheses on the semantic functions, a similar methodology can be always defined and applied to finite abstract domains in order to construct their least fully complete extensions and kernels.

Constructing Least Fully Complete Extensions. The following result gives a useful characterization of full completeness for additive functions.

Theorem 6.1 *Let* $f : C \xrightarrow{a} C$. *Then, for all* $X \in uco(C)$, $X \in \Gamma(C, f) \Leftrightarrow \forall x \in X. \vee \{y \in C \mid f(y) \leq x\} \in X$.

Thus, when f is additive, it turns out that $\mathcal{E}^{C}_{f}(X)$ is the least meet-closed set Y containing X such that $x \in Y \Rightarrow \vee\{y \mid f(y) \leq x\} \in Y$. This domain can be constructively obtained as the greatest (viz. most abstract) solution in $uco(C)$, smaller or equal to X, of the recursive abstract domain equation $Y = Y \sqcap \mathcal{F}_f(Y)$, where

$\mathcal{F}_f = \lambda X.\mathcal{M}(\{\vee\{y \in C \mid f(y) \leq x\} \mid x \in X\})$. Note that $\mathcal{F}_f : \mathit{uco}(C) \to \mathit{uco}(C)$ is monotone, and therefore $\mathcal{E}_f^C(X)$ is the limit of the (possibly transfinite) iteration sequence $Y^0 = X$, $Y^{\alpha+1} = Y^\alpha \sqcap \mathcal{F}_f(Y^\alpha)$ if α is a successor ordinal, and $Y^\alpha = \sqcap_{\delta<\alpha} Y^\delta$ if α is a limit ordinal. Clearly, if C is finite, then this constructive method is terminating. It is not difficult to prove that if f is in addition extensive and $X = \{\top, x\}$, with $x \in C \setminus \{\top\}$, is an atomic domain, then each element Y^α of the above iteration sequence having $\mathcal{E}_f^C(\{\top, x\})$ as limit, is a chain. In this case, termination for an atomic domain $\{\top, x\}$ is ensured by assuming the weaker hypothesis that $\langle \downarrow x, \leq \rangle$ is a well-ordered sublattice (i.e. it does not contain infinite descending chains) of the (possibly infinite) concrete domain C. An analogous argument led us to the constructive characterization of Proposition 5.9. The interest in least fully complete extensions of atomic domains is justified by observing that, because $\mathit{uco}(C)$ is always dual-atomic, for any abstract domain there exists a canonical decomposition involving atomic domains only: If $X \in \mathit{uco}(C)$ then $\langle \{\top, x\} \rangle_{x \in X \setminus \{\top\}}$ is such a decomposition of X. Then, by Proposition 5.6 (i), for any $X \in \mathit{uco}(C)$ and $f : C \xrightarrow{c} C$, $\mathcal{E}_f^C(X) = \sqcap_{x \in X \setminus \{\top\}} \mathcal{E}_f^C(\{\top, x\})$. We also observe that the extension of the above constructive methods to a finite set of functions F is straightforward. In fact, if $\mathit{Sol}(f, X)$ denotes the solution of the above equation for $f : C \xrightarrow{a} C$ and $X \in \mathit{uco}(C)$, then, independently from the order on which the function Ref, on the left, chooses f in F by select, for any finite set F of additive functions, $\mathit{Ref}(F, X) = \mathcal{E}_F^C(X)$.

```
fun $Ref(F, X)$ = if $F = \emptyset$ then $X$
                  else select $f$ in $F$;
                       $Ref(F, Sol(f, X))$.
```

Constructing Fully Complete Kernels. The systematic construction of fully complete kernels relies on the above procedure for least fully complete extensions. As before, we consider atomic decompositions of the input domain, and the following result characterizes the fully complete kernel of a domain X in terms of the least fully complete extensions of the factors in the atomic decomposition of X.

