Making Abstract Interpretations Complete

ROBERTO GIACOBAZZI

Università di Verona, Verona, Italy

FRANCESCO RANZATO

Università di Padova, Padova, Italy

AND

FRANCESCA SCOZZARI

École Polytechnique, Palaiseau, France

Abstract. Completeness is an ideal, although uncommon, feature of abstract interpretations, formalizing the intuition that, relatively to the properties encoded by the underlying abstract domains, there is no loss of information accumulated in abstract computations. Thus, complete abstract interpretations can be rightly understood as optimal. We deal with both pointwise completeness, involving generic semantic operations, and (least) fixpoint completeness. Completeness and fixpoint completeness are shown to be properties that depend on the underlying abstract domains only. Our primary goal is then to solve the problem of making abstract interpretations complete by minimally extending or restricting the underlying abstract domains. Under the weak and reasonable hypothesis of dealing with continuous semantic operations, we provide constructive characterizations for the least complete extensions and the greatest complete restrictions of abstract domains. As far as fixpoint completeness is concerned, for merely monotone semantic operators, the greatest restrictions of abstract domains are constructively characterized, while it is shown that the existence of least extensions of abstract domains cannot be, in general, guaranteed, even under strong hypotheses. These methodologies, which in finite settings give rise to effective algorithms, provide advanced formal tools for manipulating and comparing abstract interpretations, useful both in static program analysis and in semantics design. A number of examples illustrating these techniques are given.

Categories and Subject Descriptors: D.3.1 [**Programming Languages**]: Formal Definitions and Theory—*semantics*; F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages—*program analysis*

General Terms: Languages, Theory

Authors' addresses: R. Giacobazzi, Dipartimento Scientifico e Tecnologico, Università di Verona, Strada Le Grazie, Ca' Vignal 2, 37134 Verona, Italy, e-mail: giaco@sci.univr.it; F. Ranzato, Dipartimento di Matematica Pura ed Applicata, Università di Padova, Via Belzoni 7, 35131 Padova, Italy, e-mail: franz@math.unipd.it; F. Scozzari, Laboratoire d'Informatique, École Polytechnique, F-91128 Palaiseau Cedex, France, e-mail: scozzari@lix.polytechnique.fr.

Permission to make digital/hard copy of part or all of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery (ACM), Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.
© 2000 ACM 0004-5411/00/0300-0361 $05.00

Additional Key Words and Phrases: Abstract domain, abstract interpretation, complete core, complete shell, completeness, fixpoint completeness, integer interval analysis, logic program analysis

1. *Introduction*

According to a fine 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 designing and then validating static (compile-time) 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 model checking [Clarke et al. 1994; Dams et al. 1997; Loiseaux et al. 1995], verification of distributed memory systems [Graf 1999], process calculi [Cleaveland and Riely 1994; Venet 1996], security [Ørbæk 1995; Ørbæk and Palsberg 1997], type inference [Cousot 1997; Monsuez 1995], theorem proving [Plaisted 1981], constraint solving [Caseau 1991] and comparative semantics [Comini and Levi 1994; Cousot 2000; Cousot and Cousot 1992b, 1997; 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, for example, a program, at different levels of abstraction, is a suitable approximation of its formal semantics. The following discussion, although somewhere loosely tailored for the classical case of programming languages and their semantics approximation, is relevant in all the above fields, and potentially in every area where abstract interpretation is applicable.

1.1. A SYNOPSIS OF ABSTRACT INTERPRETATION BASICS. Let us first recall some relevant basic ideas of the abstract interpretation technique. In the classical Cousot and Cousot [1977; 1979] framework, an abstract interpretation is defined as a nonstandard, approximated (called abstract) semantics obtained from the standard (called concrete) one by substituting the actual (concrete) domains of computation and their basic (concrete) semantic operations with, respectively, *abstract domains* and corresponding *abstract semantic operations.* The basic intuition is that abstract domains are representations of some properties of interest about concrete domains' values, while abstract operations simulate, over the properties encoded by the abstract domains, the behavior of their concrete counterparts. Abstract semantics schemes are therefore parameterized with respect to abstract domains and operations, and several abstract interpretations at various levels of precision lead to hierarchies of abstract semantics, thus enabling layered modular abstract semantics design [Cousot 1996a; 1996b]. Let us consider the case of a single concrete semantic operation f: $C^n \rightarrow D$ defined over (n-tuples of) an input concrete domain C and assuming its values on an output concrete domain D. Then, an abstract interpretation can be formulated by specifying corresponding abstract domains A and B and an abstract semantic operation $f^\sharp$: $A^n \rightarrow B$. The notion of approximation is encoded by suitable partial orderings on domain's objects. Both concrete and abstract domains are assumed to be complete lattices with respect to their approximation orders.[1] The orderings on the concrete and abstract domains describe the relative precision of domain values, somehow in a dual fashion with respect to the domains of standard denotational semantics: $x \leq y$ means that x is more precise than y, that is, y carries less information than x—top elements represent lack of information. Therefore, as a very basic requirement, concrete and abstract operations preserve the approximation orderings, that is, they are monotone. Concrete and abstract universes are related by pairs of adjoint maps (also known as Galois connections): Left adjoints $\alpha_{C,A}: C \to A$ and $\alpha_{D,B}: D \to B$ are called abstraction maps, and right adjoints $\gamma_{A,C}: A \to C$ and $\gamma_{B,D}: B \to D$ are called concretization maps. Here, the intended meaning is that an abstract value $a \in A$ approximates a concrete value $c \in C$ if $c \leq_C \gamma_{A,C}(a)$, or equivalently (by adjunction), if $\alpha_{C,A}(c) \leq_A a$. The concrete value corresponding to an abstract denotation a is therefore $\gamma_{A,C}(a)$, whereas the adjunction guarantees that $\alpha_{C,A}(c)$ is the best possible approximation of c in A.

Correctness is a basic requirement of any approximation technique, and this holds also for abstract interpretation: An abstract interpretation $\langle A, B, f^\sharp\rangle$ is sound for $\langle C, D, f\rangle$, whenever for all $\langle c_1, \ldots, c_n\rangle \in C$, $\alpha_{D,B}(f(\langle c_1, \ldots, c_n\rangle)) \leq_B f^\sharp(\langle \alpha_{C,A}(c_1), \ldots, \alpha_{C,A}(c_n)\rangle)$. This is more compactly denoted by $\alpha_{D,B} \circ f \sqsubseteq f^\sharp \circ \langle \alpha_{C,A}, \ldots, \alpha_{C,A}\rangle$, where $\sqsubseteq$ denotes pointwise ordering between functions. Following a standard terminology, $f^\sharp$ is also called a correct approximation (over A and B) of f. The intuition here is clear: If a tuple $\langle c_1, \ldots, c_n\rangle$ of concrete values is approximated by $\langle a_1, \ldots, a_n\rangle$, then a concrete computation step $f(\langle c_1, \ldots, c_n\rangle)$ is still approximated on B by $f^\sharp(\langle a_1, \ldots, a_n\rangle)$. Let us consider the case of least fixpoint-based semantics, for example, of programming languages. Let $[\![\cdot]\!]$: *Program* $\to C$ be a semantic evaluation function, associating with each $P \in$ *Program* its least fixpoint semantics $[\![P]\!] \overset{\text{def}}{=} \mathit{lfp}(T_P)$, where every $T_P: C \to C$ is a concrete monotone operator. Then, an abstract interpretation is specified by an abstract domain A and by a family of monotone abstract operators $T_P^\sharp: A \to A$, indexed over programs. In this case, the abstract interpretation $\langle A, A, \{T_P^\sharp\}_{P\in \mathit{Program}}\rangle$ is fixpoint sound for $\langle C, C, \{T_P\}_{P\in \mathit{Program}}\rangle$, whenever, for all programs P, $\alpha_{C,A}(\mathit{lfp}(T_P)) \leq_A \mathit{lfp}(T_P^\sharp)$. A basic abstract interpretation result tells us that soundness implies fixpoint soundness. This is a well-known and widely applied verification technique (see, for instance, the classical strictness analysis of functional programs and groundness analysis of logic programs [Cousot and Cousot 1993; Marriott et al. 1994; Mycroft 1981]): Soundness is usually easier to check than fixpoint soundness, which may result hard to prove, due to its fixpoint nature.

1.2. Completeness in Abstract Interpretation.

In the 1990s, there have been a number of works studying *completeness* in abstract interpretation[2] (see

[1] This assumption is mainly made to keep the technical development of the article reasonably simple. We will briefly discuss in Section 8 how, thanks to a recent result by Ranzato [1999], concrete and abstract domains could be required to be mere directed-complete partial orders (CPOs), still retaining all the significant results, but paying for heavier order-theoretic details.

[2] In abstract interpretation literature, unfortunately, there is not uniformity on the terminology related to completeness issues. For instance: Cousot and Cousot [1995, Section 8] and Cousot [1997a, Section 2] alternatively use *exactness* and *faithfulness*; in abstract model checking, Clarke et al. [1994]

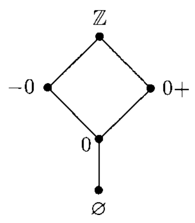


FIG. 1. The abstract domain *Sign*.

Section 7 for a discussion on completeness in abstract interpretation literature). Maintaining the above standard scenario, given an abstract interpretation $\langle A, B, f^{\#}\rangle$ sound for $\langle C, D, f\rangle$, $\alpha_{D,B}(f(\langle c_1, \ldots, c_n\rangle)) <_B f^{\#}(\langle \alpha_{C,A}(c_1), \ldots, \alpha_{C,A}(c_n)\rangle)$, for some $\langle c_1, \ldots, c_n\rangle \in C^n$, means that a strict loss of information has occurred in simulating the behavior of the concrete operation f by $f^{\#}$ over A and B. Completeness means that, relatively to the semantic properties encoded by the abstract domains, such losses of information never occur. Thus, $\langle A, B, f^{\#}\rangle$ is defined to be *complete* for $\langle C, D, f\rangle$, whenever $\alpha_{D,B} \circ f = f^{\#} \circ \langle \alpha_{C,A}, \ldots, \alpha_{C,A}\rangle$. While soundness is *the* basic requirement for any abstract interpretation, completeness is instead an ideal and uncommon situation. In this case, roughly speaking, the abstract semantics is able to take full advantage of the expressive power of the underlying abstract domains.

Let us employ the classical "rule of signs" [Cousot and Cousot 1977; 1979] as a simple example. The abstract domain *Sign* depicted in Figure 1 is used to represent the sign of sets of integers in $\langle \wp(\mathbb{Z}), \subseteq\rangle$, which plays the role of concrete domain. Objects in *Sign* are self-explanatory, namely *Sign* is related to $\wp(\mathbb{Z})$ by an obvious adjunction. Consider a concrete pointwise multiplication operation $*$: $\wp(\mathbb{Z})^2 \to \wp(\mathbb{Z})$ defined by $X * Y \overset{\text{def}}{=} \{x \cdot y \mid x \in X, y \in Y\}$—pointwise addition is similarly defined. Then, the rule of signs over the abstract domain *Sign* can be formalized by the most obvious abstract multiplication $*^{\#}$: $Sign^2 \to Sign$; for example, $-0 *^{\#} 0+ = -0$ and $0 *^{\#} 0+ = 0$. Thus, it should be evident that $\langle Sign, Sign, *^{\#}\rangle$ is complete for $\langle \wp(\mathbb{Z}), \wp(\mathbb{Z}), *\rangle$, namely, for every pair of sets of integers $Z_1, Z_2 \in \wp(\mathbb{Z})$, $\alpha(Z_1 * Z_2) = \alpha(Z_1) *^{\#} \alpha(Z_2)$. Here, this completeness relationship precisely asserts the validity of the rule of signs, that is, the sign of any integer multiplication can be exactly obtained by the rule of signs, with no loss of precision. For example, $\alpha(\{-1\} * \{-2\}) = 0+ = -0 *^{\#} -0 = \alpha(\{-1\}) *^{\#} \alpha(\{-2\})$. On the contrary, this ideal situation does not hold for addition. The obvious abstract addition $+^{\#}$: $Sign^2 \to Sign$ over *Sign* is not complete: For instance, $\alpha(\{-1\} + \{3\}) = 0+ \neq -0 +^{\#} 0+ = \mathbb{Z}$.

