Università di Pisa

Dipartimento di Informatica

Technical Report : TR-97-22

Making abstract interpretations complete

Giacobazzi Roberto, Ranzato Francesco

November 03, 1997

ADDR: Corso Italia 40,56125 Pisa,Italy. TEL: +39-50-887111. FAX: +39-50-887226

Making Abstract Interpretations Complete

Roberto Giacobazzi
Dipartimento di Informatica
Università di Pisa
Corso Italia 40, 56125 Pisa, Italy
giaco@di.unipi.it

Francesco Ranzato
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 situation where the abstract semantics is able to take full advantage of the power of representation of the underlying abstract domain. Thus, complete abstract interpretations can be rightfully considered as optimal. In this article, we develop a general theory of completeness in abstract interpretation, also dealing with the most frequent case of least fixpoint semantics. We show that both completeness and least fixpoint completeness are properties that only depend on the underlying abstract domain. In this context, we demonstrate that there always exist both the greatest complete and least fixpoint complete restrictions of any abstract domain, and for continuous semantic functions, the least complete extension exists as well. Also, under certain hypotheses, a constructive characterization of the least complete extensions and restrictions of abstract domains is given as solutions of simple abstract domain equations. These methodologies provide advanced algebraic tools for manipulating and comparing abstract interpretations, which can be fruitfully used both in program analysis and in semantics design. A number of examples illustrating these techniques are given in the fields of integer variable and logic program analysis.

Keywords: Abstract interpretation, program analysis, completeness, full completeness, complete kernel, least complete extension, integer interval analysis, logic program analysis.

1 Introduction

According to a recent and widely recognized definition: *"Abstract interpretation is a general theory for approximating the semantics of discrete dynamic systems"* [Cousot 1996a]. Abstract interpretation was originally developed by Cousot and Cousot [1977] as a unifying framework for specifying and then validating static program analyses, and in recent years has increasingly gained popularity as a general methodology for describing and formalizing approximate computations in many different areas of computer science, like for instance in theorem proving [Plaisted 1981], model checking [Clarke et al. 1994; Dams et al. 1997; Loiseaux et al. 1995], verification of distributed memory systems [Graf 1997], type inference [Cousot 1997b], constraint solving [Caseau 1991], query optimization [Helm et al. 1995], and comparative semantics [Comini and Levi 1994; Cousot and Cousot 1992c; Cousot 1997a; Giacobazzi 1996]. The success of abstract interpretation stems from its simple, but nevertheless rigorously defined, underlying idea that the specification of the behavior of a system, e.g. a program, at different levels of abstraction, is an approximation of its formal semantics. We consider here the classical case of programming languages and their semantics approximation, although the discussion that follows is also relevant in all the above fields, and in every area where abstract interpretation is applicable.

In the classical Cousot and Cousot [1977, 1979b] framework, an abstract interpretation is defined as a nonstandard approximated program semantics obtained from the standard (also called concrete) one by substituting the actual (concrete) domain of computation and its basic (concrete) semantic operations with, respectively, an *abstract domain* and corresponding *abstract semantic operations*. Let us consider the typical case of a concrete semantics specified by a concrete domain

C and a semantic function $[\![\cdot]\!] : \mathit{Program} \to C$, associating with each program $P \in \mathit{Program}$ its semantics $[\![P]\!]$. Then, an abstract interpretation can be formulated by defining a corresponding abstract domain A and an abstract semantic function $[\![\cdot]\!]^\sharp : \mathit{Program} \to A$ which approximates the concrete one. The notion of approximation is encoded in abstract interpretation by a suitable partial order on domain's objects. Both C and A are assumed to be complete lattices with respect to this approximation order. The orderings on the concrete and abstract domains describe the relative precision of domain values, somehow in a dual fashion with respect to the standard domains for denotational semantics – the top element representing no information. The relationship between the concrete and abstract setting is given by a pair of adjoint functions, respectively the abstraction map $\alpha : C \to A$ and the concretization map $\gamma : A \to C$, where $\alpha(c) \leq_A a$, or, equivalently, $c \leq_C \gamma(a)$, means that a is a correct approximation of c, and therefore $\alpha(c)$ plays the role of the best possible approximation of c in A. In this context, a typical *soundness* result for an abstract interpretation goes as follows:

$$\textit{For all programs } P,\ \alpha([\![P]\!])) \leq_A [\![P]\!]^\sharp.$$

When, as in most cases, the semantics is specified by the least fixpoint of some concrete monotone operator $T_P : C \to C$, indexed on the set of programs, i.e. $[\![P]\!] = \mathit{lfp}(T_P)$, the above soundness statement may result quite hard to prove, due to its fixpoint nature. One basic idea of abstract interpretation is to approximate the concrete least fixpoint semantics by the least fixpoint of an abstract monotone semantic operator $T_P^\sharp : A \to A$. Luckily, in this common case, there is a well-known property which is stronger than soundness but nevertheless much easier to check [Cousot and Cousot 1979b]: *For all programs* P, $\alpha \circ T_P \leq_A T_P^\sharp \circ \alpha$. In this case, $T_P^\sharp$ is called a correct approximation of T_P. Most abstract interpretations are designed and proved sound in this way, like for instance, the classical strictness analysis of functional programs and groundness analysis of logic programs (see e.g. [Cousot and Cousot 1993; Marriott et al. 1994; Mycroft 1981]).

Completeness in Abstract Interpretation. In the 90's, there have been a number of works studying the dual issue of *completeness* in abstract interpretation (see Section 8 for a discussion on completeness in abstract interpretation literature). Maintaining the above standard scenario, a typical completeness theorem is as follows:

$$\textit{For all programs } P,\ \alpha([\![P]\!]) = [\![P]\!]^\sharp.$$

Already Cousot and Cousot in their pioneering work considered such a situation (cf. [Cousot and Cousot 1979b, Theorems 7.1.0.3 and 7.1.0.4]). As above, whenever $[\![P]\!] = \mathit{lfp}(T_P)$, this least fixpoint completeness is implied by the following stronger completeness property for semantic operators: *For all programs* P, $\alpha \circ T_P = T_P^\sharp \circ \alpha$. In this latter case, we use the term *full completeness*. This terminology and the relation between completeness and full completeness is justified by the fact that in the following we will be mainly concerned with least fixpoint presentations of programming language semantics[1].

While soundness is *the* basic requirement for any abstract interpretation, completeness is instead an ideal and more rare situation. Completeness arises when no additional loss of precision is accumulated by computing the least fixpoint abstract semantics as induced by the abstract semantic operator $T_P^\sharp$ over A, and therefore the abstract semantics $\mathit{lfp}(T_P^\sharp)$ is the best possible approximation in A of the concrete semantics $\mathit{lfp}(T_P)$. Roughly speaking, this means that the abstract semantics is able to take full advantage of the power of representation of the abstract domain A. Further, full completeness means in addition that in *every step* of the Kleene iteration sequence for $\mathit{lfp}(T_P^\sharp)$ over A, no loss of precision occurs (i.e., for every ordinal δ, $\alpha(T_P^\delta(\bot_C)) = (T_P^\sharp)^\delta(\bot_A)$), which is clearly a stronger condition. The classical naïve

[1]There is not presently an agreement on a name for what we call completeness and full completeness. For instance: Cousot [1997b, Section 12], Cousot [1997a, Section 12], and Clarke et al. [1994, Section 4.3] call *exactness* our notion of (nonfull) completeness; Steffen [1989, Section 3.1] uses instead the term *full abstraction*; Comini and Levi [1994] use the term *precision*; while Dams et al. [1997, Section 4] use the term *optimality* for the same notion. We follow Cousot and Cousot [1994], Mycroft [1993], Reddy and Kamin [1992], and Sekar et al. [1997] by using the term *completeness*, which is more typically used in contrast to the well established notion of *soundness*.

"rule of signs" example [Cousot and Cousot 1977;1979b] helps in clarifying this concept. Let us represent the sign of sets of integers in $\wp(\mathbb{Z})$ by an abstract domain like $S = \{?, \geq_0, \leq_0, 0, \perp\}$ (where the meaning of the objects in S is the most obvious one), and consider an obvious abstract operation $*^\sharp$ of multiplication over S (e.g., $\geq_0 *^\sharp \leq_0 = \leq_0$ and $\geq_0 *^\sharp 0 = 0$) which correctly approximates the concrete operation $X * Y = \{x \cdot y \mid x \in X,\ y \in Y\}$. Then, it is evident that $*^\sharp$ is fully complete for the concrete multiplication operation $*$. This means that for every pair of sets of integers $Z_1, Z_2 \in \wp(\mathbb{Z})$, $\alpha_S(Z_1 * Z_2) = \alpha_S(Z_1) *^\sharp \alpha_S(Z_2)$. For example, $\alpha_S(\{-1\} * \{-2\}) = \geq_0 = \leq_0 *^\sharp \leq_0 = \alpha_S(\{-1\}) *^\sharp \alpha_S(\{-2\})$, i.e. the sign of any integer multiplication can be exactly obtained by the above rule of signs, with no loss of precision. By contrast, this does not hold for addition: The obvious abstract addition $+^\sharp$ over S is not fully complete, for instance $\alpha_S(\{-1\} + \{3\}) = \geq_0 \neq \leq_0 +^\sharp \geq_0 = ?$. Moreover, let us also consider the monotone function $f : \wp(\mathbb{Z}) \to \wp(\mathbb{Z})$ defined by $f(X) = \{0\} \cup \{x + 2 \mid x \in X\}$ and the abstract function $f^\sharp : S \to S$ defined as $\{? \mapsto ?,\ \geq_0 \mapsto \geq_0,\ \leq_0 \mapsto ?,\ 0 \mapsto \geq_0,\ \perp \mapsto 0\}$. It is not too difficult to observe that $f^\sharp$ is fully complete for f, and therefore completeness holds as well, i.e. $\alpha_S(\mathit{lfp}(f)) = \mathit{lfp}(f^\sharp)$: In fact, the least fixpoints of f and $f^\sharp$ are $\mathit{lfp}(f) = \{x \in \mathbb{Z} \mid x \geq 0,\ x \text{ even}\}$ and $\mathit{lfp}(f^\sharp) = \geq_0$. Instead, by considering $g^\sharp$ defined by $\{? \mapsto ?,\ \geq_0 \mapsto ?,\ \leq_0 \mapsto ?,\ 0 \mapsto \geq_0,\ \perp \mapsto 0\}$, $g^\sharp$ still is a correct approximation of f, and therefore soundness holds, but $g^\sharp$ is neither fully complete nor complete for f. In fact, it turns out that $\alpha_S(f(\{0,1\})) = \geq_0 \neq ? = g^\sharp(\alpha_S(\{0,1\}))$ and $\alpha_S(\mathit{lfp}(f)) = \geq_0 \neq ? = \mathit{lfp}(g^\sharp)$.

Completeness is typically recurrent in comparative semantics, i.e. in the study of semantics of programming languages at different levels of abstraction. This is probably due to the fact that for many pairs of concrete semantics (e.g. trace-based operational and denotational), their corresponding domains are both sufficiently rich and hence, intuitively, quite close, so that the relationship of completeness holds; instead, whenever one of the two domains is substantially more simple, like in a program analysis, completeness holds in much more rare cases. For instance, Cousot and Cousot [1992c] and Cousot [1997a] consider some classical semantics of programming languages, like denotational, predicate transformer and axiomatic, as complete abstractions of a generalized SOS trace-based operational program semantics (see Section 8 for other examples of complete abstract interpretations). Although in practical program analysis systems completeness arises in more rare cases, complete abstract interpretations still represent an absolute upper bound on the precision of the analysis, and therefore they can be rightfully considered as a ground optimal basis to design hierarchies of semantics approximations, offering the possibility to tune the analysis in accuracy and costs. This is the case for instance in abstract interpretation and analysis of reactive systems [Dams et al. 1997] and in abstract model checking [Clarke et al. 1994], where complete abstractions play a key role, providing the systematic derivation of optimal abstract (simplified) systems which simulate the original concrete one and satisfy as many as possible of its properties. For these reasons, it is widely recognized that completeness plays a central role in abstract interpretation, both in theory and in its practical applications. The problem of achieving completeness for an abstract interpretation by enhancing either its abstract semantic operators or its abstract domain, has been investigated by a number of authors (see Section 8 for more details). While some solutions have been successfully achieved for some specific abstract interpretations and analyses, in most cases by exploiting ad hoc techniques, the more general and ambitious 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. This article proposes a solution to this problem.

Completeness as a Domain Property. The following important point requires particular attention. It is well known since [Cousot and Cousot 1979b] that any abstract domain A defined in terms of a pair of adjoint functions α and γ as before, induces the *best correct approximation* $T_P^{best_A}$ of any concrete semantic operator T_P, which is defined as: $T_P^{best_A} = \alpha \circ T_P \circ \gamma$. This corresponds to perform an abstraction of the concrete semantic operator T_P computed on the objects approximated by each abstract value. For instance, in the above rule of signs, the best correct approximation of an integer multiplication $-1 * -2$ can be obtained by looking for the common sign in S of the concrete multiplications $n * m$ with n and m being nonpositive numbers, which is $\leq_0 \ *^\sharp \ \leq_0 = \geq_0$. Consequently, a domain A always induces an (automatically) sound abstract semantics $\lambda P.\, \mathit{lfp}(T_P^{best_A})$. By contrast, in general, not all abstract domains admit a complete abstract semantics. However, if A admits a complete (respectively, fully complete) abstract semantic operator $T_P^\sharp$ then the best correct approximation $T_P^{best_A}$ is complete (fully complete) as well; otherwise, $T_P^{best_A}$ would not be "the best". Hence, the following important characterization holds:

It is possible to define a (fully) complete abstract semantic operator on an abstract domain A if and only if $T_P^{best_A}$ is (fully) complete.

This means that completeness and full completeness are properties which depend only on the underlying abstract domain, i.e. on the pair α and γ.

Contributions of this Work. In this article, we develop an algebraic theory of domain completeness in the classical abstract interpretation framework. As the above observations suggest, we attack this problem from a domain perspective, by considering completeness as an abstract domain property, i.e. A is (fully) complete for T_P iff $T_P^{best_A}$ is (fully) complete. It is known that the collection of all possible abstract domains of a given concrete domain C gives rise to the so-called *lattice of abstract interpretations* of C, here denoted by $\mathcal{L}_C$, where, for any two abstract domains A and B, $A \sqsubseteq B$ holds when A is more precise (i.e. less abstract) than B (roughly, when $B \subseteq A$) [Cousot and Cousot 1979b]. Thus, we concentrate on the set of all the domains in the lattice $\mathcal{L}_C$, which are complete and fully complete for a given family of semantic operators. We denote these sets by $\Delta(C)$ and $\Gamma(C)$ respectively. We first prove that both $\Delta(C)$ and $\Gamma(C)$ are always complete meet subsemilattices of $\mathcal{L}_C$. This has an immediate practical application: Since the *glb* in $\mathcal{L}_C$ corresponds to the widely known reduced product operation for combining domains [Cousot and Cousot 1979b], this means that the reduced product of abstract domains preserves both the properties of completeness and full completeness. As far as the *lub* of $\mathcal{L}_C$ (also known as least common abstraction operation) is concerned, we show that, in general, $\Delta(C)$ is not a join subsemilattice of $\mathcal{L}_C$, even when very restrictive hypotheses on C and on the semantic operators are imposed. By contrast, we prove that, when each concrete semantic operator is *(Scott-)continuous* on C, $\Gamma(C)$ is instead 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.

Exploiting the above results on the lattice structure of the sets of complete and fully complete abstract domains, we define three systematic operators that transform any given abstract domain A of input into a complete or fully complete one, which is as close as possible, with respect to precision, to A. There are two possibilities for doing this: Either by *refining* the domain, i.e. by enhancing its precision by adding new elements, or by *simplifying* it, i.e. by taking out from the domain some information which may cause incompleteness. In fact, we prove that these operators are instances of the generic operators for domain refinement and simplification introduced by Filé et al. [1996] and Giacobazzi and Ranzato [1997]. Again, this can be explained with the rule of signs. It is clear that the simple domain $S^{0+} = \{\geq_0, 0, ?\}$, which is strictly more abstract than S above, i.e. $S^{0+} \subset S$, is not fully complete for integer multiplication: $\alpha_{S^{0+}}(\{-1\} * \{-2\}) = +$, whilst $\alpha_{S^{0+}}(\{-1\}) *^\sharp \alpha_{S^{0+}}(\{-2\}) = \,?\, *^\sharp \,?\, = \,?$. The largest fully complete domain for integer multiplication which is contained in S^{0+} is $S^0 = \{0, ?\}$. In fact, $\alpha_{S^{0+}}(\{n\} * \{m\}) = 0$ iff $n = 0 \ \vee \ m = 0$ iff $\alpha_{S^{0+}}(\{n\}) *^\sharp \alpha_{S^{0+}}(\{m\}) = 0$. Analogously, it is not too hard to observe that S above is the smallest fully complete domain which contains S^{0+}. More in general, we define the *complete* and *fully complete kernel* operators $\mathbb{K}$ and $\mathcal{K}$. These are domain simplifications such that, for any fixed 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, respectively fully complete, for any function in F. On the other hand, we define a dual *least fully complete extension* operator $\mathcal{E}$: Here, for any fixed set of concrete continuous functions F, $\mathcal{E}(A)$ is the most abstract domain which is an extension (i.e. more precise) of A and fully complete for any function in F. By the aforementioned negative findings on the structure of $\Delta(C)$, an analogous least complete extension operator is not generally definable. We prove that 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 independently refining these simpler domains. We present a constructive method for building the least fully complete extension and fully complete kernel of any abstract domain, under the hypothesis that the semantic operators are additive functions. These fully complete domains are obtained as the solution of a simple recursive domain equation, defined on the lattice of abstract interpretations $\mathcal{L}_C$. Under the additional hypothesis that the semantic functions are extensive, the above constructive method can be efficiently implemented by a "divide et impera" algorithm. All the development of our theory is gradually illustrated by simple examples, like the rule of signs for integer variables.

Although the main contribution of this paper is in the general theory of abstract interpretation, we present some applications in the field of static program analysis. We choose program analysis in order to show that our operators for the systematic construction of complete abstract domains may offer the right technology to solve the important problem, observed by Cousot [1996b, Section 3], of designing optimal abstract algebras, offering the best cost-precision compromise, for program analysis. Clearly, our results can also be applied in all the fields where abstract interpretation is used, like for instance in those areas mentioned in the beginning. In this article we consider the following applications:

- In the context of domain design for complete abstract interpretations, we start by considering the simple rule of signs analysis. As we sketched above, the rule of signs domain for integer variable analysis is obviously not fully complete for the addition operation. We characterize the least fully complete extension for the addition of the rule of signs domain: It turns out that this is precisely the well known Cousot and Cousot [1977] domain of integer intervals. This result provides an interesting constructive characterization of the lattice of intervals as the most abstract solution of a simple domain equation involving the domain of signs and integer addition.

- In the context of fixpoint based approximation of program semantics, we show how our least fully complete extension operator can be fruitfully exploited for carefully tuning the efficiency/precision trade-off when improving the precision of an analysis by widening operators or abstract domain refinements. It is often the case that a domain refinement step overflows the necessary information for an analysis, by adding some costly unnecessary information; as well as it may happen that, in order to improve a given analysis, a widening operator is designed on a domain which may be too concrete, again containing redundant information. We apply our operators to reduce the amount of redundancy in these constructions. Clearly, to be an abstract domain is a relative notion. Thus, our systematic operators can be applied to refine or simplify domains for analysis relatively to other more precise – but still approximated – domains. Here, our tools of completeness allows us 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. This is particularly important both for removing costly and redundant information from refined domains and to design optimal (most abstract) domains for building widening operators.

- In the context of abstract domain refinements, we apply the above efficiency-oriented strategy for refining domains to compare the expressive power of some well-known abstract domains for ground-dependency analysis of logic languages, like *Def*, *Pos* and their common disjunctive completion $\mathbb{P}(Def)$ (cf. [Armstrong et al. 1997; Cortesi et al. 1996; Filé and Ranzato 1998; Giacobazzi and Ranzato 1998; Marriott and Søndergaard 1993]). More in detail, considering a standard bottom-up least fixpoint semantics of reference like those of Barbuti et al. [1993], Codish et al. [1994], and Marriott et al. [1994], we prove that the least complete and fully complete extensions of *Def* with respect to the disjunctive completion $\mathbb{P}(Def)$, both coincide with *Pos*. Therefore, an intelligent disjunctive completion refinement can be defined so that the refinement of *Def* turns out to be *Pos* instead of the canonical $\mathbb{P}(Def)$.

The practical usefulness of such an approach can be fully appreciated by considering that $\mathbb{P}(Def)$ is enormously less efficient than Pos (the former has an exponential size with respect to the latter one), whilst, thanks to full completeness, Pos is just thinly less precise than $\mathbb{P}(Pos)$.

Structure of the Article. The rest of the article is organized as follows. Section 2 contains the basic notations and notions on abstract interpretation used throughout the paper. The basic concepts of completeness and full completeness in abstract interpretation are introduced in Section 3, where it is also shown that these are abstract domain properties. The lattices of complete and fully complete abstract domains are studied in Section 4. The operators of (fully) complete kernel and least (fully) complete extension on abstract domains are introduced and studied in Section 5. In Section 6 we provide the constructive characterization of the least fully complete extensions and fully complete kernels of abstract domains. Section 7, structured in three subsections, contains the applications. Finally, Section 8 addresses related literature and Section 9 closes by sketching some further research directions. A preliminary version of this article will appear in the Proc. of the Sixth Internat. Conf. on Algebraic Methodology and Software Technology, to be held in Sydney, Australia, on December 1997.

2 Abstract Interpretation Basics

We present in this section a succinct overview of the basic notions on closure operators and abstract interpretation used throughout the article. For more details about closure operators the reader is referred to [Davey and Priestley 1990; Morgado 1960; Ward 1942], while for abstract interpretation to [Cousot and Cousot 1977; 1979b; 1992a].

Basic Mathematical Notation. If S and T are sets, then $\wp(S)$ denotes the powerset of S, $|S|$ the cardinality of S, $S \setminus T$ denotes the set-difference between S and T, $S \subset T$ denotes strict inclusion, and for a function $f : S \to T$ and $X \subseteq S$, $f(X) = \{f(x) \mid x \in X\}$. By $g \circ f$ we denote the composition of the functions f and g, i.e., $\forall x.\ (g \circ f)(x) = g(f(x))$. If P is a pre-ordered set and $I \subseteq P$ then the downward closure of I is defined as $\downarrow I = \{x \in P \mid \exists y \in I.\ x \leq_P y\}$, and for $x \in P$, $\downarrow x$ is a shorthand for $\downarrow\{x\}$. The upward closure $\uparrow$ is dually defined. If C is a complete lattice then $MI(C)$ denotes the set of (*completely*) *meet-irreducible* elements of C, i.e. $MI(C) = \{x \in C \mid \forall S \subseteq C.\ x = \wedge S \Rightarrow x \in S\}$. If P is a poset with bottom $\perp$ then an element $a \in P$ is an *atom* if a covers $\perp$, i.e., $a \neq \perp$ and for all $x \in P$, $\perp < x \leq a$ implies $x = a$. By $At(P)$ we denote the set of all the atoms of P. A complete lattice C is *atomic* if every element $x \in C$ different from $\perp$ is the *lub* of the atoms which precede x, i.e. $x = \vee\{a \in At(C) \mid a \leq x\}$. *Dual-atoms* and *dual-atomicity* are dually defined. By $C \cong D$ we denote that C and D are isomorphic ordered structures. If C is a complete lattice and $n \in \mathbb{N}$, then C^n denotes the usual direct product, which still is a complete lattice under the canonical componentwise ordering induced from C. If S is any set, P a poset, and $f, g : S \to P$ then we write $f \leq_P g$ if for all $x \in S$, $f(x) \leq_P g(x)$. Let C and D be complete lattices. Then, $C \xrightarrow{\mathrm{m}} D$, $C \xrightarrow{\mathrm{c}} D$, and $C \xrightarrow{\mathrm{a}} D$ denote, respectively, the set of all monotone, (Scott-)continuous and (completely) additive (i.e. preserving all *lub*'s, empty set included) functions from C to D. Recall that $f \in C \xrightarrow{\mathrm{c}} D$ iff f preserves *lub*'s of chains iff f preserves *lub*'s of directed subsets. We denote by $lfp(f)$ the least fixpoint, when it exists, of a function f. If $f : C \to C$ then $f + id : C \to C$ is defined as $f + id = \lambda x. f(x) \vee x$. Ord denotes the proper class of ordinals, where $\omega \in Ord$ is the first limit ordinal. For a complete lattice C and $x \in C$, the ordinal α (upper) power of a function $f : C \to C$ for x is denoted by $f^\alpha(x)$ and it is defined as x if $\alpha = 0$, $f(f^{\alpha-1}(x))$ if α is a successor ordinal, and $\vee_{\delta<\alpha} f^\delta(x)$ if α is a limit ordinal. When $x = \perp_C$, sometimes we simply use f^α. The lower power of f for x is denoted by $f^{\downarrow\alpha}(x)$ and it is defined dually.

Closure Operators and Galois Connections. Abstract interpretation theory can be equivalently formulated either in terms of Galois connections or in terms of closure operators [Cousot and Cousot 1979b]. Let us recall these concepts. An (*upper*) *closure operator*, or simply a *closure*, on a poset $\langle P, \leq \rangle$ is an operator $\rho : P \to P$ monotonic, idempotent and extensive (viz. $\forall x \in P.\ x \leq \rho(x)$). The set of all closure operators on P is denoted by $uco(P)$. Let $\langle C, \leq, \wedge, \vee, \top, \bot \rangle$ be a complete lattice. Then, each closure operator $\rho \in uco(C)$ is uniquely determined by the set of its fixpoints, which is its image $\rho(C)$. A subset $X \subseteq C$ is the set of fixpoints of a closure operator on C iff X is meet-closed, or, equivalently, a *Moore-family* of C, i.e. $X = \{\wedge S \mid S \subseteq X\}$, where $\top = \wedge\emptyset$. In this case, $\rho_X = \wedge\{x \in X \mid y \leq x\}$ is the corresponding closure operator on C. For any $X \subseteq C$, $\mathcal{M}(X) = \{\wedge S \mid S \subseteq X\}$ is the *Moore-closure* of X in C, i.e. the least subset of C containing X which is a Moore-family of C. Thus, $\rho(C)$ is a complete meet subsemilattice of C, but, in general, it is not a complete sublattice of C, since the *lub* in $\rho(C)$ might be different from that in C. It turns out that $\rho(C)$ is a complete sublattice of C iff ρ is additive. Often, we will find particularly convenient to identify closure operators with their sets of fixpoints, using as notation capital Latin letters; instead, when viewing closures as functions, we will use Greek letters to denote them. In the following, we will keep this soft ambiguity by using both notations, and leave to the reader to distinguish their use as functions or sets, according to the context.