Theorem 6.2 *If $f : C \xrightarrow{c} C$ then $\mathcal{K}_f^C(X) = \sqcap\{\mathcal{E}_f^C(\{\top, x\}) \mid x \in X, \mathcal{E}_f^C(\{\top, x\}) \subseteq X\}$.*

The above result hints a systematic constructive method to design fully complete of abstract domains. The correctness of the algorithm on the left follows by Theorem 6.2. The function *choose* selects and removes an arbitrary element from its input domain. Clearly, when f is additive, the least fully complete extension of the atomic domains can be obtained by the constructive method above.

```
Input : Finite domain $X$
Output : Fully complete kernel $K$ of $X$
$K := \{\top\}$;
$A := X$;
while $A \neq \{\top\}$ do
   $x := choose(A \setminus \{\top\})$;
   if $\mathcal{E}_f^C(\{\top, x\}) \subseteq X$ then
            $K := K \sqcap \mathcal{E}_f^C(\{\top, x\})$
endwhile
```

7 Reconstructing the Integer Interval Domain

We observed in Example 4.2 that *Sign* is not fully complete for integer addition $\oplus$. Instead, it is an easy task to prove that the Cousot and Cousot integer interval domain *Int* of Example 5.8 is fully complete for addition, i.e. $\mathit{Int} \in \Gamma(\wp(\mathbb{Z}), \oplus)$. We can go beyond this expected result. Since $\oplus$ is additive, and therefore continuous, the least complete extension of *Sign* with respect to $\oplus$ does exist, and the following theorem actually proves that it coincides with *Int*.

Theorem 7.1 $\mathcal{E}_\oplus^{\wp(\mathbb{Z})}(\mathit{Sign}) = \mathit{Int}$.

Therefore, the domain *Int* for integer interval analysis can be reconstructed by a systematic domain refinement from the rule of signs domain *Sign*.

8 Intelligently Refining Domains: The Case of Groundness Analysis

We have seen that for a fixed concrete least fixpoint semantics $\mathcal{S} = \langle C, f \rangle$, any abstraction $D \in uco(C)$ and its corresponding best correct approximation $f^D : D \to D$ automatically induce a correct abstract semantics $\mathcal{I}^D = \langle D, f^D \rangle$. Clearly, being abstract or concrete is a relative notion, and therefore if $A \in uco(C)$ is a domain abstracting D, viz. $D \sqsubseteq A$, we can reason on the completeness relationships between A and D, where $\mathcal{I}^D$ plays the role of the relative concrete semantics for A. In such a scenario, we simply say that A is complete, resp. fully complete, for D when $A \in \Delta(D, f^D)$, resp. $A \in \Gamma(D, f^D)$. In particular, this situation arises whenever $\Re : uco(C) \to uco(C)$ is some (possibly nonidempotent) domain refinement, and therefore, for some abstract domain A, $C \sqsubseteq \Re(A) \sqsubseteq A$. In this case, we take into consideration the possible completeness relations of A wrt the relative concrete semantics $\langle \Re(A), f^{\Re(A)} \rangle$. Following the notation at the end of Section 5, we denote by $\mathcal{E}_f^{\Re(A)}$ the corresponding completeness transformer. This is a safely defined least fully complete extension operator under the weak hypothesis that the best correct approximation $f^{\Re(A)} : \Re(A) \to \Re(A)$ is continuous. In general, we have that $\Re(A) \sqsubseteq \mathcal{E}_f^{\Re(A)}(A) \sqsubseteq A$. We say that $\Re(A)$ is *too refined* when $\Re(A) \sqsubset \mathcal{E}_f^{\Re(A)}(A)$ holds. The intuition behind this definition is as follows. When, in the above scenario, it happens that A is fully complete for D, then we can reasonably consider the abstract interpretation $\mathcal{I}^A = \langle A, f^A \rangle$ as just thinly less precise than $\mathcal{I}^D$. In other terms, the gap of precision between the abstract interpretations $\mathcal{I}^A$ and $\mathcal{I}^D$ is very narrow, and therefore an efficiency/precision trade-off would suggest to prefer $\mathcal{I}^A$ to the more costly $\mathcal{I}^D$. In particular, whenever $\Re(A) \sqsubset \mathcal{E}_f^{\Re(A)}(A)$, one should prefer the thinly less refined domain $\mathcal{E}_f^{\Re(A)}(A)$ rather than the canonical refinement $\Re(A)$. Thus, under the weak hypothesis that each best correct approximation $f^{\Re(A)}$ is continuous, an intelligent "efficiency-oriented" version $\Re^\star$ of a refinement $\Re$ can therefore be defined as $\Re^\star = \lambda A. \mathcal{E}_f^{\Re(A)}(A)$. In the following, we illustrate a practical example of this idea in the field of ground-dependency analysis for logic programs.