Likewise, fixpoint completeness is dual to fixpoint soundness. A least fixpoint abstract interpretation $\langle A, \{T_P^{\#}\}_{P\in Program}\rangle$ is *fixpoint complete* for $\langle C, \{T_P\}_{P\in Program}\rangle$ whenever, for all programs P, $\alpha_{C,A}(lfp(T_P)) = lfp(T_P^{\#})$. Thus, fixpoint completeness means that there is no information loss when globally looking at least fixpoints. As first noted by Cousot and Cousot [1979], completeness implies fixpoint completeness. Analogously to soundness, completeness is, in general, easier to prove than fixpoint completeness.

also use exactness while Dams et al. [1997] and Cleaveland et al. [1995] use optimality. Here, we adopt the term completeness, as it is typically used in contrast to the notion of soundness.

Let us consider the above rule of signs example. Let f: $\wp(\mathbb{Z}) \to \wp(\mathbb{Z})$ be defined by $f \overset{\text{def}}{=} \lambda X.\{0\} \cup \{x + 2 \mid x \in X\}$, and let $f^\sharp$: *Sign* $\to$ *Sign* be defined by $f^\sharp \overset{\text{def}}{=} \{\mathbb{Z} \mapsto \mathbb{Z}, 0+ \mapsto 0+, -0 \mapsto \mathbb{Z}, 0 \mapsto 0+, \varnothing \mapsto 0\}$. Both f and $f^\sharp$ are monotone, and it is not too difficult to observe that $f^\sharp$ is complete for f, and therefore fixpoint complete as well, that is, $\alpha(lfp(f)) = lfp(f^\sharp)$. In fact, it turns out that $lfp(f) = \{x \in \mathbb{Z} \mid x \geq 0, x \text{ even}\}$ and $lfp(f^\sharp) = 0+$. Instead, by considering an "ill-defined" abstract operation $g^\sharp$: *Sign* $\to$ *Sign* given by $\{\mathbb{Z} \mapsto \mathbb{Z}, 0+ \mapsto \mathbb{Z}, -0 \mapsto \mathbb{Z}, 0 \mapsto 0+, \varnothing \mapsto 0\}$, $g^\sharp$ is still a correct approximation of f, and therefore fixpoint soundness holds, but $g^\sharp$ is neither complete nor fixpoint complete for f. In fact, it turns out that $\alpha(f(\{0, 1\})) = 0+ \neq \mathbb{Z} = g^\sharp(\alpha(\{0, 1\}))$ and $\alpha(lfp(f)) = 0+ \neq \mathbb{Z} = lfp(g^\sharp)$.

Completeness is typically recurrent in comparative semantics, that is, in studying formal semantics (e.g., of programming languages) at different levels of abstraction. For many pairs of concrete semantics (e.g., trace-based operational and denotational), their corresponding computational domains are sufficiently rich of information so that, whenever their relationship is formalized by abstract interpretation, completeness holds. For instance, Cousot and Cousot [1992b; 1995] and Cousot [1997a] consider some classical program semantics, like denotational, predicate transformer and axiomatic, as complete abstractions of a generalized SOS trace-based operational program semantics, while Cousot and Cousot [1997] present several complete abstractions of algebraic polynomial systems (see Section 7 for other examples of complete abstract interpretations). Completeness for classes of expressive temporal logic formulae, also known as strong preservation, is highly desirable in abstract model-checking, in order to avoid of getting "don't know" answers whenever a system is correct [Cleaveland et al. 1995; Dams 1996; Dams et al. 1997]. Instead, on the static program analysis side, decidability issues often force to sacrifice completeness to achieve termination and/or efficiency. Clearly, if some computational program property, as formalized by the least fixpoint of an operator $T_P^\sharp$ on some finite abstract domain A, is undecidable, then undecidability surely prevents completeness for every $T_P^\sharp$. Although practical program analysis systems are rarely complete for their reference concrete semantics, they may instead be complete relatively to some, approximated and possibly decidable, intermediate abstractions. As we will see more in detail in Section 6.2, in this context, completeness turns out to be a useful tool in order to tune static analyses in accuracy and cost. These argumentations probably stimulated the trend of research on completeness in abstract interpretation mentioned above.

The problem of achieving completeness for an abstract interpretation by enhancing either its abstract semantic operations or its abstract domains, has been investigated by a number of authors (see Section 7). 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 problem of making a generic abstract interpretation complete in the best possible way—that is, by minimally extending or restricting the underlying abstract domains and operations—is still, to the best of our knowledge, open. This article puts forward a solution to this problem.

1.3. Completeness is an Abstract Domain Property.

It is well known since Cousot and Cousot [1979] that, given f: $C^n \to D$, any pair of abstract domains A and B induces the so-called canonical *best correct approximation* $f^{b_{A,B}}$: $A^n \to B$ of f defined by $f^{b_{A,B}} \stackrel{\text{def}}{=} \alpha_{D,B} \circ f \circ \langle \gamma_{A,C}, \ldots, \gamma_{A,C} \rangle$. This terminology is justified by the fact that any $f^\sharp$: $A^n \to B$ is a correct approximation of (i.e., sound for) f iff $f^{b_{A,B}} \sqsubseteq f^\sharp$. Consequently, any pair of input and output abstract domains always induces an (automatically) sound abstract operation defined over them. This is not in general true for completeness: Not every pair of abstract domains induces a complete abstract operation. However, whenever there exists a (fixpoint) complete abstract operation $f^\sharp$ over $\langle A, B \rangle$ then the best correct approximation $f^{b_{A,B}}$ is (fixpoint) complete as well. Hence, the following characterization holds:

It is possible to define a (fixpoint) complete abstract semantic operation on the abstract domains A and B if and only if $f^{b_{A,B}}$ is (fixpoint) complete.

This means that both completeness and fixpoint completeness are properties which depend on the underlying abstract domains only.

1.4. MAKING ABSTRACT INTERPRETATIONS 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 $\mathfrak{L}_C$, where, for any two abstract domains A, $B \in \mathfrak{L}_C$, $A \sqsubseteq B$ holds when A is more precise (i.e., less abstract) than B (roughly, when $B \subseteq A$) [Cousot and Cousot 1979]. Thus, for example, statements like "there exists the most abstract domain X abstracting C and such that X satisfies property $\mathcal{P}$" are formalized by exploiting *lub*'s of $\mathfrak{L}_C$: "$\sqcup\{X \in \mathfrak{L}_C \mid X$ satisfies $\mathcal{P}\}$ satisfies $\mathcal{P}$". As the above observations hint, we attack the problem of making abstract interpretations complete from a nonrestrictive domain perspective, by considering completeness as an abstract domain property. Thus, we will say that a pair of abstract domains $\langle A, B \rangle$ is complete for f iff $f^{b_{A,B}}$ is complete for f, and similarly for fixpoint completeness.

Given a concrete interpretation f: $C^n \to D$, and a pair of input and output abstractions $\langle A, B \rangle \in \mathfrak{L}_C \times \mathfrak{L}_D$, our goal is to answer to the following questions:

(i) Does there exist the most abstract domain $A^e \in \mathfrak{L}_C$ extending A such that $\langle A^e, B \rangle$ is complete for f? Does there exist the most concrete domain $B^r \in \mathfrak{L}_D$ restricting B such that $\langle A, B^r \rangle$ is complete for f?

Analogous dual questions, where the roles of A and B are exchanged, are formulated and considered. Moreover, whenever f: $C^n \to C$, that is, input and output concrete domains coincide, and $A \in \mathfrak{L}_C$, we also consider the following problems:

(ii) Does there exist the most abstract domain $A^e \in \mathfrak{L}_C$ extending A such that $\langle A^e, A^e \rangle$ is complete for f? Does there exist the most concrete domain $A^r \in \mathfrak{L}_C$ restricting A such that $\langle A^r, A^r \rangle$ is complete for f?

If problem (i) admits solutions then, following a suggestive terminology, A^e is called the *complete shell* of A (for f) relative to B, and B^r the *complete core* of B (for f) relative to A. We will prove that the dual problems of the complete shell of B relative to A and of the complete core of A relative to B either admit trivial solutions or do not admit solutions, and therefore are meaningless. In problem (ii), we are interested in extending or restricting the abstract domain $A \in \mathfrak{L}_C$, simultaneously used as input and output abstract domain. In this case, whenever they exist, A^e and A^r are called, respectively, the *absolute complete shell* and the *absolute complete core* of A (for f). On the fixpoint side, the following fixpoint completeness problems are analogous to (ii). Let $f\colon C \to C$ be a monotone operator.

(iii) Does there exist the most abstract domain $A^e \in \mathfrak{L}_C$ extending A such that A^e is fixpoint complete for f? Does there exist the most concrete domain $A^r \in \mathfrak{L}_C$ restricting A such that A^r is fixpoint complete for f?

Accordingly, solutions to (iii), when they exist, are called respectively, the *fixpoint complete shell* and the *fixpoint complete core* of A (for f).

Complete shells are an enhancement of an abstract domain A which is as close as possible to A, while complete cores act exactly in the opposite direction. This way of aiming at minimally transforming abstract domains follows a general pattern introduced by Filé et al. [1996] and Giacobazzi and Ranzato [1998b]: In this sense, complete shells and cores are instances of, respectively, abstract domain refinements and simplifications.

Under the weak and reasonable hypothesis of dealing with *continuous* semantic functions, we give a key constructive characterization of complete abstract interpretations. More precisely, given a continuous concrete operation $f\colon C^n \to D$, we show that $\langle A, B\rangle$ (viz., $f^{b_{A,B}}\colon A^n \to B$) is complete for f iff A is more concrete than a certain domain $R_f(B)$ depending on B iff B is more abstract than a certain domain $L_f(A)$ depending on A. In particular, the mappings $L_f\colon \mathfrak{L}_C \to \mathfrak{L}_D$ and $R_f\colon \mathfrak{L}_D \to \mathfrak{L}_C$ form an adjunction. This result allows us to solve constructively the problems (i) and (ii) above:

(1) The complete shell of A relative to B is the most abstract domain which contains both A and $R_f(B)$ (hence, this is the so-called reduced product of A and $R_f(B)$); the complete core of B relative to A is the most concrete domain contained in both B and $L_f(A)$ (with suitable uniform representations, this is simply the set-intersection of B and $L_f(A)$).
(2) Absolute complete shells and cores are constructively characterized as, respectively, greatest fixpoints and least fixpoints of suitable operators on the lattice of abstract interpretations, whose definitions rely on R_f and L_f above.

As far as fixpoint completeness is concerned, we will first give some negative examples that prevent the possibility of finding some reasonable sufficient conditions on the concrete domain and/or on the semantic operators ensuring the existence of fixpoint complete shells. Note that this does not mean that problem (iii) never admits solutions for shells, because specific fixpoint complete shells of some abstract domains may well exist. On the other hand, a constructive characterization of fixpoint completeness allows us to show that fixpoint complete cores, for merely monotone semantic operators, can be always constructively obtained. It should be remarked that we give *constructive* results of existence for (fixpoint) complete shells and cores. This means that our results provide methodologies to compute "by hand" relative or absolute complete shells and cores, and fixpoint complete cores of abstract domains, and that in finite settings, all these methods provide terminating algorithms to automatically make abstract interpretations complete.