If C is a complete lattice then $\langle uco(C), \sqsubseteq, \sqcap, \sqcup, \lambda x.\top, \lambda x.x \rangle$ is a complete lattice, where for every $\rho, \eta \in uco(C)$, $\{\rho_i\}_{i\in I} \subseteq uco(C)$ and $x \in C$:

- $\rho \sqsubseteq \eta$ iff $\forall x \in C.\ \rho(x) \leq \eta(x)$, or, equivalently, $\rho \sqsubseteq \eta$ iff $\eta(C) \subseteq \rho(C)$;
- $(\sqcap_{i\in I}\rho_i)(x) = \wedge_{i\in I}\rho_i(x)$;
- $(\sqcup_{i\in I}\rho_i)(x) = x \Leftrightarrow \forall i \in I.\ \rho_i(x) = x$;
- $\lambda x.\top$ is the top element, whereas $\lambda x.x$ is the bottom element.

For a closure operator $\rho \in uco(C)$ and $Y \subseteq C$, the following two properties hold:

(i) $\rho(\wedge\rho(Y)) = \wedge\rho(Y)$;

(ii) $\rho(\vee Y) = \rho(\vee\rho(Y))$.

If $\rho \in uco(C)$ then for any $x, y \in C$, we write $x \approx_\rho y$ iff $\rho(x) = \rho(y)$. As a consequence of property (ii) above, $\approx_\rho$ is a join-complete congruence relation on C, i.e. for all $X \subseteq C$ and $y \in C$, if $\forall x \in X.\ x \approx_\rho y$ then $\vee X \approx_\rho y$. It is also known that, for any complete lattice C, $uco(C)$ is dual-atomic, where the dual-atoms are all and only those closures $\varphi_x = \{\top, x\}$, for any $x \in C \setminus \{\top\}$, i.e. if $y \leq x$ then $\varphi_x(y) = x$ otherwise $\varphi_x(y) = \top$. We call any φ_x simply atomic closure.

A *lower closure operator* $\psi : P \to P$ on a poset P is monotonic, idempotent and reductive (viz. $\forall x \in P.\ \psi(x) \leq x$). The complete lattice of all lower closure operators on a complete lattice C is denoted by $lco(C)$. All the properties of lower closure operators can be derived by duality from those above for upper closures ($uco(C)$ and $lco(C)$ are dually isomorphic complete lattices).

If C and A are posets, and $\alpha : C \to A$, $\gamma : A \to C$ are monotone maps such that $\forall x \in C.\ x \leq_C \gamma(\alpha(x))$ and $\forall y \in A.\ \alpha(\gamma(y)) \leq_A y$, then the quadruple (α, C, A, γ) is a *Galois connection* (GC for short) between C and A. If in addition $\forall y \in A.\ \alpha(\gamma(y)) = y$, then (α, C, A, γ) is a *Galois insertion* (GI for short) of A in C. In a GI, γ is 1-1 and α is onto. We also recall that the notion of GC is equivalent to that of *adjunction*: If $\alpha : C \to A$ and $\gamma : A \to C$ then (α, C, A, γ) is a GC iff $\forall x \in C. \forall y \in A.\ \alpha(x) \leq_A y \Leftrightarrow x \leq_C \gamma(y)$. The map α (γ) is called the *left-* (*right-*)*adjoint* to γ (α). In any GC (α, C, A, γ) between complete lattices, α is additive and γ is co-additive.

Abstract Domains. As already recalled in the introduction, in the standard abstract interpretation framework, the *concrete* and *abstract domains*, C and A, are required to be complete lattices related by a Galois connection (α, C, A, γ). Following the standard terminology, α and γ are called the *abstraction* and *concretization* maps, respectively, and A is also called an *abstraction* of C. The intuition is that the abstract domain is a representation of some approximate properties of the values of the concrete domain. Both on the concrete and on the abstract domain, a partial order relation describing the relative precision of the values is defined: $x \leq y$ means that x is more precise than y. The concretization map gives the concrete value corresponding to an abstract denotation (i.e. its semantics), whereas for a concrete value the abstraction map gives its best (with respect to the ordering of A) abstract approximation. Thus, an abstract value $y \in A$ approximates a concrete value $x \in C$ if $x \leq_C \gamma(y)$, or equivalently (by adjunction), if $\alpha(x) \leq_A y$. If (α, C, A, γ) is a GI, each value of the abstract domain A is useful in the representation of the concrete domain C, because all the elements of A represent distinct members of C, being α onto. It is known that any GC may be lifted to a GI identifying in an equivalence class those values of the abstract domain with the same concrete meaning. This process is known as *reduction* of the abstract domain.

It is well known since [Cousot and Cousot 1979b] that abstract domains can be equivalently specified either as Galois insertions or as (sets of fixpoints of) upper closures 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 $\rho_A(C) \cong A$. Actually, the "logical meaning" of an abstract domain A in a GI (α, C, A, γ) is exactly captured by the associated closure $\gamma \circ \alpha$ on the concrete domain, since using a different but lattice-theoretically isomorphic domain changes nothing in the meaning of abstract objects. Thus, the closure operator approach is particularly convenient when reasoning about properties of abstract domains independently from the representation of their objects. Hence, we will identify $uco(C)$ with the so-called *lattice of abstract interpretations* $\mathcal{L}_C$ of C (cf. [Cousot and Cousot 1977, Section 7] and [Cousot and Cousot 1979b, Section 8]), i.e. the complete lattice of all possible abstract domains (modulo isomorphic representation of their objects) of the concrete domain C. The ordering on $uco(C)$ corresponds precisely to the standard order used to compare abstract domains with regard to their precision: A_1 is *more precise* than A_2 (or A_2 is an abstraction of A_1) 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. $\sqcup_{i \in I} A_i$ is the *least common abstraction* of all A_i's; (ii) $\sqcap_{i \in I} A_i$ is (isomorphic to) the well-known *reduced product* (basically cartesian product plus reduction) of all the A_i's, or, equivalently, it is the most abstract among the domains (abstracting C) which are more concrete than every A_i. It is helpful to note that the reduced product can be also characterized as the Moore-closure of set-union, i.e. $\sqcap_{i \in I} A_i = \mathcal{M}(\cup_{i \in I} A_i)$.

Let us also recall the notion of abstract domain *complementation*. If L is a meet semilattice with bottom then the *pseudocomplement* of $x \in L$, if it exists, is the (unique) element $x^* \in L$ such that $x \wedge x^* = \bot$ and $\forall y \in L.\ (x \wedge y = \bot) \Rightarrow (y \leq x^*)$. In a complete lattice L, if the pseudocomplement x^* exists then $x^* = \vee\{y \in L \mid x \wedge y = \bot\}$. If every $x \in L$ has the pseudocomplement, L is *pseudocomplemented*. Giacobazzi et al. [1996] proved that whenever C is a meet-continuous complete lattice (i.e., for any chain $Y \subseteq C$ and $x \in C$, $x \wedge (\vee Y) = \vee_{y \in Y}(x \wedge y)$), $uco(C)$ is pseudocomplemented. This result has been applied by Cortesi et al. [1997] in order to define a notion of domain complementation in abstract interpretation. Starting from any two domains C and A, where A abstracts C, i.e. $C \sqsubseteq A$, and C is meet-continuous, the *complement* of A in C is defined as the unique most abstract domain $C \sim A$ such that $(C \sim A) \sqcap A = C$. Note that meet-continuity is a sufficiently weak hypothesis to include most abstract domains used in practice. Complementation is important for *abstract domain decompositions*: 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 their reduced product gives A, i.e. $A = \sqcap_{i \in I} D_i$ [Cortesi et al. 1997]. Thus, by complementation, if $C \sqsubseteq A$ then $\langle C \sim A, A \rangle$ is a binary decomposition for C. The advantage of domain decomposition is twofold: (1) it provides more compact representations for complex domains, enhancing space saving techniques, and (2) it simplifies verification problems for complex domains, by decomposing them into simpler problems for their factors.

3 Completeness in Abstract Interpretation

Let *Program* denotes the set of (syntactically well-formed) programs. The concrete standard or collecting semantics is in general specified by a semantic function $[\![\cdot]\!] : \mathit{Program} \to C$, 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 $[\![\cdot]\!]^\sharp : \mathit{Program} \to A$, 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 a semantic function $[\![\cdot]\!]$ obviously depends on the considered programming language and on the semantics style adopted (e.g. denotational or operational). We follow here a customary *least fixpoint* semantic approach, which is general enough to include most kinds of semantic specifications (see [Cousot and Cousot 1995]). A concrete semantics is therefore specified by a pair $\langle C, T\rangle$, where C is a complete lattice and $T : \mathit{Program} \to (C \xrightarrow{\mathrm{m}} C)$. For $P \in \mathit{Program}$, we use T_P to denote more compactly $T(P)$. The least fixpoint semantics of any program P is defined as $[\![P]\!] = \mathit{lfp}(T_P) \in C$. On the abstract side, for some $T^\sharp : \mathit{Program} \to (A \xrightarrow{\mathrm{m}} A)$, 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 $\mathcal{S}^\sharp = \langle A, T^\sharp\rangle$, where A is implicitly related to C by a GI (α, C, A, γ), we say that $\mathcal{S}^\sharp$ is 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, $\alpha \circ T_P \circ \gamma \leq_C T^\sharp_P$. In fact, a standard well-known result of abstract interpretation (cf. [Cousot and Cousot 1979b, Theorem 7.1.0.2]) ensures that this condition on T implies the soundness for $\mathcal{S}^\sharp$. We then say that $\mathcal{S}^\sharp$ is a *fully sound* abstraction of $\mathcal{S}$ if for all $P \in \mathit{Program}$, $\alpha \circ T_P \leq_A T^\sharp_P \circ \alpha$. Often, we will slightly abuse on terminology, by saying that some abstract semantic function $T^\sharp_P$ is (fully) sound for T_P if $(\alpha \circ T_P \leq_A T^\sharp_P \circ \alpha)$ $\alpha(\mathit{lfp}(T_P)) \leq_A \mathit{lfp}(T^\sharp_P)$. Clearly, soundness is in general a strictly weaker property than full soundness (for instance, $\lambda x. \alpha(\mathit{lfp}(T_P))$ is, in general, sound for T_P but not fully sound).

In abstract interpretation, the term completeness is used dually to the above notion of soundness [Mycroft 1993]. Again, one distinguishes between a weaker form of completeness, involving least fixpoints only, and a stronger one (but much easier to verify) involving semantic functions. Then, we say that $\mathcal{S}^\sharp$ is a (*fully*) *complete* abstraction of $\mathcal{S}$ if for all $P \in \mathit{Program}$, $(T^\sharp_P \circ \alpha \leq_A \alpha \circ T_P)$ $\mathit{lfp}(T^\sharp_P) \leq_A \alpha(\mathit{lfp}(T_P))$. A standard abstract interpretation result (cf. [Cousot and Cousot 1979b, Theorems 7.1.0.4]) guarantees that full completeness implies completeness, while Example 4.4 below shows that the converse does not hold.

Because soundness is always required in abstract interpretation, in the following we slightly abuse terminology and say that $\mathcal{S}^\sharp$ is (fully) complete for $\mathcal{S}$ if for all $P \in \mathit{Program}$, $(\alpha \circ T_P = T^\sharp_P \circ \alpha)$ $\alpha(\mathit{lfp}(T_P)) = \mathit{lfp}(T^\sharp_P)$. Moreover, as above for soundness, we also use such notions of completeness and full completeness locally for some abstract semantic function $T^\sharp_P$.

Completeness as a Domain Property. We have seen that an abstract operator $T^\sharp_P : A \to A$ is a correct approximation of some $T_P : C \to C$ if $\alpha \circ T_P \circ \gamma \leq_C T^\sharp_P$. It is then clear (and well-known since [Cousot and Cousot 1979b, Corollary 7.2.04]) that the abstract domain A induces a *best* correct approximation of T_P given by $T_P^{\mathit{best}_A} = \alpha \circ T_P \circ \gamma$. Consequently, A always induces an (automatically) fully sound abstract semantics $\langle A, \lambda P.\, T_P^{\mathit{best}_A}\rangle$. By contrast, of course, 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 note that the best correct approximation $T_P^{\mathit{best}_A}$ of T_P relative to A is fully complete as well, since it indeed coincides with $T^\sharp_P$: In fact, if $T^\sharp_P$ is fully complete then $T^\sharp_P = T^\sharp_P \circ \alpha \circ \gamma = \alpha \circ T_P \circ \gamma = T_P^{\mathit{best}_A}$. Likewise, if $T^\sharp_P$ is complete then $T_P^{\mathit{best}_A}$ is complete as well: $\alpha(\mathit{lfp}(T_P)) \leq_A \mathit{lfp}(T_P^{\mathit{best}_A}) \leq_A \mathit{lfp}(T^\sharp_P) = \alpha(\mathit{lfp}(T_P))$. Thus, we get the following important characterization:

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.

In other terms, completeness and full completeness are *abstract domain properties*. Therefore, in the following we say that an abstract domain A is (fully) complete for T_P if $T_P^{best_A}$ is (fully) complete for T_P.

4 The Lattice of Complete Abstract Interpretations

We have seen in the previous section that one can consider, without loss of generality, completeness and full completeness for best correct approximations only. The following result gives an alternative useful characterization for these notions.

Proposition 4.1 *Let (α, C, A, γ) be a GI and $f : C \xrightarrow{\text{m}} C$.*

(i) *f^{best_A} is fully complete iff $(\gamma \circ \alpha) \circ f = (\gamma \circ \alpha) \circ f \circ (\gamma \circ \alpha)$.*

(ii) *f^{best_A} is complete iff $\gamma(\alpha(\mathit{lfp}(f))) = \mathit{lfp}(\gamma \circ \alpha \circ f)$.*

Proof. (i) Since γ is 1-1, f^{best_A} is fully complete iff $\alpha \circ f = (\alpha \circ f \circ \gamma) \circ \alpha$ iff $(\gamma \circ \alpha) \circ f = (\gamma \circ \alpha) \circ f \circ (\gamma \circ \alpha)$.
(ii) ($\Rightarrow$) Since $\gamma \circ \alpha \in \mathit{uco}(C)$, $f \leq_C \gamma \circ \alpha \circ f$. Hence, $\gamma(\alpha(\mathit{lfp}(f))) \leq_C \gamma(\alpha(\mathit{lfp}(\gamma \circ \alpha \circ f))) = \mathit{lfp}(\gamma \circ \alpha \circ f)$. Moreover, by hypothesis, $\gamma(\alpha(f(\gamma(\alpha(\mathit{lfp}(f)))))) = \gamma(\alpha(\mathit{lfp}(f)))$, and therefore $\gamma(\alpha(\mathit{lfp}(f))) = \mathit{lfp}(\gamma \circ \alpha \circ f)$.
($\Leftarrow$) The inequality $\alpha(\mathit{lfp}(f)) \leq_A \mathit{lfp}(\alpha \circ f \circ \gamma)$ always holds. Further, by hypothesis, we have that $\gamma(\alpha(f(\gamma(\alpha(\mathit{lfp}(f)))))) = \gamma(\alpha(\mathit{lfp}(f)))$, and since γ is 1-1, $\alpha(f(\gamma(\alpha(\mathit{lfp}(f))))) = \alpha(\mathit{lfp}(f))$, hence proving that $\alpha(\mathit{lfp}(f)) = \mathit{lfp}(\alpha \circ f \circ \gamma)$. ■

Recall that $\gamma \circ \alpha \in \mathit{uco}(C)$ is the upper closure associated with the abstract domain A (cf. Section 2). Since, for the closure operator approach, the best correct approximation of f for $\rho \in \mathit{uco}(C)$ is given by $f^{best_\rho} = \rho \circ f : \rho(C) \to \rho(C)$, and ρ plays the role of the abstraction map, the above fact simply states the equivalence of the notions of completeness and full completeness under the GI and closure operator approaches to abstract domain specification. Thus, in the following of this section, we will study completeness and full completeness relatively to closure operators and generic (monotone) functions from a purely algebraic point of view. In particular, the notion of full completeness is slightly generalized to cope with generic (possibly nonmonotone) n-ary functions.[2] A generic tuple of objects is denoted by $\vec{o}$, $\vec{o}_i$ denotes its i-th component, and $\vec{o}[X/i]$ denotes the tuple obtained from $\vec{o}$ by replacing $\vec{o}_i$ with X.

Definition 4.2 Let C be a complete lattice.

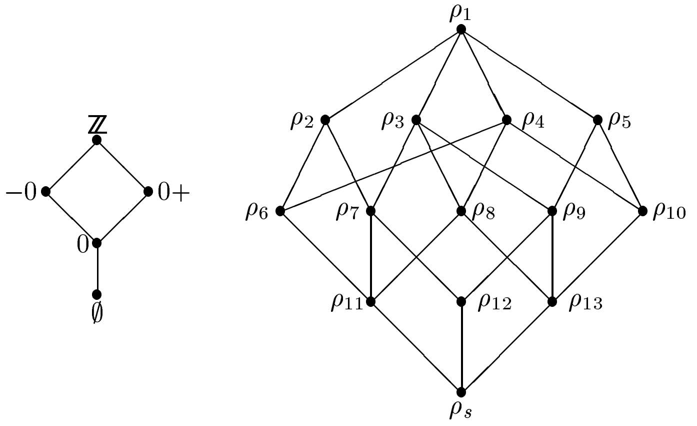
(i) Given $f : C^n \to C$ ($n \geq 1$), $\rho \in \mathit{uco}(C)$ is *fully complete* for f if for any $\vec{x} \in C^n$, $\rho(f(\vec{x})) = \rho(f(\rho(\vec{x}_1), ..., \rho(\vec{x}_n)))$.

(ii) Given $f \in C \xrightarrow{\text{m}} C$, $\rho \in \mathit{uco}(C)$ is *complete* for f if $\rho(\mathit{lfp}(f)) = \mathit{lfp}(\rho \circ f)$. □

By a slight abuse of notation, we denote the condition in the above point (i) simply by $\rho \circ f = \rho \circ f \circ \rho$. It is worth noting that Definition 4.2 (i) encompasses also functions of type $f : C \to (C \to \cdots (C \to C) \cdots)$, by considering the "Curryied" version of f. Moreover, observe that $\mathit{lfp}(\rho \circ f)$ in Definition 4.2 (ii) could be equivalently replaced by $\mathit{lfp}(\rho \circ f \circ \rho)$: In fact, by extensivity of ρ, $\mathit{lfp}(\rho \circ f) \leq_C \mathit{lfp}(\rho \circ f \circ \rho)$ always holds, and by idempotency of ρ, $\rho(f(\rho(\mathit{lfp}(\rho \circ f)))) = \rho(f(\mathit{lfp}(\rho \circ f))) = \mathit{lfp}(\rho \circ f)$.

In the following, we will be mainly interested in the set of all complete or fully complete abstract domains for a given semantic function. Therefore, in view of the above two types of completeness,

[2]It would not be difficult (but notational tedious) to consider even more general functions with n components in input and m in output.



$$
\begin{array}{rcl}
\rho_1 & = & \{\mathbb{Z}\} \\
\rho_2 & = & \{\mathbb{Z}, 0+\} \\
\rho_3 & = & \{\mathbb{Z}, 0\} \\
\rho_4 & = & \{\mathbb{Z}, \emptyset\} \\
\rho_5 & = & \{\mathbb{Z}, -0\} \\
\rho_6 & = & \{\mathbb{Z}, 0+, \emptyset\} \\
\rho_7 & = & \{\mathbb{Z}, 0+, 0\} \\
\rho_8 & = & \{\mathbb{Z}, 0, \emptyset\} \\
\rho_9 & = & \{\mathbb{Z}, -0, 0\} \\
\rho_{10} & = & \{\mathbb{Z}, -0, \emptyset\} \\
\rho_{11} & = & \{\mathbb{Z}, 0+, 0, \emptyset\} \\
\rho_{12} & = & \{\mathbb{Z}, -0, 0+, 0\} \\
\rho_{13} & = & \{\mathbb{Z}, -0, 0, \emptyset\} \\
\rho_s & = & \mathit{Sign}
\end{array}
$$

Figure 1: The lattices *Sign* and $uco(\mathit{Sign})$.

we will use the following notation. Let C denote a concrete domain (viz. a complete lattice) of reference. For a given function $f \in C^n \to C$, we define $\Gamma(C,f) \subseteq uco(C)$ to be the set of fully complete closures on C for f, namely:

$$\Gamma(C,f) = \{\rho \in uco(C) \mid \rho \circ f = \rho \circ f \circ \rho\}.$$

If $f : C \xrightarrow{\mathrm{m}} C$ then we define $\Delta(C,f) \subseteq uco(C)$ as the set of complete closures on C for f, namely:

$$\Delta(C,f) = \{\rho \in uco(C) \mid \rho(\mathit{lfp}(f)) = \mathit{lfp}(\rho \circ f)\}.$$

Also, if $\eta \in uco(C)$ 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). It is helpful to recall (see e.g. [Cousot 1978, Theorem 4.2.0.4.7] for a detailed proof) that if $\eta \in uco(C)$ then $uco(\eta(C))$ is isomorphic to the principal filter of $uco(C)$ generated by η, i.e. $\langle uco(\eta(C)), \sqsubseteq \rangle \cong \langle \uparrow \eta, \sqsubseteq \rangle$, and therefore $\rho \in uco(\eta(C))$ iff $\rho \in uco(C)$ and $\rho \sqsubseteq \eta$. By this remark, we get that $\Gamma^{\uparrow\eta}(C,f) = \Gamma(C,f) \cap \uparrow\eta$ and $\Delta^{\uparrow\eta}(C,f) = \Delta(C,f) \cap \uparrow\eta$. Using the more familiar terminology of abstract interpretation, $\Gamma^{\uparrow\eta}(C,f)$ and $\Delta^{\uparrow\eta}(C,f)$ are the sets of abstract domains which are more abstract than η and, respectively, fully complete and complete for f. Moreover, if (α, C, A, γ) is a GI such that $\gamma \circ \alpha = \eta$, then $\Gamma^{\uparrow A}(C,f)$ and $\Delta^{\uparrow A}(C,f)$ can be used as 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 (by a slight abuse of notation) $\Gamma(C,F) = \cap_{f \in F} \Gamma(C,f)$ and $\Delta(C,G) = \cap_{g \in G} \Delta(C,g)$. Obviously, $\Gamma(C,\emptyset) = \Delta(C,\emptyset) = uco(C)$.

We can then restate the basic Cousot and Cousot [1979b] result on completeness using our notation as follows.

Proposition 4.3 (Cousot and Cousot 1979b) *If $F \subseteq C \xrightarrow{\mathrm{m}} C$ then $\Gamma(C,F) \subseteq \Delta(C,F)$.*

Let us now illustrate the notions introduced so far through the simple and classical example of the rule of signs for integer numbers [Cousot and Cousot 1977].

Example 4.4 Let us consider the abstract domain *Sign* depicted in Figure 1, which is an obvious abstraction of $\langle \wp(\mathbb{Z}), \subseteq \rangle$ (concretization and abstraction maps are the most obvious ones). 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 Mycroft [1993], 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, as still Mycroft [1993] observes, *Sign* (i.e. ρ_s) is not fully complete for integer addition $\oplus : \wp(\mathbb{Z})^2 \to \wp(\mathbb{Z})$, where $X \oplus Y = \{n + m \mid n \in X, \ m \in Y\}$. 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_{ev} that selects, e.g., even numbers, i.e. $f_{ev} = \lambda X. X \cap \mathbb{Z}_{even}$. Then, while *Sign* is not fully complete for f_{ev} (e.g., $\rho_s(f_{ev}(\{-1,2\})) = 0+ \neq \mathbb{Z} = \rho_s(f_{ev}(\rho_s(\{-1,2\})))$), it is instead complete: $\rho_s(\mathit{lfp}(f_{ev})) = \rho_s(\emptyset) = \emptyset = \mathit{lfp}(\rho_s \circ f_{ev})$.
Let us now consider the lattice $uco(Sign)$ of all possible abstractions of *Sign*, depicted in Figure 1, 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:

- $\Delta^{\uparrow Sign}(\wp(\mathbb{Z}), sq) = uco(Sign) \setminus \{\rho_5\}$: 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;
- $\Gamma^{\uparrow Sign}(\wp(\mathbb{Z}), sq) = uco(Sign) \setminus \{\rho_5, \rho_{10}\}$: For ρ_5 and ρ_{10}, just consider $X = \{0\}$.

As an important consequence, 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 the set of fully complete abstractions of *Sign* for the binary integer multiplication is $\Gamma^{\uparrow Sign}(\wp(\mathbb{Z}), *) = uco(Sign) \setminus \{\rho_2, \rho_5, \rho_6, \rho_{10}\}$, which is a complete sublattice of $uco(Sign)$. □

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

Proposition 4.5 *Let $f : C^n \to C$, $g : C \to C$, and $h : C \xrightarrow{\mathrm{m}} C$.*

(i) $\lambda x. \top_C, \lambda x. x \in \Gamma(C, f) \cap \Delta(C, h)$.

(ii) *For all* $c \in C$, $\Gamma(C, \lambda\vec{x}. c) = \Gamma(C, \lambda x. x) = uco(C) = \Delta(C, \lambda x. x) = \Delta(C, \lambda x. c)$.

(iii) $\Gamma(C, \lambda\vec{x}. \vee_{i=1}^{n} x_i) = uco(C)$.

(iv) $\rho \in \Gamma(C, \lambda\vec{x}. \wedge_{i=1}^{n} x_i) \Leftrightarrow (\forall X \subseteq C.\ |X| < \omega \Rightarrow \rho(\wedge X) = \wedge\rho(X))$.

(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* $\rho, \eta \in uco(C)$, $\rho \circ \eta = \eta \circ \rho \Leftrightarrow \rho \in \Gamma(C, \eta)\ \&\ \eta \in \Gamma(C, \rho)$.

(viii) *For* $i \in [1, n]$, $\Gamma(C, f) = \Gamma(C, \{\lambda\vec{x} \in C^{n-1}. f(\langle x_1, ..., x_{i-1}, c, x_i, ..., x_{n-1}\rangle) : C^{n-1} \to C\}_{c \in C})$.

(ix) $\Gamma(C, h) \subseteq \Gamma(C, h + id) \subseteq \Delta(C, h + id) \subseteq \Delta(C, h)$.