The Intelligent Disjunctive Completion of ***Def*** **is** ***Pos*****.** *Def* and *Pos* are two well-known finite abstract domains of propositional formulae widely used for ground-dependency analysis of logic programs [20]. We refer e.g. to [2] for the details of their definitions. These abstract domains can be viewed as abstractions of the standard concrete domain $\langle \wp(Atom), \subseteq \rangle$ used in a collecting bottom-up semantics for logic programs (as usual, the set of atoms $Atom$ is considered up to renaming). It turns out that $\wp(Atom) \sqsubset Pos \sqsubset Def$. For instance, a pair $\langle p(x,y,z), x \wedge (y \leftrightarrow z) \rangle \in Def \subset Pos$ represents each atom $p(t_1, t_2, t_3)$ such that for any its instance $p(s_1, s_2, s_3)$: (i) s_1 is ground; (ii) $var(s_2) = var(s_3)$. In particular, $p(a,b,c)$ and $p(a,x,g(x))$ satisfy this property (where, as usual, $a, b, c, \ldots$ denote ground terms).

The *disjunctive completion* [9] is a well known abstract domain refinement that enhances an abstract domain so that it becomes disjunctive, i.e. so that the corresponding concretization map is additive. When the concrete domain C is completely distributive, as any powerset $\langle \wp(X), \subseteq \rangle$ is, the disjunctive completion $\mathbb{P}(A)$ of an abstract domain A can be obtained by quotienting the powerset of A for the equivalence relation $\{\langle S, T \rangle \mid S, T \subseteq \wp(A),\ \vee_C \gamma(S) = \vee_C \gamma(T)\}$ (see [15] for more details). The concretization of an equivalence class $[S]$ of the disjunctive completion $\mathbb{P}(A)$ is obviously defined as $\vee_C \gamma(S)$, while the abstraction of $[S]$ in A is given by $\vee_A S$. It has been shown in [15, Theorem 5.3] that *Pos* is not disjunctive, i.e. $\mathbb{P}(Pos) \sqsubset Pos$, while [19, Theorem 7.2] proved that $\mathbb{P}(Def) = \mathbb{P}(Pos)$. Thus, it turns out that $\mathbb{P}(Def) = \mathbb{P}(Pos) \sqsubset Pos \sqsubset Def$. For example, by considering the following abstract predicates $\langle p(x,y), x \rangle$ and $\langle p(x,y), x \to y \rangle$, one has $\gamma_{Def}(\langle p(x,y), x \rangle) \cup \gamma_{Def}(\langle p(x,y), x \to y \rangle) \subset$

$\gamma_{Def}(\langle p(x,y), x \vee (x \to y)\rangle) = \gamma_{Def}(\langle p(x,y), \mathit{true}\rangle)$ (it is enough to consider the atom $p(v,w)$), and this shows that *Def* and *Pos* are not disjunctive.

Following the standard bottom-up approach (e.g. [3]), we consider the well-known s-semantics $\mathcal{S} = \langle \wp(\mathit{Atom}), T_P^s\rangle$ of [13] as the concrete least fixpoint semantics of reference, which is fully abstract for the observable given by the computed answer substitutions of a logic program. Then, the abstract domains *Def*, *Pos* and their common disjunctive completion $\mathbb{P}(\mathit{Def})$ induce the abstract semantics $\mathit{Def}^s = \langle \mathit{Def}, T_P^{\mathit{Def}}\rangle$, $\mathit{Pos}^s = \langle \mathit{Pos}, T_P^{\mathit{Pos}}\rangle$, $\mathbb{P}(\mathit{Def})^s = \langle \mathbb{P}(\mathit{Def}), T_P^{\mathbb{P}(\mathit{Def})}\rangle$, where each abstract semantic operator is defined as the best correct approximation of T_P^s on the corresponding abstract domain. Let us turn to discuss the completeness issues between these abstract semantics. It is known [20] that *Def* is not complete (and therefore fully complete) for *Pos* and $\mathbb{P}(\mathit{Def})$, i.e. $\mathit{Def} \notin \Gamma(\mathit{Pos}, \{T_P^{\mathit{Pos}}\}_{P \in \mathit{Program}}) \cup \Gamma(\mathbb{P}(\mathit{Def}), \{T_P^{\mathbb{P}(\mathit{Def})}\}_{P \in \mathit{Program}})$, while [15, Proposition 5.13] proved that *Pos* is fully complete (and hence complete) for $\mathbb{P}(\mathit{Def})$, i.e. $\mathit{Pos} \in \Gamma(\mathbb{P}(\mathit{Def}), \{T_P^{\mathbb{P}(\mathit{Def})}\}_{P \in \mathit{Program}})$.