At this point, let us give a simple example concerning classical Mycroft's strictness analysis for functional programs [Burn et al. 1986; Mycroft 1981]. Consider the following function F of type $\mathsf{Nat} \times \mathsf{Nat} \to \mathsf{Bool}$:

$$F(\langle x, y\rangle) \overset{\text{def}}{=} \textit{if } (x = 3 \textit{ and } y = 3) \textit{ then true else } \bot$$

Following Burn et al. [1986, Section 4], from F one gets in the most natural way its denotational "collecting" semantics f: $\mathbf{P}(\mathbb{N}_\bot \times \mathbb{N}_\bot) \to \mathbf{P}(\mathit{Bool}_\bot)$, where $\mathbf{P}$ is the Hoare powerdomain operator and $\bot$ denotes undefinedness (i.e., both nontermination and error). Let $S \overset{\text{def}}{=} \{0 < 1\}$ be the basic strictness domain, abstracting both $\mathbf{P}(\mathbb{N}_\bot)$ and $\mathbf{P}(\mathit{Bool}_\bot)$, and such that $S \times S$ abstracts $\mathbf{P}(\mathbb{N}_\bot \times \mathbb{N}_\bot)$. Concretization and abstraction maps are the usual ones, for example, $\gamma(\langle 0, 0\rangle) = \{\langle\bot, \bot\rangle\}$ and $\gamma(\langle 0, 1\rangle) = \{\langle\bot, x\rangle \mid x \in \mathbb{N}_\bot\}$. Then, the best correct approximation f^b: $S \times S \to S$ of f is as follows: $f^b = \{\langle 0, 0\rangle \mapsto 0, \langle 0, 1\rangle \mapsto 0, \langle 1, 0\rangle \mapsto 0, \langle 1, 1\rangle \mapsto 1\}$. Clearly, it turns out that f^b is not complete: For instance, $\alpha(f(\{\langle\bot, \bot\rangle, \langle 4, 5\rangle\})) = \alpha(\{\bot\}) = 0$, while $f^b(\alpha(\{\langle\bot, \bot\rangle, \langle 4, 5\rangle\})) = f^b(\langle 1, 1\rangle) = 1$. These phenomena of incompleteness in strictness analysis are analyzed in depth by Reddy and Kamin [1993] and Sekar et al. [1997], who, however, do not investigate the issue of achieving completeness by minimally modifying the abstract domains. Since the collecting semantics f is obviously continuous, our methodologies allow us, for example, to constructively derive the complete shell, denoted by $\mathcal{S}(S \times S)$, of the input abstract domain $S \times S$ (here, $S \times S$ is thought of as a whole), relative to S for the collecting semantics f. It should be clear that by adding a point to $S \times S$ which is able to represent the information that the first and second components are surely not simultaneously equal to $3 \in \mathbb{N}_\bot$, one gets an abstract domain inducing a complete abstract interpretation. Indeed, our results permit to constructively derive that $\mathcal{S}(S \times S) = (S \times S) \cup \{\langle\neq, \neq\rangle\}$, where $\gamma(\langle\neq, \neq\rangle) = (\mathbb{N}_\bot \times \mathbb{N}_\bot)\setminus\{\langle 3, 3\rangle\}$. In this way, one gets a best correct approximation f^{b*}: $\mathcal{S}(S \times S) \to S$ such that $f^{b*}(\langle\neq, \neq\rangle) = 0$, and therefore completeness has been achieved.

1.5. Applications. Our results might find applications in every field where Galois connection-based abstract interpretation is used, like for instance in the areas mentioned at the beginning. In this article, we present some applications in the field of static program analysis, which exemplify some possible uses of our results.

—As we sketched above, the rule of signs abstract domain for integer variable analysis is obviously not complete for the addition operation. We characterize the absolute complete shell of the rule of signs domain for integer addition: It turns out that this is precisely the well-known Cousot and Cousot [1976; 1977] domain of integer intervals, thus providing an interesting constructive characterization of the lattice of intervals.

—In the context of systematic design of program analyses, we show how the notion of absolute complete shell can be exploited for carefully tuning the efficiency/precision trade-off when improving the precision of an analysis by abstract domain refinements. It may happen that an abstract domain refinement step overflows the necessary information for an analysis, by adding some costly unnecessary information. In such cases, our completeness tools allow us to modify such a refinement operator by devising an intelligent refinement strategy which is able to take sensibly into account efficiency/precision issues.

—The above efficiency-oriented strategy for refining abstract domains is applied to some well-known abstract domains for ground-dependency analysis of logic languages, like *Def*, *Pos* and their common disjunctive completion $\mathbb{P}(\mathit{Def})$.[3] More in detail, by considering a standard bottom-up least fixpoint concrete semantics, we prove that the absolute complete shell of *Def* with respect to the disjunctive completion $\mathbb{P}(\mathit{Def})$ coincides with *Pos.* Thus, an intelligent disjunctive completion refinement should refine *Def* to *Pos* rather than to the canonical $\mathbb{P}(\mathit{Def})$. The usefulness of such an approach can be appreciated by considering that $\mathbb{P}(\mathit{Def})$ is enormously less efficient than *Pos* (the former has an exponential size with respect to the latter), while *Pos* turns out to be complete for $\mathbb{P}(\mathit{Def})$.

This article is an expanded and revised version of two conference articles appeared in the Proceedings of the Sixth International Conference on Algebraic Methodology and Software Technology, held in Sydney, Australia, on December 1997, and in the Proceedings of the Twenty-third International Symposium on Mathematical Foundations of Computer Science, held in Brno, Czech Republic, on August 1998.

2. *Preliminary Notions*

In this section we introduce some notations and recall the basic notions used throughout the article. For more details about closure operators the reader is referred to Abramsky and Jung [1994], Davey and Priestley [1990], and Ward [1942], while for abstract interpretation to Cousot and Cousot [1977; 1979; 1992a].

2.1. Basic Mathematical Notation. If S and T are sets, then $\wp(S)$ denotes the powerset of S, $|S|$ the cardinality of S, $S \smallsetminus T$ the set-difference between S and T, $S \subset T$ strict inclusion, $S \times T$ the cartesian product, and for a function f: $S \rightarrow T$ and $X \subseteq S$, $f(X) \overset{\text{def}}{=} \{f(x) \mid x \in X\}$. By $g \circ f$ we denote the composition of the functions f and g, that is, $g \circ f \overset{\text{def}}{=} \lambda x.g(f(x))$. For any set S and $n \in \mathbb{N} \smallsetminus \{0\}$, S^n denotes the nth cartesian self product of S. A generic tuple in S^n is denoted by $\vec{x}$, $\vec{x}_i$ or x_i denote its ith component, and $\vec{x}[y/i]$ denotes the tuple obtained from $\vec{x}$ by replacing $\vec{x}_i$ with y. If f: $S \rightarrow T$, then $\langle f, \ldots, f\rangle$: $S^n \rightarrow T^n$ denotes the component-wise extension of f, that is, $\langle f, \ldots, f\rangle = \lambda\vec{x}.\langle f(\vec{x}_1), \ldots, f(\vec{x}_n)\rangle$. *Ord* denotes the proper class of ordinals, where $\omega \in \mathit{Ord}$ is the first infinite ordinal.

$\langle P, \leq\rangle$ denotes a poset P with ordering relation $\leq$, while $\langle C, \leq, \vee, \wedge, \top, \bot\rangle$ denotes a complete lattice C, with ordering $\leq$, *lub* $\vee$, *glb* $\wedge$, greatest element (top) $\top$, and least element (bottom) $\bot$. Often, $\leq_P$ will be used to denote the underlying ordering of a poset P, and $\vee_C$, $\wedge_C$, $\top_C$ and $\bot_C$ to denote the basic operations and elements of a complete lattice C. The notation $C \cong D$ denotes that C and D are isomorphic ordered structures. Let P be a poset and $S \subseteq P$.

[3] See, for example, Armstrong et al. [1998], Cortesi et al. [1996], Filé and Ranzato [1999], Giacobazzi and Ranzato [1998a], and Marriott and Søndergaard [1993].

Then, $\max(S) \overset{\text{def}}{=} \{x \in S \mid \forall y \in S.\ x \leq_P y \Rightarrow x = y\}$ denotes the set of maximal elements of S in P; also, the downward closure of S is defined by $\downarrow S \overset{\text{def}}{=} \{x \in P \mid \exists y \in S.\ x \leq_P y\}$, and for $x \in P$, $\downarrow x$ is a shorthand for $\downarrow \{x\}$, while the upward closure $\uparrow$ is dually defined. If C is a complete lattice, then a cartesian product C^n is still a complete lattice under the canonical componentwise ordering induced from C, where *lub* and *glb* are defined componentwise.

We use the symbol $\sqsubseteq$ to denote pointwise ordering between functions: If S is any set, P a poset, and f, g: $S \to P$ then $f \sqsubseteq g$ if for all $x \in S$, $f(x) \leq_P g(x)$. Let C and D be complete lattices. Then, $C \overset{\text{m}}{\to} D$ and $C \overset{\text{c}}{\to} D$ denote, respectively, the set of all monotone and (Scott-)continuous functions from C to D. Recall [Abramsky and Jung 1994] that $f \in C \overset{\text{c}}{\to} D$ iff f preserves *lub*'s of (nonempty) chains iff f preserves *lub*'s of directed subsets. Co-continuity is defined dually to continuity. It is useful to recall (see for example, Abramsky and Jung [1994, Lemma 3.2.6]) that $f \in C^n \overset{\text{c}}{\to} \text{D}$ ($f \in C^n \overset{\text{m}}{\to} D$) iff f is continuous (monotone) in each component separately. Also, f: $C \to D$ is (completely) additive if f preserves *lub*'s of all subsets of C (emptyset included), while co-additivity is dually defined. We denote by *lfp*(f) and *gfp*(f), respectively, the least and greatest fixpoint, when they exist, of an operator f on a poset. The well-known Knaster-Tarski's theorem states that any monotone operator f: $C \overset{\text{m}}{\to} C$ on a complete lattice C admits both least and greatest fixpoints, and the following characterizations hold:

$$lfp(f) = \wedge_C\{x \in C \mid f(x) \leq_C \text{x}\}; \qquad gfp(f) = \vee_C\{x \in C \mid x \leq_C f(x)\}.$$

Let us note that if f, g: $C \overset{\text{m}}{\to} C$ and $f \sqsubseteq g$ then $lfp(f) \sqsubseteq lfp(g)$. It is well known that if f: $C \to C$ is continuous then $lfp(f) = \vee_{i \in \mathbb{N}} f^i(\bot_C)$, where, for any $i \in \mathbb{N}$ and $x \in C$, the i-th power of f in x is inductively defined as follows: $f^0(x) = x$; $f^{i+1}(x) = f(f^i(x))$. Dually, if f: $C \to C$ is co-continuous then $gfp(f) = \wedge_{i \in \mathbb{N}} f^i(\top_C)$. $\{f^i(\bot_C)\}_{i \in \mathbb{N}}$ and $\{f^i(\top_C)\}_{i \in \mathbb{N}}$ are called, respectively, the upper and lower Kleene's iteration sequences of f.

2.2. Closure Operators and Galois Connections. Abstract domains can be equivalently formulated either in terms of Galois connections or in terms of closure operators [Cousot and Cousot 1979]. Let us recall these notions. An (*upper*) *closure operator*, or simply a *closure*, on a poset P is an operator ρ: $P \to P$ monotone, idempotent and extensive (i.e., $\forall x \in P.\ x \leq_P \rho(x)$). Dually, *lower* closure operators are monotone, idempotent, and restrictive. The set of all closure operators on P is denoted by $uco(P)$. Let $\langle C, \leq, \vee, \wedge, \top, \bot \rangle$ be a complete lattice. A basic property of closure operators is that each closure $\rho \in uco(C)$ is uniquely determined by the set of its fixpoints, which coincides with its image $\rho(C)$, as follows: $X \subseteq C$ is the set of fixpoints of a closure operator on C iff X is a *Moore-family* of C, that is, $X = \mathcal{M}(X) \overset{\text{def}}{=} \{\wedge S \mid S \subseteq X\}$—where $\wedge\varnothing = \top \in \mathcal{M}(X)$. In this case, $\rho_X = \lambda y.\ \wedge \{x \in X \mid y \leq x\}$ is the corresponding closure operator on C. For any $X \subseteq C$, $\mathcal{M}(X)$ is called the *Moore-closure* of X in C, that is, $\mathcal{M}(X)$ is the least (with respect to set inclusion) subset of C which contains X and is a Moore-family of C. Sometimes, we will use $\mathcal{M}_C$ to emphasize the complete lattice C of reference. It turns out that $\langle \rho(C), \leq \rangle$ is a complete meet subsemilattice of C (i.e., $\wedge$ is its *glb*), but, in general, it is not a complete sublattice of C, since the *lub* in $\rho(C)$—defined by $\lambda Y \subseteq \rho(C).\rho(\vee Y)$—might be different from that in C. In fact, $\rho(C)$ 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, possibly using as notation capital Latin letters. Hence, a notation like $X \in uco(C)$ means that $X \subseteq C$ is a Moore-family of C, that is, X is the set of fixpoints of a closure on C. This does not give rise to ambiguity, since one can readily distinguish the use of closures as functions or sets according to the context.