Proof. (i) and (ii): Clear.
(iii): By property (ii) of upper closures in Section 2.
(iv): By property (i) of upper closures in Section 2.
(v): Let $\rho \in \Gamma(C, \{f, g\})$ and $\vec{x} \in C^n$. Then,

$$\begin{aligned} \rho(g(f(\vec{x}))) &= \rho(g(\rho(f(\vec{x})))) \\ &= \rho(g(\rho(f(\rho(\vec{x}_1), ..., \rho(\vec{x}_n))))) \\ &= \rho(g(f(\rho(\vec{x}_1), ..., \rho(\vec{x}_n)))), \end{aligned}$$

and therefore $\rho \in \Gamma(C, g \circ f)$.
(vi): $\rho \circ \eta \circ f = \rho \circ \eta \circ f \circ \eta = \eta \circ \rho \circ f \circ \eta = \eta \circ \rho \circ f \circ \rho \circ \eta = \rho \circ \eta \circ f \circ \rho \circ \eta$.
(vii): If $\rho \circ \eta = \eta \circ \rho$ then, by idempotency of ρ and η, $\rho \in \Gamma(C, \eta)$ and $\eta \in \Gamma(C, \rho)$. Conversely, if $\rho \in \Gamma(C, \eta)$ and $\eta \in \Gamma(C, \rho)$ then $\rho \circ \eta \leq \eta \circ \rho \circ \eta = \eta \circ \rho \leq= \rho \circ \eta \circ \rho = \rho \circ \eta$.
(viii): Clear, by definition.
(ix): First note that $h + id : C \xrightarrow{\mathrm{m}} C$. That $\Gamma(C, h + id) \subseteq \Delta(C, h + id)$ follows by Proposition 4.3. Let $\rho \in \Gamma(C, h)$ and $x \in C$. Then, by exploiting property (ii) in Section 2, $\rho(h(x) \vee x) = \rho(\rho(h(x)) \vee \rho(x)) = \rho(\rho(h(\rho(x))) \vee \rho(\rho(x))) = \rho(h(\rho(x)) \vee \rho(x))$, and therefore $\rho \in \Gamma(C, h + id)$. Let now $\rho \in \Delta(C, h + id)$. Notice that $\rho \circ h \leq \rho \circ (h + id)$, and therefore $\mathit{lfp}(\rho \circ h) \leq \mathit{lfp}(\rho \circ (h + id)) = \rho(\mathit{lfp}(h + id)) = \rho(\mathit{lfp}(h))$, and this suffices to conclude. ■

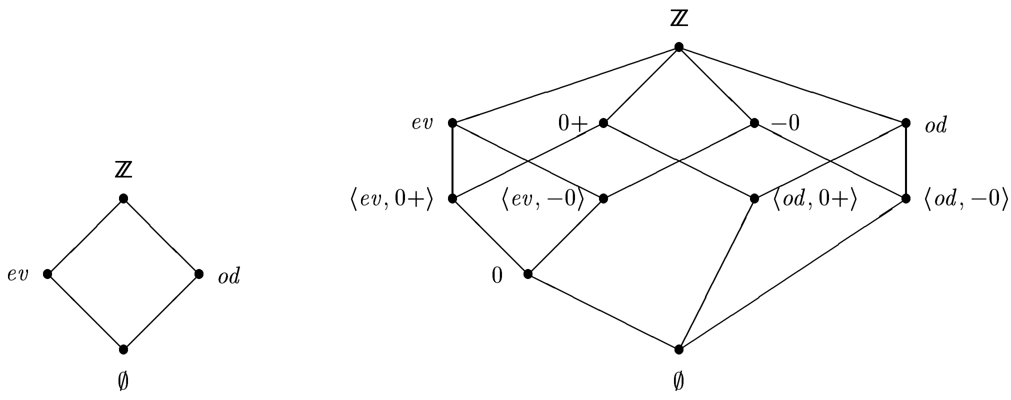


Figure 2: The lattices *Parity* and $Sign \sqcap Parity$.

In particular, point (iii) says that any abstract domain is fully complete for the concrete *lub* operation, which is often used in program analysis frameworks, e.g. for logic programs [Debray 1994]; while it is fully complete for the concrete *glb* operation iff the corresponding closure operator is finitely co-additive, as stated in point (iv). This latter result is particularly important in logic programming where the concrete *glb* operation plays often the role of constraint unification. Point (v) states that full completeness for a composite function $f \circ g$ can be compositionally checked on f and g. This is important in order to verify full completeness for semantics inductively specified on the syntax of programs, as it is usually done in denotational semantics. Point (vi) is an instance of a more general result later in Theorem 4.11: In fact it is known (cf. [Morgado 1960]) that if two closures ρ and η are such that $\rho \circ \eta = \eta \circ \rho$ then $\rho \circ \eta = \rho \sqcup \eta$. Point (vii) characterizes the set of fully complete abstractions for a function which is itself a closure operator. Finally, point (viii) says that the property of full completeness for an n-ary function can be verified componentwise.

We now consider both $\Gamma(C,f)$ and $\Delta(C,f)$ equipped with the pointwise partial order $\sqsubseteq$ of relative precision between abstract domains, inherited from the lattice of abstract interpretations $uco(C)$. We are firstly interested in studying the lattice-theoretic properties of such posets.

4.1 Reduced Product of Complete Domains

Our first finding is that $\Gamma(C,f)$ and $\Delta(C,f)$ 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 reduced product was already observed by Cortesi et al. [1997, Proposition 5.2.3].

Theorem 4.6 *Let* $f : C^n \xrightarrow{\text{m}} C$ *and* $g : C \xrightarrow{\text{m}} C$*. Then* $\Gamma(C,f), \Delta(C,g) \in lco(uco(C))$.

Proof. Let $\{\rho_i\}_{i\in I} \subseteq \Gamma(C,f)$ and $x \in C$. For simplicity of notation, we assume $n = 1$, i.e. f unary; the extension to the general case is straightforward. It is sufficient to show that $\wedge_{i\in I}\rho_i(f(\wedge_{k\in I}\rho_k(x))) \leq \wedge_{i\in I}\rho_i(f(x))$, because the other inequality always holds. Thus, for any $j \in I$, we have to prove that $\wedge_{i\in I}\rho_i(f(\wedge_{k\in I}\rho_k(x))) \leq \rho_j(f(x))$: By hypothesis and monotonicity of f and closure operators, $\wedge_{i\in I}\rho_i(f(\wedge_{k\in I}\rho_k(x))) \leq \rho_j(f(\wedge_{k\in I}\rho_k(x))) \leq \rho_j(f(\rho_j(x))) = \rho_j(f(x))$. Now, let $\{\rho_i\}_{i\in I} \subseteq \Delta(C,g)$ and $x \in C$, and recall that $\lambda h.\, lfp(h)$ is monotone. Then, for all $j \in I$, $lfp((\sqcap_{i\in I}\rho_i) \circ g) \leq lfp(\rho_j \circ g)$, and therefore, $lfp((\sqcap_{i\in I}\rho_i) \circ g) \leq \wedge_{i\in I} lfp(\rho_i \circ g) = \wedge_{i\in I}\rho_i(lfp(g)) = (\sqcap_{i\in I}\rho_i)(lfp(g))$. Since the other inequality always holds, this concludes the proof. ∎

Corollary 4.7 *Let* $F \subseteq C^n \xrightarrow{\text{m}} C$ *and* $G \subseteq C \xrightarrow{\text{m}} C$*. Then* $\Gamma(C,F), \Delta(C,G) \in lco(uco(C))$.

Proof. Because set-intersection of sets of fixpoints of lower closures is still a set of fixpoint of a lower closure. ■

Example 4.8 It is straightforward to prove that the domain *Parity* for parity analysis of integer variables in Figure 2, which is an abstraction of $\langle\wp(\mathbb{Z}),\subseteq\rangle$, is fully complete for integer multiplication $\lambda XY.\,X * Y$, as defined in Example 4.4, where *Sign* was also observed to be fully complete. Thus, by Theorem 4.6, the reduced product $\mathit{Sign} \sqcap \mathit{Parity}$ depicted in Figure 2 is fully complete for integer multiplication as well. □

4.2 Least Common Abstraction of Complete Domains

Let us now turn to the *lub*, namely to the least common abstraction of complete domains. We have already observed in Example 4.4 that, in general, a subset of complete closures $\Delta(C,f)$ is not closed under *lub*'s. Even more, the following contrived example shows that this remains true also when C and f satisfy very strong hypothesis.

Example 4.9 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 $\mathit{lfp}(f) = 0$. Next, consider the closures $\rho_1, \rho_2 \in \mathit{uco}(C)$ given by $\rho_1 = \{1,3,4\}$ and $\rho_2 = \{2,3,4\}$. It is not difficult to verify that $\rho_1, \rho_2 \in \Delta(C,f)$: Indeed, $\rho_k(\mathit{lfp}(f)) = k = \mathit{lfp}(\rho_k \circ f)$. It turns out that their *lub* $\rho_1 \sqcup \rho_2 = \{3,4\}$ does not belong to $\Delta(C,f)$. In fact, $(\rho_1 \sqcup \rho_2)(\mathit{lfp}(f)) = 3$, while $(\rho_1 \sqcup \rho_2) \circ f = \{0 \mapsto 3, 1 \mapsto 3, 2 \mapsto 3, 3 \mapsto 4, 4 \mapsto 4\}$ and therefore $\mathit{lfp}((\rho_1 \sqcup \rho_2) \circ f) = 4$. □

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

Example 4.10 Let C be the $\omega + 2$ ordinal 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 4.5 (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 4.10 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.[3]

Theorem 4.11 *If* $f : C^n \xrightarrow{c} C$ *then* $\Gamma(C,f) \in \mathit{uco}(\mathit{uco}(C))$.

The proof of this theorem is quite intricate. We demonstrate this result by gradually proving a number of lemmata. The first one is due to Cousot and Cousot [1979a, Theorem 4.3] and gives a constructive characterization for the *lub* operation $\sqcup$ on closure operators.

Lemma 4.12 (Cousot and Cousot 1979a) *Let* $\{\rho_i\}_{i\in I} \subseteq \mathit{uco}(C)$. *Then, there exists* $\delta \in \mathit{Ord}$ *such that for any* $x \in C$, $(\sqcup_{i\in I}\rho_i)(x) = \vee_{\beta<\delta}(\lambda y.\ \vee_{i\in I}\ \rho_i(y))^{\beta}(x)$.

The next lemma demonstrates that for a continuous unary function $f : C \xrightarrow{c} C$, the *lub* of a nonempty family of fully complete closures $\{\rho_i\}_{i\in I}$ for f is still fully complete for f iff the following condition holds:

$$\forall x \in C.\ (\sqcup_{i\in I}\rho_i)(f(x)) = (\sqcup_{i\in I}\rho_i)(f(\vee_{i\in I}\rho_i(x))). \qquad (\ddagger)$$

Lemma 4.13 *Let* $f : C \xrightarrow{c} C$ *and* $\{\rho_i\}_{i\in I} \subseteq \Gamma(C,f)$ *with* $I \neq \emptyset$. *Then,* $\sqcup_{i\in I}\rho_i \in \Gamma(C,f)$ *iff* $(\ddagger)$ *holds.*

[3] Amato and Levi [1997] have independently made an observation similar to Example 4.10, and stated an analogous but weaker result to Theorem 4.11 for additive functions.

Proof. ($\Rightarrow$) Assume $(\sqcup_{i\in I}\rho_i) \in \Gamma(C,f)$ and let $x \in C$. Then,

$$\begin{array}{rcll}(\sqcup_{i\in I}\rho_i)(f(x)) & = & (\sqcup_{i\in I}\rho_i)(f((\sqcup_{i\in I}\rho_i)(x))) & \\ & \geq & (\sqcup_{i\in I}\rho_i)(f(\vee_{i\in I}\rho_i(x))) & (\text{since } \lambda x.\ \vee_{i\in I}\ \rho_i(x) \leq \lambda x.\,(\sqcup_{i\in I}\rho_i)(x)) \\ & \geq & (\sqcup_{i\in I}\rho_i)(f(x)) & (\text{since } \lambda x.\, x \leq \lambda x.\ \vee_{i\in I}\ \rho_i(x)),\end{array}$$

and therefore, (‡) holds.
($\Leftarrow$) Assume (‡) holds. We prove by transfinite induction that for any $\alpha \in \mathit{Ord}$:

$$(\sqcup_{i\in I}\rho_i)(f(x)) = (\sqcup_{i\in I}\rho_i)(f((\lambda y.\ \vee_{i\in I}\ \rho_i(y))^{\alpha}(x))).$$

($\alpha = 0$): Trivial.

($\alpha + 1$): We have that:

$$\begin{array}{lcl}(\sqcup_{i\in I}\rho_i)(f((\lambda y.\ \vee_{i\in I}\ \rho_i(y))^{\alpha+1}(x))) & = & (\text{by definition}) \\ (\sqcup_{i\in I}\rho_i)(f(\vee_{i\in I}\rho_i((\lambda y.\ \vee_{i\in I}\ \rho_i(y))^{\alpha}(x)))) & = & (\text{by } (\ddagger)) \\ (\sqcup_{i\in I}\rho_i)(f((\lambda y.\ \vee_{i\in I}\ \rho_i(y))^{\alpha}(x))) & = & (\text{by induction}) \\ (\sqcup_{i\in I}\rho_i)(f(x)). & & \end{array}$$

(α limit ordinal): Let $x \in C$. First note that $\{(\lambda y.\ \vee_{i\in I}\ \rho_i(y))^{\alpha}(x)\}_{\beta<\alpha}$ is an increasing chain, since, by extensivity of closures, $x \leq (\lambda y.\ \vee_{i\in I}\ \rho_i(y))(x)$. Hence, by continuity of f:

$$(\sqcup_{i\in I}\rho_i)(f(\vee_{\beta<\alpha}(\lambda y.\ \vee_{i\in I}\ \rho_i(y))^{\beta}(x))) = (\sqcup_{i\in I}\rho_i)(\vee_{\beta<\alpha}f((\lambda y.\ \vee_{i\in I}\ \rho_i(y))^{\beta}(x))).$$

Moreover,

$$\begin{array}{lcl}(\sqcup_{i\in I}\rho_i)(\vee_{\beta<\alpha}f((\lambda y.\ \vee_{i\in I}\ \rho_i(y))^{\beta}(x))) & = & (\text{by (i) in Section 2}) \\ (\sqcup_{i\in I}\rho_i)(\vee_{\beta<\alpha}(\sqcup_{i\in I}\rho_i)(f((\lambda y.\ \vee_{i\in I}\ \rho_i(y))^{\beta}(x)))) & = & (\text{by induction}) \\ (\sqcup_{i\in I}\rho_i)(\vee_{\beta<\alpha}(\sqcup_{i\in I}\rho_i)(f(x))) & = & (\text{by (i) in Section 2}) \\ (\sqcup_{i\in I}\rho_i)(f(x)). & & \end{array}$$

By Lemma 4.12, there exists $\delta \in \mathit{Ord}$ such that $(\sqcup_{i\in I}\rho_i)(x) = \vee_{\beta<\delta}(\lambda y.\ \vee_{i\in I}\ \rho_i(y))^{\beta}(x)$. Consider therefore $\{f((\lambda y.\ \vee_{i\in I}\ \rho_i(y))^{\beta}(x))\}_{\beta<\delta}$, which is an increasing chain in C. Since for all $\alpha \in \mathit{Ord}$, $f(x) \approx_{\sqcup_{i\in I}\rho_i} f((\lambda y.\ \vee_{i\in I}\ \rho_i(y))^{\alpha}(x))$, $\approx_{\sqcup_{i\in I}\rho_i}$ is a join-complete congruence relation and f is continuous, we get:

$$f(x) \approx_{\sqcup_{i\in I}\rho_i} \vee_{\beta<\delta}f((\lambda y.\ \vee_{i\in I}\ \rho_i(y))^{\beta}(x)) = f(\vee_{\beta<\delta}(\lambda y.\ \vee_{i\in I}\ \rho_i(y))^{\beta}(x)) = f((\sqcup_{i\in I}\rho_i)(x)).$$

Thus, we get $f(x) \approx_{\sqcup_{i\in I}\rho_i} f((\sqcup_{i\in I}\rho_i)(x))$, and this concludes the proof. ∎

The next lemma proves that for finite subsets of $\Gamma(C,f)$, property (‡) holds.

Lemma 4.14 *If $\{\rho_i\}_{i\in I} \subseteq \Gamma(C,f)$ and $0 < |I| < \omega$, then (‡) holds.*

Proof. Let $x \in C$. The inequality $(\sqcup_{i\in I}\rho_i)(f(x)) \leq (\sqcup_{i\in I}\rho_i)(f(\vee_{i\in I}\rho_i(x)))$ always holds, by monotonicity and extensivity. Then, we prove by induction on the cardinality $|I| = n$ that

$$(\sqcup_{i\in I}\rho_i)(f(x)) \geq (\sqcup_{i\in I}\rho_i)(f(\vee_{i\in I}\rho_i(x))).$$

($n = 1$): Trivial, by full completeness.

($n + 1$): Assume that $\{\rho_i\}_{i\in I} = \{\rho_i\}_{i\in \bar{I}} \cup \{\bar{\rho}\}$ for some $\bar{I} \subset I$ such that $|\bar{I}| = n$. Let $x \in C$. Then,

$$(\sqcup_{i\in I}\rho_i)(f(\vee_{i\in I}\rho_i(x))) = (\sqcup_{i\in I}\rho_i)(f((\vee_{i\in \bar{I}}\rho_i(x)) \vee \bar{\rho}(x))).$$

Moreover, by monotonicity and extensivity:

$$(\vee_{i\in \bar{I}}\rho_i(x)) \vee \bar{\rho}(x) \leq (\vee_{i\in \bar{I}}\rho_i(x)) \vee \bar{\rho}(\vee_{i\in \bar{I}}\rho_i(x)) = \bar{\rho}(\vee_{i\in \bar{I}}\rho_i(x)). \quad (*)$$

Therefore, by monotonicity, we have the following inequalities:

$$\begin{array}{lll}
(\sqcup_{i\in I}\rho_i)(f(\vee_{i\in I}\rho_i(x))) & \leq & \text{(by } (*)) \\
(\sqcup_{i\in I}\rho_i)(f(\bar{\rho}(\vee_{i\in \bar{I}}\rho_i(x)))) & = & \text{(since } (\sqcup_{i\in I}\rho_i)\circ\bar{\rho} = \sqcup_{i\in I}\rho_i) \\
(\sqcup_{i\in I}\rho_i)(\bar{\rho}(f(\bar{\rho}(\vee_{i\in \bar{I}}\rho_i(x))))) & = & \text{(by full completeness of } \bar{\rho} \text{ for } f) \\
(\sqcup_{i\in I}\rho_i)(\bar{\rho}(f(\vee_{i\in \bar{I}}\rho_i(x)))) & = & \text{(since } (\sqcup_{i\in I}\rho_i)\circ\bar{\rho} = \sqcup_{i\in I}\rho_i) \\
(\sqcup_{i\in I}\rho_i)(f(\vee_{i\in \bar{I}}\rho_i(x))). & &
\end{array}$$

By inductive hypothesis, $(\sqcup_{i\in \bar{I}}\rho_i)(f(\vee_{i\in \bar{I}}\rho_i(x))) \leq (\sqcup_{i\in \bar{I}}\rho_i)(f(x))$. Moreover,

$$\begin{array}{lll}
(\sqcup_{i\in I}\rho_i)(f(\vee_{i\in \bar{I}}\rho_i(x))) & = & \text{(since } (\sqcup_{i\in I}\rho_i)\circ(\sqcup_{i\in \bar{I}}\rho_i) = \sqcup_{i\in I}\rho_i) \\
((\sqcup_{i\in I}\rho_i)\circ(\sqcup_{i\in \bar{I}}\rho_i))(f(\vee_{i\in \bar{I}}\rho_i(x))) & \leq & \text{(by induction)} \\
((\sqcup_{i\in I}\rho_i)\circ(\sqcup_{i\in \bar{I}}\rho_i))(f(x)) & = & \text{(since } (\sqcup_{i\in I}\rho_i)\circ(\sqcup_{i\in \bar{I}}\rho_i) = \sqcup_{i\in I}\rho_i) \\
(\sqcup_{i\in I}\rho_i)(f(x)), & &
\end{array}$$

and therefore, $(\sqcup_{i\in I}\rho_i)(f(\vee_{i\in I}\rho_i(x))) \leq (\sqcup_{i\in I}\rho_i)(f(x))$. ∎

The next lemma demonstrates our claim for continuous unary functions.

Lemma 4.15 *If $f \in C \xrightarrow{c} C$ and $\{\rho_i\}_{i\in I} \subseteq \Gamma(C,f)$ then $\sqcup_{i\in I}\rho_i \in \Gamma(C,f)$.*

Proof. If $I = \emptyset$ then $\sqcup_{i\in I}\rho_i = \lambda x.\, x \in \Gamma(C,f)$, by Proposition 4.5 (i). Then, let us assume $|I| \geq 1$. Thus, by Lemma 4.13, it is enough to show that (‡) holds. We write $J \subseteq_f I$ when $J \subseteq I$ and $|J| < \omega$. Let $x \in C$. For any $J \subseteq_f I$, we have:

$$\begin{array}{lll}
(\sqcup_{i\in I}\rho_i)(f(x)) & = & \text{(since } (\sqcup_{i\in I}\rho_i)\circ(\sqcup_{j\in J}\rho_j) = \sqcup_{i\in I}\rho_i) \\
(\sqcup_{i\in I}\rho_i)((\sqcup_{j\in J}\rho_j)(f(x))) & = & \text{(by Lemma 4.14)} \\
(\sqcup_{i\in I}\rho_i)((\sqcup_{j\in J}\rho_j)(f(\vee_{j\in J}\rho_j(x)))) & = & \text{(since } (\sqcup_{i\in I}\rho_i)\circ(\sqcup_{j\in J}\rho_j) = \sqcup_{i\in I}\rho_i) \\
(\sqcup_{i\in I}\rho_i)(f(\vee_{j\in J}\rho_j(x))). & &
\end{array}$$

Thus, for any $J \subseteq_f I$, $f(\vee_{j\in J}\rho_j(x)) \approx_{\sqcup_{i\in I}\rho_i} f(x)$. Hence, by join-completeness of $\approx_{\sqcup_{i\in I}\rho_i}$, we get $f(x) \approx_{\sqcup_{i\in I}\rho_i} \vee_{J\subseteq_f I} f(\vee_{j\in J}\rho_j(x))$. Moreover, $\{\vee_{j\in J}\rho_j(x) \mid J \subseteq_f I\}$ is clearly a directed subset of C. By continuity, f also preserves *lub*'s of directed subsets. Thus, $\vee_{J\subseteq_f I} f(\vee_{j\in J}\rho_j(x)) = f(\vee_{J\subseteq_f I}\vee_{j\in J}\rho_j(x)) = f(\vee_{i\in I}\rho_i(x))$, and therefore $f(x) \approx_{\sqcup_{i\in I}\rho_i} f(\vee_{i\in I}\rho_i(x))$, which concludes the proof. ∎

We are now in the position to give the proof of Theorem 4.11.

Proof of Theorem 4.11. Notice that for any $i \in [1,n]$ and $\vec{x} \in C^n$ $f^i_{\vec{x}} = \lambda y.\, f(\vec{x}[y/i])$ is a continuous function, i.e., $f^i_{\vec{x}} : C \xrightarrow{c} C$. We use the following shorthand: For any $\rho \in uco(C)$, $\rho(\vec{x}) = \langle \rho(x_1), ..., \rho(x_n)\rangle$. Let us then show that for any $\rho \in uco(C)$, $\rho \in \Gamma(C,f)$ iff for any $i \in [1,n]$ and $\vec{x} \in C^n$, $\rho \in \Gamma(C, f^i_{\vec{x}})$.
(⇒) Let $y \in C$. $\rho(f^i_{\vec{x}}(y)) \leq \rho(f^i_{\vec{x}}(\rho(y)))$ always holds. Moreover, by extensivity and monotonicity, $\rho(f^i_{\vec{x}}(\rho(y))) \leq \rho(f^i_{\rho(\vec{x})}(\rho(y))) = \rho(f^i_{\vec{x}}(y))$.
(⇐) Let $\vec{x} \in C^n$. Then, by applying n times the hypothesis,

$$\begin{array}{lll}
\rho(f(\vec{x})) & = & \rho(f(\rho(x_1), x_2, ..., x_n)) \\
 & = & \rho(f(\rho(x_1), \rho(x_2), ..., x_n)) \\
 & = & \cdots \\
 & = & \rho(f(\rho(\vec{x}))),
\end{array}$$

namely $\rho \in \Gamma(C,f)$.
Thus, $\Gamma(C,f) = \cap\{\Gamma(C, f^i_{\vec{x}}) \mid i \in [1,n],\ \vec{x} \in C^n\}$. The thesis follows by Lemma 4.15, since for any i and $\vec{x}$, $\Gamma(C, f^i_{\vec{x}}) \in uco(uco(C))$, and set-intersection of (sets of fixpoints of) upper closures gives (a set of fixpoints of) an upper closure. ∎

Corollary 4.16 *If* $F \subseteq C^n \xrightarrow{c} C$ *then* $\Gamma(C, F) \in uco(uco(C))$.

Proof. Because set-intersection of sets of fixpoints of upper closures is still a set of fixpoints of an upper closure. ■

Thus, the following result is a direct consequence of Corollaries 4.7 and 4.16.

Corollary 4.17 *If* $F \subseteq C^n \xrightarrow{c} C$ *then* $\Gamma(C, F)$ *is a complete sublattice of* $uco(C)$.

Obviously, continuity is a well-known and sufficiently weak hypothesis, so that our result on full completeness is widely applicable in the field of programming language semantics and analysis. As a consequence of Corollary 4.17, we get that when $uco(C)$ is pseudocomplemented (and, as recalled in Section 2, this holds when C is meet-continuous) then $\Gamma(C, F)$ is pseudocomplemented as well.

Corollary 4.18 *If* $uco(C)$ *is pseudocomplemented and* $F \subseteq C^n \xrightarrow{c} C$ *then* $\Gamma(C, F)$ *is pseudocomplemented.*

Proof. By Proposition 4.5 (i), $\lambda x. x$ is the bottom in $\Gamma(C, F)$, and by Corollary 4.17, $\sqcap$ and $\sqcup$ are the *glb* and the *lub* of $\Gamma(C, F)$. Let $\rho \in \Gamma(C, F)$ and let ρ^* be the pseudocomplement of ρ in $uco(C)$. Then, from $\lambda x. x \sqsubseteq \sqcup\{\eta \in \Gamma(C, F) \mid \rho \sqcap \eta = \lambda x. x\} \sqsubseteq \rho^*$, we get $\lambda x. x \sqsubseteq (\sqcup\{\eta \in \Gamma(C, F) \mid \rho \sqcap \eta = \lambda x. x\}) \sqcap \rho \sqsubseteq \rho^* \sqcap \rho = \lambda x. x$, i.e. $\sqcup\{\eta \in \Gamma(C, F) \mid \rho \sqcap \eta = \lambda x. x\}$ is the pseudocomplement of ρ in $\Gamma(C, F)$. ■

However, in general, $\Gamma(C, F)$ is not a sub-pseudocomplemented lattice of $uco(C)$. In fact, with reference to Example 4.4, for $\rho_7 \in \Gamma^{\uparrow Sign}(\wp(\mathbb{Z}), sq)$, its pseudocomplement in $uco(Sign)$ is ρ_{10}, whereas in $\Gamma^{\uparrow Sign}(\wp(\mathbb{Z}), sq)$ is ρ_{13}. Moreover, we can also observe that, in general, $\Gamma(C, f)$ and $\Delta(C, g)$ (for suitable f and g) are neither atomic nor dual-atomic: It is enough to consider $\Gamma^{\uparrow Sign}(\wp(\mathbb{Z}), sq)$ and $\Delta^{\uparrow Sign}(\wp(\mathbb{Z}), sq)$ of Example 4.4, to verify these negative claims.