Example 8.1 Consider the programs P and Q below.

$$\begin{array}{lll@{\qquad\qquad}lll}
P: & p(x,y) :- & q(x,y), r(x,y). & Q: & p(x,a) :- & p(x,z), p(y,x).\\
 & q(a,x) :- & & & p(x,x) :- & \\
 & q(x,a) :- & & & p(a,y) :- & \\
 & r(x,x) :- & & & &
\end{array}$$

It is easy to see that $\langle p(x,y), x \leftrightarrow y\rangle \in \mathit{lfp}(T_P^{\mathit{Def}})$, whilst $\langle p(x,y), x \wedge y\rangle \in \mathit{lfp}(T_P^{\mathit{Pos}}) = \mathit{lfp}(T_P^{\mathbb{P}(\mathit{Def})})$. Thus, in P, by either *Pos* or $\mathbb{P}(\mathit{Def})$, we are able to infer that any computed answer substitution for the predicate p will ground both its arguments, while in *Def* we can only conclude that the first argument is ground iff the second is ground. Moreover, the Kleene iterations for T_Q^{Pos} and $T_Q^{\mathbb{P}(\mathit{Def})}$ are as follows: $(T_Q^{\mathit{Pos}})^0 = \langle p(x,y), \mathit{false}\rangle$, $(T_Q^{\mathit{Pos}})^1 = \langle p(x,y), y \to x\rangle$, $(T_Q^{\mathit{Pos}})^2 = \langle p(x,y), \mathit{true}\rangle$ (least fixpoint); $(T_Q^{\mathbb{P}(\mathit{Def})})^0 = \langle p(x,y), \mathit{false}\rangle$, $(T_Q^{\mathbb{P}(\mathit{Def})})^1 = \langle p(x,y), [x \leftrightarrow y, x]\rangle$, $(T_Q^{\mathbb{P}(\mathit{Def})})^2 = \langle p(x,y), [x \leftrightarrow y, x, y]\rangle$ (least fixpoint). Thus, in Q, by $\mathbb{P}(\mathit{Def})$ we are able to infer that in each computer answer substitution for p, either its first argument is ground or its second argument is ground or they are equivalent. Instead, by using *Pos* we get no ground-dependency information. Nevertheless, by abstracting in *Pos* the final output $\langle p(x,y), [x \leftrightarrow y, x, y]\rangle$ of $\mathbb{P}(\mathit{Def})$ we get exactly the output $\langle p(x,y), \mathit{true}\rangle$ of *Pos*. □

We sharp the aforementioned result of [15, Proposition 5.13], and we show that *Pos* is the least complete and fully complete extension of *Def* in $\mathbb{P}(\mathit{Def})$, i.e., by letting $F = \{T_P^{\mathbb{P}(\mathit{Pos})}\}_{P \in \mathit{Program}}$, the following theorem holds.

Theorem 8.2 $\mathcal{E}_F^{\mathbb{P}(\mathit{Def})}(\mathit{Def}) = \mathbb{E}_F^{\mathbb{P}(\mathit{Def})}(\mathit{Def}) = \mathit{Pos}$.