If C is a complete lattice, then $uco(C)$ ordered pointwise is a complete lattice as well. It will be denoted by $\langle uco(C), \sqsubseteq, \sqcup, \sqcap, \lambda x.\top, \lambda x.x\rangle$, where for every ρ, $\eta \in uco(C)$, $\{\rho_i\}_{i \in I} \subseteq uco(C)$ and $x \in C$:

—$\rho \sqsubseteq \eta$ iff $\forall y \in C.\ \rho(y) \leq \eta(y)$ 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.

Thus, the *glb* in $uco(C)$ is defined pointwise, while the *lub* of a set of closures $\{\rho_i\}_{i\in I} \subseteq uco(C)$ is the closure whose set of fixpoints is given by the set-intersection $\cap_{i\in I}\rho_i(C)$. In the following, we will make use of the following properties: For ρ, $\eta \in uco(C)$ and $Y \subseteq C$,

(i) $\rho(\wedge\rho(Y)) = \wedge\rho(Y)$;
(ii) $\rho(\vee Y) = \rho(\vee\rho(Y))$;
(iii) $\eta \sqsubseteq \rho \Leftrightarrow \eta \circ \rho = \rho \Leftrightarrow \rho \circ \eta = \rho$.

If A and C are posets, and α: $C \overset{\text{m}}{\to}$ A and γ: $A \overset{\text{m}}{\to}$ C are monotone functions such that $\lambda x.x \sqsubseteq \gamma \circ \alpha$ and $\alpha \circ \gamma \sqsubseteq \lambda x.x$, then the quadruple (α, C, A, γ) is called a Galois connection (GC for short) between C and A. If in addition $\alpha \circ \gamma = \lambda x.x$, 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. Let us also recall that the notion of GC is equivalent to that of adjunction: If α: $C \to A$ and γ: $A \to C$, 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 $\alpha(\gamma)$ is called the *left-(right-)adjoint* to $\gamma(\alpha)$. In any GC (α, C, A, γ) between complete lattices, α is additive and γ is co-additive. Also, if (α, C, A, γ) is a GI and C is a complete lattice then A is a complete lattice, too.

2.3. The Lattice of Abstract Interpretations. We assume the standard abstract interpretation framework, where concrete and abstract domains, C and A, are complete lattices related by abstraction and concretization maps forming a GC (α, C, A, γ). Following a standard terminology, A is called an abstraction of C, and C a concretization of A. If (α, C, A, γ) is a GI, then each value of the abstract domain A is useful in representing C, because all the elements of A represent distinct members of C, being γ 1-1. Any GC can be lifted to a GI identifying in an equivalence class those values of the abstract domain with the same concretization. This process is known as reduction of the abstract domain. It is well known since [Cousot and Cousot 1979] 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 ι: $\rho(C) \to A$ and ι^{-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 $\rho_A(C) \cong A$;

furthermore, these two constructions are the inverse of each other. Given an abstract domain A specified by a GI (α, C, A, γ), its associated closure $\gamma \circ \alpha$ on C can be thought of as the "logical meaning" in C of A, since this is shared by any other representation for the objects of A. 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* $\mathfrak{L}_C$ *of abstract interpretations* of C (cf. Cousot and Cousot [1977, Section 7], Cousot and Cousot [1979, Section 8], and Mycroft [1981, Section 2.4.2]), that is, the complete lattice of all possible abstract domains (modulo isomorphic representation of their objects) of the concrete domain C. The pointwise ordering on $uco(C)$ corresponds precisely to the standard ordering used to compare abstract domains with respect to their precision: A_1 is more precise than A_2 (i.e., A_2 is an abstraction of A_1) iff $A_1 \sqsubseteq A_2$ in $uco(C)$. *Lub*'s and *glb*'s on $uco(C)$ have therefore the following reading as operators on domains. Let $\{A_i\}_{i \in I} \subseteq uco(C)$: (i) $\sqcup_{i \in I} A_i$ is the most concrete among the domains in $\mathfrak{L}_C$ which are abstractions of all the A_i's, that is, $\sqcup_{i \in I} A_i$ is the *least* (with respect to $\sqsubseteq$) *common abstraction* of all the 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 in $\mathfrak{L}_C$ which are more concrete than every A_i. Let us remark that the reduced product can be also characterized as Moore-closure of set-union, that is, $\sqcap_{i \in I} A_i = \mathcal{M}(\cup_{i \in I} A_i)$. If $B = \sqcap_{i \in I} A_i$, then $\langle A_i \rangle_{i \in I}$ is called a (conjunctive) decomposition of the abstract domain B—Cortesi et al. [1997] present systematic methodologies for decomposing abstract domains.

3. *Completeness in Abstract Interpretation*

Let $f: C^n \overset{\mathrm{m}}{\to} D$ $(n \geq 1)$ be a concrete semantic operation defined over the concrete domains C and D. Let an abstract interpretation be specified by the GIs $(\alpha_{C,A}, C, A, \gamma_{A,C})$ and $(\alpha_{D,B}, D, B, \gamma_{B,D})$ and by a corresponding abstract semantic operation $f^\sharp: A^n \overset{\mathrm{m}}{\to} B$. Then, $f^\sharp$ is *sound* for (or is a *correct approximation* of) f if $\alpha_{D,B} \circ f \sqsubseteq f^\sharp \circ \langle \alpha_{C,A}, \ldots, \alpha_{C,A} \rangle$. Whenever $f: C \overset{\mathrm{m}}{\to} C$ and $f^\sharp: A \overset{\mathrm{m}}{\to} A$, $f^\sharp$ is *fixpoint sound* for f if $\alpha_{C,A}(lfp(f)) \leq_A lfp(f^\sharp)$. As we recalled in the introduction, a well-known basic result of abstract interpretation [Cousot and Cousot 1979, Theorem 7.1.0.4] (this has been later rediscovered as μ-fusion rule or transfer lemma, cf. Mathematics of Program Construction Group [1995, Sect. 3] and von Karger [1998, Theorem 3.1.1]) states that soundness implies fixpoint soundness. It is worth remarking that fixpoint soundness is in general a strictly weaker property than soundness—for instance, $\lambda x.\alpha_{C,A}(lfp(f)): A \to A$ is always fixpoint sound for $f: C \overset{\mathrm{m}}{\to} C$, but, in general, is not sound.

In abstract interpretation, completeness is meant as the natural strengthening of the notion of soundness, requiring its reverse relation to hold. Then, $f^\sharp$ is complete for f whenever $\alpha_{D,B} \circ f = f^\sharp \circ \langle \alpha_{C,A}, \ldots, \alpha_{C,A} \rangle$. Moreover, whenever $f: C \overset{\mathrm{m}}{\to} C$ and $f^\sharp: A \overset{\mathrm{m}}{\to} A$, $f^\sharp$ is *fixpoint complete* for f if $\alpha_{C,A}(lfp(f)) = lfp(f^\sharp)$. By a standard result (cf. Cousot and Cousot [1979, Theorems 7.1.0.4]; analogous technical results appear also in Apt and Plotkin [1986, Fact 2.3], de Bakker et al. [1983, Lemma 4.3], Mathematics of Program Construction Group [1995, Section 3], and von Karger [1998, Theorem 3.1.1]), completeness implies fixpoint completeness, while Example 3.4 below shows that the converse does not hold.

3.1. Completeness Is a Domain Property. By exploiting the properties of adjunctions, one can easily derive that $f^\sharp$ is a correct approximation of f if and only if $\alpha_{D,B} \circ f \circ \langle \gamma_{A,C}, \ldots, \gamma_{A,C} \rangle \sqsubseteq f^\sharp$. Thus, following a standard terminology [Cousot and Cousot 1979, Section 7.2],

$$f^{b_{A,B}} \stackrel{\text{def}}{=} \alpha_{D,B} \circ f \circ \langle \gamma_{A,C}, \ldots, \gamma_{A,C} \rangle : A^n \xrightarrow{\text{m}} B$$

is called the *best correct approximation* (where best refers to the pointwise order) of f relatively to the abstract domains A and B. Hence, the best correct approximation $f^{b_{A,B}}$, whose definition depends on the chosen input and output abstract domains A and B, is always automatically sound, and therefore fixpoint sound. By contrast, of course, this is not true for completeness, that is, given A and B, it may well happen that it is not possible to define a complete (or even merely fixpoint complete) abstract operation over A and B—indeed, this is the most common situation.

Furthermore, given A and B, let $f^\sharp : A^n \to B$ be complete for f. Then, it turns out that the best correct approximation $f^{b_{A,B}}$ of f is complete as well, and indeed it coincides with $f^\sharp$, as shown by the following equalities:

$$\begin{array}{rcl} f^\sharp & = & (\text{since } \alpha_{C,A} \circ \gamma_{A,C} = \lambda x.x) \\ f^\sharp \circ \langle \alpha_{C,A} \circ \gamma_{A,C}, \ldots, \alpha_{C,A} \circ \gamma_{A,C} \rangle & = & (\text{by completeness of } f^\sharp) \\ \alpha_{D,B} \circ f \circ \langle \gamma_{A,C}, \ldots, \gamma_{A,C} \rangle & = & \\ f^{b_{A,B}}. & & \end{array}$$

Likewise, if $f^\sharp : A \xrightarrow{\text{m}} \text{A}$ is fixpoint complete then $f^{b_{A,A}}$ is fixpoint complete as well:

$$\begin{array}{rl} \alpha_{C,A}(lfp(f)) \leq_A & (\text{since } f^{b_{A,A}} \text{ is always fixpoint sound}) \\ lfp(f^{b_{A,A}}) \leq_A & (\text{since } f^{b_{A,A}} \sqsubseteq f^\sharp) \\ lfp(f^\sharp) = & (\text{by fixpoint completeness of } f^\sharp) \\ \alpha_{C,A}(lfp(f)). & \end{array}$$

Thus, the possibility of defining complete or fixpoint complete abstract operations depends on the underlying abstract domains only, that is, the following characterizations hold:

(1) It is possible to define a complete abstract operation on the abstract domains A and B if and only if the best correct approximation induced by A and B is complete;
(2) It is possible to define a fixpoint complete abstract operator on an abstract domain A if and only if the best correct approximation induced by A is fixpoint complete.

In other terms, completeness and fixpoint completeness are *abstract domain properties*. Therefore, in the following, given the abstract domains, we refer to their completeness or fixpoint completeness in order to refer to the corresponding properties of the associated best correct approximations.

3.2. Completeness by Closures. Let us first give the following useful technical lemma characterizing both completeness and fixpoint completeness for best correct approximations.

Lemma 3.1. *Let* $(\alpha_{C,A}, C, A, \gamma_{A,C})$ *and* $(\alpha_{D,B}, D, B, \gamma_{B,D})$ *be GIs,* $f\colon C^n \xrightarrow{m} D$ *and* $g\colon C \xrightarrow{m} C$.

(*i*) $f^{b_{A,B}}$ *is complete for f iff* $(\gamma_{B,D} \circ \alpha_{D,B}) \circ f = (\gamma_{B,D} \circ \alpha_{D,B}) \circ f \circ \langle \gamma_{A,C} \circ \alpha_{C,A}, \ldots, \gamma_{A,C} \circ \alpha_{C,A} \rangle$.

(*ii*) $g^{b_{A,A}}$ *is fixpoint complete for g iff* $\gamma_{A,C}(\alpha_{C,A}(lfp(g))) = lfp(\gamma_{A,C} \circ \alpha_{C,A} \circ g)$.

Proofs.