5 Completeness by Domain Transformers

The notion of *abstract domain refinement* has been studied by Filé et al. [1996], and more recently by Giacobazzi and Ranzato [1997], as a formalization and generalization for most of the systematic operators enhancing the precision of abstract domains (e.g. reduced product and disjunctive completion [Cousot and Cousot 1979b]). For a fixed concrete domain C, a (unary) abstract domain refinement is defined as a mapping $\Re : uco(C) \to uco(C)$, such that $\Re$ is monotone, and, for any $A \in uco(C)$, $\Re(A)$ 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 $uco(C)$, i.e. mappings belonging to $lco(uco(C))$. A dual theory holds for abstract domain *simplifications*, that are defined as upper closures belonging to $uco(uco(C))$ (cf. [Giacobazzi and Ranzato 1997]). Following this framework and exploiting the results of Section 4, in this section we introduce the notions of *complete* and *fully complete kernel* simplifications, with respect to any set F of monotone concrete functions, and the notion of *least fully complete extension* refinement, with respect to any set F of continuous concrete 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.

5.1 Complete Kernel Operators

Let us introduce the following basic definitions.

Definition 5.1 Let $F \subseteq C^n \to C$ and $G \subseteq C \xrightarrow{\mathrm{m}} C$.

(i) $\mathcal{K}_F^C : uco(C) \to uco(C)$ is defined as $\mathcal{K}_F^C(X) = \sqcap\{Y \in uco(C) \mid X \sqsubseteq Y,\ Y \in \Gamma(C, F)\}$.

(ii) $\mathbb{K}_G^C : uco(C) \to uco(C)$ is defined as $\mathbb{K}_G^C(X) = \sqcap\{Y \in uco(C) \mid X \sqsubseteq Y,\ Y \in \Delta(C, G)\}$. $\square$

When $F = \{f\}$, we simply use $\mathcal{K}_f^C$ and $\mathbb{K}_f^C$. The following observation is immediate.

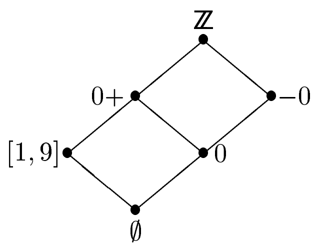
Lemma 5.2 *For any $F \subseteq C^n \to C$ and $G \subseteq C \xrightarrow{\mathrm{m}} C$, $\mathcal{K}_F^C, \mathbb{K}_G^C \in uco(uco(C))$.*

Thus, $\mathcal{K}_F^C$ and $\mathbb{K}_G^C$ are idempotent abstract domain simplifications (cf. [Giacobazzi and Ranzato 1997]). We then define $\mathcal{K}_F^C$, respectively $\mathbb{K}_G^C$, to be *safely defined* when for any $X \in uco(C)$, $\mathcal{K}_F^C(X) \in \Gamma(C, F)$, respectively $\mathbb{K}_G^C(X) \in \Delta(C, G)$, i.e. when $\mathcal{K}_F^C(X)$ and $\mathbb{K}_G^C(X)$ are, respectively, the greatest fully complete for (every function in) F and complete for (every function in) G, restrictions of X. 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 Corollary 4.7, $\mathbb{K}_G^C$ is always safely defined, while the weak hypothesis of monotonicity on F is enough for $\mathcal{K}_F^C$.

Proposition 5.3 *For any $F \subseteq C^n \xrightarrow{\mathrm{m}} C$ and $G \subseteq C \xrightarrow{\mathrm{m}} C$, $\mathcal{K}_F^C$ and $\mathbb{K}_G^C$ are safely defined.*

Example 5.4 Let us consider Example 4.4. We have seen that $\{\mathbb{Z}, -0, \emptyset\}$ is not fully complete for the square operation sq, and $\{\mathbb{Z}, -0\}$ is not even complete for sq. By Proposition 5.3, we know that there exist their corresponding greatest (fully) complete restrictions. It is immediate to check straight from the diagram of $uco(Sign)$ in Figure 1 that $\mathcal{K}_{sq}^{\wp(\mathbb{Z})}(\{\mathbb{Z}, -0, \emptyset\}) = \{\mathbb{Z}, \emptyset\}$ and $\mathbb{K}_{sq}^{\wp(\mathbb{Z})}(\{\mathbb{Z}, -0\}) = \{\mathbb{Z}\}$. $\square$

Example 5.5 Let us consider the following abstract domain $Sign^+$, introduced by Mycroft [1993, Sect. 3.1]. With respect to $Sign$ of Example 4.4, $Sign^+$ includes a new element denoting the integer interval $[1, 9]$.



Denote by ρ_s^+ the closure associated with $Sign^+$ (i.e. such that $\rho_s^+(\wp(\mathbb{Z})) \cong Sign^+$). It is not difficult to check that $Sign^+$ is fully complete for $\lambda X. \{n\} * X$ iff $n \notin [2, 9]$: In fact, for any $n \in [2, 9]$, there exists $X \subseteq [1, 9] \subset \mathbb{Z}$ such that $\rho_s^+(\{n\} * X) \subset \rho_s^+(\{n\} * \rho_s^+(X))$ (e.g., if $n = 2$ then $X = [1, 2]$, and if $n = 9$ then $X = [1, 1]$). It is then easily seen 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 $n \in [2, 9]$. What about binary multiplication $*$? Obviously, by the above considerations, $Sign^+$ is not fully complete for $*$, and indeed $\mathcal{K}_*^{\wp(\mathbb{Z})}(Sign^+) = Sign$. $\square$

5.2 The Least Fully Complete Extension Operator

Dually to what done above, we give the following definition.

Definition 5.6 Let $F \subseteq C^n \to C$ and $G \subseteq C \xrightarrow{\mathrm{m}} C$.

(i) $\mathcal{E}_F^C : uco(C) \to uco(C)$ is defined as $\mathcal{E}_F^C(X) = \sqcup\{Y \in uco(C) \mid Y \sqsubseteq X,\ Y \in \Gamma(C, F)\}$.

(ii) $\mathbb{E}_G^C : uco(C) \to uco(C)$ is defined as $\mathbb{E}_G^C(X) = \sqcup\{Y \in uco(C) \mid Y \sqsubseteq X,\ Y \in \Delta(C,G)\}$. $\square$

As before, whenever $F = \{f\}$ we simply use $\mathcal{E}_f^C$ and $\mathbb{E}_f^C$. Both $\mathcal{E}_F^C$ and $\mathbb{E}_G^C$ are abstract domain refinements in the sense of Filé et al. [1996], as the following straightforward fact states.

Lemma 5.7 *For any $F \subseteq C^n \to C$ and $G \subseteq C \xrightarrow{\text{m}} C$, $\mathcal{E}_F^C, \mathbb{E}_G^C \in lco(uco(C))$.*

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, i.e. when, for any $X \in uco(C)$, $\mathcal{E}_F^C(X)$ and $\mathbb{E}_G^C(X)$ are, respectively, the least fully complete for F and complete for G extensions of X. In such a 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 4.4 and 4.9, 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, with reference to Example 4.4, it is easily seen from the Hasse diagram of $uco(Sign)$ in Figure 1, 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 $\mathbb{E}_{sq}^{\wp(\mathbb{Z})}(\{\mathbb{Z}, -0\}) = \{\mathbb{Z}, -0\}$. Nevertheless, it is worth remarking that the least complete extension of a given abstract domain may equally exist, i.e. it may well happen that for some $X \in uco(C)$, $\mathbb{E}_G^C(X) \in \Delta(C,G)$ – indeed, this is the most frequent situation (e.g., we will demonstrate later in Section 7.3 that the least complete extension of an important abstract domain for logic program ground-dependency analysis indeed exists), since unfavorable cases are rather pathological.

By contrast, when F consists of continuous functions, we know, from Corollary 4.16, that for any $X \in uco(C)$, $\mathcal{E}_F^C(X)$ actually provides the least fully complete extension of X.

Proposition 5.8 *For any $F \subseteq C^n \xrightarrow{\text{c}} C$, $\mathcal{E}_F^C$ is safely defined.*

Example 5.9 Let us consider Example 4.4. Since sq and $*$ are obviously continuous, by Proposition 5.8, every abstract domain of $\wp(\mathbb{Z})$ admits the least fully complete extension for sq and $*$. For instance, $\mathcal{E}_{sq}^{\wp(\mathbb{Z})}(\{\mathbb{Z}, -0\}) = \{\mathbb{Z}, -0, 0\}$ and $\mathcal{E}_{*}^{\wp(\mathbb{Z})}(\{\mathbb{Z}, 0+, \emptyset\}) = \{\mathbb{Z}, 0+, 0, \emptyset\}$. $\square$

5.3 Algebraic Properties

In this section, we study the basic algebraic properties of the above operators $\mathcal{K}$, $\mathcal{E}$, $\mathbb{K}$, and $\mathbb{E}$ with respect to some well known operations for combining domains, like reduced product. These results will be particularly important in the following in order to simplify the construction of complete domains. Let us first give the following list of simple properties.

Proposition 5.10 *Let $F \subseteq C \xrightarrow{\text{m}} C$, $G \subseteq C^n \to C$, $H \subseteq C \xrightarrow{\text{c}} C$, and $X, Y \in uco(C)$.*

(i) $\mathcal{E}_F^C(X) \sqsubseteq \mathbb{E}_F^C(X) \sqsubseteq X \sqsubseteq \mathbb{K}_F^C(X) \sqsubseteq \mathcal{K}_F^C(X)$.

(ii) *If $X \sqsubseteq Y$ then, $Y \sqsubseteq \mathcal{K}_G^C(X) \Rightarrow \mathcal{K}_G^C(X) = \mathcal{K}_G^C(Y)$, and $\mathcal{E}_G^C(Y) \sqsubseteq X \Rightarrow \mathcal{E}_G^C(X) = \mathcal{E}_G^C(Y)$. Analogous statements hold for $\mathbb{K}_F^C$ and $\mathbb{E}_F^C$.*

(iii) $\mathcal{K}_F^C(\mathbb{K}_F^C(X)) = \mathcal{K}_F^C(X) = \mathbb{K}_F^C(\mathcal{K}_F^C(X))$ *and* $\mathcal{E}_F^C(\mathbb{E}_F^C(X)) = \mathcal{E}_F^C(X)$.

(iv) $\mathbb{E}_H^C(\mathcal{E}_H^C(X)) = \mathcal{E}_H^C(X)$.

Proof. (i) By Proposition 4.3.
(ii) By Lemma 5.2, from $X \sqsubseteq Y \sqsubseteq \mathcal{K}_G^C(X)$, by applying $\mathcal{K}_G^C$, we get $\mathcal{K}_G^C(X) \sqsubseteq \mathcal{K}_G^C(Y) \sqsubseteq \mathcal{K}_G^C(X)$. The other statements can be similarly proved.
(iii) By point (i), $X \sqsubseteq \mathbb{K}_F^C(X) \sqsubseteq \mathcal{K}_F^C(X)$, and hence, by point (ii), $\mathcal{K}_F^C(\mathbb{K}_F^C(X)) = \mathcal{K}_F^C(X)$.

Moreover, by Propositions 4.3 and 5.3, $\mathcal{K}_F^C(X) \in \Delta(C, F)$, and therefore $\mathcal{K}_F^C(X) = \mathbb{K}_F^C(\mathcal{K}_F^C)$. The remaining statement and point (iv) can be similarly proved. ∎

Intuitively, the operator $\mathcal{K}_F^C$ acts "inversely" to $\mathcal{E}_F^C$, and the same holds for the pair $\mathbb{K}_G^C$ and $\mathbb{E}_G^C$. This intuition is made precise by the following result proving that these two pairs of functions form an adjunction on the lattice of abstract interpretations.

Lemma 5.11 *Let $F \subseteq C^n \to C$ and $G \subseteq C \xrightarrow{\text{m}} C$. Then both $(\mathcal{K}_F^C, uco(C), uco(C), \mathcal{E}_F^C)$ and $(\mathbb{K}_G^C, uco(C), uco(C), \mathbb{E}_G^C)$ are Galois connections.*

Proof. By Lemmata 5.2 and 5.7, $\mathcal{K}_F^C$ and $\mathcal{E}_F^C$ are monotone, and, by definition, for any $X \in uco(C)$, $X \sqsubseteq \mathcal{K}_F^C(X) = \mathcal{E}_F^C(\mathcal{K}_F^C(X))$ and $\mathcal{K}_F^C(\mathcal{E}_F^C(X)) = \mathcal{E}_F^C(X) \sqsubseteq X$. Analogous observations hold for $\mathbb{K}_G^C$ and $\mathbb{E}_G^C$. ∎

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

Proposition 5.12 *Let $F \subseteq C^n \to C$ and $G \subseteq C \xrightarrow{\text{m}} C$.*

(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 and $X, Y \in uco(C)$ with $X \sqsubseteq Y$, then:*

- $\mathcal{E}_F^C(X) = \mathcal{E}_F^C(X \sim Y) \sqcap \mathcal{E}_F^C(Y)$;
- $\mathbb{E}_G^C(X) = \mathbb{E}_G^C(X \sim Y) \sqcap \mathbb{E}_G^C(Y)$.

Proof. (i): Because left- and right-adjoints are, respectively, additive and co-additive.
(ii): By meet-continuity of C, $X \sim Y$ exists. Since $(X \sim Y) \sqcap Y = X$, the result follows by (i). ∎

In particular, by point (i), the task of computing the least (fully) complete extension of a domain X can always be accomplished by independently computing the least (fully) complete extensions of simpler domains, which represent a decomposition of X. This decomposition can be obtained by complementation, and point (ii) above is an example of this idea. 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 an abstract domain can be verified. This because the concrete domain may be far too concrete for this task, i.e. it may contain too many objects. In the following, we show that the task of refining or simplifying a domain A abstracting C by our operators $\mathcal{E}$, $\mathbb{E}$, $\mathcal{K}$, and $\mathbb{K}$ can always be done, with the same result, on any (fully) complete abstraction D of C, which is more concrete than A. As we will see later, this is an important property in order to make easier the design of complete domains for abstract interpretation.

From what seen at the beginning of Section 4, we know that for a given domain A abstracting C and a concrete function $f : C^n \to C$ (when needed, it is assumed $n = 1$), the best correct approximation of f for A is $f^{best_A} : A^n \to A$ defined as $f^{best_A} = \rho_A \circ f$, where, ρ_A is the closure associated with A and, in the right hand side, f is thought of restricted to A^n. If F is a set of functions, then we denote by F^{best_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^{best_A}, instead of, respectively, C and f. Note that when f is monotone then f^{best_A} is monotone as well. Thus, for $F \subseteq C^n \to C$ and $G \subseteq C \xrightarrow{\text{m}} C$, and for any $A \in uco(C)$, we can define the operators of Definitions 5.1 and 5.6 relatively to $\Gamma(A, F^{best_A})$ and $\Delta(A, G^{best_A})$. All these operators have type $uco(A) \to uco(A)$, and we denote them 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 indeed considered.

By Proposition 5.3, we have that $\mathbb{K}_G^A$ is always a complete kernel operator, and for $F \subseteq C^n \xrightarrow{\text{m}} C$, $\mathcal{K}_G^A$ is a fully complete kernel operator. Moreover, the following fact holds.

Lemma 5.13 *If $f : C^n \xrightarrow{\text{c}} C$ and $A \in \Gamma(C, f)$, then $f^{best_A} : A^n \xrightarrow{\text{c}} A$.*

Proof. For simplicity of notation, we consider $n = 1$ (the general case is a straightforward extension). Let $A = \eta(C)$, for $\eta \in uco(C)$, and $Y \subseteq \eta(C)$ be a chain. Therefore, $f^{best_A} = f^{best_\eta} = \eta \circ f$. Then, we have:

$$\begin{array}{rcll} f^{best_\eta}(\vee_{\eta(C)} Y) & = & \eta(f(\vee_{\eta(C)} Y)) & \\ & = & \eta(f(\eta(\vee Y))) & \text{(by definition of the } lub \vee_{\eta(C)}) \\ & = & \eta(f(\vee Y)) & \text{(by full completeness of } \eta) \\ & = & \eta(\vee f(Y)) & \text{(by continuity of } f \text{ on } C) \\ & = & \eta(\vee \eta(f(Y))) & \text{(by (i) in Section 2)} \\ & = & \vee_{\eta(C)} \eta(f(Y)) & \text{(by definition of the } lub \vee_{\eta(C)}) \\ & = & \vee_{\eta(C)} f^{best_\eta}(Y), & \end{array}$$

and therefore f^{best_η} is continuous on $\eta(C)$. ∎

Thus, if $F \subseteq C^n \xrightarrow{c} C$ and A is fully complete for F, then, by Proposition 5.8, $\mathcal{E}_F^A$ is a safely defined least fully complete extension operator.

The following important characterization holds.

Theorem 5.14 *Let $F \subseteq C^n \to C$, $G \subseteq C \xrightarrow{m} C$, and $A \in uco(C)$.*

(i) $A \in \Gamma(C, F)$ *iff* $\forall X \in \uparrow A.\ \mathcal{K}_F^C(X) = \mathcal{K}_F^A(X)$ *iff* $\forall X \in \uparrow A.\ \mathcal{E}_F^C(X) = \mathcal{E}_F^A(X)$.

(ii) $A \in \Delta(C, G)$ *iff* $\forall X \in \uparrow A.\ \mathbb{K}_G^C(X) = \mathbb{K}_G^A(X)$ *iff* $\forall X \in \uparrow A.\ \mathbb{E}_G^C(X) = \mathbb{E}_G^A(X)$.

Proof. Let $\eta \in uco(C)$ such that $A = \eta(C)$.
(i): ($\Rightarrow$) Let $f \in F$. Then,

$$\begin{array}{rcll} \Gamma(\eta(C), f^{best_\eta}) & = & \{\psi \in uco(\eta(C)) \mid \psi \circ \eta \circ f =_{\eta(C)^n} \psi \circ \eta \circ f \circ \psi\} & \\ & = & \{\psi \in uco(\eta(C)) \mid \psi \circ \eta \circ f \circ \eta = \psi \circ \eta \circ f \circ \psi \circ \eta\} & \text{(since } \eta \circ f \circ \eta = \eta \circ f) \\ & = & \{\psi \in uco(\eta(C)) \mid \psi \circ \eta \circ f = \psi \circ \eta \circ f \circ \psi \circ \eta\} & \text{(since } \psi \circ \eta = \psi) \\ & = & \{\psi \in uco(\eta(C)) \mid \psi \circ f = \psi \circ f \circ \psi\} & \\ & = & \{\psi \in uco(C) \mid \eta \sqsubseteq \psi,\ \psi \in \Gamma(C, f)\}. & \end{array}$$

Thus, for any $\rho \in uco(\eta(C))$,

$$\begin{array}{rcl} \mathcal{K}_F^\eta(\rho) & = & \sqcap\{\psi \in uco(\eta(C)) \mid \rho \sqsubseteq \psi,\ \psi \in \Gamma(\eta(C), F^{best_\eta})\} \\ & = & \sqcap\{\psi \in uco(C) \mid \rho \sqsubseteq \psi,\ \psi \in \Gamma(C, F)\} \\ & = & \mathcal{K}_F^C(\rho), \\ \mathcal{E}_F^\eta(\rho) & = & \sqcup\{\psi \in uco(\eta(C)) \mid \psi \sqsubseteq \rho,\ \psi \in \Gamma(\eta(C), F^{best_\eta})\} \\ & = & \sqcup\{\psi \in uco(C) \mid \eta \sqsubseteq \psi \sqsubseteq \rho,\ \psi \in \Gamma(C, F)\}. \end{array}$$

Hence, since $\eta \in \Gamma(C, F)$, we also get that $\mathcal{E}_F^\eta(\rho) = \mathcal{E}_F^C(\rho)$.
($\Leftarrow$) By Proposition 4.5 (i), $\mathcal{K}_F^\eta(\eta) = \eta$, and therefore $\mathcal{K}_F^C(\eta) = \eta$, i.e. $\eta \in \Gamma(C, F)$.
(ii): We have already observed that $G^{best_\eta} \subseteq \eta(C) \xrightarrow{m} \eta(C)$, and therefore $\mathbb{K}_G^\eta$ and $\mathbb{K}_G^\eta$ are both defined.
($\Rightarrow$) For any $f \in G$, we have that

$$\begin{array}{rcll} \Delta(\eta(C), f^{best_\eta}) & = & \{\psi \in uco(\eta(C)) \mid \psi(lfp(f^{best_\eta})) = lfp(\psi \circ f^{best_\eta})\} & \\ & = & \{\psi \in uco(\eta(C)) \mid \psi(lfp(\eta \circ f)) = lfp(\psi \circ \eta \circ f)\} & \text{(since } \eta \in \Delta(C, f)) \\ & = & \{\psi \in uco(\eta(C)) \mid \psi(\eta(lfp(f))) = lfp(\psi \circ \eta \circ f)\} & \text{(since } \psi \circ \eta = \psi) \\ & = & \{\psi \in uco(\eta(C)) \mid \psi(lfp(f)) = lfp(\psi \circ f)\} & \\ & = & \{\psi \in uco(C) \mid \eta \sqsubseteq \psi,\ \psi \in \Delta(C, f)\}. & \end{array}$$

Thus, analogously to the previous point, for any $\rho \in uco(\eta(C))$, we get $\mathbb{K}_G^\eta(\rho) = \mathbb{K}_G^C(\rho)$ and $\mathbb{E}_G^\eta(\rho) = \mathbb{E}_G^C(\rho)$.
($\Leftarrow$) By Proposition 4.5 (i), $\mathbb{K}_G^\eta(\eta) = \eta$, and therefore $\mathbb{K}_G^C(\eta) = \eta$, i.e. $\eta \in \Delta(C, G)$. ∎

Let us consider the case of $\mathcal{E}^C_F$ (for the other operators, analogous considerations apply). In general, for a given $X \in \uparrow A$, it should be easier to calculate $\mathcal{E}^A_F(X)$ than $\mathcal{E}^C_F(X)$, since both A and F^{best_A} might be considerably less complex than C and F – it could even happen that $\mathcal{E}^A_F(X)$ is decidable whereas $\mathcal{E}^C_F(X)$ is not (this could be the case whenever A is finite and C is not). Thus, by the above theorem, knowing in advance that some abstract domain A is fully complete may simplify a lot the task of computing a least fully complete extension $\mathcal{E}^C_F(X)$ of an abstraction X of A. The following example allows us to observe this phenomenon, and shows how to practically exploit the properties in Proposition 5.12 to design the least fully complete extension of Mycroft's domain $Sign^+$ for integer multiplication.

Example 5.15 Consider the domain $Sign^+$ in Example 5.5. We constructively define the least fully complete extension of $Sign^+$ for integer multiplication. Firstly observe that if X is some set and $\cdot : X \times X \to X$ is a commutative binary operation on X then $\odot : \wp(X) \times \wp(X) \to \wp(X)$ defined by $A \odot B = \{a \cdot b \in X \mid a \in A,\ b \in B\}$ is also commutative (and, obviously, monotone). This step of lifting an operation to the powerset is typical in collecting semantics (cf. [Cousot and Cousot 1977; Mycroft and Nielson 1983]). In this case, we can go beyond Proposition 4.5 (viii), and prove the following result.

Proposition 5.16 $\Gamma(\wp(X), \odot) = \Gamma(\wp(X), \{\lambda A. \{x\} \odot A : \wp(X) \to \wp(X)\}_{x \in X})$.

Proof. $\Gamma(\wp(X), \odot) \subseteq \Gamma(\wp(X), \{\lambda A. \{x\} \odot A : \wp(X) \to \wp(X)\}_{x \in X})$ trivially holds. Let $A, B \subseteq X$ and $\rho \in \Gamma(\wp(X), \{\lambda A. \{x\} \odot A : \wp(X) \to \wp(X)\}_{x \in X})$. Then, by exploiting Proposition 4.5 (iii) and point (ii) in Section 2, one gets the following equalities.

$$\begin{array}{rcl}
\rho(\rho(A) \odot \rho(B)) & = & \rho(\cup_{a \in \rho(A)} \{a\} \odot \rho(B)) \\
& = & \rho(\cup_{a \in \rho(A)} \rho(\{a\} \odot \rho(B))) \\
& = & \rho(\cup_{a \in \rho(A)} \rho(\{a\} \odot B)) \\
& = & \rho(\cup_{a \in \rho(A)} \{a\} \odot B) \\
& = & \rho(\rho(A) \odot B) \\
& = & \rho(B \odot \rho(A)) \\
& = & \rho(\cup_{b \in B} \{b\} \odot \rho(A)) \\
& = & \rho(\cup_{b \in B} \rho(\{b\} \odot \rho(A))) \\
& = & \rho(\cup_{b \in B} \rho(\{b\} \odot A)) \\
& = & \rho(\cup_{b \in B} \{b\} \odot A) \\
& = & \rho(B \odot A) \\
& = & \rho(A \odot B).
\end{array}$$

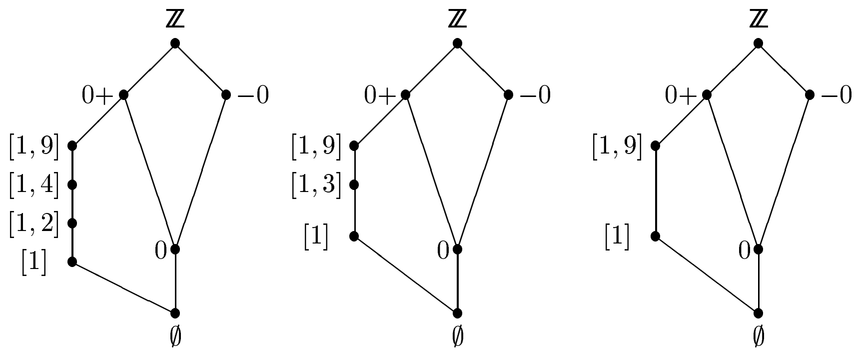
Therefore, $\rho \in \Gamma(\wp(X), \odot)$. ■

Thus, as far as integer multiplication is concerned, it is sufficient to compute the fully complete extension of $Sign^+$ for any $f_n = \lambda X. \{n\} * X$ with $n \in \mathbb{Z}$. As we observed in Example 5.5, $Sign^+$ is fully complete for $f_n = \lambda X. \{n\} * X$ (which is obviously continuous) iff $n \notin [2, 9]$. Consider then any $n \in [2, 9]$. Because, as shown in Example 5.5, $\mathcal{K}^{\wp(\mathbb{Z})}_{f_n}(Sign^+) = Sign$ and $Sign^+ \sim Sign = \{\mathbb{Z}, [1, 9]\}$, then, by applying Proposition 5.12 (ii), we get

$$\mathcal{E}^{\wp(\mathbb{Z})}_{f_n}(Sign^+) = \mathcal{E}^{\wp(\mathbb{Z})}_{f_n}(Sign^+ \sim \mathcal{K}^{\wp(\mathbb{Z})}_{f_n}(Sign^+)) \sqcap \mathcal{E}^{\wp(\mathbb{Z})}_{f_n}(\mathcal{K}^{\wp(\mathbb{Z})}_{f_n}(Sign^+)) = \mathcal{E}^{\wp(\mathbb{Z})}_{f_n}(\{\mathbb{Z}, [1, 9]\}) \sqcap Sign.$$

We compute the least fully complete extension $\mathcal{E}^{\wp(\mathbb{Z})}_{f_n}(\{\mathbb{Z}, [1, 9]\})$ of the atomic domain $\{\mathbb{Z}, [1, 9]\}$ by a recursive computation, as shown by the next proposition. Here, we exploit Theorem 5.14 (i), because we consider as concrete domain the more abstract (and simple) domain of integer intervals *Int* (cf. [Cousot and Cousot 1977]), which is easily observed to be fully complete for binary integer multiplication. The abstract domain *Int* is defined as:

$$\begin{array}{rcl}
Int & = & \{[a, b] \mid a, b \in \mathbb{Z}, a \le b\} \cup \{[-\infty, b] \mid b \in \mathbb{Z}\} \cup \\
& & \{[a, +\infty] \mid a \in \mathbb{Z}\} \cup \{[-\infty, +\infty]\} \cup \{\bot\}.
\end{array}$$



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

Figure 3: Least fully complete extensions of $Sign^+$ with $n \in [2, 9]$.