To conclude, let us draw the practical consequences of Theorem 8.2, in view of the general observations made at the beginning of this section. If we want to systematically design an abstract domain for disjunctive ground-dependency analysis starting from the existing domain *Def*, then, according to the standard procedure of refining *Def* to its disjunctive completion, we should use the disjunctive domain $\mathbb{P}(\mathit{Def})$. However, according to our definitions, $\mathbb{P}(\mathit{Def})$ is too refined, since, by Theorem 8.2, $\mathcal{E}^{\mathbb{P}(\mathit{Def})}(\mathit{Def}) = \mathit{Pos}$ is a proper abstraction of $\mathbb{P}(\mathit{Def})$. By contrast, following the "efficiency-oriented" strategy of refinement outlined above, the intelligent disjunctive completion is defined as $P^\star = \lambda A.\mathcal{E}_F^{\mathbb{P}(A)}(A)$, and therefore we get $P^\star(\mathit{Def}) = \mathit{Pos}$. This still is a systematic step of refinement, and by doing this, it is important to remark that we dramatically gain in efficiency: While $\mathbb{P}(\mathit{Def})$ has an exponential size wrt Def, i.e. $|\mathbb{P}(\mathit{Def})| = O(2^{|\mathit{Def}|})$, by contrast, it is easy to verify that $|\mathit{Pos}| = O(|\mathit{Def}|)$ yet maintaining a relationship of full completeness between Pos and $\mathbb{P}(\mathit{Def})$.

9 Related Work

The notion of completeness in abstract interpretation has been first considered by Cousot and Cousot in [9], where the basic properties of complete abstractions in the approximation of fixpoint-based semantics have been proved. Completeness arises typically among (concrete) semantics of programming languages at different levels of abstraction. Cousot and Cousot proved in [10] that some classical semantics of programming languages are (fully) complete abstractions of a generalized SOS operational semantics, and Cousot [7] formalized the relationships between some well known type inference systems as complete (called exact) abstract interpretations. In logic program semantics, Giacobazzi [16] proved that fully complete abstractions relating semantics provide a corresponding hierarchy of models for positive logic programs, while Comini and Levi [5] based their taxonomy of observables (perfect, denotational, and semi-denotational) on the notion of full completeness, applied to some basic operators for building SLD-trees. Recently, Amato and Levi [1] studied the lattice structure of these observables, which is basically the lattice of full complete abstractions of the above additive operators for building SLD-trees, and independently formulated an analogous but weaker result to our Theorem 4.7, for additive operators only. In program analysis, Sekar et al. [24] focussed on completeness of Mycroft's [21] strictness analysis of functional programs. Their approach is rather different from ours: They identified the greatest class Cl of programs such that the strictness analysis of P is complete iff $P \in \mathit{Cl}$. In our terminology, they have found the greatest set of programs Cl such that $\mathit{Strictness} \in \Delta(C, \{T_P\}_{P \in \mathit{Cl}})$, where C and T_P are, respectively, the concrete domain and the semantic transformer in the standard collecting denotational semantics of P. Reddy and Kamin [23] generalized [24] to first-order and typed higher-order functional languages. Steffen [25] is one of the first authors isolating completeness as a key property for analysis. His approach is "observation directed", in the sense that the design of a fully abstract (complete) abstract interpretation is directed by a certain observation level. This differs from our approach, which is, as the standard Cousot and Cousot theory of abstract interpretation, "semantics directed". A categorical generalization of Steffen's work has been successively studied in [26]. Colby [4] isolated the phenomenon of accumulated imprecision in abstract interpretation, which is essentially the same as the lack of full completeness. To overcome these problems, Colby proposes to consider a new enhanced so-called transfer relations language to express the net behavior of finite control paths in the operational semantics of programming languages, and he shows that this allows to solve the problem in some relevant examples. Colby's approach is therefore orthogonal to ours: The whole semantic metalanguage is changed to gain precision. Finally, the most related work is certainly that of Mycroft [22]. He considers a notion of completeness which is essentially equal to ours, except that he develops his theory using a predicate-based approach to abstract interpretation (a kind of logical view of classical abstract interpretation). Moreover, he argues that the well-known state minimization algorithm for finite deterministic automata can be used to produce a canonical (fully) complete abstract interpretation by removing useless domain elements. Although the technical approach followed by Mycroft is quite different from ours, the idea of systematically defining a canonical simplest complete abstract interpretation is basically the same idea of our complete kernel operators. This relationship certainly deserves further investigation in order to see if these two systematic methodologies actually have the same behavior. On the other hand, Mycroft does not consider the dual problem analogous to our least (fully) complete extension operators.

Acknowledgments. We are grateful to Francesca Scozzari for her contribution to Theorem 4.7.