(i) $f^{b_{A,B}}$ is complete iff $\alpha_{D,B} \circ f = (\alpha_{D,B} \circ f \circ \langle \gamma_{A,C}, \ldots, \gamma_{A,C} \rangle) \circ \langle \alpha_{C,A}, \ldots, \alpha_{C,A} \rangle$, and hence, since $\gamma_{B,D}$ is 1-1, iff $(\gamma_{B,D} \circ \alpha_{D,B}) \circ f = (\gamma_{B,D} \circ \alpha_{D,B}) \circ f \circ \langle \gamma_{A,C} \circ \alpha_{C,A}, \ldots, \gamma_{A,C} \circ \alpha_{C,A} \rangle$.

(ii) ($\Rightarrow$) Since $\gamma_{A,C} \circ \alpha_{C,A} \in uco(C)$, $g \sqsubseteq \gamma_{A,C} \circ \alpha_{C,A} \circ g$. Hence, $lfp(g) \leq_C lfp(\gamma_{A,C} \circ \alpha_{C,A} \circ g)$, from which $\gamma_{A,C}(\alpha_{C,A}(lfp(g))) \leq_C \gamma_{A,C}(\alpha_{C,A}(lfp(\gamma_{A,C} \circ \alpha_{C,A} \circ g)))$. Moreover, because in any Galois insertion we have $\gamma_{A,C} \circ \alpha_{C,A} \circ \gamma_{A,C} = \gamma_{A,C}$, then we have the following equalities

$$\begin{aligned} \gamma_{A,C}(\alpha_{C,A}(lfp(\gamma_{A,C} \circ \alpha_{C,A} \circ g))) &= \\ \gamma_{A,C}(\alpha_{C,A}((\gamma_{A,C} \circ \alpha_{C,A} \circ g)(lfp(\gamma_{A,C} \circ \alpha_{C,A} \circ g)))) &= \\ (\gamma_{A,C} \circ \alpha_{C,A} \circ g)(lfp(\gamma_{A,C} \circ \alpha_{C,A} \circ g)) &= \\ lfp(\gamma_{A,C} \circ \alpha_{C,A} \circ g), & \end{aligned}$$

and therefore $\gamma_{A,C}(\alpha_{C,A}(lfp(g))) \leq_C lfp(\gamma_{A,C} \circ \alpha_{C,A} \circ g)$. Let us prove the reverse inequality. By hypothesis, $\alpha_{C,A}(lfp(g)) = lfp(\alpha_{C,A} \circ g \circ \gamma_{A,C})$. Thus, $(\alpha_{C,A} \circ g \circ \gamma_{A,C})(\alpha_{C,A}(lfp(g))) = \alpha_{C,A}(lfp(g))$, and $\gamma_{A,C}(\alpha_{C,A}(g(\gamma_{A,C}(\alpha_{C,A}(lfp(g)))))) = \gamma_{A,C}(\alpha_{C,A}(lfp(g)))$. Hence, $\gamma_{A,C}(\alpha_{C,A}(lfp(g)))$ is a fixpoint of $\gamma_{A,C} \circ \alpha_{C,A} \circ g$, and therefore $lfp(\gamma_{A,C} \circ \alpha_{C,A} \circ g) \leq_C \gamma_{A,C}(\alpha_{C,A}(lfp(g)))$.

($\Leftarrow$) The inequality $\alpha_{C,A}(lfp(g)) \leq_A lfp(\alpha_{C,A} \circ g \circ \gamma_{A,C})$ always holds. Further, by hypothesis, we have that $\gamma_{A,C}(\alpha_{C,A}(g(\gamma_{A,C}(\alpha_{C,A}(lfp(g)))))) = \gamma_{A,C}(\alpha_{C,A}(lfp(g)))$, and therefore, since $\gamma_{A,C}$ is 1-1, $\alpha_{C,A}(g(\gamma_{A,C}(\alpha_{C,A}(lfp(g))))) = \alpha_{C,A}(lfp(g))$. Thus, $\alpha_{C,A}(lfp(g))$ is a fixpoint of $\alpha_{C,A} \circ g \circ \gamma_{A,C}$, and therefore $lfp(\alpha_{C,A} \circ g \circ \gamma_{A,C}) \leq_C \alpha_{C,A}(lfp(g))$. $\square$

Recall from Section 2 that $\gamma_{A,C} \circ \alpha_{C,A} \in uco(C)$ and $\gamma_{B,D} \circ \alpha_{D,B} \in uco(D)$ are, respectively, the closures on C and D associated with the abstract domains A and B. Let us observe that when abstract domains are specified by closures, the best correct approximation of a semantic operation $f\colon C^n \to D$ for $\rho \in uco(C)$ and $\eta \in uco(D)$, is given by $f^{b_{\rho,\eta}} = \eta \circ f\colon \rho(C)^n \to \eta(D)$. Thus, the above lemma simply states the equivalence of the notions of completeness and fixpoint completeness under the GI and closure operator approaches to abstract domain specification. Lemma 3.1 and the characterizations given in Section 3.1 justify the following nonrestrictive closure-based definitions of completeness and fixpoint completeness.[4]

[4] It would not be hard (but notationally tedious) to consider even more general semantic operations with n components in input and m in output.

Definition 3.2. Let C and D be complete lattices, $\rho \in uco(C)$, and $\eta \in uco(D)$.

(i) If $f\colon C^n \overset{\mathrm{m}}{\to} D$, then the pair $\langle \rho, \eta \rangle$ is *complete* for f if $\eta \circ f = \eta \circ f \circ \langle \rho, \ldots, \rho \rangle$.
(ii) If $g \in C \overset{\mathrm{m}}{\to} C$, then ρ is *fixpoint complete* for g if $\rho(lfp(g)) = lfp(\rho \circ g)$.

Sometimes, we will denote more compactly the condition in the above point (i) by $\eta \circ f = \eta \circ f \circ \vec{\rho}$. It is worth remarking that the above definition of completeness encompasses the case where $f\colon C^n \to C$ and one is interested in two different abstractions of input and output, that is, $\rho, \eta \in uco(C)$ with $\rho \neq \eta$. When $C = D$ and $\langle \rho, \rho \rangle$ is complete for f, by a slight abuse of terminology, often we say that ρ is complete for f. Alternatively, completeness and fixpoint completeness could be algebraically defined as follows:

LEMMA 3.3. *Let $f\colon C^n \overset{\mathrm{m}}{\to} D$, $g\colon C \overset{\mathrm{m}}{\to} C$, $\rho \in uco(C)$ and $\eta \in uco(D)$.*

(*i*) *$\langle \rho, \eta \rangle$ is complete for f if and only if $f \circ \langle \rho, \ldots, \rho \rangle \sqsubseteq \eta \circ f$.*
(*ii*) *ρ is fixpoint complete for g if and only if $\rho(lfp(g)) = lfp(\rho \circ g \circ \rho)$.*

PROOF

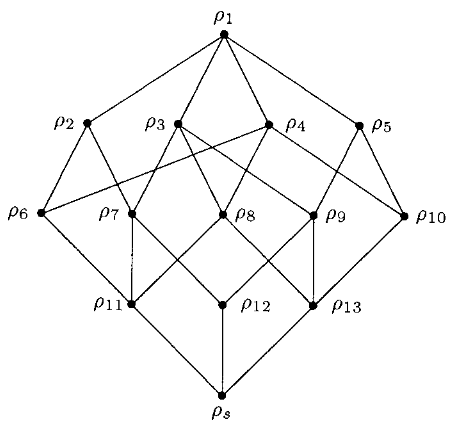
(i) ($\Rightarrow$) By extensivity of η.
($\Leftarrow$) By applying η to both sides, $\eta \circ f \circ \langle \rho, \ldots, \rho \rangle \sqsubseteq \eta \circ \eta \circ f = \eta \circ f$. The reverse inequality follows by extensivity of ρ.
(ii) Let us show that $lfp(\rho \circ g) = lfp(\rho \circ g \circ \rho)$. Since, by extensivity of ρ, $\rho \circ g \sqsubseteq \rho \circ g \circ \rho$, $lfp(\rho \circ g) \leq lfp(\rho \circ g \circ \rho)$. On the other hand, by exploiting the idempotency of ρ, $(\rho \circ g \circ \rho)(lfp(\rho \circ g)) = (\rho \circ g \circ \rho)((\rho \circ g)(lfp(\rho \circ g))) = (\rho \circ g)((\rho \circ g)(lfp(\rho \circ g))) = lfp(\rho \circ g)$, that is, $lfp(\rho \circ g)$ is a fixpoint of $\rho \circ g \circ \rho$, and therefore $lfp(\rho \circ g \circ \rho) \leq lfp(\rho \circ g)$. $\square$

The above notions of completeness are extended in the most natural way to generic sets of functions. If $F \subseteq C^n \overset{\mathrm{m}}{\to} D$, then $\langle \rho, \eta \rangle$ is complete for F if $\langle \rho, \eta \rangle$ is complete for all $f \in F$. Also, if $G \subseteq C \overset{\mathrm{m}}{\to} C$, then ρ is fixpoint complete for G if ρ is fixpoint complete for all $g \in G$. Throughout the article, we will adopt the following useful notations:

—$\Gamma(C, D, F) \overset{\text{def}}{=} \{\langle \rho, \eta \rangle \in uco(C) \times uco(D) \mid \langle \rho, \eta \rangle \text{ is complete for } F\}$;
—$\Delta(C, G) \overset{\text{def}}{=} \{\rho \in uco(C) \mid \rho \text{ is fixpoint complete for } G\}$.

For singleton sets of functions, we simply write $\Gamma(C, D, f)$ and $\Delta(C, g)$. Let us now see the simple example of the rule of signs for integer numbers [Cousot and Cousot 1977; 1979], already sketched in the introduction.

Example 3.4. Let us consider the abstract domain *Sign* depicted in Figure 1, which is an obvious abstraction of the concrete domain $\langle \wp(\mathbb{Z}), \subseteq \rangle$—concretization and abstraction maps are the most obvious ones—and let ρ_s denote the closure on $\langle \wp(\mathbb{Z}), \subseteq \rangle$ corresponding to *Sign.* It is easy to check that ρ_s is complete for the multiplication $*\colon \wp(\mathbb{Z})^2 \to \wp(\mathbb{Z})$ given by $X * Y \overset{\text{def}}{=} \{n \cdot m \mid n \in X, m \in Y\}$. This formalizes the "rule of signs", that is, the sign of an integer multiplication can be retrieved by the rule of signs. Instead, as Mycroft [1993] observes, ρ_s is not complete for integer addition $\oplus\colon \wp(\mathbb{Z})^2 \to \wp(\mathbb{Z})$, where $X \oplus Y \overset{\text{def}}{=} \{n + m \mid n \in X, m \in Y\}$. For instance, $\rho_s(\rho_s(\{-3\}) \oplus \rho_s(\{4\})) = \rho_s(\{x \mid x \leq 0\} \oplus$



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

FIG. 2. The lattice *uco*(*Sign*).

$\{x \mid x \geq 0\}) = \rho_s(\mathbb{Z}) = \mathbb{Z}$, whereas $\rho_s(\{-3\} \oplus \{4\}) = \rho_s(\{1\}) = \{x \mid x \geq 0\}$.

Consider the following program fragment:

```
    x := 1;
1:
    while $-100 \leq x \leq 100$ do
2:
        x := 2 * x
3:
    endw
4:
```

According to classical data-flow analysis techniques, the collection of all the values assumed by the integer variable x within the while-loop, that is, at program point 2, can be obtained as least fixpoint of the operator π_2: $\wp(\mathbb{Z}) \to (\mathbb{Z})$ defined by $\pi_2 \overset{\text{def}}{=} \lambda X.\{1\} \cup ((\{2\} * X) \cap [-100, 100])$. In this case, while *Sign* is not complete for π_2 (e.g., $\rho_s(\pi_2(\{2, -51\})) = \rho_s(\{1, 4\}) = 0+ \neq \mathbb{Z} = \rho_s(\pi_2(\mathbb{Z})) = \rho_s(\pi_2(\rho_s(\{2, -51\}))))$, it is instead fixpoint complete because we have $\rho_s(\mathit{lfp}(\pi_2)) = \rho_s(\{1, 2, 4, 8, 16, 32, 64\}) = 0+ = \mathit{lfp}(\rho_s \circ \pi_2)$.