Obviously, objects of *Int* having as extremes $-\infty$ and/or $+\infty$ represent infinite intervals. It turns out that $\langle Int, \subseteq \rangle$ is a complete lattice, and, being closed under set-intersection, *Int* is an abstraction of $\wp(\mathbb{Z})$. The closure operator $\eta_{Int} \in uco(\wp(\mathbb{Z}))$ such that $Int \cong \eta_{Int}(\wp(\mathbb{Z}))$ is then defined as follows: For any $X \in \wp(\mathbb{Z})$, $\eta_{Int}(X) = [min(X), max(X)]$, where *min* and *max* are the *glb* and *lub* on the complete lattice $\mathbb{Z} \cup \{-\infty, +\infty\}$.

Proposition 5.17 $Int \in \Gamma(\wp(\mathbb{Z}), *)$.

Proof. By contradiction, assume that $\eta_{Int} \notin \Gamma(\wp(\mathbb{Z}), *)$. Then, by Proposition 5.16, there exist $n \in \mathbb{Z}$ and $X \in \wp(\mathbb{Z})$ such that $\eta_{Int}(\{n\} * X) \subset \eta_{Int}([n, n] * \eta_{Int}(X))$. For $n = 0$ obviously we get a contradiction. So, let $n \neq 0$, and assume that $-\infty < min(X)$ and $max(X) < +\infty$. This implies that there exists $a \in [min(X), max(X)]$ such that either $n * a < n * x$ or $n * x < n * a$ for any $x \in X$. Hence, for any $x \in X$, either $a < x$ or $x < a$, in both cases a contradiction. If $min(X) = -\infty$ and/or $max(X) = +\infty$, we similarly get a contradiction. ∎

Therefore, for any $n \in \mathbb{Z}$, $Int \in \Gamma(\wp(\mathbb{Z}), f_n)$. Thus, by Theorem 5.14 (i), for any $n \in \mathbb{Z}$ and X such that $Int \sqsubseteq X$, $\mathcal{E}^{\wp(\mathbb{Z})}_{fn}(X) = \mathcal{E}^{Int}_{fn}(X)$. The following proposition characterizes the least fully complete extension of any atomic domain in $uco(Int)$ of the form $\{\mathbb{Z}, [1, m]\}$. We can assume w.l.o.g. 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 5.18 $\mathcal{E}^{Int}_{fn}(\{\mathbb{Z}, [1, m]\}) = \{\mathbb{Z}, [1, m]\} \sqcap \mathcal{E}^{Int}_{fn}(\{\mathbb{Z}, [1, \lfloor \frac{m}{n} \rfloor]\})$.

Proof. First observe that $\mathcal{E}^{Int}_{fn}(\{\mathbb{Z}, [1, 1]\}) = \{\mathbb{Z}, [1, 1]\}$. Note that, by definition of reduced product and by a straightforward inductive argument, for any $m \geq 2$ and $n \in [2, m]$, the above recursive definition defines a finite chain $\{\mathbb{Z}, [1, m], [1, \lfloor \frac{m}{n} \rfloor], [1, \lfloor \frac{m}{n^2} \rfloor], \ldots, [1, \lfloor \frac{m}{n^{k-1}} \rfloor], [1, 1]\}$ such that $\lfloor \frac{m}{n^k} \rfloor = 1$ for some $k \in \mathbb{N}$. Let us denote by X such a domain. Thus, $X = \{\mathbb{Z}\} \cup \{[1, \lfloor \frac{m}{n^h} \rfloor]\}_{0 \leq h \leq k}$. Clearly, being closed by set intersection, X is a Moore family of *Int*, i.e. $X \in uco(Int)$. We first prove that for any $n \in [2, m]$, $X \in \Gamma(Int, f_n)$. Assume, by contradiction, that $X \notin \Gamma(Int, f_n)$ for some $n \in [2, m]$. Then, there exists $I \in Int$ such that, by denoting ρ_X the closure operator such that $X = \rho_X(Int)$, $\rho_X(\{n\} * I) \subset \rho_X(\{n\} * \rho_X(I))$. If $\rho_X(I) = \mathbb{Z}$ or $\rho_X(I) = [1, m]$ then we get a contradiction in both cases because $max(I) > \lfloor \frac{m}{n} \rfloor$, and therefore $\rho_X(\{n\} * I) = \rho_X(\{n\} * \rho_X(I)) = \mathbb{Z}$. Thus, assume that $\rho_X(I) \subset [1, m]$, i.e. $I = [l, u]$, with $1 \leq l \leq u < m$. Then, there exists $1 \leq h \leq k$ such that $\rho_X(I) = [1, \lfloor \frac{m}{n^h} \rfloor]$, and therefore $\rho_X(\{n\} * \rho_X(I)) = [1, \lfloor \frac{m}{n^{h-1}} \rfloor]$. Moreover, $\rho_X(\{n\} * I) = [1, \lfloor \frac{m}{n^h} \rfloor]$, because, by hypothesis, the upper bound u' of the interval $\{n\} * I$ is such that $u \leq u' < \lfloor \frac{m}{n^{h-1}} \rfloor$. This implies that $n * u \leq \lfloor \frac{m}{n^h} \rfloor$, which in turn implies that $u \leq \lfloor \frac{m}{n^{h+1}} \rfloor$. But this is again a contradiction because this implies that

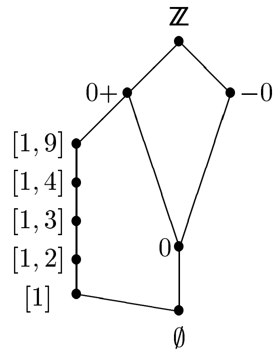


Figure 4: $\mathcal{E}_*^{\wp(\mathbb{Z})}(Sign^+)$.

$\rho_X(I) \subset [1, \lfloor \frac{m}{n^h} \rfloor]$. The proof ends by observing that, for any $x \in X \setminus \{\mathbb{Z}\}$, $X \setminus \{x\}$ is not fully complete for f_n: To verify this, it is sufficient to consider, for any $x = [1, \lfloor \frac{m}{n^h} \rfloor]$ with $0 \leq h \leq k$, the unit interval $I = [\lfloor \frac{m}{n^h} \rfloor]$. ■

The above recursive definition has a solution which can be obtained finitely for any finite $n \leq m$. In the case of $Sign^+$, i.e. $m = 9$, it provides the domains depicted in Figure 3, which are the least fully complete extensions of $Sign^+$ for $n = 2, 3$ and $5 \leq n \leq 9$. In particular, if $F = \{f_n \mid n \in \mathbb{Z}\}$ then $\mathcal{E}_F^{\wp(\mathbb{Z})}(Sign^+) = \mathcal{E}_{\{f_2, f_3\}}^{Int}(Sign^+)$, which is the domain depicted in Figure 4, and, by Proposition 5.16, $\mathcal{E}_*^{\wp(\mathbb{Z})}(Sign^+) = \mathcal{E}_F^{\wp(\mathbb{Z})}(Sign^+)$. □

6 Systematic Construction of Complete Domains

In the previous section we have shown a constructive method to design the least fully complete extension of a simple domain for sign analysis with respect to integer multiplication. However, due to the extremely weak hypotheses of Theorems 4.6 and 4.11, it is in general a hard task to develop a similar constructive method to design fully complete extensions and kernels of generic domains and semantic functions. This is even harder when dealing with the weaker notions of (nonfully) complete kernels and extensions. In this section, we show that under some hypotheses on the semantic functions, a similar constructive methodology can be applied to construct the least fully complete extensions and kernels of abstract domains.

6.1 Constructing Least Fully Complete Extensions

Let us concentrate on full completeness for a single function $f : C \xrightarrow{c} C$. The extension to sets of functions will be straightforward. The following result gives a useful characterization of full completeness for additive functions.

Theorem 6.1 *If $f : C \xrightarrow{a} C$ and $X \in uco(C)$ then, $X \in \Gamma(C, f) \Leftrightarrow \forall x \in X. \vee \{y \in C \mid f(y) \leq x\} \in X$.*

Proof. Let $\rho \in \Gamma(C, f)$ such that $X = \rho(C)$.
($\Rightarrow$) Let $x \in \rho(C)$. We prove that $\rho(\vee\{y \in C \mid f(y) \leq x\}) = \vee\{y \in C \mid f(y) \leq x\}$. Because ρ is extensive, $\rho(\vee\{y \in C \mid f(y) \leq x\}) \geq \vee\{y \in C \mid f(y) \leq x\}$. Moreover, from the additivity of f follows that $f(\vee\{y \in C \mid f(y) \leq x\}) \leq x$. Thus, because $\rho \in \Gamma(C, f)$, we have that $\rho(f(\rho(\vee\{y \in C \mid f(y) \leq x\}))) = \rho(f(\vee\{y \in C \mid f(y) \leq x\})) \leq \rho(x) = x$. Hence, we obtain $f(\rho(\vee\{y \in C \mid f(y) \leq x\})) \leq x$, which implies that $\rho(\vee\{y \in C \mid f(y) \leq x\}) \in \{y \in C \mid f(y) \leq x\}$, namely $\rho(\vee\{y \in C \mid f(y) \leq x\}) \leq \vee\{y \in C \mid f(y) \leq x\}$.

($\Leftarrow$) Assume that for all $x \in \rho(C)$, $\vee\{y \in C \mid f(y) \leq x\} \in \rho(C)$, and suppose that $\rho \notin \Gamma(C,f)$. Then, by definition, there exists $z \in C$ such that $\rho(f(z)) < \rho(f(\rho(z)))$. Because $\rho(f(z)) \in \rho(C)$, then, by hypothesis, $\vee\{y \in C \mid f(y) \leq \rho(f(z))\} \in \rho(C)$, namely $\rho(\vee\{y \in C \mid f(y) \leq \rho(f(z))\}) = \vee\{y \in C \mid f(y) \leq \rho(f(z))\}$. Note that, because $f(z) \leq \rho(f(z))$ then $z \in \{y \in C \mid f(y) \leq \rho(f(z))\}$. Therefore, $\rho(z) \leq \rho(\vee\{y \in C \mid f(y) \leq \rho(f(z))\})$. Moreover, by additivity of f, $f(\vee\{y \in C \mid f(y) \leq \rho(f(z))\}) \leq \rho(f(z))$. From the above relations we get the following inequalities:

$$\begin{array}{rcl}\rho(f(\rho(z))) & \leq & \rho(f(\rho(\vee\{y \in C \mid f(y) \leq \rho(f(z))\}))) \\ & = & \rho(f(\vee\{y \in C \mid f(y) \leq \rho(f(z))\})) \\ & \leq & \rho(\rho(f(z))) \\ & = & \rho(f(z)).\end{array}$$

Thus, we get $\rho(f(z)) = \rho(f(\rho(z)))$, which is a contradiction. ∎

Although additivity is a much more restrictive condition than continuity, a number of important program analyses satisfy this condition. For instance, liveness analysis of first-order functional languages of Nielson and Nielson [1989] gives rise to additive semantic functions, as well as all the backward analyses formulated in terms of concrete context domains and context functions (basically additive functions) of Hughes [1988]. This is also the case of abstract and concrete unification and projection in ground-dependency analysis of logic languages [Armstrong et al. 1997; Cortesi et al. 1996; Marriott and Søndergaard 1993], and of dependency analysis for imperative languages of Nielson and Nielson [1992]. Moreover, in the field of comparative semantics for logic programs, all the operators for building SLD-trees of Comini and Levi [1994] are additive. [Nielson and Nielson 1992] thoroughly discuss this issue of additivity for semantic functions, and argue that many interesting program analyses satisfy this condition.

By Theorem 6.1, whenever f is additive, $\mathcal{E}_f^C(X)$ is the least Moore-family Y such that $X \subseteq Y$ and $x \in Y \Rightarrow \vee\{y \mid f(y) \leq x\} \in Y$ holds. This domain can be constructively obtained as the greatest (viz. most abstract) solution in $uco(C)$, i.e. the greatest fixpoint solution, of the recursive abstract domain equation

$$Y = X \sqcap \mathcal{C}_f(Y), \qquad (*)$$

where $\mathcal{C}_f = \lambda X.\, \mathcal{M}(\{\vee\{y \in C \mid f(y) \leq x\} \mid x \in X\})$. The following result is immediate by definition.

Lemma 6.2 *$\mathcal{C}_f : uco(C) \to uco(C)$ is monotone.*

Therefore, $\mathcal{E}_f^C(X)$ is the limit of the (possibly transfinite) iteration sequence $X^0 = \{\top\}$, $X^{\alpha+1} = X \sqcap \mathcal{C}_f(X^\alpha)$, and for α limit ordinal, $X^\alpha = \sqcap_{\delta<\alpha} X^\delta$, i.e. there exists some (possibly limit) ordinal $\delta \in Ord$ such that $\mathcal{E}_f^C(X) = X^\delta = (\lambda Y.\, X \sqcap \mathcal{C}_f(Y))^{\downarrow\delta}(\{\top\})$. In the following of this section $\{X^\alpha\}_{\alpha \in Ord}$ denotes the above transfinite sequence for X. Obviously, if C is finite, then the above constructive method becomes effective.

Theorem 6.3 *If $f : C \xrightarrow{a} C$ and $X \in uco(C)$ then there exists $\delta \in Ord$ such that $\mathcal{E}_f^C(X) = X^\delta$.*

Proof. Because $f : C \xrightarrow{a} C$, then f is continuous as well, and therefore $\mathcal{E}_f^C(X)$ is safely defined. Thus, by Theorem 6.1, $\mathcal{E}_f^C(X) = \sqcup\{Y \in uco(C) \mid Y \sqsubseteq X,\ \mathcal{C}_f(Y) \subseteq Y\} = \sqcup\{Y \in uco(C) \mid Y \sqsubseteq X,\ Y \sqsubseteq \mathcal{C}_f(Y)\}$. But this is nothing else than the greatest fixpoint of $\mathcal{C}_f$ smaller than or equal to X, i.e. being $\mathcal{C}_f$ monotone on $uco(C)$ by Lemma 6.2, $\mathcal{E}_f^C(X) = (\lambda Y.\, X \sqcap \mathcal{C}_f(Y))^{\downarrow\delta}(\{\top\})$ for some $\delta \in Ord$. ∎

The extension of the above constructive method to a set of additive functions F is straightforward. In fact, by Theorem 6.1, for any $X \in uco(C)$, $X \in \Gamma(C,F) \Leftrightarrow \forall x \in X. \forall f \in F. \vee\{y \in C \mid f(y) \leq x\} \in X$. Thus, $\mathcal{E}_F^C(X)$ simply is the greatest fixpoint solution in $uco(C)$ of the equation $Y = X \sqcap \mathcal{C}_F(Y)$, where $\mathcal{C}_F = \lambda X.\, \mathcal{M}(\{\vee\{y \in C \mid f(y) \leq x\} \mid x \in X, f \in F\})$.

As observed in Proposition 5.12 (i), it is always possible to compute the least fully complete extension of a domain A in a compositional fashion, as reduced product of the least fully complete extensions of the factors in a decomposition of A. Clearly, the best decompositions for this task are those having the least number of factors and involving the most abstract domains as factors. In Section 2, we have recalled that, for any concrete domain C, the lattice of abstract interpretations $uco(C)$ is always dual-atomic, where a (dual-)atomic domain is $\{x, \top\}$, for any $x \in C \setminus \{\top\}$. Thus, for any abstract domain there exists a canonical decomposition, involving atomic domains only, which we call *atomic decomposition*: If $X \in uco(C)$ then $X = \sqcap_{x \in X \setminus \{\top\}} \{\top, x\}$, and therefore $\langle \{\top, x\} \rangle_{x \in X \setminus \{\top\}}$ is a decomposition of X. However, it may well happen that some atomic domains in an atomic decomposition can be redundant. This happens when x is meet-reducible in X, i.e. there exists $Y \subseteq X$ such that $x = \wedge Y$ and $x \notin Y$. We say that a decomposition $\langle D_i \rangle_{i \in I}$ for an abstract domain X is *irredundant* when for any $j \in I$, $X \neq \sqcap_{i \in I \setminus \{j\}} D_i$. A simple way to obtain irredundant atomic decompositions of abstract domains is by restricting ourself to algebraic complete lattices C as concrete domains (cf. [Cortesi et al. 1997]). It is worth noting that algebraic lattices are weak enough to include most relevant domains typically used for programming language semantics (see e.g. [Abramsky and Jung 1994]). Any algebraic complete lattice C is meet-generated by its meet-irreducible elements, i.e. $\forall x \in C.\ x = \wedge(\uparrow x \cap MI(C))$ (cf. e.g. [Gierz et al. 1980]). If C is algebraic then in the above atomic decomposition it suffices to consider atomic domains indexed by meet-irreducible elements only: In fact, it turns out that $\langle \{\top, x\} \rangle_{x \in MI(X)}$ is an atomic irredundant decomposition of X. Notice that, since X is a complete meet subsemilattice of C, $MI(X) = X \cap MI(C)$ and $\top \notin MI(X)$.

By the above observations and Proposition 5.12 (i), we get that for any $X \in uco(C)$ and $F \subseteq C \xrightarrow{c} C$, $\mathcal{E}_F^C(X) = \sqcap_{x \in X \setminus \{\top\}} \mathcal{E}_F^C(\{\top, x\})$, and for C algebraic, $\mathcal{E}_F^C(X) = \sqcap_{x \in MI(X)} \mathcal{E}_F^C(\{\top, x\})$. Thus, the problem of designing the least fully complete extension of a domain can be reduced to the problem of designing the least fully complete extensions of all the atomic domains in its atomic decomposition. This task can be made simpler when, in addition to additivity, the semantic functions are extensive.

Lemma 6.4 *Let $f : C \xrightarrow{a} C$ be extensive and $X = \{\top, x\}$ be an atomic domain for some $x \in C \setminus \{\top\}$. Then, for any $\alpha \in Ord$, X^α is a decreasing chain such that for any $y \in X^\alpha \setminus \{\top\}$, $y \leq x$. Moreover, for any $n < \omega$, X^n is finite.*

Proof. We proceed by transfinite induction.

($\alpha = 0$): $X^0 = \{\top\}$ is a trivial (decreasing finite) chain.

($\alpha + 1$): If $\alpha = 0$ then $X^1 = X = \{\top, x\}$ satisfies the thesis. Thus, let us assume that $\alpha > 0$. For each $z \in C$, let us define $c_z = \vee\{y \in X \mid f(y) \leq z\}$. We have that $X^{\alpha+1} = X \sqcap \mathcal{C}_f(X^\alpha) = \{\top, x\} \sqcap \mathcal{M}(\{c_z \mid z \in X^\alpha\})$. Notice that, by definition of reduced product, $\{\top, x\} \sqcap \mathcal{M}(\{c_z \mid z \in X^\alpha\}) = \mathcal{M}(\{c_z \mid z \in X^\alpha\}) \cup \{x \wedge y \mid y \in \mathcal{M}(\{c_z \mid z \in X^\alpha\})\}$. Since, by inductive hypothesis, X^α is a decreasing chain, then $\{c_z \mid z \in X^\alpha\}$ is a decreasing chain as well. Moreover, it is a routine task to verify that the Moore-closure of a chain in a complete lattice still is a chain. Thus, the Moore-closure $\mathcal{M}(\{c_z \mid z \in X^\alpha\})$ still is a decreasing chain. Since f is extensive, for any $z \in X^\alpha$, $c_z \leq z$. Thus, by inductive hypothesis, for any $z \in X^\alpha \setminus \{\top\}$, $c_z \leq x$. Hence, for any $w \in \mathcal{M}(\{c_z \mid z \in X^\alpha\}) \setminus \{\top\}$, $w \leq x$. Thus, we get that $X^{\alpha+1} = \{\top, x\} \cup \mathcal{M}(\{\vee\{z \in C \mid f(z) \leq y\} \mid y \in X^\alpha\})$. Hence, $X^{\alpha+1}$ is a decreasing chain satisfying the thesis. Moreover, note that if X^α is finite then $X^{\alpha+1}$ is finite as well.

(α limit ordinal): We have that $X^\alpha = \sqcap_{\beta < \alpha} X^\beta = \mathcal{M}(\cup_{\beta < \alpha} X^\beta)$. Since, by inductive hypothesis, for any $\beta < \alpha$, X^β is a decreasing chain, $\cup_{\beta < \alpha} X^\beta$ is a decreasing chain as well. Moreover, by inductive hypothesis, if $y \in (\cup_{\beta < \alpha} X^\beta) \setminus \{\top\}$, then $y \leq x$. Thus, $X^\alpha = \mathcal{M}(\cup_{\beta < \alpha} X^\beta)$ is a decreasing chain, and for any $y \in \mathcal{M}(\cup_{\beta < \alpha} X^\beta) \setminus \{\top\}$, $y \leq x$. ■

Thus, when F is a set of additive and extensive functions, for any $X \in uco(C)$, $\mathcal{E}_F^C(X)$ is the reduced product of the set of chains $\{\mathcal{E}_F^C(\{\top, x\})\}_{x \in X \setminus \{\top\}}$. Clearly, if C is algebraic, these chains can originate from meet-irreducible elements only.

Lemma 6.5 *Let $f : C \xrightarrow{a} C$ be extensive and $X = \{\top, x\}$ be an atomic domain for some $x \in C \setminus \{\top\}$. Then, for any $0 < n < \omega$, $X^{n+1} = X^n \cup \{\vee\{z \in C \mid f(z) \leq \wedge X^n\}\}$.*

Input : An atomic domain $\{\top, x\}$ such that $\langle \downarrow x, \leq \rangle$ is well-ordered
Output : The least fully complete extension E of $\{\top, x\}$ for f additive and extensive
 	
$E := \{\top, x\}$;
$y := x$;
while $\vee \{z \mid f(z) \leq y\} \neq y$ **do**
  $E := E \cup \{\vee\{z \mid f(z) \leq y\}\}$;
  $y := \vee\{z \mid f(z) \leq y\}$
endwhile

Table 1: An algorithm computing least fully complete extensions of atomic domains.

Proof. As in the above proof, for each $z \in C$, let us define $c_z = \vee\{y \in X \mid f(y) \leq z\}$. We proceed by induction.

($n = 1$): Note that $X^1 = X = \{\top, x\}$, and $c_\top = \top$. Thus, $X^2 = X \sqcap \mathcal{M}(\{c_z \mid z \in X^1\}) = X \sqcap \mathcal{M}(\{\top, c_x\})$. Moreover, by extensivity of f, $c_x \leq x$. Hence, $X^2 = \{\top, x, c_x\} = X^1 \cup \{c_{\wedge X^1}\}$.

($n + 1$): Observe that in the case successor of the proof of Lemma 6.4, we have shown that $X^{n+1} = \{\top, x\} \cup \mathcal{M}(\{c_z \mid z \in X^n\})$. Moreover, again by Lemma 6.4, X^n is a finite chain. Thus, $\{c_z \mid z \in X^n\}$ is a finite chain as well, and therefore $\mathcal{M}(\{c_z \mid z \in X^n\}) = \{c_z \mid z \in X^n\}$. Since $n > 0$, analogously we have that $X^n = \{\top, x\} \cup \{c_z \mid z \in X^{n-1}\}$. By inductive hypothesis, $X^{n+1} = \{\top, x\} \cup \{c_z \mid z \in X^{n-1}\} \cup \{c_{c_{\wedge X^{n-1}}}\}$, and therefore $X^{n+1} = X^n \cup \{c_{c_{\wedge X^{n-1}}}\}$. On the other hand, note that $\wedge X^n = c_{\wedge X^{n-1}}$, and therefore $c_{c_{\wedge X^{n-1}}} = c_{\wedge X^n}$. Hence, this shows that $X^{n+1} = X^n \cup \{c_{\wedge X^n}\}$. ∎

Thus, for an atomic domain $\{\top, x\}$ such that the principal ideal $\langle \downarrow x, \leq \rangle$ generated by x is well-ordered (i.e. it does not contain infinite descending chains), Lemma 6.4 suggests the iterative and convergent method in Table 1 for computing its least fully complete extension with respect to an additive and extensive function. The following result proves the correctness of the algorithm in Table 1.

Theorem 6.6 *Let $f : C \xrightarrow{a} C$ be extensive and $X = \{\top, x\} \in uco(C)$ be an atomic domain for some $x \in C \setminus \{\top\}$. Then, for every $n < \omega$, $X^n \in \Gamma(C, f) \Leftrightarrow \vee\{z \in C \mid f(z) \leq \wedge X^n\} \in X^n$.*

Proof. The thesis is trivially true for $n = 0$ because $X^0 = \{\top\}$. For $n = 1$, i.e. $X^1 = \{\top, x\}$, the thesis follows straight from Theorem 6.1. Thus, let us assume $n > 1$. One implication is immediate because, by definition, for any $n < \omega$, $X^n \in uco(C)$, and therefore $\wedge X^n \in X^n$. Hence, by Theorem 6.1, if $X^n \in \Gamma(C, f)$ then $\vee\{z \in C \mid f(z) \leq \wedge X^n\} \in X^n$. On the other hand, if $\vee\{z \in C \mid f(z) \leq \wedge X^n\} \in X^n$ then, by Lemma 6.5, $X^{n+1} = X^n$, i.e. X^n is a fixpoint of the equation $(*)$, and therefore $X^n \in \Gamma(C, f)$. ∎

The hypothesis of extensivity is not strictly necessary in order to effectively apply the algorithm in Table 1. In fact, it would not be too hard to see that in order to apply that algorithm, it is sufficient that, in every iteration, for any $y \in E$, $\vee\{z \mid f(z) \leq y\} \leq y$. The following example reconstructs the result in Example 5.15 as an instance of the more general method for constructing fully complete domains described above.