References

1. G. Amato and G. Levi. Properties of the lattice of observables in logic programming. In *Proc. APPIA-GULP-PRODE'97*, pp. 175–187, 1997.
2. T. Armstrong, K. Marriott, P. Schachte, and H. Søndergaard. Two classes of Boolean functions for dependency analysis. To appear in *Sci. Comput. Program.*, 1997.
3. R. Barbuti, R. Giacobazzi, and G. Levi. A general framework for semantics-based bottom-up abstract interpretation of logic programs. *ACM TOPLAS*, 15(1):133–181, 1993.
4. C. Colby. Accumulated imprecision in abstract interpretation. In *Proc. 1st ACM AAS'97*, pp. 77–89, 1997.
5. M. Comini and G. Levi. An algebraic theory of observables. In *Proc. ILPS'94*, pp. 172–186, 1994.
6. A. Cortesi, G. Filé, R. Giacobazzi, C. Palamidessi, and F. Ranzato. Complementation in abstract interpretation. *ACM TOPLAS*, 19(1):7–47, 1997.
7. P. Cousot. Types as abstract interpretations. In *Proc. ACM POPL'97*, pp. 316–331, 1997.
8. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. ACM POPL'77*, pp. 238–252, 1977.
9. P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Proc. ACM POPL'79*, pp. 269–282, 1979.
10. P. Cousot and R. Cousot. Inductive definitions, semantics and abstract interpretation. In *Proc. ACM POPL'92*, pp. 83–94, 1992.
11. P. Cousot and R. Cousot. Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and PER analysis of functional languages). In *Proc. IEEE ICCL'94*, pp. 95–112, 1994.
12. P. Cousot and R. Cousot. Compositional and inductive semantic definitions in fixpoint, equational, constraint, closure-condition, rule-based and game-theoretic form. In *Proc. CAV'95*, LNCS 939, pp. 293–308, 1995.
13. M. Falaschi, G. Levi, M. Martelli, and C. Palamidessi. Declarative modeling of the operational behavior of logic languages. *Theor. Comput. Sci.*, 69(3):289–318, 1989.
14. G. Filé, R. Giacobazzi, and F. Ranzato. A unifying view of abstract domain design. *ACM Comput. Surv.*, 28(2):333–336, 1996.
15. G. Filé and F. Ranzato. The powerset operator on abstract interpretations. To appear in *Theor. Comput. Sci.*, 1998.
16. R. Giacobazzi. "Optimal" collecting semantics for analysis in a hierarchy of logic program semantics. In *Proc. STACS '96*, LNCS 1046, pp. 503–514, 1996.
17. R. Giacobazzi, C. Palamidessi, and F. Ranzato. Weak relative pseudo-complements of closure operators. *Algebra Universalis*, 36(3):405–412, 1996.
18. R. Giacobazzi and F. Ranzato. Refining and compressing abstract domains. In *Proc. ICALP'97*, LNCS 1256, pp. 771–781, 1997.
19. R. Giacobazzi and F. Ranzato. Optimal domains for disjunctive abstract interpretation. To appear in *Sci. Comput. Program.*, 1998.
20. K. Marriott and H. Søndergaard. Precise and efficient groundness analysis for logic programs. *ACM Lett. Program. Lang. Syst.*, 2(1-4):181–196, 1993.
21. A. Mycroft. *Abstract interpretation and optimising transformations for applicative programs.* PhD thesis, Univ. of Edinburgh, 1981.
22. A. Mycroft. Completeness and predicate-based abstract interpretation. In *Proc. ACM PEPM '93*, pp. 179–185, 1993.
23. U.S. Reddy and S.N. Kamin. On the power of abstract interpretation. In *Proc. IEEE ICCL '92*, 1992.
24. R.C. Sekar, P. Mishra, and I.V. Ramakrishnan. On the power and limitation of strictness analysis. *J. ACM*, 44(3), 1997.
25. B. Steffen. Optimal data flow analysis via observational equivalence. In *Proc. MFCS '89*, LNCS 379, 1989.
26. B. Steffen, C.B. Jay, and M. Mendler. Compositional characterization of observable program properties. *AFCET*, 26:403–424, 1992.
27. M. Ward. The closure operators of a lattice. *Ann. Math.*, 43(2):191–196, 1942.