Let us now consider the lattice $\langle \mathit{uco}(\mathit{Sign}), \sqsubseteq\rangle$ of all possible abstractions of *Sign*, as depicted in Figure 2, and the monotone unary square operation $sq \overset{\text{def}}{=} \lambda X.X * X$: $\wp(\mathbb{Z}) \to \wp(\mathbb{Z})$. It is a routine task to check that all the abstractions in *uco*(*Sign*), except ρ_5, are fixpoint complete for sq, that is, $\Delta(\wp(\mathbb{Z}), sq) \cap \uparrow \mathit{Sign} = \mathit{uco}(\mathit{Sign})\setminus\{\rho_5\}$: In fact, $\rho_5(\mathit{lfp}(sq)) = \rho_5(\varnothing) = \{x \mid x \leq 0\}$, while $\mathit{lfp}(\rho_5 \circ sq) = \mathbb{Z}$, and this holds for ρ_5 only. As a consequence, let us remark that $\Delta(\wp(\mathbb{Z}), sq)$ is not a complete sublattice of $\mathit{uco}(\wp(\mathbb{Z}))$: In fact, $\rho_9, \rho_{10} \in \Delta(\wp(\mathbb{Z}), sq)$, whereas $\rho_9 \sqcup \rho_{10} = \rho_5 \notin \Delta(\wp(\mathbb{Z}), sq)$.

Throughout the article, given f: $C^n \to D$, $\vec{x} \in C^n$ and $i \in [1, n]$, we will use the following compact notation:

$$f^{\,i}_{\vec{x}} \overset{\text{def}}{=} \lambda z.f(\vec{x}[z/i]): C \to D.$$

Thus, any $f^i_{\vec{x}}$ is simply one possible i-th unary restriction of f. Then, obviously, for $n = 1$, that is, when $f\colon C \to D$, $f^1_x = f$. The next result summarizes some helpful basic properties of completeness and fixpoint completeness.

PROPOSITION 3.5

(*i*) *Let* $F \subseteq C^n \overset{m}{\to} D$ and $G \subseteq C \overset{m}{\to} C$. For all $\rho \in \mathrm{uco}(C)$ *and* $\eta \in uco(D)$, $\langle \rho, \lambda x.\top_D\rangle$, $\langle \lambda x.x, \eta\rangle \in \Gamma(C, D, F)$ *and* $\lambda x.\top_C$, $\lambda x.x \in \Delta(C, G)$.

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

(*iii*) $\Gamma(C, C, \lambda\vec{x}.x_i) = \{\langle \rho, \eta\rangle \in uco(C) \times uco(C) \mid \rho \sqsubseteq \eta\}$ *and* $\Delta(C, \lambda x.x) = uco(C)$.

(*iv*) $\Gamma(C, C, \lambda\vec{x}.\ \vee_{i=1}^n x_i) = \{\langle \rho, \eta\rangle \in uco(C) \times uco(C) \mid \rho \sqsubseteq \eta\}$.

(*v*) *For any* $n \geq 2$, $\langle \rho, \rho\rangle \in \Gamma(C, C, \lambda\vec{x}.\ \wedge_{i=1}^n x_i)$ *iff* ρ *is finitely co-additive.*

(*vi*) *Let* $f\colon C^n \overset{m}{\to} D$ *and* $g\colon D \overset{m}{\to} E$. *If* $\langle \rho, \eta\rangle \in \Gamma(C, D, f)$ *and* $\langle \eta, \mu\rangle \in \Gamma(D, E, g)$ *then* $\langle \rho, \mu\rangle \in \Gamma(C, E, g \circ f)$.

(*vii*) *Let* $F \subseteq C^n \overset{m}{\to} D$. *If* $\langle \rho, \eta\rangle \in \Gamma(C, D, F)$, $\rho \sqsupseteq \delta \in uco(C)$ *and* $\eta \sqsubseteq \beta \in uco(D)$, *then* $\langle \delta, \beta\rangle \in \Gamma(C, D, F)$.

(*viii*) *If* $f\colon C^n \overset{m}{\to} D$, *then* $\Gamma(C, D, f) = \Gamma(C, D, \{f^i_{\vec{x}}\colon C \to D \mid i \in [1, n], \vec{x} \in C^n\})$.

(*ix*) *If* $F \subseteq C \overset{m}{\to} C$, *then* $\Gamma(C, C, F) \subseteq \Delta(C, F)$.

PROOF. (i) and (ii) Clear.

(iii) We have that $\langle \rho, \eta\rangle \in \Gamma(C, C, \lambda\vec{x}.x_i)$ iff $\eta \circ \lambda\vec{x}.x_i = \eta \circ \lambda\vec{x}.x_i \circ \vec{\rho}$ iff $\eta = \eta \circ \rho$ iff, by property (iii) in Section 2, $\rho \sqsubseteq \eta$. For the other equality, if $\rho \in uco(C)$ then $\rho(lfp(\lambda x.x)) = \rho(\bot_C) = lfp(\rho) = lfp(\rho \circ \lambda x.x)$, and therefore $\rho \in \Delta(C, \lambda x.x)$.

(iv) We have that $\langle \rho, \eta\rangle \in \Gamma(C, C, \lambda\vec{x}.\vee_{i=1}^n x_i)$ iff $\forall\vec{x} \in C^n.\eta(\vee_{i=1}^n x_i) = \eta(\vee_{i=1}^n \rho(x_i))$. Let us see that this latter equality holds iff $\rho \sqsubseteq \eta$.

($\Rightarrow$) For any $x \in C$, $\eta(x) = \eta(\rho(x))$, and hence, by property (iii) in Section 2, $\rho \sqsubseteq \eta$.

($\Leftarrow$) $\eta(\vee_{i=1}^n \rho(x_i)) \leq \eta(\vee_{i=1}^n \eta(x_i)) =$, by property (ii) of closures in Section 2, $= \eta(\vee_{i=1}^n x_i)$, while the reverse inequality trivially holds.

(v) $\langle \rho, \rho\rangle \in \Gamma(C, C, \lambda\vec{x}.\wedge_{i=1}^n x_i) \Leftrightarrow \forall\vec{x} \in C^n.\rho(\wedge_{i=1}^n x_i) = \rho(\wedge_{i=1}^n \rho(x_i)) =$, by property (i) of closures in Section 2, $= \wedge_{i=1}^n \rho(x_i)$, and this means that ρ is finitely co-additive.

(vi) Let $\langle \rho, \eta\rangle \in \Gamma(C, D, f)$, $\langle \eta, \mu\rangle \in \Gamma(D, E, g)$ and $\vec{x} \in C^n$. Then,

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

and therefore $\langle \rho, \mu\rangle \in \Gamma(C, E, g \circ f)$.

(vii) Let $\vec{x} \in C^n$. Then, for any $f \in F$,

$$\begin{aligned}
\beta(f(\vec{x})) &= \text{(since, by (iii) in Section 2, } \eta \sqsubseteq \beta \Rightarrow \beta \circ \eta = \beta) \\
\beta(\eta(f(\vec{x}))) &= \text{(since } \langle \rho, \eta \rangle \text{ is complete for } f) \\
\beta(\eta(f(\rho(\vec{x}_1), \ldots, \rho(\vec{x}_n)))) &= \text{(since, by (iii) in Section 2, } \eta \sqsubseteq \beta \Rightarrow \beta \circ \eta = \beta) \\
\beta(f(\rho(\vec{x}_1), \ldots, \rho(\vec{x}_n))) &\geq \text{(since } \rho \sqsupseteq \delta) \\
\beta(f(\delta(\vec{x}_1), \ldots, \delta(\vec{x}_n))).
\end{aligned}$$

Since $\beta(f(\vec{x})) \leq \beta(f(\delta(\vec{x})))$ trivially holds, we get that $\langle \delta, \beta \rangle \in \Gamma(C, D, F)$.

(viii) We show that $\langle \rho, \eta \rangle \in \Gamma(C, D, f)$ iff $\langle \rho, \eta \rangle \in \Gamma(C, D, \{f^i_{\vec{x}}: C \to D \mid i \in [1, n], \vec{x} \in C^n\})$.

($\Rightarrow$) We use the following shorthand: $\rho(\vec{x}) = \langle \rho(x_1), \ldots, \rho(x_n) \rangle$. Let $y \in C$. Then, $\eta(f^i_{\vec{x}}(y)) \leq \eta(f^i_{\vec{x}}(\rho(y)))$ always holds. On the other hand, by extensivity of ρ, $f^i_{\vec{x}} \sqsubseteq f^i_{\rho(\vec{x})}$, therefore $\eta(f^i_{\vec{x}}(\rho(y))) \leq \eta(f^i_{\rho(\vec{x})}(\rho(y)))$ and by hypothesis, $\eta(f^i_{\rho(\vec{x})}(\rho(y))) = \eta(f(\rho(x_1), \ldots, \rho(x_{i-1}), \rho(y), \rho(x_{i+1}), \ldots, \rho(x_n))) = \eta(f(\vec{x}[y/i])) = \eta(f^i_{\vec{x}}(y))$.

($\Leftarrow$) Let $\vec{x} \in C^n$. Then, by applying n times the hypothesis,

$$\begin{aligned}
\eta(f(\vec{x})) &= \eta(f(\rho(x_1), x_2, \ldots, x_n)) \\
&= \eta(f(\rho(x_1), \rho(x_2), \ldots, x_n)) \\
&= \cdots \\
&= \eta(f(\rho(\vec{x}))),
\end{aligned}$$

namely $\langle \rho, \eta \rangle \in \Gamma(C, D, f)$.

(ix) It means that completeness implies fixpoint completeness [Cousot and Cousot 1979, Theorem 7.1.0.4]. □

Let us comment on the most significant points of the above proposition. Point (iv) says that considering the finite *lub* as a concrete operation, a pair of abstract domains $\langle \rho, \eta \rangle$ is complete for it iff $\rho \sqsubseteq \eta$—hence, any pair $\langle \rho, \rho \rangle$ is always complete for the *lub*. Point (v) instead says that by considering the finite *glb* as a concrete operation, any closure ρ preserving finite *glb*'s gives rise to a pair $\langle \rho, \rho \rangle$ complete for the *glb*. These observations are particularly useful, since both *glb* and *lub* are commonly used as concrete operations in program analysis frameworks—for example, *lub*'s are commonly used to model multiple computation paths, while a concrete *glb* plays often the role of constraint unification (e.g., in logic programming). Let us also comment on point (vi): This states that completeness for a composite function $f \circ g$ can be compositionally checked on f and g. This could be helpful in order to check for completeness of semantics inductively specified on the syntax of programs, like in denotational semantics. Point (vii) is an observation that will be particularly important later on: It says that completeness of a pair $\langle \rho, \eta \rangle$ of domains is preserved when going towards the concretization direction on the input component ρ and, dually, when going towards the abstraction direction on the output component η. Finally, point (viii) reduces the completeness of n-ary functions to completeness of a suitable set of unary functions.

4. *Constructive Characterizations of Completeness*

In this section, we characterize in a "constructive" fashion both complete and fixpoint complete abstract domains. First, it is shown that a pair of abstract domains $\langle\rho, \eta\rangle$ is complete for a set of continuous semantic operations F iff ρ contains a certain set of points depending on η, and, dually, iff η is contained in a certain set of points which depends on ρ. Second, it is shown that ρ is fixpoint complete for any given set of semantic operators F iff ρ is contained in a certain set of points depending on ρ itself.

The proof concerning completeness makes use of the axiom of choice, as given by the following variant known as Hausdorff's maximal principle [Birkhoff 1967, page 192]. Let us point out that a chain Y in a poset P is maximal (with respect to set inclusion) whenever for any other chain Y' in P, $Y \subseteq Y'$ implies $Y = Y'$.