Example 6.7 Reconsider the problem of computing the least fully complete extension of Mycroft's domain Sign^+ for integer multiplication of Example 5.15. We observed that this can be solved by computing $\mathcal{E}^{Int}_{f_n}(\{\mathbb{Z}, [1, m]\})$ with $n \in [2, m]$, and Proposition 5.18 gives a constructive method to accomplish this task. Although f_n is not in general extensive, it is easy to show that Proposition 5.18 leads to a procedure which is indeed an instance of the algorithm in Table 1. This follows from the fact that, for any $m \geq 1$ and $n \in [2, m]$, $\vee\{I \in \mathit{Int} \mid n * I \subseteq [1, m]\} = [1, \lfloor \frac{m}{n} \rfloor]$

Input : A finite domain X
Output : The fully complete kernel K of X for f
 	
$K := \{\top\}$;
$A := X$;
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

Table 2: An algorithm computing fully complete kernels.

and $[1, \lfloor \frac{m}{n} \rfloor] \subseteq [1, m]$. Moreover, although *Int* is an infinite lattice, for each $m \geq 1$, $\langle \downarrow [1, m], \subseteq \rangle$ is a finite sublattice of $\langle Int, \subseteq \rangle$. Therefore, $\mathcal{E}_{fn}^{Int}(\{\mathbb{Z}, [1, m]\})$ in Example 5.15 can be computed in $\lfloor \frac{m}{n} \rfloor$ steps by the algorithm in Table 1. □

6.2 Constructing Fully Complete Kernels

The systematic construction of fully complete kernels relies on the algorithms for least fully complete extension of the preceding section. This because, by Lemma 5.11, $\mathcal{E}$ and $\mathcal{K}$ form an adjunction on the lattice of abstract interpretations, and one adjoint function uniquely determines the other. In this case, $\mathcal{K}_f^C(X) = \sqcap\{ Y \in uco(C) \mid X \sqsubseteq \mathcal{E}_f^C(Y)\}$.

The following lemma characterizes full completeness of an abstract domain X in terms of the least fully complete extensions of the factors in its atomic decomposition.

Lemma 6.8 *Let $f : C^n \xrightarrow{c} C$. Then, $X \in \Gamma(C,f)$ iff $\forall x \in X \setminus \{\top\}.\ \mathcal{E}_f^C(\{\top, x\}) \subseteq X$ iff $\forall x \in X \setminus \{\top\}.\ \{\top, x\} \subseteq \mathcal{K}_f^C(X)$.*

Proof. ($\Rightarrow$) Assume $X \in \Gamma(C,f)$ and let $x \in X \setminus \{\top\}$. Then, $\mathcal{E}_f^C(X) = X$. Because $X \sqsubseteq \{\top, x\}$, $\mathcal{E}_f^C(X) \sqsubseteq \mathcal{E}_f^C(\{\top, x\})$. Thus, $X \sqsubseteq \mathcal{E}_f^C(\{\top, x\})$, i.e. $\mathcal{E}_f^C(\{\top, x\}) \subseteq X$.
($\Leftarrow$) Assume that for all $x \in X \setminus \{\top\}$, $\mathcal{E}_f^C(\{\top, x\}) \subseteq X$. We know that $X = \sqcap_{x \in X \setminus \{\top\}} \{\top, x\}$. Because, by Proposition 5.12 (i), $\mathcal{E}$ is co-additive, $\mathcal{E}_f^C(X) = \sqcap_{x \in X \setminus \{\top\}} \mathcal{E}_f^C(\{\top, x\})$, and therefore $X \sqsubseteq \mathcal{E}_f^C(X)$. Moreover, $\mathcal{E}_f^C(X) \sqsubseteq X$ always holds. Thus, $X \in \Gamma(C,f)$. The rest of the proof follows by Lemma 5.11. ■

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 6.9 *Let $f : C^n \xrightarrow{c} C$. For any $X \in uco(C)$, $\mathcal{K}_f^C(X) = \sqcap\{\mathcal{E}_f^C(\{\top, x\}) \mid x \in X \setminus \{\top\},\ \mathcal{E}_f^C(\{\top, x\}) \subseteq X\}$.*

Proof. Let $K = \sqcap\{\mathcal{E}_f^C(\{\top, x\}) \mid x \in X \setminus \{\top\},\ \mathcal{E}_f^C(\{\top, x\}) \subseteq X\}$. Since $K \subseteq X$ and $K \in \Gamma(C,f)$, we have that $\mathcal{K}_f^C(X) \sqsubseteq K$. Assume, by contradiction, that $\mathcal{K}_f^C(X) \sqsubset K$. Then, there exists $y \in \mathcal{K}_f^C(X) \subseteq X$ but $y \notin K$ (note that $y \neq \top$). This implies that $y \notin \cup\{\mathcal{E}_f^C(\{\top, x\}) \mid x \in X \setminus \{\top\},\ \mathcal{E}_f^C(\{\top, x\}) \subseteq X\}$. Since $\mathcal{K}_f^C(X) \in \Gamma(C,f)$, then, by Lemma 6.8, $\mathcal{E}_f^C(\{\top, y\}) \subseteq \mathcal{K}_f^C(X)$, and therefore, $\mathcal{E}_f^C(\{\top, y\}) \subseteq X$, which is a contradiction. ■

Thus, the procedure in Table 2, whose correctness is an immediate consequence of Theorem 6.9, computes the fully complete kernel of an abstract domain X for any function $f : C^n \xrightarrow{c} C$, once that the least fully complete extensions $\mathcal{E}_f^C(\{\top, x\})$ of the atomic domains $\{\top, x\}$ (with $x \in X \setminus \{\top\}$) are known. The function *choose* is intended here to select and remove an arbitrary element from its input domain. Clearly, when f is additive or in addition extensive, the least fully complete extension of atomic domains can be obtained by the constructive methods mentioned above.

7 Applications

In this section, we present some applications of the techniques developed above. It is first shown in Section 7.1 how to reconstruct the Cousot and Cousot [1977] abstract domain of integer intervals as the least fully complete extension of *Sign* for the addition of sets of integers. Next, in Section 7.2 we show how our least fully complete extension operator can be fruitfully exploited in tuning the efficiency/precision trade-off when systematically improving the precision of an analysis either by a widening operator or by an abstract domain refinement. This is more practically illustrated in Section 7.3, where we apply an intelligent efficiency-oriented disjunctive completion refinement in the field of ground-dependency analysis for logic programs.

7.1 Reconstructing the Integer Interval Domain

We show that the least fully complete extension of the simple rule of signs domain *Sign* with respect to addition of sets of integers is precisely the Cousot and Cousot [1977] lattice of integer intervals *Int* considered in Example 5.15. This demonstrates that the domain for interval analysis can be fully reconstructed by a systematic domain refinement from *Sign*. This result gives a new interpretation to the lattice for integer intervals analysis, as the simplest and most precise analysis of sign for integer addition.

As observed in Example 4.4, *Sign* is not fully complete for integer addition. Instead, as proved by the following lemma, the abstract domain *Int* is fully complete for addition.

Lemma 7.1 $\mathit{Int} \in \Gamma(\wp(\mathbb{Z}), \oplus)$.

Proof. By Proposition 4.5 (viii), it is enough to prove that for any $n \in \mathbb{Z}$ and $X \subseteq \mathbb{Z}$, $\eta_{\mathit{Int}}(\{n\} \oplus X) = \eta_{\mathit{Int}}(\{n\} \oplus \eta_{\mathit{Int}}(X))$. But, assuming that $\eta_{\mathit{Int}}(\{n\} \oplus X) \subset \eta_{\mathit{Int}}(\{n\} \oplus \eta_{\mathit{Int}}(X))$ for some $n \in \mathbb{Z}$ and $X \in \wp(\mathbb{Z})$, evidently leads to a contradiction. ■

We go beyond the above expected result, and prove that *Int* is indeed the most abstract domain which includes *Sign* and for which the above result holds. Notice that $\oplus$ is an additive function, and therefore continuous. Thus, the least complete extension of *Sign* with respect to $\oplus$ does exist (by Theorem 4.11).

Theorem 7.2 $\mathcal{E}_{\oplus}^{\wp(\mathbb{Z})}(\mathit{Sign}) = \mathit{Int}$.

Proof. Since $\mathit{Int} \sqsubset \mathit{Sign}$, and, by Lemma 7.1, $\mathit{Int} \in \Gamma(\wp(\mathbb{Z}), \oplus)$, we get $\mathit{Int} \sqsubseteq \mathcal{E}_{\oplus}^{\wp(\mathbb{Z})}(\mathit{Sign})$. Moreover, by Theorem 5.14 (i), $\mathcal{E}_{\oplus}^{\wp(\mathbb{Z})}(\mathit{Sign}) = \mathcal{E}_{\oplus}^{\mathit{Int}}(\mathit{Sign})$, where in $\mathcal{E}_{\oplus}^{\mathit{Int}}(\mathit{Sign})$, addition $\oplus$ is restricted to intervals. Thus, let us assume by contradiction that $\mathcal{E}_{\oplus}^{\mathit{Int}}(\mathit{Sign}) \subset \mathit{Int}$. For any $n \in \mathbb{Z}$, define $f_n = \lambda X. [n, n] \oplus X : \mathit{Int} \to \mathit{Int}$. Similarly to Proposition 5.16, one can easily prove that $\Gamma(\mathit{Int}, \oplus) = \Gamma(\mathit{Int}, \{f_n\}_{n \in \mathbb{Z}})$. Thus, there exists $D \in \mathit{uco}(\mathit{Int})$ such that $\mathit{Sign} \subseteq D$, $\forall n \in \mathbb{Z}.\ D \in \Gamma(\mathit{Int}, f_n)$, and $\mathit{Int} \neq D$. Hence, $D \subset \mathit{Int}$. Since *Int* is meet-generated by its set $\mathit{MI}(\mathit{Int})$ of meet-irreducible elements, there exists $I \in \mathit{MI}(\mathit{Int})$ such that $I \notin D$ (otherwise, $\mathit{MI}(\mathit{Int}) \subseteq D$, and therefore $\mathit{Int} = D$). Note that, because $D \in \mathit{uco}(\mathit{Int})$, then $I \neq [-\infty, +\infty]$. Let $\eta_D \in \mathit{uco}(\mathit{Int})$ be the closure operator such that $\eta_D(\mathit{Int}) = D$. Observe that the (non top) meet-irreducible elements of *Int* are all and only those infinite intervals of the form $[-\infty, m]$ or $[m, +\infty]$, with $m \in \mathbb{Z}$. Thus, we can distinguish the following two cases.
(a) $I = [m, +\infty]$. Since $D \in \Gamma(\mathit{Int}, \{f_n\}_{n \in \mathbb{Z}})$, then $\forall n \in \mathbb{Z}.\ \eta_D([n, n] \oplus I) = \eta_D([n, n] \oplus \eta_D(I))$. Moreover, because $D \sqsubseteq \mathit{Sign}$, then $[0, +\infty] \in D$. We have that $[m, +\infty] \subset \eta_D([m, +\infty]) = [m', +\infty]$, where $m' < m$, and possibly $m' = -\infty$. If $m' \neq -\infty$, then the following equalities hold:

$$\begin{array}{rcl} [0, +\infty] & = & \eta_D([0, +\infty]) \\ & = & \eta_D([-m, -m] \oplus [m, +\infty]) \\ & = & \eta_D([-m, -m] \oplus \eta_D([m, +\infty])) \\ & = & \eta_D([m' - m, +\infty]). \end{array}$$

Thus, $[m' - m, +\infty] \subseteq \eta_D([m' - m, +\infty]) = [0, +\infty]$, and since $m' - m < 0$, this is an obvious contradiction. If instead $m' = -\infty$, i.e. $\eta_D([m, +\infty]) = [-\infty, +\infty]$, then

$$\begin{array}{rcl} [0,+\infty] & = & \eta_D([0,+\infty]) \\ & = & \eta_D([-m,-m] \oplus [m,+\infty]) \\ & = & \eta_D([-m,-m] \oplus \eta_D([m,+\infty])) \\ & = & \eta_D([-\infty,+\infty]) \\ & = & [-\infty,+\infty], \end{array}$$

which still is an evident contradiction.
(b) $I = [-\infty, m]$. The proof is analogous to the previous case, by observing that because $D \sqsubseteq$ *Sign*, then $[-\infty, 0] \in D$. ∎

The above proof is not constructive. Since $\oplus$ is additive, a constructive proof of the above result can be given in view of Theorem 6.1. In particular, an equational presentation of the domain of integer intervals can be given as follows: *Int is the most abstract solution of the system of recursive domain equations:* $\{X = Sign \sqcap \mathcal{C}_{\lambda X.\, n \oplus X}(X)\}_{n \in \mathbb{N}}$.

7.2 Towards Optimal Abstract Interpretations

Our fully complete refinement provides a systematic specification of an optimal (most abstract) fully complete domain from a given non fully complete one, with respect to any concrete domain C of reference. In the examples we have seen so far, C was always the concrete domain of the collecting or standard semantics. In this case, the fully complete refinement may result a domain too close to the concrete one, i.e. an infinite, often intractable, domain. For instance, in the previous section we have seen that the fully complete refinement of the 5-point lattice of signs is the lattice of integer intervals, which does not satisfy the ascending chain condition (ACC). The idea of using our domain transformer for designing optimal abstract interpretations comes from the observation that completeness is a relative notion, representing a situation of optimality between one domain with respect to a more concrete one. Thus, the absolute and ideal optimality given by completeness with respect to the concrete domain of the collecting or standard semantics can be scaled down by considering instead domains at different levels of abstraction playing the role of the concrete domain of reference. Within this context, in this section we show that the least fully complete extension operator is an important tool for supporting the systematic design of abstract domains for program analysis, in particular for reducing and optimizing the possible overflow in the process of refining an abstract interpretation.

Given some abstract interpretation $\mathbb{I}$ with underlying domain A, a more precise abstract interpretation $\mathbb{I}'$ can be systematically obtained either by refining A or by designing a suitable widening operator on the concrete domain which do better than A. We apply our least fully complete extension operator to these two cases, to design both optimal widenings and refinements.

7.2.1 Simplifying Widening Operators

Widening operators have been introduced in abstract interpretation by Cousot and Cousot [1977], to accelerate or even ensure convergence in least fixpoint approximation on abstract domains that are either too complex or do not satisfy the ACC. In the latter case, widenings do ensure that termination occurs, whereas standard Galois insertion-based abstract interpretations do not. A *widening* is a binary (possibly noncommutative) operator $\nabla : C \times C \to C$ such that:

1. For any $x, y \in C$, $x \leq x \nabla y$ and $y \leq x \nabla y$;
2. For any (increasing) chain $\{x_i\}_{i<\omega}$, the chain $\{y_i\}_{i<\omega}$ recursively defined by:

$$\begin{array}{rcl} y_0 & = & x_0 \\ y_{i+1} & = & y_i \nabla x_{i+1} \end{array}$$

   is not strictly increasing, i.e. $\exists k < \omega.\ y_k = y_{k+1}$.

In most cases, if $c \in C$ then $\lambda x.\, c \nabla x$ is monotone, while no hypothesis is assumed on the function $\lambda x.\, x \nabla c$. If $\mathcal{S} = \langle C, T \rangle$ is a concrete semantics and P is some program, Cousot and Cousot [1977] proved that the iteration sequence with widening $\{T_P^{\nabla i}\}_{i<\omega}$ defined as:

$$\begin{array}{rcl} T_P^{\nabla 0} & = & \perp \\ T_P^{\nabla i+1} & = & T_P^{\nabla i} \nabla T_P(T_P^{\nabla i}) \end{array}$$

converges in a finite number of steps, and it is an upper approximation of $lfp(T_P)$, i.e. if T_P^{∇} denotes the least element $T_P^{\nabla k}$ in the above iteration sequence such that $T_P^{\nabla k} = T_P^{\nabla k+1}$, then $lfp(T_P) \leq T_P^{\nabla}$.

It is known that widening-based abstract interpretations are in general more precise than any (finitely convergent) Galois insertion-based one. This is due to the fact that widening-based abstract interpretations can take full advantage of a possibly infinite domain not satisfying the ACC, whilst, in order to be convergent for any input program, a Galois insertion-based abstract interpretation needs to be defined on an abstract domain which satisfies the ACC. This limits the precision of program analyses specified in terms of Galois insertion-based abstract interpretations. This result has been formally proved by Cousot and Cousot [1992b]. Therefore, given any abstract domain A satisfying the ACC, it is always possible to improve the precision of the analysis induced by A, by designing a suitable widening on the concrete domain C, and then abstracting the result of the widening-based analysis back on A. More formally, if $\rho \in uco(C)$ is such that $\rho(C) \cong A$ and $f : C \xrightarrow{\mathrm{m}} C$, we may reasonably have that $\rho(lfp(f)) \leq \rho(f^{\nabla}) \leq lfp(f^{best_A})$ for some suitable widening ∇ on C. Whenever $\rho(f^{\nabla}) < lfp(f^{best_A})$, we say that ∇ *improves on the precision of* A.

Clearly, if $A \in \Gamma(C, f)$ then we cannot improve on the precision of A by any clever widening on C; in fact, in this case, for any improving widening ∇, $\rho(lfp(f)) = \rho(f^{\nabla}) = lfp(f^{best_A})$. Thus, whenever $\mathcal{E}_f^C(A)$ satisfies the ACC, $\mathcal{E}_f^C(A)$ is a sufficiently abstract domain which both improves on the precision of A and ensures termination. However, $\mathcal{E}_f^C(A)$ may not satisfy the ACC – indeed, this is the most frequent situation in program analysis (e.g., in the previous section we showed that $\mathcal{E}_{\oplus}^{\wp(\mathbb{Z})}(Sign) = Int$). In this case, in order to improve on the precision of the domain A, yet ensuring termination, we combine our systematic methodologies for designing complete abstract domains with widening-based abstract interpretation. The idea is that, given a widening ∇ on the concrete domain which improves on the precision of A, i.e. such that $\rho(f^{\nabla}) < lfp(f^{best_A})$, we systematically design a new simplified widening operator on a suitable domain X abstracting C, such that a widening-based analysis on X is still terminating, yet improving on A with the same precision as ∇ does.

First, observe that it is always possible to transfer a widening ∇ defined on C, to a generic abstract domain $A \in uco(C)$. This can be simply obtained by considering the best correct approximation of ∇ in A. The proof of the following result is immediate.

Proposition 7.3 *Let ∇ be a widening on C and $\rho \in uco(C)$. Then, $\nabla^{\rho} = \lambda x, y.\, \rho(x \nabla y)$ is a widening on $\rho(C)$.*

Thus, if $f : C \xrightarrow{\mathrm{m}} C$ and $\rho \in uco(C)$, then $f^{\nabla} \leq f^{\nabla^{\rho}}$, and therefore $\rho(f^{\nabla}) \leq \rho(f^{\nabla^{\rho}})$. The fact that $\rho(f^{\nabla}) < \rho(f^{\nabla^{\rho}})$ basically amounts to say that A is not complete for ∇. In the following, we characterize these "optimal" widenings in a practical way by means of our least fully complete extension operator.

Theorem 7.4 *Let ∇ be a widening on C such that for any $c \in C$, $\lambda x.\, c \nabla x : C \xrightarrow{\mathrm{c}} C$, and $f : C \xrightarrow{\mathrm{c}} C$. If $\rho \in \Gamma(C, \{\nabla, f\})$ then $\rho(f^{\nabla}) = \rho(f^{\nabla^{\rho}})$.*

Proof. Assume that $\rho \in \Gamma(C, \{\nabla, f\})$. We prove that for any $i < \omega$, $\rho(f^{\nabla}) = \rho(f^{\nabla^{\rho}})$.
$(i = 0)$: Obvious, because $\rho(f^{\nabla 0}) = \rho(\perp) = \rho(f^{\nabla^{\rho} 0})$.

$(i+1)$: From $\rho \in \Gamma(C, \{\nabla, f\})$ and by inductive hypothesis, we get:

$$\begin{array}{rcl}
\rho(f^{\nabla i+1}) & = & \rho(f^{\nabla i} \nabla f(f^{\nabla i})) \\
& = & \rho(\rho(f^{\nabla i}) \nabla \rho(f(f^{\nabla i}))) \\
& = & \rho(\rho(f^{\nabla i}) \nabla \rho(f(\rho(f^{\nabla i})))) \\
& = & \rho(\rho(f^{\nabla^\rho i}) \nabla \rho(f(\rho(f^{\nabla^\rho i})))) \\
& = & \rho(\rho(f^{\nabla^\rho i}) \nabla \rho(f(f^{\nabla^\rho i}))) \\
& = & \rho(f^{\nabla^\rho i} \nabla f(f^{\nabla^\rho i})) \\
& = & \rho(\rho(f^{\nabla^\rho i} \nabla f(f^{\nabla^\rho i}))) \\
& = & \rho(f^{\nabla^\rho i} \nabla^\rho f(f^{\nabla^\rho i})) \\
& = & \rho(f^{\nabla^\rho i+1}).
\end{array}$$

By Proposition 7.3, ∇^ρ is a widening, and therefore there exists a suitably large $k < \omega$ such that $\rho(f^{\nabla^\rho}) = \rho(f^{\nabla k}) = \rho(f^{\nabla})$. ∎

Thus, given a semantics $\langle C, T\rangle$, an abstract domain A, and a widening ∇ improving on the precision of A, the domain $\mathcal{E}^C_{\{T,\nabla\}}(A)$ exists unique by Corollary 4.16 and it is precisely the most abstract domain which induces a simpler widening which improves on the precision of A as ∇ does. Moreover, if $\eta = \mathcal{E}^C_{\{T,\nabla\}}(A)$, then $\mathcal{E}^C_{\{T,\nabla\}}(A)$ is the most abstract domain such that the above holds and $f^{\nabla^\eta} = \eta(f^{\nabla})$. In this case, the fact that $\mathcal{E}^C_{\{T,\nabla\}}(A)$ is the most abstract domain which includes A and is fully complete for f and ∇ ensures that the new widening will be specified in the simplest possible domain. This allows us to simplify considerably widening-based abstract interpretations:

> *Instead of using a concrete widening $\nabla : C \times C \to C$ when improving the analysis of a Galois insertion-based abstract interpretation with domain A, use the more simple and efficient ∇^X with $X = \mathcal{E}^C_{\{T,\nabla\}}(A)$.*

Moreover, note that the complexity of a widening is usually related with the length of maximal chains of the form: $y_0 = x_0$ and $y_{i+1} = y_i \nabla x_{i+1}$ for $i \geq 0$, with $\{x_i\}_{i<\omega}$ being a chain in C: Smaller is the length of these chains and quicker is termination. Hence, by Theorem 7.4, the widening induced on the fully complete domain $\mathcal{E}^C_{\{T,\nabla\}}(A)$, although being simpler and having a lower complexity than ∇, since $\mathcal{E}^C_{\{T,\nabla\}}(A) \subseteq C$, is just thinly less precise than ∇.

7.2.2 Intelligently Refining Abstract Domains

It is well known that the complexity and efficiency of an abstract interpretation strictly depend on the underlying abstract domain. A typical measure of complexity is in fact the maximal length of the chains in the abstract domain (e.g. see Deutsch [1997]). Thus, a systematic step of abstract domain refinement might considerably increase the global cost of the analysis, perhaps so much that it becomes even unacceptable. Even worse, such an enhanced abstract interpretation might become undecidable. For instance, this could be the case for the well-known disjunctive completion refinement [Cousot and Cousot 1979b;1994], that does not preserve the ACC property of abstract domains. This refinement operator enhances an abstract domain by introducing the necessary structure to model precisely the concrete disjunctive information, like the alternative computation paths in, for instance, the branches of a conditional choice (see the important examples in strictness and comportment analysis for functional programs discussed by Jensen [1992] and Cousot and Cousot [1994]). A further example could be the *Heyting completion* refinement [Giacobazzi and Scozzari 1997], which generalizes the Cousot and Cousot [1979b] *reduced cardinal power* of abstract domains, and enhances an abstract domain with suitable relational (functional) properties.

A number of systematic refinement strategies are known (cf. [Cousot and Cousot 1979b; Filé et al. 1996; Giacobazzi and Ranzato 1997]), but none of them provide the right balance between efficiency and precision. Let us first generalize and formalize the above scenario. The situation is close to that considered in Section 5.3. Let $\mathcal{S} = \langle C, f\rangle$ be a concrete least fixpoint semantics, where $f : C \xrightarrow{\mathrm{m}} C$, and let D be an abstraction of C. The best correct approximation f^{best_D} of f

for D automatically induces a correct abstract semantics $\mathcal{S}^D = \langle D, f^{best_D} \rangle$. Thus, if A is another abstraction of C such that $D \sqsubseteq A$ then we can reason on the completeness relationships between A and D, where $\mathcal{S}^D$ plays the role of the relative concrete semantics – i.e., A is interpreted as an abstraction of D (viz. $A \in uco(D)$). Then, we simply say that A is complete, resp. fully complete, for (or with respect to) D to mean that $A \in \Delta(D, f^{best_D})$, resp. $A \in \Gamma(D, f^{best_D})$. In particular, such a situation arises whenever $\Re : uco(C) \to uco(C)$ is some (possibly nonidempotent) domain refinement, and therefore, for some abstract domain A we get $C \sqsubseteq \Re(A) \sqsubseteq A$. In this case, we take into consideration the possible completeness relations of A with respect to the relative concrete semantics $\langle \Re(A), f^{best_{\Re(A)}} \rangle$. In the following, we simplify the notation, and by omitting the semantic function, we denote the corresponding least fully complete extension operator simply by $\mathcal{E}^{\Re(A)}$. Note that in general a domain refinement $\Re$ is intended to transform "analyses into analyses", i.e., at least decidability should be preserved by $\Re$; therefore, A and $\Re(A)$ should be, roughly speaking, quite close, and completeness should not be an exceptional event.

In general, we have that $\Re(A) \sqsubseteq \mathcal{E}^{\Re(A)}(A) \sqsubseteq A$. We say that $\Re(A)$ is *too refined* when the least fully complete extension of A with respect to $\Re(A)$, when it exists, is a proper abstraction of $\Re(A)$, i.e. when $\Re(A) \sqsubset \mathcal{E}^{\Re(A)}(A)$ holds. The intuition behind this definition is as follows. If for any two abstract domains A and A' (thought of as underlying domains of two abstract interpretations $\mathcal{I}$ and $\mathcal{I}'$ of a common concrete semantics) such that $A' \sqsubseteq A$, it happens that A is fully complete with respect to A', then, we can reasonably consider $\mathcal{I}$ as just thinly less precise than $\mathcal{I}'$. In such a case, an efficiency/precision trade-off would then suggest to prefer $\mathcal{I}$ to $\mathcal{I}'$, especially when $\mathcal{I}'$ is much less efficient than $\mathcal{I}$. In other terms, since, thanks to full completeness, the gap of precision between $\mathcal{I}$ and $\mathcal{I}'$ is very narrow, in general, one should be inclined to choose $\mathcal{I}$ rather than the more costly $\mathcal{I}'$. Thus, in the above definition, whenever $\mathcal{E}^{\Re(A)}(A) \neq \Re(A)$, one should prefer the thinly less refined domain $\mathcal{E}^{\Re(A)}(A)$ rather than the fully refined domain $\Re(A)$; in particular, if A is already fully complete with respect to $\Re(A)$ (i.e. $\mathcal{E}^{\Re(A)}(A) = A$), then A actually does not need to be refined by $\Re$. This explains why in these cases we term $\Re(A)$ too refined.

Therefore, an "efficiency-oriented" version $\Re^\star$ of the refinement $\Re$ can be defined as $\Re^\star = \lambda A.\, \mathcal{E}^{\Re(A)}(A)$. This strategy may in fact lead to significant savings. For instance, in the above example of the disjunctive completion refinement, it might happen that A is a domain satisfying the ACC and its disjunctive completion $\mathbb{P}(A)$ does not satisfy any more this desirable, and sometimes essential, property, whilst, in contrast, our strategy provides an intelligent disjunctive completion $\mathbb{P}^\star(A)$ that still satisfies the ACC property. In the next section, we present a relevant example of this phenomenon.

7.3 Intelligent Disjunctive Refinement of Ground-Dependency Analysis

Def and *Pos* are two well-known and widely used abstract domains for ground-dependency analysis of logic languages [Armstrong et al. 1997; Cortesi et al. 1996; Marriott and Søndergaard 1993]. We consider here the simple case of positive logic programs, but it is worth remarking that minor variants of *Def* and *Pos* have also been used for the analysis of constraint and concurrent [Codish et al. 1994] logic languages. We consider the standard *s-semantics* by Falaschi et al. [1989] as the concrete least fixpoint semantics. The s-semantics is fully abstract for the observable given by the computed answer substitutions of a positive logic program. It is the standard base semantics for a bottom-up abstract interpretation framework for logic program analysis, such as those of Barbuti et al. [1993], Codish et al. [1994], and Marriott et al. [1994].

The Least Fixpoint s-Semantics. Let us succinctly recall the definition of the s-semantics. Let *Var* be a countable set of variables, Σ be an alphabet of constant and function symbols, and Π be an alphabet of predicate symbols, that give rise to a fixed first order language L. If $p \in \Pi$ then $\#(p) \in \omega$ denotes the arity of p. The set of atoms and (positive) logic programs built from L are denoted by *Atom* and *Program*, respectively. For any syntactic object o, $var(o)$ denotes the set of variables occurring in o. A substitution σ is denoted by its set of nontrivial bindings, i.e. $\sigma = \{x/\sigma(x) \mid \sigma(x) \neq x\}$. The empty substitution is denoted by ϵ. The set of idempotent substitutions is denoted by *Sub*. If σ is a substitution and E is any syntactic entity,

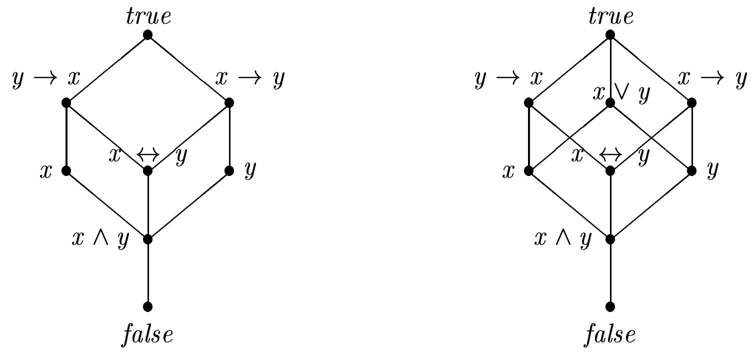


Figure 5: The domains Def_2 and Pos_2 for $\mathit{VI} = \{x, y\}$.

then both $\sigma(E)$ and $E\sigma$ stand for the result of applying σ to E. If o and o' are two syntactic objects, then o is more general than o', denoted $o' \preceq o$, iff there exists a substitution σ such that $o' = o\sigma$. Also, o and o' are equivalent up to renaming, denoted $o \simeq o'$, iff $o \preceq o'$ and $o' \preceq o$. The obvious lifting of $\preceq$ to the quotient $\mathit{Atom}_{/\simeq}$ gives rise to a partial order. With abuse of notation, such a poset will be still denoted by Atom, and we will let any atom denote its equivalence class by renaming. For a syntactic object o and a set of atoms $I \subseteq \mathit{Atom}$, we denote by $\langle a_1, \ldots, a_n \rangle \ll_o I$ $(n \geq 0)$, that $a_1, \ldots, a_n$ are elements of I renamed apart from o and from each other (i.e. $\forall i, j \in [1, n].\ \mathit{var}(a_i) \cap \mathit{var}(o) = \emptyset \ \&\ (i \neq j \Rightarrow \mathit{var}(a_i) \cap \mathit{var}(a_j) = \emptyset)$). For two tuples of atoms $\bar{s}$ and $\bar{t}$, if $\bar{s}$ and $\bar{t}$ are unifiable then $\mathit{mgu}(\bar{s}, \bar{t})$ provides an idempotent most general unifier (it is well known that they are all equivalent up to renaming, cf. [Lassez et al. 1988]), while it is undefined otherwise.

The concrete domain of the s-semantics is given by the set of all so-called s-interpretations ordered by subset inclusion, i.e. it is the complete lattice $\langle \wp(\mathit{Atom}), \subseteq \rangle$. The semantic transformer $T^s : \mathit{Program} \to (\wp(\mathit{Atom}) \xrightarrow{c} \wp(\mathit{Atom}))$ is defined as follows: For any $P \in \mathit{Program}$,

$$T^s_P(I) = \left\{ h\theta \in \mathit{Atom} \;\middle|\; \begin{array}{l} c = h \leftarrow b_1, \ldots, b_n \in P \\ \langle b'_1, \ldots, b'_n \rangle \ll_c I,\ n \geq 0 \\ \theta = \mathit{mgu}(\langle b_1, \ldots, b_n \rangle, \langle b'_1, \ldots, b'_n \rangle) \end{array} \right\}.$$

It turns out that, for any program P, T^s_P is a continuous operator, and therefore the s-semantics of P is defined as $[\![P]\!] = \mathit{lfp}(T^s_P) = \cup_{n \in \omega} (T^s_P)^n(\emptyset)$. Thus, our concrete interpretation is given by $\mathcal{S} = \langle \wp(\mathit{Atom}), T^s \rangle$.

The Abstract Domains *Def* and *Pos*. We now briefly recall the definitions of the abstract domains *Def* and *Pos*. For more details we refer to [Armstrong et al. 1997; Marriott and Søndergaard 1993]. Let $\mathit{VI} \subseteq \mathit{Var}$ be a (nonempty) finite subset of variables of interest. *Pos* is the finite lattice (indeed Boolean lattice) of *positive* Boolean functions on VI, where a Boolean function is positive if it gives *true* for the truth assignment where each variable is *true*. Obviously, the order of *Pos* is given by logical consequence, denoted by $\preceq$ ($\prec$ denotes strict ordering). *Def* is the finite lattice of positive Boolean functions on VI whose set of models is closed under intersection. Boolean functions in *Def* are called *definite*. Note that the bottom Boolean function *false* does not belong to *Pos* (and *Def*). However, since *false* is useful in order to represent precisely the empty set of substitutions (and therefore to represent the information of reachability), we add *false* to both *Def* and *Pos*, and we still use *Def* and *Pos* to denote these lattices. The *glb*'s of *Pos* and *Def* are both given by logical conjunction, while only the *lub* of *Pos* is given by logical disjunction; the *lub* of *Def* can be defined in the usual way in terms of logical conjunction: $f_1 \vee_{\mathit{Def}} f_2 = \wedge\{f \in \mathit{Def} : f_i \preceq f\ (i = 1, 2)\}$. Thus, it turns out that *Def* is a meet subsemilattice of Pos, and hence it is an abstraction of Pos. It is well known that Boolean functions can be represented by means of propositional formulae. Thus, in the following, we will use propositional formulae over VI to represent Boolean functions in Pos and Def. If $|VI| = n$ then we will often use the notations Def_n and Pos_n to emphasize the size of the underlying set of variables. The lattices Def_2 and Pos_2 are depicted in Figure 5 for the case of two variables of interest $VI = \{x, y\}$.

Def and Pos are canonically viewed as abstractions of the powerset of all the idempotent substitutions $\langle \wp(Sub), \subseteq \rangle$. The abstraction and concretization maps between Pos, Def and $\wp(Sub)$ are well known, and can be found, e.g., in [Armstrong et al. 1997; Marriott and Søndergaard 1993]. For instance, assuming $VI = \{x, y, z, u\}$, the formula $x \wedge (y \leftrightarrow z)$ is an element of Pos (and Def) that represents the substitutions σ such that for any instance σ' of σ: (i) the term $\sigma'(x)$ is ground; (ii) $\sigma'(y)$ is ground iff also $\sigma'(z)$ is ground. In particular,[4] $\sigma_1 = \{x/a, y/b, z/c\}$ and $\sigma_2 = \{x/a, y/w, z/w, v/u\}$ satisfy this property. Thus, if $(\alpha_{Def}, \wp(Sub), Def, \gamma_{Def})$ and $(\alpha_{Pos}, \wp(Sub), Pos, \gamma_{Pos})$ are the corresponding GIs, $\{\sigma_1, \sigma_2\} \subseteq \gamma_{Pos}(x \wedge (y \leftrightarrow z)) = \gamma_{Def}(x \wedge (y \leftrightarrow z))$.

The Disjunctive Completion Refinement. Let us then recall how the disjunctive completion refinement is defined. For more details see [Cousot and Cousot 1979b; Filé and Ranzato 1998; Giacobazzi and Ranzato 1998]. Here, we only consider the simple case of a concrete domain C which is completely distributive (as any powerset $\langle \wp(X), \subseteq \rangle$ is). For a GC (α, C, A, γ), we define the following equivalence relation on $\wp(A)$: For all $S, T \subseteq A$, $S \stackrel{\vee}{\simeq} T \;\Leftrightarrow\; \vee_C \gamma(S) = \vee_C \gamma(T)$. The disjunctive completion $\mathbb{P}(A)$ of A is defined as the quotient of $\wp(A)$ for the equivalence relation $\stackrel{\vee}{\simeq}$. $\mathbb{P}(A)$ is a complete lattice under the following ordering relation: $[S] \leq [T] \;\Leftrightarrow\; \forall s \in S. \exists t \in T.\; s \leq_A t$ (other equivalent definitions can be given, see [Filé and Ranzato 1998]). $\mathbb{P}(A)$ is related to the concrete domain C by the GI $(\alpha^\star, C, \mathbb{P}(A), \gamma^\star)$, where, for any equivalence class $[S] \in \mathbb{P}(A)$, $\gamma^\star([S]) = \vee_C \gamma(S))$, while $\alpha^\star$ is defined as the usual left adjoint of $\gamma^\star$. It turns out that the disjunctive completion operator P actually is an idempotent refinement. When $\langle C, \leq_C \rangle = \langle \wp(X), \subseteq \rangle$, for some set X, we have that, for any $Z \in \wp(X)$, $\alpha^\star(Z) = [\alpha(Z)]$ (cf. [Filé and Ranzato 1998, Proposition 3.7]). A is viewed as an abstraction of its disjunctive completion $\mathbb{P}(A)$ in the most obvious way by the following GI: $(\lambda[S].\, \vee_A S, \mathbb{P}(A), A, \lambda a. [a])$.

Filé and Ranzato [1998] investigated the disjunctive completion of Pos as a suitable domain for disjunctive ground-dependency analysis for logic programs. Filé and Ranzato [1998, Theorem 5.3] proved that $\mathbb{P}(Pos) \sqsubset Pos$, namely there is a strict improvement in precision by lifting Pos to its disjunctive completion. For example, by considering as variables of interest $VI = \{x, y\}$, we have that the logical disjunction $(x \to y) \vee (y \to x)$ in Pos does not represent the concrete disjunction of the two formulae $x \to y$ and $y \to x$, i.e. the union of their concretizations. In fact, $\gamma_{Pos}(x \to y) \cup \gamma_{Pos}(y \to x) \subset \gamma_{Pos}((x \to y) \vee (y \to x)) = \gamma_{Pos}(true)$: $\sigma = \{x/v, y/w\}$ is such that $\sigma \in \gamma_{Pos}((x \to y) \vee (y \to x)) \setminus \gamma_{Pos}(x \to y) \cup \gamma_{Pos}(y \to x)$. Moreover, Giacobazzi and Ranzato [1998, Theorem 7.2] demonstrated that $\mathbb{P}(Def) = \mathbb{P}(Pos)$. Thus, the disjunctive ground-dependency analysis of logic programs can always be obtained by disjunctive completion of Def, without losing precision and at a lower cost with respect to the disjunctive completion of Pos. Moreover, Giacobazzi and Ranzato [1998] completed the understanding of the problem by proving that Def is the most abstract domain having the same disjunctive completion as Pos has. The disjunctive completion of Def_2 (and Pos_2) is depicted in Figure 6.

Abstracting the s-Semantics. Let us see now how to lift Def, Pos and $\mathbb{P}(Def)$ in order to get corresponding abstractions Def^s, Pos^s and $\mathbb{P}(Def)^s$ of the concrete domain $\wp(Atom)$ of the s-semantics. We consider the case of Def, since the other two are analogous. The abstract domain Def^s is defined as follows:

$$Def^s = \{\langle (p, f_p) \rangle_{p \in \Pi} \;\mid\; \#(p) = n \;\Rightarrow\; f \in Def_n\}.$$

Thus, a typical element of Def^s is a tuple, indexed on the alphabet Π, of pairs (p, f_p), where p is a predicate symbol and f_p represents the Def-ground-dependency information on p. By

[4] By $a, b, c, \ldots$, we denote ground terms.

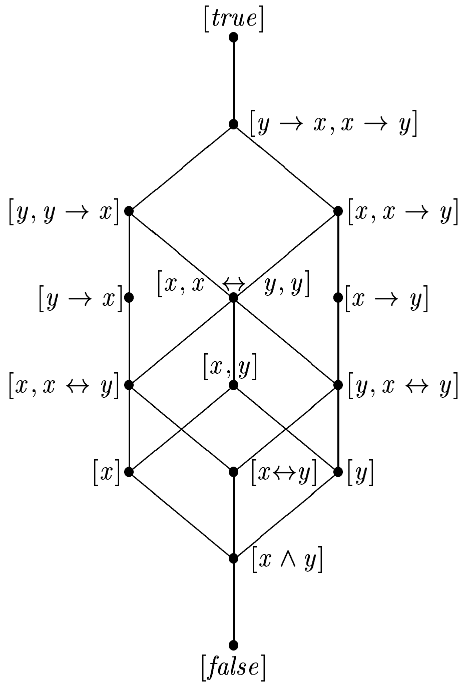


Figure 6: The disjunctive completion $\mathbb{P}(\mathit{Def}_2) = \mathbb{P}(\mathit{Pos}_2)$ for $\mathit{VI} = \{x, y\}$.

convention, we define $\mathit{Def}_0 = \{\mathit{true}\}$, and therefore for a constant predicate symbol p of arity zero, the corresponding component is always (p, true). Def^s is ordered componentwise in the following way: For any $F, G \in \mathit{Def}^s$, $F \leq G$ iff $\forall (p, f_p) \in F, (p, g_p) \in G.\ f_p \preceq g_p$. Obviously, Def^s still is a (finite) lattice, where *lub* and *glb* are defined componentwise from those of Def. The idempotent substitution associated to an atom $p(t_1, ..., t_n) \in \mathit{Atom}$, with $\#(p) \geq 1$, is given by $\mathit{sub}(p(t_1, ..., t_n)) = \{x_1/t_1, ..., x_n/t_n\}$, where $x_1, ..., x_n$ are fresh variables, while if $\#(p) = 0$ then $\mathit{sub}(p) = \epsilon$. It is then straightforward to design the GI $(\alpha_{\mathit{Def}^s}, \wp(\mathit{Atom}), \mathit{Def}^s, \gamma_{\mathit{Def}^s})$: For any $\langle (p, f_p) \rangle_{p \in \Pi} \in \mathit{Def}^s$,

$$\gamma_{\mathit{Def}^s}(\langle (p, f_p) \rangle_{p \in \Pi}) = \{p(t_1, ..., t_n) \in \mathit{Atom} \mid \mathit{sub}(p(t_1, ..., t_n)) \in \gamma_{\mathit{Def}_n}(f_p)\},$$

where it is understood that the underlying set of variables of interest for Def_n is $\{x_1, ..., x_n\}$. Also, for an atom $p(t_1, ..., t_n) \in \mathit{Atom}$ the corresponding abstraction in Def is simply given by $\alpha_{\{\cdot\}}(p(t_1, ..., t_n)) = \alpha_{\mathit{Def}_n}(\{\{x_1/t_1, ..., x_n/t_n\}\})$. Clearly, if $\#(p) = 0$ then its abstraction results to be *true*; moreover, $\alpha_{\{\cdot\}}$ is well defined, i.e. if $p(\bar{t}) \simeq p(\bar{s})$ then $\alpha_{\{\cdot\}}(p(\bar{t})) = \alpha_{\{\cdot\}}(p(\bar{t}))$. Thus, for any s-interpretation $I \in \wp(\mathit{Atom})$, its abstraction in Def^s is defined as follows:

$$\alpha_{\mathit{Def}^s}(I) = \langle (p, \vee_{\mathit{Def}} \{\alpha_{\{\cdot\}}(p(\bar{t})) \mid p(\bar{t}) \in P\}) \rangle_{p \in \Pi}.$$

Note that if some predicate p is missing in I, then the corresponding component in $\alpha_{\mathit{Def}^s}(I)$ is given by *false*, if $\#(p) > 0$, by *true* otherwise. The fact that $(\alpha_{\mathit{Def}^s}, \wp(\mathit{Atom}), \mathit{Def}^s, \gamma_{\mathit{Def}^s})$ is a GI is straightforward to prove, as this is a direct consequence of the GI of Def in $\wp(\mathit{Sub})$. By following the same way of proceeding, one also get the GIs $(\alpha_{\mathit{Pos}^s}, \wp(\mathit{Atom}), \mathit{Pos}^s, \gamma_{\mathit{Pos}^s})$ and $(\alpha_{\mathbb{P}(\mathit{Def})^s}, \wp(\mathit{Atom}), \mathbb{P}(\mathit{Def})^s, \gamma_{\mathbb{P}(\mathit{Def})^s})$. By a slight abuse of notation, for the sake of conciseness, in the following we omit the superscript s when denoting such abstract domains. Obviously, it turns out that $\wp(\mathit{Atom}) \sqsubseteq \mathbb{P}(\mathit{Def}) \sqsubseteq \mathit{Pos} \sqsubseteq \mathit{Def}$. The following example helps in clarifying the formalizations introduced above.

Example 7.5 Let $\Pi = \{p/2, q/1, r/0, s/3\}$ be the fixed alphabet of predicate symbols of the language L. Consider the following two s-interpretations: $I_1 = \{p(a, y), p(x, x), q(x)\}$ and $I_2 =$

$\{p(f(x,x),a), p(a,f(x,y)), q(a), r\}$. Then, it is immediate to verify that:

- $\alpha_{Def}(I_1) = \langle (p, x_2 \to x_1), (q, true), (r, true), (s, false)\rangle$;
  $\alpha_{Def}(I_2) = \langle (p, true), (q, x_1), (r, true), (s, false)\rangle$.
- $\alpha_{Pos}(I_1) = \alpha_{Def}(I_1)$; $\alpha_{Pos}(I_2) = \langle (p, x_1 \vee x_2), (q, x_1), (r, true), (s, false)\rangle$.
- $\alpha_{\mathbb{P}(Def)}(I_1) = \langle (p, [x_1, x_1 \leftrightarrow x_2]), (q, true), (r, true), (s, false)\rangle$; $\alpha_{\mathbb{P}(Def)}(I_2) = \alpha_{Pos}(I_2)$. □

Therefore, from the concrete s-semantics $\mathcal{S}$ we get the abstract semantics: $\mathcal{S}^{Def} = \langle Def, T^{Def}\rangle$, $\mathcal{S}^{Pos} = \langle Pos, T^{Pos}\rangle$, $\mathcal{S}^{\mathbb{P}(Def)} = \langle \mathbb{P}(Def), T^{\mathbb{P}(Def)}\rangle$, where each abstract semantic transformer is defined as the corresponding best correct approximation of T^s (e.g., for any P, $T_P^{\mathbb{P}(Def)} = \alpha_{\mathbb{P}(Def)} \circ T_P^s \circ \gamma_{\mathbb{P}(Def)}$).

The Intelligent Disjunctive Completion of *Def* is *Pos*. Let us turn to discuss the completeness issues between these abstract domains. It is known [Marriott and Søndergaard 1993] that *Def* is not complete (and therefore fully complete) with respect to *Pos* and $\mathbb{P}(Def)$, as shown by the following example.

Example 7.6 Consider the program P below.

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

It is easy to see that $(p, x_1 \leftrightarrow x_2) \in lfp(T_P^{Def})$, whilst $(p, x_1 \wedge x_2) \in lfp(T_P^{Pos}) = lfp(T_P^{\mathbb{P}(Def)})$. Thus, by using *Pos* (or $\mathbb{P}(Def)$) we are able to infer that any computed answer substitution for a query $p(x,y)$ will ground both arguments of p, while by using *Def* we can only conclude that the first argument will be ground iff the second will be. Since $x_1 \wedge x_2 \in Def$, this shows that $Def \notin \Delta(Pos, \{T_P^{Pos}\}_{P\in Program}) \cup \Delta(\mathbb{P}(Def), \{T_P^{\mathbb{P}(Def)}\}_{P\in Program})$, and hence $Def \notin \Gamma(Pos, \{T_P^{Pos}\}_{P\in Program}) \cup \Gamma(\mathbb{P}(Def), \{T_P^{\mathbb{P}(Def)}\}_{P\in Program})$. □

On the other hand, Filé and Ranzato [1998, Proposition 5.13] proved that *Pos* is fully complete with respect to $\mathbb{P}(Def)$, i.e., the following result holds.

Theorem 7.7 (Filé and Ranzato 1998) $Pos \in \Gamma(\mathbb{P}(Def), \{T_P^{\mathbb{P}(Def)}\}_{P\in Program})$.

As a consequence, *Pos* is also complete with respect to $\mathbb{P}(Def)$. Let us see an example.

Example 7.8 Let us consider the following program P.

$$\begin{array}{lll} p(x,a) & :- & p(x,z), p(y,x). \\ p(x,x) & :- & \\ p(a,y) & :- & \end{array}$$

It is a routine task to see that the Kleene iterations for T_P^{Pos} and $T_P^{\mathbb{P}(Def)}$ are as follows:

- $(T_P^{Pos})^0 = (p, false)$; $(T_P^{Pos})^1 = (p, x_2 \to x_1)$; $(T_P^{Pos})^2 = (p, true)$ (least fixpoint).
- $(T_P^{\mathbb{P}(Def)})^0 = (p, false)$; $(T_P^{\mathbb{P}(Def)})^1 = (p, [x_1 \leftrightarrow x_2, x_1])$; $(T_P^{\mathbb{P}(Def)})^2 = (p, [x_1 \leftrightarrow x_2, x_1, x_2])$ (least fixpoint).

Notice that, in accordance with the above completeness theorem, we have that, for $i \in \{0,1,2\}$, $(T_P^{Pos})^i = \vee (T_P^{\mathbb{P}(Def)})^i$. Thus, using $\mathbb{P}(Def)$ we are able to infer that in each computer answer substitution for the predicate p, either its first argument is ground or its second argument is ground or they are equivalent (namely, they are bound to the same variables). Instead, by using *Pos* we get no ground-dependency information. Nevertheless, by abstracting in *Pos* the final output $(p, [x_1 \leftrightarrow x_2, x_1, x_2])$ of $\mathbb{P}(Def)$ we exactly get the output $(p, true)$ of *Pos*, as one expect from Theorem 7.7. □

Let $F = \{T_P^{\mathbb{P}(Pos)}\}_{P\in Program}$. Since each concrete semantic transformer T_P^s is continuous, by Theorem 4.11 and Lemma 5.13, we know that the least fully complete extension $\mathcal{E}_F^{\mathbb{P}(Pos)}(Def)$ of Def with respect to $\mathbb{P}(Pos)$ exists. Then, we sharp Theorem 7.7 by proving that $\mathcal{E}_F^{\mathbb{P}(Pos)}(Def)$ actually coincides with Pos.

Theorem 7.9 $\mathcal{E}_F^{\mathbb{P}(Pos)}(Def) = Pos$.

Proof. By Theorem 7.7, we have that

$$Pos \sqsubseteq \mathcal{E}_F^{\mathbb{P}(Def)}(Def) = \sqcup\{A \in uco(\mathbb{P}(Def)) \mid A \sqsubseteq Def,\ A \in \Gamma(\mathbb{P}(Def), \{T_P^{\mathbb{P}(Def)}\}_{P\in Program})\}.$$

Let us suppose, by contradiction, that $Pos \sqsubset \mathcal{E}_F^{\mathbb{P}(Pos)}(Def)$, i.e. that there exists some $D \in uco(\mathbb{P}(Def))$ such that:

(i) $D \in \Gamma(\mathbb{P}(Def), \{T_P^{\mathbb{P}(Def)}\}_{P\in Program})$;

(ii) $D \sqsubseteq Def$;

(iii) $D \not\sqsubseteq Pos$.

Then, by (iii), there exists some predicate $p \in \Pi$ such that $R = \langle ..., (p, [S_p]), ...\rangle \in \mathbb{P}(Def)$, $\rho_{Pos}(R) = R$ and $R <_D \rho_D(R)$. Thus, by (ii), $R <_{Def} \rho_{Def}(R)$ also holds. So, let $[S_p]$ be represented by some formula $f_p \in Pos$, and note that $[S_p] = [f_p]$ and $f_p \notin Def$. Since Pos can be obtained by closing under logical disjunction Def (cf. [Armstrong et al. 1997]), this implies that $f_p = g \vee h$, where $g, h \in Def$. Thus, by (ii), up to isomorphic representations, $g, h \in D$. Notice that Def is closed under the "iff" logical connective $\leftrightarrow$, and therefore we have that $g \wedge h, g \leftrightarrow h \in Def$, and, in turn, $g \wedge h, g \leftrightarrow h \in D$. Let us consider now the program $P_f = \{p(y_1, ..., y_n) :- p(s_1, ..., s_n).\}$ such that the Def-abstraction of $p(s_1, ..., s_n)$ is $g \leftrightarrow h$, i.e. such that $\{y_1, ..., y_n\} \cap var(\langle s_1, ..., s_n\rangle) = \emptyset$ and $\alpha_{\{.\}}(p(s_1, ..., s_n)) = g \leftrightarrow h$. Note that this is always possible, since for any formula $h \in Def$ there exists a suitable idempotent substitution $\sigma_h \in Sub$ such that $\alpha_{Def}(\{\sigma_h\}) = h$ (cf. [Filé and Ranzato 1998, Lemma 5.4]). It is then clear that

$$T_{P_f}^{\mathbb{P}(Def)}(R) = R \wedge \langle ..., (p, [g \leftrightarrow h]), ...\rangle = \langle ..., (p, [f_p \wedge (g \leftrightarrow h))], ...\rangle = \langle ..., (p, [g \wedge h]), ...\rangle.$$

Therefore, since $g \wedge h \in D$, we also get that $\rho_D(T_{P_f}^{\mathbb{P}(Def)}(R)) = \langle ..., (p, [g \wedge h]), ...\rangle$. Now, let $\rho_D([f_p]) = [a, b]$, for some $a, b \in Def$. Hence, $[f_p] = [g \vee h] <_{\mathbb{P}(Def)} [a, b]$, and, by definition of the ordering of the disjunctive completion $\mathbb{P}(Def)$, this means that either $g \vee h \prec_{Def} a$ or $g \vee h \prec_{Def} b$. W.l.o.g., let $g \vee h \prec_{Def} a$. Thus, there exists a truth interpretation m such that $m \models a$ and $m \not\models g \vee h$, and therefore $m \not\models g$ and $m \not\models h$. Thus, we get that

$$T_{P_f}^{\mathbb{P}(Def)}(\rho_D(R)) = \langle ..., (p, [a, b]), ...\rangle \wedge \langle ..., (p, [g \leftrightarrow h]), ...\rangle = \langle ..., (p, [a \wedge (g \leftrightarrow h), b \wedge (g \leftrightarrow h)]), ...\rangle.$$

Since $g \wedge h \preceq a \wedge (g \leftrightarrow h)$, we then have that

$$T_{P_f}^{\mathbb{P}(Def)}(\rho_D(R)) \geq \langle ..., (p, [a \wedge (g \leftrightarrow h)]), ...\rangle \geq \langle ..., (p, [g \wedge h]), ...\rangle,$$

and therefore, since $a \wedge (g \leftrightarrow h) \in D$,

$$\rho_D(T_{P_f}^{\mathbb{P}(Def)}(\rho_D(R))) \geq \langle ..., (p, [a \wedge (g \leftrightarrow h)]), ...\rangle \geq \langle ..., (p, [g \wedge h]), ...\rangle.$$

Moreover, note that $g \wedge h \prec_{Def} a \wedge (g \leftrightarrow h)$, because $m \models a \wedge (g \leftrightarrow h)$ but $m \not\models g \wedge h$. Thus, we get that $\rho_D(T_{P_f}^{\mathbb{P}(Def)}(R)) < \rho_D(T_{P_f}^{\mathbb{P}(Def)}(\rho_D(R)))$, i.e. D is not fully complete with respect to $\mathbb{P}(Def)$, which, by (i), is a contradiction. ■

As a consequence of the proof of the above theorem, we also get that the least complete extension $\mathbb{E}_F^{\mathbb{P}(Pos)}(Def)$ of Def in $\mathbb{P}(Pos)$ does exist (although no general theorem ensures this), and it still coincides with Pos.