HAUSDORFF'S MAXIMAL PRINCIPLE. *Every chain in a poset P can be extended to a maximal chain in P.*

We will adopt the following useful notation: For any $f\colon C \to D$ and $y \in D$, $f^{-1}(\downarrow y) \overset{\text{def}}{=} \{x \in C \mid f(x) \leq_D y\}$, that is, $f^{-1}(\downarrow y)$ is the counterimage for f of $\downarrow y$. We will exploit Hausdorff's maximal principle in a form given by the following lemma:

LEMMA 4.1. *Let $f\colon C \overset{c}{\to} D$ and $y \in D$. If $x \in f^{-1}(\downarrow y)$, then there exists $z \in max(f^{-1}(\downarrow y))$ such that $x \leq_C z$.*

PROOF. If $x \in f^{-1}(\downarrow y)$, then, by Hausdorff's maximal principle, there exists a maximal chain $Z \subseteq f^{-1}(\downarrow y)$ which contains x. Then, $\vee_D f(Z) \leq_D y$, and, by continuity of f, $f(\vee_C Z) = \vee_D f(Z) \leq_D y$. Thus, $\vee_C Z \in f^{-1}(\downarrow y)$. Moreover, if $w \in f^{-1}(\downarrow y)$ is such that $\vee_C Z \leq_C w$, we have that $Z \cup \{w\}$ is a chain in $f^{-1}(\downarrow y)$ containing Z, and hence, by maximality of Z in $f^{-1}(\downarrow y)$, $w \in Z$. Thus, $w = \vee_C Z$, and therefore $\vee_C Z \in \max(f^{-1}(\downarrow y))$ (and $x \leq_C \vee_C Z$). $\square$

In order to give the general characterization of completeness, let us first prove the following particular case involving a single unary operation.

LEMMA 4.2. *Let $f\colon C \overset{c}{\to} D$, $\rho \in uco(C)$, and $\eta \in uco(D)$. $\langle\rho, \eta\rangle \in \Gamma(C, D, f)$ iff $\cup_{y\in\eta} max(f^{-1}(\downarrow y)) \subseteq \rho$. Moreover, $\{y \in D \mid max(f^{-1}(\downarrow y)) \subseteq p\} \in uco(D)$.*

PROOF

($\Rightarrow$) Let $y \in \eta$ and $m \in \max(\{x \in C \mid f(x) \leq_D y\})$. Let us show that $\rho(m) = m$. Since $f(m) \leq_D y$, by hypothesis, we have that $\eta(f(\rho(m))) = \eta(f(m)) \leq_D \eta(y) = y$, and therefore, by extensivity of η, $f(\rho(m)) \leq_D y$. Thus, $\rho(m) \in \{x \in C \mid f(x) \leq_D y\}$. By extensivity of ρ, $m \leq_C \rho(m)$, and therefore, by maximality of m, $m = \rho(m)$.

($\Leftarrow$) Let $z \in C$, and let us prove that $\eta(f(\rho(z))) \leq_D \eta(f(z))$, since the reverse inequality always holds. Let us consider the set $f^{-1}(\downarrow \eta(f(z))) = \{x \in C \mid f(x) \leq_D \eta(f(z))\}$. Then, by extensivity of η, $f(z) \leq_D \eta(f(z))$, that is, $z \in f^{-1}(\downarrow \eta(f(z)))$. By Lemma 4.1, there exists $w \in \max(f^{-1}(\downarrow \eta(f(z))))$ such that $z \leq_C w$. Thus, by hypothesis, $\rho(w) = w$. Each of the following inequalities is justified on the left, and implies its successive:

$$\begin{array}{lrcl}
\text{since } z \leq_C w = \rho(w): & \rho(z) & \leq_C & w \\
\text{by monotonicity of } f: & f(\rho(z)) & \leq_D & f(w) \\
\text{since } w \in f^{-1}(\downarrow \eta(f(z))): & f(\rho(z)) & \leq_D & \eta(f(z)) \\
\text{by monotonicity of } \eta: & \eta(f(\rho(z))) & \leq_D & \eta(\eta(f(z))) \\
\text{by idempotency of } \eta: & \eta(f(\rho(z))) & \leq_D & \eta(f(z)).
\end{array}$$

Thus, the last inequality closes this implication.

Next, we show that $\mathcal{M}(\{y \in D \mid \max(f^{-1}(\downarrow y)) \subseteq \rho\}) = \{y \in D \mid \max(f^{-1}(\downarrow y)) \subseteq \rho\}$. First, since $\max(f^{-1}(\downarrow \top_D)) = \max(\{x \in C \mid f(x) \leq_D \top_D\}) = \{\top_C\}$, we have that $\top_D \in \{y \in D \mid \max(f^{-1}(\downarrow y)) \subseteq \rho\}$. Now, let $Z \subseteq \{y \in D \mid \max(f^{-1}(\downarrow y)) \subseteq \rho\}$, with $Z \neq \varnothing$, and let $w = \wedge Z$. We show that $\max(f^{-1}(\downarrow w)) \subseteq \rho$. Let $x \in \max(f^{-1}(\downarrow w))$. Since $f^{-1}(\downarrow w) = \cap_{z\in Z} f^{-1}(\downarrow z)$, we have that $x \in \cap_{z \in Z} f^{-1}(\downarrow z)$. Hence, by Lemma 4.1, for any $z \in Z$, there exists some $m_z \in \max(f^{-1}(\downarrow z))$ such that $x \leq_C m_z$, and therefore $x \leq_C \wedge_{z\in Z} m_z$. On the other hand, for any $u \in Z$, $f(\wedge_{z\in Z} m_z) \leq_D f(m_u) \leq_D u$, from which we get $f(\wedge_{z\in Z} m_z) \leq_D \wedge Z = w$. Hence, $\wedge_{z\in Z} m_z \in f^{-1}(\downarrow w)$, and therefore, by maximality of x, from $x \leq_C \wedge_{z \in Z} m_z$, we get $\wedge_{z\in Z} m_z = x$. Thus, since any m_z belongs to ρ and ρ is Moore-closed, we conclude that $x \in \rho$. □

The next theorem extends the previous lemma to sets of continuous n-ary functions, by exploiting Proposition 3.5 (viii).

THEOREM 4.3. *Let* $F \subseteq C^n \xrightarrow{c} D$, $\rho \in uco(C)$, *and* $\eta \in uco(D)$. *Then, the following are equivalent:*

(*i*) $\langle \rho, \eta \rangle \in \Gamma(C, D, F)$;
(*ii*) $\bigcup_{f\in F, i\in[1,n], \vec{x}\in C^n, y\in\eta} max((f^i_{\vec{x}})^{-1}(\downarrow y)) \subseteq \rho$;
(*iii*) $\eta \subseteq \{y \in D \mid \bigcup_{f\in F, i\in[1,n], \vec{x}\in C^n} max((f^i_{\vec{x}})^{-1}(\downarrow y)) \subseteq \rho\}$.

Moreover, $\{y \in D \mid \bigcup_{f\in F, i\in[1,n], \vec{x}\in C^n} max((f^i_{\vec{x}})^{-1}(\downarrow y)) \subseteq \rho\} \in uco(D)$.

PROOF. (i) ⇔ (ii) By Proposition 3.5 (viii), for any $f \in F$, we have that $\Gamma(C, D, f) = \cap\{\Gamma(C, D, f^i_{\vec{x}}) \mid i \in [1, n], \vec{x} \in C^n\}$. Since f is continuous, every $f^i_{\vec{x}}$: $C \xrightarrow{c} D$ is continuous. Hence, by Lemma 4.2, for any $i \in [1, n]$ and $\vec{x} \in C^n$, $\langle \rho, \eta \rangle \in \Gamma(C, D, f^i_{\vec{x}})$ iff $\cup_{y\in\eta} \max((f^i_{\vec{x}})^{-1}(\downarrow y)) \subseteq \rho$. Thus,

$$\langle \rho, \eta \rangle \in \Gamma(C, D, f) \Leftrightarrow \bigcup_{i\in[1,n], \vec{x}\in C^n, y\in\eta} \max((f^i_{\vec{x}})^{-1}(\downarrow y)) \subseteq \rho.$$

Moreover, by definition, $\Gamma(C, D, F) = \cap_{f\in F}\Gamma(C, D, f)$, and therefore

$$\langle \rho, \eta \rangle \in \Gamma(C, D, F) \Leftrightarrow \bigcup_{f\in F, i\in[1,n], \vec{x}\in C^n, y\in\eta} \max((f^i_{\vec{x}})^{-1}(\downarrow y)) \subseteq \rho.$$

(ii) ⇔ (iii) This equivalence follows by a straightforward set-theoretic argument.

To conclude, let us show that $\{y \in D \mid \bigcup_{f\in F, i\in[1,n], \vec{x}\in C^n} \max((f^i_{\vec{x}})^{-1}(\downarrow y)) \subseteq \rho\} \in uco(D)$. To this aim, note that $\{y \in D \mid \bigcup_{f\in F, i\in[1,n], \vec{x}\in C^n} \max(f^i_{\vec{x}})^{-1}(\downarrow y)) \subseteq \rho\} = \bigcap_{f\in F, i\in[1,n], \vec{x}\in C^n} \{y \in D \mid \max((f^i_{\vec{x}})^{-1}(\downarrow y)) \subseteq \rho\}$, and, as above, the hypotheses allow us to apply Lemma 4.2. Thus, for any $f \in F$, $i \in [1, n]$ and $\vec{x} \in C^n$, $\{y \in D \mid \max((f^i_{\vec{x}})^{-1}(\downarrow y)) \subseteq \rho\} \in uco(D)$. Hence, since sets of fixpoints of closures are closed under set-intersection (cf. Section 2), this concludes the proof. □

It is important to remark that whenever the semantic operations are (completely) additive rather than continuous, max's in the above statement of Theorem 4.3 actually are *lub*'s (of C). This because, in Lemma 4.2, if $f: C \to D$ is additive, then, for any $y \in D$, $\vee_C f^{-1}(\downarrow y) \in f^{-1}(\downarrow y)$, and therefore $\max(f^{-1}(\downarrow y)) = \{\vee_C f^{-1}(\downarrow y)\}$. Hence, this observation will allow us to simplify the notation when dealing with additive semantic operations, as it will be the case in Section 6.1.

It is worthwhile to give an example showing that, whenever the semantic operations fail to be continuous, in general, Theorem 4.3 does not hold.

Example 4.4. Let us consider as concrete domain C the $\omega + 2$ ordinal, that is, $C = \{x \in Ord \mid x < \omega\} \cup \{\omega, \omega + 1\}$, and let $f: C \to C$ be the closure $f \in uco(C)$ whose set of fixpoints is given by $f(C) \overset{\text{def}}{=} \{x \mid x < \omega\} \cup \{\omega + 1\}$. Thus, f is the identity on $C \setminus \{\omega\}$ whereas it maps ω to the top $\omega + 1$. Next, consider the closure $\rho \in uco(C)$ given by $\rho \overset{\text{def}}{=} \{\omega, \omega + 1\}$. Note that $\rho(f(0)) = \omega$ and $\rho(f(\rho(0))) = \omega + 1$, and therefore $\langle \rho, \rho \rangle \notin \Gamma(C, C, f)$. On the other hand, we have that:

—$\max(f^{-1}(\downarrow(\omega + 1))) = \max(C) = \{\omega + 1\} \subseteq \rho$,

—$\max(f^{-1}(\downarrow \omega)) = \max(\{x \in C \mid f(x) \leq_C \omega\}) = \max(\{x \in C \mid x < \omega\}) = \varnothing \subseteq \rho$,

and therefore $\cup_{y \in \rho} \max(f^{-1}(\downarrow y)) \subseteq \rho$. Note that f lacks of the continuity property, and therefore this example is consistent with Theorem 4.3.

While Theorem 4.3 shows that $\{y \in D \mid \cup_{f \in F, i \in [1,n], \vec{x} \in C^n} \max((f^i_{\vec{x}})^{-1}(\downarrow y)) \subseteq \rho\}$ is the set of fixpoints of a closure on D, in general, this is not true for the set $\cup_{f \in F, i \in [1,n], \vec{x} \in C^n, y \in \eta} \max((f^i_{\vec{x}})^{-1}(\downarrow y))$, as shown by the following example.