Theorem 7.10 $\mathbb{E}_F^{\mathbb{P}(Pos)}(Def) = Pos$.

Proof. With reference to the proof of Theorem 7.9, consider instead the following program $P_f = \{p(\bar{y}) : -\, p(\bar{u}).,\ p(\bar{s}) : -,\ p(\bar{t}) : -\ \}$, where $\bar{y}$ is a tuple of fresh variables, and $\alpha_{\{\cdot\}}(p(\bar{u})) = g \leftrightarrow h$, $\alpha_{\{\cdot\}}(p(\bar{s})) = g$ and $\alpha_{\{\cdot\}}(p(\bar{t})) = h$. We need to note that $[f_p] = [g \vee h] = [g, h]$, because $g, h \in Def$ and $f_p \in Pos$ (this is a consequence of a more general fact by Filé and Ranzato [1998, Theorem 5.6]). Observe now that $(T_{P_f}^{\mathbb{P}(Def)})^1 = \langle ..., (p, [g, h]), ... \rangle$ and

$$(T_{P_f}^{\mathbb{P}(Def)})^2 = \langle ..., (p, [g \wedge (g \leftrightarrow h), h \wedge (g \leftrightarrow h)]), ... \rangle = \langle ..., (p, [g \wedge h]), ... \rangle = lfp(T_{P_f}^{\mathbb{P}(Def)}).$$

On the other hand,

$$(\rho_D \circ T_{P_f}^{\mathbb{P}(Def)})^1 = \langle ..., (p, \rho_D([g, h]), ... \rangle = \langle ..., (p, [a, b], ... \rangle$$

and

$$(\rho_D \circ T_{P_f}^{\mathbb{P}(Def)})^2 = \langle ..., (p, \rho_D([a, b] \wedge [g \leftrightarrow h]), ... \rangle,$$

and therefore, by the same reasoning of the proof of Theorem 7.9,

$$(\rho_D \circ T_{P_f}^{\mathbb{P}(Def)})^2 = \langle ..., (p, [a \wedge (g \leftrightarrow h), b \wedge (g \leftrightarrow h)]), ... \rangle = lfp(\rho_D \circ T_{P_f}^{\mathbb{P}(Def)}).$$

Thus, as in the proof of Theorem 7.9, we get $\rho_D(lfp(T_{P_f}^{\mathbb{P}(Def)})) < lfp(\rho_D \circ T_{P_f}^{\mathbb{P}(Def)})$, and this contradiction concludes the proof. ∎

To conclude this section, let us discuss the practical consequences of Theorem 7.9, in view of the general observations made at the beginning of this section. If we want to systematically design an abstract domain for disjunctive groundness 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 $\mathbb{P}(Def)$. However, according to our definitions, $\mathbb{P}(Def)$ is too refined, since, by Theorem 7.9, $\mathcal{E}_F^{\mathbb{P}(Pos)}(Def) = Pos$ is a proper abstraction of $\mathbb{P}(Def)$. By contrast, following the "efficiency-oriented" strategy of refinement outlined above, the intelligent disjunctive completion is defined as $\mathbb{P}^\star = \lambda A.\, \mathcal{E}_F^{\mathbb{P}(A)}(A)$, and therefore we would instead refine Def to 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 $\mathbb{P}(Def)$ has an exponential size with respect to Def, i.e. $|\mathbb{P}(Def)| = O(2^{|Def|})$, by contrast, it is easy to verify that $|Pos| = O(|Def|)$ yet maintaining a relationship of full completeness between Pos and $\mathbb{P}(Def)$.

8 Related Work

Completeness in abstract interpretation has been first considered by Cousot and Cousot [1979b], who studied the basic properties of complete abstractions in the approximation of least fixpoint semantics. After that seminal paper, a number of works considered complete abstract interpretations in specific applications. It would be impossible to give a full account of all these works; therefore, we consider only some representative works in different areas where abstract interpretation is traditionally used.

In comparative semantics of programming languages, Cousot and Cousot [1992c] and Cousot [1997a] proved that some classical semantics of programming languages, like (demoniac, erratic and angelic) relational, denotational, predicate transformer and axiomatic, are all (fully) complete abstractions of a generalized common SOS operational semantics. In the field of logic program semantics, Giacobazzi [1996] proved that fully complete abstractions relating semantics provide a corresponding hierarchy of models for positive logic programs, while Comini and Levi [1994] based their taxonomy of observables (perfect, denotational, and semi-denotational) on the notion of full completeness, applied to some basic additive operators for building SLD-trees. Recently, Amato and Levi [1997] studied the lattice-theoretic structure of these observables, which basically is the lattice of full complete abstractions for the above additive operators for building SLD-trees, and independently formulated an analogous but weaker result to our Theorem 4.11, for additive operators only. Their proof is not constructive, and they did not provide any constructive method for building fully complete abstract domains for additive functions.

In program analysis and type inference, Cousot [1997b] formalized Hindley monotype semantics as a (non fully) complete abstraction of the more concrete Church/Curry monotype semantics. Sekar et al. [1997] focussed on completeness of Mycroft's [1980, 1981] strictness analysis for functional programs. Their approach is rather different from ours: In fact, they identified the greatest class Cl of functional programs (roughly, all the programs whose computational behavior is unaffected under any variation of constants) such that the strictness analysis of a program P is complete iff $P \in Cl$. Thus, in our terminology, they have found the greatest set of programs Cl such that $\mathit{Strictness} \in \Delta(C, \{T_P\}_{P \in Cl})$, where C is the concrete domain of the standard collecting denotational semantics and T_P is the standard least fixpoint semantic operator for a functional program P. Reddy and Kamin [1992] generalize the above approach of Sekar et al., which first appeared in POPL '91, by studying completeness with respect to a so-called similarity concrete semantics, which is defined in a denotational-like environment. They obtained two completeness results, respectively, for first-order and typed higher-order functional languages, that in a certain precise sense subsume the completeness result of Sekar et al. Thus, also their approach substantially differs from our perspective. Steffen [1989] is one of the first authors isolating completeness as a key property for analysis. He introduces a concept of observation equivalence for abstract interpretations, which is similar to our notion of completeness. However, these approaches have a quite different viewpoint. Steffen's 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 investigated by Steffen et al. [1992]. More recently, Colby [1997] isolated the phenomenon of accumulated imprecision in abstract interpretation. This is essentially the same as the lack of completeness for the considered domain: An error is accumulated in the least fixpoint evaluation of an abstract semantics, analogously to the accumulated rounding error in numerical algorithms. To overcome these problems, Colby [1997] proposed 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 showed 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. We believe instead that the accumulated imprecision is inherently an abstract domain property, and, as we proved, precision can be enforced in any abstract domain of any abstract interpretation by either refining or simplifying them.

Completeness is particularly frequent in abstract model checking and in analysis of reactive systems. In abstract model checking, Clarke et al. [1994] observed that when dealing with fully complete abstract models (that they term exact), a given temporal CTL* formula can be equivalently checked either at the concrete or at the abstract level. Clarke et al. argue that this is an optimal situation, which however is unlikely to be useful in practice for reducing the complexity of verification. Completeness also plays a central role in [Dams et al. 1997] in abstract interpretation for checking properties of reactive systems, where the authors consider it a very desirable situation and hence call it optimality. In Section 5 of that work, the authors give some specific sufficient conditions so that the computed abstract model is complete. Furthermore, Dams et al. observe in Section 6 that the construction of a complete, i.e. optimal, abstract model may require a refinement of the precision of the underlying abstract domain, although they claim that abstract interpretation does not offer a systematic approach to accomplish this task. We believe that all the works on systematic domain refinement in abstract interpretation, e.g. [Cousot and Cousot 1979b, 1992a, 1994; Filé et al. 1996; Giacobazzi and Ranzato 1995, 1997, 1998; Giacobazzi and Scozzari 1997], and in particular the present work on refinements for achieving completeness, can be extremely helpful to both specify optimal abstract models strongly preserving a given property of interest and to tune the accuracy and cost in abstract interpretation of reactive systems.

Completeness is also frequent in complexity studies on program analysis systems. Debray [1995] considered a notion of exactness which is strictly stronger than completeness. He defined the complexity of a dataflow analysis algorithm A as the complexity of typical programs in the class of exactness of A. If A computes the least fixpoint of an abstract operator $T_P^\sharp$, then the above class of exactness is precisely the set of all programs $P \in \mathit{Program}$ such that $\mathit{lfp}(T_P) = \gamma(\mathit{lfp}(T_P^\sharp))$. Although this notion of exactness is evidently stronger than completeness, the author proves that it still represents a reasonable absolute (i.e. not relative to other algorithms) measure of precision for practical purposes. Recently, Deutsch [1997] improved on the precision of Park and Goldberg [1992] escape analysis of strongly typed functional languages, by specifying a complete abstraction which is polynomial in the first-order case. This represents a strong improvement with respect to the original exponential proposal. Similarly, Bagnara et al. [1997] obtained an abstract unification algorithm for pair-sharing analysis of logic programs, which is quadratic in complexity. This is obtained by characterizing a fully complete abstraction of the set-sharing domain, which leads traditionally to exponential algorithms for abstract unification.

Few works in abstract interpretation theory devote particular attention to completeness by its own. The most related work is certainly that of Mycroft [1993], that inspired our approach. Mycroft 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 operator. 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.

9 Conclusion and Further Work

This article investigated the completeness issue in abstract interpretation theory from a (non restrictive) domain perspective. We proposed a number of abstract domain operators that, under very weak hypotheses, systematically refine and simplify domains, so that they give rise to complete abstract interpretations. This is, to the best of our knowledge, a novel approach to the completeness problem, and it provides an appropriate basis for a range of further research directions. Among them, we sketch some lines of research that in our opinion deserve a careful investigation. We think that our completeness transformers may lead to design a somehow *optimal* (abstract interpretation-based) program analysis for a fixed property π of interest. Let us assume that $\mathcal{S}$ is a given concrete semantics, and let π be an abstraction of $\mathcal{S}$ playing the role of the basic property to analyze (e.g., a plain groundness domain may play the role of π in the context of Section 7.3). We claim that an optimal abstract interpretation for π may be devised by characterizing the least fully complete extension of π in $\mathcal{S}$ which in addition is finitely computable. If such an abstract semantics exists, then it should deserve to be rightfully termed optimal for the property π. Further, we believe that our complete and fully complete kernel operators can be fruitfully exploited in combination with some complexity measures in order to derive complete (i.e. most precise) polynomial complexity analyses from exponential ones. In this context, our operators can be used to systematically and often constructively specify optimal analyses, or more in general approximate semantics (e.g. for typing), from simple basic properties of interest.

Acknowledgments

We are particularly grateful to Francesca Scozzari for many valuable discussions and for her contribution to Theorem 4.11.

References

Abramsky, S. and Jung, A. 1994. Domain theory. In *Handbook of Logic in Computer Science*, S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, Eds. Oxford University Press, Oxford, U.K., 2–168.

Amato, G. and Levi, G. 1997. Properties of the lattice of observables in logic programming. In *Proc. of the Italian-Portuguese-Spanish Joint Conf. on Declarative Programming,* (*APPIA-GULP-PRODE '97*). 175–187.

Armstrong, T., Marriott, K., Schachte, P., and Søndergaard, H. 1997. Two classes of Boolean functions for dependency analysis. To appear in *Sci. Comput. Program.* Preliminary version in *Proc. SAS '94*, LNCS 864, pages 266–280, Springer-Verlag, Berlin.

Bagnara, R., Hill, P. M., and Zaffanella, E. 1997. Set-sharing is redundant for pair sharing. In *Proc. of the 4th Internat. Static Analysis Symp.* (*SAS '97*), P. Van Hentenryck, Ed. Lecture Notes in Computer Science, vol. 1302. Springer-Verlag, 53–67.

Barbuti, R., Giacobazzi, R., and Levi, G. 1993. A general framework for semantics-based bottom-up abstract interpretation of logic programs. *ACM Trans. Program. Lang. Syst. 15,* 1, 133–181.

Caseau, Y. 1991. Abstract interpretation of constraints on order-sorted domains. In *Proc. of the 1991 Internat. Logic Programming Symp.* (*ILPS '91*). The MIT Press, Cambridge, Mass., 435–452.

Clarke, E. M., Grumberg, O., and Long, D. E. 1994. Model checking and abstraction. *ACM Trans. Program. Lang. Syst. 16,* 5, 1512–1542.

Codish, M., Dams, D., and Yardeni, E. 1994. Bottom-up abstract interpretation of logic programs. *Theor. Comput. Sci. 124,* 1, 93–126.

Codish, M., Falaschi, M., and Marriott, K. 1994. Suspension analyses for concurrent logic programs. *ACM Trans. Program. Lang. Syst. 16,* 3, 649–686.

Colby, C. 1997. Accumulated imprecision in abstract interpretation. In *Proc. of the 1st ACM Workshop on Automatic Analysis of Software* (*AAS '97*), R. Cleaveland and D. Jackson, Eds. ACM, 77–89.

Comini, M. and Levi, G. 1994. An algebraic theory of observables. In *Proc. of the 1994 Internat. Logic Programming Symp.* (*ILPS '94*), M. Bruynooghe, Ed. The MIT Press, Cambridge, Mass., 172–186.

Cortesi, A., Filé, G., Giacobazzi, R., Palamidessi, C., and Ranzato, F. 1997. Complementation in abstract interpretation. *ACM Trans. Program. Lang. Syst. 19,* 1, 7–47.

Cortesi, A., Filé, G., and Winsborough, W. 1996. Optimal groundness analysis using propositional logic. *J. Logic Program. 27,* 2, 137–167.

Cousot, P. 1978. Méthodes Itératives de Construction et d'Approximation de Points Fixes d'Opérateurs Monotones sur un Treillis, Analyse Sémantique des Programmes. Ph.D. thesis, Université Scientifique et Médicale de Grenoble, Grenoble, France.

Cousot, P. 1996a. Abstract interpretation. *ACM Comp. Surv. 28,* 2, 324–328.

Cousot, P. 1996b. Program analysis position statement: The abstract interpretation perspective. In *ACM Workshop on Strategic Directions in Computing Research,* MIT Lab. for Comput. Sci., Boston, June 1996. Electronically available abstract in *ACM Comput. Surv. 28A*, 4, 1996.

Cousot, P. 1997a. Constructive design of a hierarchy of semantics of a transition system by abstract interpretation (Invited Paper). In *Proc. of the 13th Internat. Symp. on Mathematical Foundations of Programming Semantics* (*MFPS '97*), S. Brookes and M. Mislove, Eds. Electronic Notes in Theoretical Computer Science, vol. 6. Elsevier, Amsterdam.

Cousot, P. 1997b. Types as abstract interpretations (Invited Paper). In *Conference Record of the 24th ACM Symp. on Principles of Programming Languages* (*POPL '97*). ACM Press, New York, 316–331.

Cousot, P. and Cousot, R. 1977. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the 4th ACM Symp. on Principles of Programming Languages* (*POPL '77*). ACM Press, New York, 238–252.

Cousot, P. and Cousot, R. 1979a. A constructive characterization of the lattices of all retractions, preclosure, quasi-closure and closure operators on a complete lattice. *Portug. Math. 38,* 2, 185–198.

Cousot, P. and Cousot, R. 1979b. Systematic design of program analysis frameworks. In *Conference Record of the 6th ACM Symp. on Principles of Programming Languages* (*POPL '79*). ACM Press, New York, 269–282.

Cousot, P. and Cousot, R. 1992a. Abstract interpretation and application to logic programs. *J. Logic Program. 13,* 2-3, 103–179.

Cousot, P. and Cousot, R. 1992b. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation (Invited Paper). In *Proc. of the 4th Internat. Symp. on Programming Language Implementation and Logic Programming* (*PLILP '92*), M. Bruynooghe and M. Wirsing, Eds. Lecture Notes in Computer Science, vol. 631. Springer-Verlag, Berlin, 269–295.

Cousot, P. and Cousot, R. 1992c. Inductive definitions, semantics and abstract interpretation. In *Conference Record of the 19th ACM Symp. on Principles of Programming Languages* (*POPL '92*). ACM Press, New York, 83–94.

Cousot, P. and Cousot, R. 1993. Galois connection based abstract interpretation for strictness analysis (Invited Paper). In *Proc. of the Internat. Conf. on Formal Methods in Programming and their Applications* (*FMPA '93*), D. Bjørner, M. Broy, and I. Pottosin, Eds. Lecture Notes in Computer Science, vol. 735. Springer-Verlag, Berlin, 98–127.

Cousot, P. and Cousot, R. 1994. Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and PER analysis of functional languages) (Invited Paper). In *Proc. of the 1994 IEEE Internat. Conf. on Computer Languages* (*ICCL '94*). IEEE Computer Society Press, Los Alamitos, Calif., 95–112.

Cousot, P. and Cousot, R. 1995. Compositional and inductive semantic definitions in fixpoint, equational, constraint, closure-condition, rule-based and game-theoretic form (Invited Paper). In *Proc. of the 7th Internat. Conf. on Computer Aided Verification* (*CAV '95*), P. Wolper, Ed. Lecture Notes in Computer Science, vol. 939. Springer-Verlag, Berlin, 293–308.

Dams, D., Gerth, R., and Grumberg, O. 1997. Abstract interpretation of reactive systems. *ACM Trans. Program. Lang. Syst. 19,* 2, 253–291.

Davey, B. A. and Priestley, H. A. 1990. *Introduction to Lattices and Order.* Cambridge University Press, Cambridge, U.K.

Debray, S. K. 1994. Formal bases for dataflow analysis of logic programs. In *Advances in Logic Programming,* Chapter 3, G. Levi, Ed. Clarendon Press, Oxford, U.K., 115–182.

Debray, S. K. 1995. On the complexity of dataflow analysis of logic programs. *ACM Trans. Program. Lang. Syst. 17,* 2, 331–365.

Deutsch, A. 1997. On the complexity of escape analysis. In *Conference Record of the 24th ACM Symp. on Principles of Programming Languages* (*POPL '97*). ACM Press, New York, 358–371.

Falaschi, M., Levi, G., Martelli, M., and Palamidessi, C. 1989. Declarative modeling of the operational behavior of logic languages. *Theor. Comput. Sci. 69,* 3, 289–318.

Filé, G., Giacobazzi, R., and Ranzato, F. 1996. A unifying view of abstract domain design. *ACM Comput. Surv. 28,* 2, 333–336.

Filé, G. and Ranzato, F. 1998. The powerset operator on abstract interpretations. To appear in *Theor. Comput. Sci.* Preliminary version in *Proc. ILPS '94*, pages 655–669, The MIT Press, Cambridge, Mass.

Giacobazzi, R. 1996. "Optimal" collecting semantics for analysis in a hierarchy of logic program semantics. In *Proc. of the 13th Internat. Symp. on Theoretical Aspects of Computer Science* (*STACS '96*), C. Puech and R. Reischuk, Eds. Lecture Notes in Computer Science, vol. 1046. Springer-Verlag, Berlin, 503–514.

Giacobazzi, R., Palamidessi, C., and Ranzato, F. 1996. Weak relative pseudo-complements of closure operators. *Algebra Universalis 36,* 3, 405–412.

Giacobazzi, R. and Ranzato, F. 1995. Functional dependencies and Moore-set completions of abstract interpretations and semantics. In *Proc. of the 1995 Internat. Logic Programming Symp.* (*ILPS '95*), J. Lloyd, Ed. The MIT Press, Cambridge, Mass., 321–335.

Giacobazzi, R. and Ranzato, F. 1997. Refining and compressing abstract domains. In *Proc. of the 24th Internat. Colloq. on Automata, Languages and Programming* (*ICALP '97*), P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, Eds. Lecture Notes in Computer Science, vol. 1256. Springer-Verlag, Berlin, 771–781.

Giacobazzi, R. and Ranzato, F. 1998. Optimal domains for disjunctive abstract interpretation. To appear in *Sci. Comput. Program.* Preliminary version in *Proc. ESOP '96*, LNCS 1058, pages 141–155, Springer-Verlag, Berlin.

Giacobazzi, R. and Scozzari, F. 1997. Intuitionistic implication in abstract interpretation. In *Proc. of the 9th Internat. Symp. on Programming Languages, Implementations, Logics and Programs* (*PLILP'97*), H. Glaser, P. Hartel, and H. Kuchen, Eds. Lecture Notes in Computer Science, vol. 1292. Springer-Verlag, Berlin, 175–189.

Gierz, G., Hofmann, K. H., Keimel, K., Lawson, J. D., Mislove, M., and Scott, D. S. 1980. *A Compendium of Continuous Lattices.* Springer-Verlag, Berlin.

Graf, S. 1997. Characterization of a sequentially consistent memory and verification of a cache memory by abstraction. To appear in *Distributed Computing*. Preliminary version in *Proc. CAV '94*, LNCS 818, Springer-Verlag, Berlin.

Helm, R., Marriott, K., and Odersky, M. 1995. Spatial query optimization: From boolean constraints to range queries. *J. Comput. Syst. Sci. 51,* 2, 197–210.

Hughes, R. J. M. 1988. Backwards analysis of functional programs. In *Partial Evaluation and Mixed Computation,* Proc. IFIP TC2 Workshop, D. Bjørner, A. P. Ershov, and N. D. Jones, Eds. North-Holland, Amsterdam, 155–169.

Jensen, T. P. 1992. Disjunctive strictness analysis. In *Proc. of the 7th IEEE Symp. on Logic in Computer Science* (*LICS '92*). IEEE Computer Society Press, Los Alamitos, Calif., 174–185.

Lassez, J.-L., Maher, M. J., and Marriott, K. 1988. Unification revisited. In *Foundations of Deductive Databases and Logic Programming*, J. Minker, Ed. Morgan Kaufmann, Los Altos, Calif., 587–625.

Loiseaux, C., Graf, S., Sifakis, J., Bouajjani, A., and Bensalem, S. 1995. Property preserving abstractions for the verification of concurrent systems. *Formal Methods Syst. Des. 6*, 11–44.

Marriott, K. and Søndergaard, H. 1993. Precise and efficient groundness analysis for logic programs. *ACM Lett. Program. Lang. Syst. 2,* 1-4, 181–196.

Marriott, K., Søndergaard, H., and Jones, N. D. 1994. Denotational abstract interpretation of logic programs. *ACM Trans. Program. Lang. Syst. 16,* 3, 607–648.

Morgado, J. 1960. Some results on the closure operators of partially ordered sets. *Portug. Math. 19,* 2, 101–139.

Mycroft, A. 1980. The theory and practice of transforming call-by-need into call-by-value. In *Proc. of the 4th Internat. Symp. on Programming*, B. Robinet, Ed. Lecture Notes in Computer Science, vol. 83. Springer-Verlag, Berlin, 270–281.

Mycroft, A. 1981. Abstract interpretation and optimising transformations for applicative programs. Ph.D. thesis, CST-15-81, Univ. of Edinburgh, Edinburgh, Scotland.

Mycroft, A. 1993. Completeness and predicate-based abstract interpretation. In *Proc. of the ACM Symp. on Partial Evaluation and Program Manipulation* (*PEPM '93*). ACM Press, New York, 179–185.

Mycroft, A. and Nielson, F. 1983. Strong abstract interpretation using power domains. In *Proc. of the 10th Internat. Colloq. on Automata, Languages and Programming* (*ICALP '83*), J. Díaz, Ed. Lecture Notes in Computer Science, vol. 154. Springer-Verlag, Berlin, 536–547.

Nielson, H. R. and Nielson, F. 1989. Transformations on higher-order functions. In *Proc. of the 4th Internat. Conf. on Functional Programming Languages and Computer Architecture* (*FPCA '89*). ACM Press, New York, 129–143.

Nielson, H. R. and Nielson, F. 1992. Bounded fixed point iteration. In *Conference Record of the 19th ACM Symp. on Principles of Programming Languages* (*POPL '92*). ACM Press, New York, 71–82.

Park, Y. and Goldberg, B. 1992. Escape analysis on lists. In *Proc. of the 1992 ACM Conf. on Programming Language Design and Implementation* (*PLDI '92*). ACM Press, New York, 116–127.

Plaisted, D. 1981. Theorem proving with abstraction. *Artif. Intell. 16*, 47–108.

Reddy, U. S. and Kamin, S. N. 1992. On the power of abstract interpretation. In *Proc. of the 1992 IEEE Internat. Conf. on Computer Languages* (*ICCL '92*). IEEE Computer Society Press, Los Alamitos, Calif.

Sekar, R. C., Mishra, P., and Ramakrishnan, I. V. 1997. On the power and limitation of strictness analysis. *J. ACM 44,* 3, 505–525.

Steffen, B. 1989. Optimal data flow analysis via observational equivalence. In *Proc. of the 14th Internat. Symp. on Mathematical Foundations of Computer Science* (*MFCS '89*). Lecture Notes in Computer Science, vol. 379. Springer-Verlag, Berlin.

Steffen, B., Jay, C. B., and Mendler, M. 1992. Compositional characterization of observable program properties. *Informatique Theéorique et Applications* (*AFCET*) *26*, 403–424.

Ward, M. 1942. The closure operators of a lattice. *Ann. Math. 43,* 2, 191–196.