Example 4.5. Consider *Sign* in Figure 1 as concrete domain, the function f: $Sign \to Sign$ defined by $f \overset{\text{def}}{=} \{\varnothing \mapsto \varnothing, 0 \mapsto 0, -0 \mapsto 0+, 0+ \mapsto 0+, \mathbb{Z} \mapsto \mathbb{Z}\}$, and the closure $\eta \overset{\text{def}}{=} \{\mathbb{Z}, 0+\} \in uco(Sign)$. We have that

—$\max(f^{-1}(\downarrow \mathbb{Z})) = \max(\{x \in Sign \mid f(x) \leq_{Sign} \mathbb{Z}\}) = \max(Sign) = \{\mathbb{Z}\}$,

—$\max(f^{-1}(\downarrow 0+)) = \max(\{x \in Sign \mid f(x) \leq_{Sign} 0+\}) = \max(Sign \setminus \{\mathbb{Z}\}) = \{-0, 0+\}$.

Hence, $\cup_{y \in \eta} \max(f^{-1}(\downarrow y)) = \{-0, 0+, \mathbb{Z}\}$, which is not a Moore-family of (i.e., a closure on) *Sign*, although f is monotone, and thus continuous.

As Theorem 4.3 suggests, we then introduce the following mappings between lattices of abstract interpretations.

Definition 4.6. Given $F \subseteq C^n \overset{\text{m}}{\to} D$, define

—L_F: $uco(C) \to uco(D)$ where

$$L_F(\rho) \overset{\text{def}}{=} \{y \in D \mid \cup_{f \in F, i \in [1,n], \vec{x} \in C^n} max((f^i_{\vec{x}})^{-1}(\downarrow y)) \subseteq \rho\};$$

—R_F: $uco(D) \to uco(C)$ where

$$R_F(\eta) \overset{\text{def}}{=} \mathcal{M}(\textstyle\bigcup_{f\in F, i\in[1,n],\, \vec{x}\in C^n{}_{d, y\in\eta}} \max((f^i_{\vec{x}})^{-1}(\downarrow y))).$$

In this way, we get the following consequence of Theorem 4.3.

COROLLARY 4.7. *Let $F \subseteq C^n \overset{c}{\to} D$. Then, $\langle \rho, \eta\rangle \in \Gamma(C, D, F) \Leftrightarrow L_F(\rho) \sqsubseteq \eta \Leftrightarrow \rho \sqsubseteq R_F(\eta)$, and therefore $(L_F, uco(C), uco(D), R_F)$ is an adjunction.*

PROOF. Firstly, notice that for any closure $\rho \in uco(C)$ and any arbitrary set of points $S \subseteq C$, the following equivalence holds: $S \subseteq \rho \Leftrightarrow \rho \sqsubseteq \mathcal{M}(S)$. Thus, Theorem 4.3 can be restated as follows: for any $\rho \in uco(C)$ and $\eta \in uco(D)$,

$$\langle \rho, \eta\rangle \in \Gamma(C, D, F) \Leftrightarrow L_F(\rho) \sqsubseteq \eta \Leftrightarrow \rho \sqsubseteq R_F(\eta),$$

and therefore $(L_F, uco(C), uco(D), R_F)$ turns out to be an adjunction. □

The following further consequence of the constructive characterization theorem shows that *lub*'s of pairs of complete abstract domains are complete, too.[5]

COROLLARY 4.8. *Assume $F \subseteq C^n \overset{m}{\to} D$ and $\{\langle \rho_i, \eta_i\rangle\}_{i\in I} \subseteq \Gamma(C, D, F)$. Then $\langle \sqcup_{i\in I}\rho_i, \sqcup_{i\in I}\eta_i\rangle \in \Gamma(C, D, F)$.*

PROOF. By Proposition 3.5 (vii), we get $\{\langle \rho_i, \sqcup_{j\in I}\eta_j\rangle\}_{i\in I} \subseteq \Gamma(C, D, F)$. By Corollary 4.7, $\langle \sqcup_{i\in I}\rho_i, \sqcup_{i\in I}\eta_i\rangle \in \Gamma(C, D, F)$ iff $L_F(\sqcup_{i\in I}\rho_i) \sqsubseteq \sqcup_{i\in I}\eta_i$. Since L_F is the left-adjoint of an adjunction between complete lattices, L_F is (completely) additive, and therefore $\langle \sqcup_{i\in I}\rho_i, \sqcup_{i\in I}\eta_i\rangle \in \Gamma(C, D, F)$ iff $\sqcup_{i\in I}L_F(\rho_i) \sqsubseteq \sqcup_{i\in I}\eta_i$. Since, by Corollary 4.7, for any $j \in I$, $L_F(\rho_j) \sqsubseteq \sqcup_{i\in I}\eta_i$, we get that $\langle \sqcup_{i\in I}\rho_i, \sqcup_{i\in I}\eta_i\rangle \in \Gamma(C, D, F)$. □

When the semantic operations are not continuous, in general, the above result is not true, as shown by the following continuation of Example 4.4.[6]

Example 4.9. Let C and f as in Example 4.4, and consider $\rho_1, \rho_2 \in uco(C)$ defined by $\rho_1 \overset{\text{def}}{=} \{x < \omega \mid x \text{ is even}\} \cup \{\omega, \omega + 1\}$ and $\rho_2 \overset{\text{def}}{=} \{x < \omega \mid x \text{ is odd}\} \cup \{\omega, \omega + 1\}$. It is routine to verify that $\langle \rho_i, \rho_i\rangle_{i=1,2} \in \Gamma(C, C, f)$. Consider the closure $\rho \in uco(C)$ of Example 4.4, which is such that $\langle \rho, \rho\rangle \notin \Gamma(C, C, f)$, and observe that $\rho_1 \sqcup \rho_2 = \rho$. As noted in Example 4.4, f lacks of the continuity property, and therefore this example is coherent with Corollary 4.8. □

Let us remark that an analogous dual proof to that of Corollary 4.8 would also allow us to prove that $\langle \sqcap_{i\in I}\rho_i, \sqcap_{i\in I}\eta_i\rangle \in \Gamma(C, D, F)$. However, this can be proved for merely monotone semantic operations without resorting to Theorem 4.3.

PROPOSITION 4.10. *Let $F \subseteq C^n \overset{m}{\to} D$ and $\{\langle \rho_i, \eta_i\rangle\}_{i\in I} \subseteq \Gamma(C, D, F)$. Then, $\langle \sqcap_{i\in I}\rho_i, \sqcap_{i\in I}\eta_i\rangle \in \Gamma(C, D, F)$.*

PROOF. By Proposition 3.5 (vii), we get $\{\langle \sqcap_{j\in I}\rho_j, \eta_i\rangle\}_{i\in I} \subseteq \Gamma(C, D, F)$. Let $\varphi = \sqcap_{i\in I}\rho_i$, and let us show that $\langle \varphi, \sqcap_{i\in I}\eta_i\rangle \in \Gamma(C, D, F)$. Let $\vec{x} \in C^n$. It is sufficient to show that $\wedge_{i\in I}\eta_i(f(\varphi(\vec{x}_1), \ldots, \varphi(\vec{x}_n))) \leq_D \wedge_{i\in I}\eta_i(f(\vec{x}))$, because the reverse inequality trivially holds. Thus, for any $j \in I$, we have to prove that

[5] Amato and Levi [1997, Theorem 2.3] have independently stated an analogous, but weaker, result to Corollary 4.8 for additive functions of type $C^n \to C$.

[6] Amato and Levi [1997, Example 2.2] independently made an observation similar to Example 4.9.

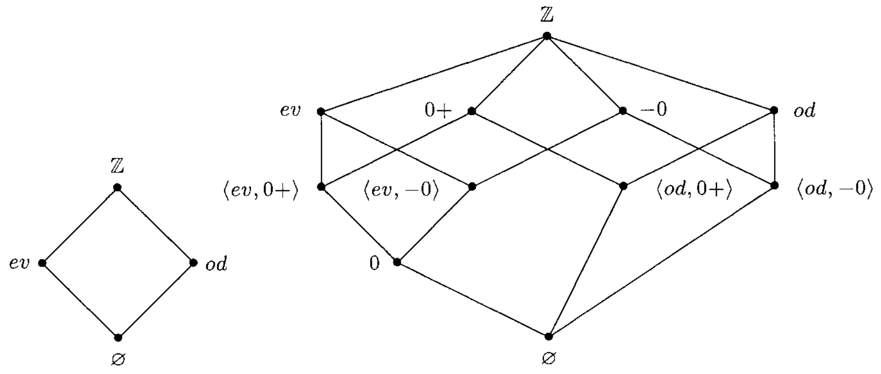


FIG. 3. The lattices *Parity*, on the left, and *Sign* $\sqcap$ *Parity*, on the right.

$\wedge_{i\in I}\eta_i(f(\varphi(\vec{x}_1), \ldots, \varphi(\vec{x}_n))) \leq_D \eta_j(f(\vec{x}))$:

$$\begin{array}{rcl} \wedge_{i\in I}\eta_i(f(\varphi(\vec{x}_1), \ldots, \varphi(\vec{x}_n))) & \leq_D & \\ \eta_j(f(\varphi(\vec{x}_1), \ldots, \varphi(\vec{x}_n))) & = & (\text{by completeness of } \langle \varphi, \eta_j\rangle) \\ \eta_j(f(\vec{x})) & & \end{array}$$

and this closes the proof. □

Example 4.11. Consider the simple abstract domain *Parity* for parity analysis of integer variables shown in Figure 3 [Cousot and Cousot 1979, Example 10.1.0.3], which is an obvious abstraction of $\langle \wp(\mathbb{Z}), \subseteq\rangle$. It is straightforward to check that $\langle$*Parity, Parity*$\rangle$ is complete for the integer multiplication $\lambda\langle X, Y\rangle.X * Y$, as defined in Example 3.4. Also, in Example 3.4 we observed that $\langle$*Sign, Sign*$\rangle$ $\in \Gamma(\wp(\mathbb{Z}), \wp(\mathbb{Z}), *)$. Let us consider their reduced product *Sign* $\sqcap$ *Parity* depicted in Figure 3 (see Cousot and Cousot [1979, Example 10.1.0.3] for an example of an enhanced sign-parity analysis using *Sign* $\sqcap$ *Parity*). Then, by Proposition 4.10, we get that $\langle$*Sign* $\sqcap$ *Parity*, *Sign* $\sqcap$ *Parity*$\rangle$ is complete for integer multiplication as well. On the other hand, the least common abstraction between *Sign* and *Parity* is simply given by *Sign* $\sqcup$ *Parity* $= \{\mathbb{Z}, \varnothing\}$. By Corollary 4.8, *Sign* $\sqcup$ *Parity* is complete for $*$, too. □

Let us now turn to fixpoint completeness. For any set of merely monotone semantic operators, the following result characterizes fixpoint complete abstract domains in a simple way.

THEOREM 4.12. *Let* $F \subseteq C \overset{\text{m}}{\to} C$ *and* $\rho \in uco(C)$. *Then, the following are equivalent:*

(i) $\rho \in \Delta(C, F)$;
(ii) *For all* $f \in F$, $\rho(lfp(f)) = \rho(f(\rho(lfp(f))))$;
(iii) $\rho \subseteq \{y \in C \mid \forall f \in F.\ \rho(lfp(f)) \leq y \Rightarrow \rho(f(\rho(lfp(f)))) \leq y\}$.

Moreover, $\{y \in C \mid \forall f \in F.\ \rho(lfp(f)) \leq y \Rightarrow \rho(f(\rho(lfp(f)))) \leq y\} \in uco(C)$.

PROOF
(i) $\Leftrightarrow$ (ii) By definition, $\rho \in \Delta(C, F)$ iff for all $f \in F$, $\rho(lfp(f)) = lfp(\rho \circ f)$. Then, $\rho(lfp(f))$ is a fixpoint of $\rho \circ f$, and this gives the "$\Rightarrow$" direction. On the

other hand, $\rho(lfp(f)) = \rho(f(\rho(lfp(f))))$ means that $\rho(lfp(f))$ is a fixpoint of $\rho \circ f$, and therefore $lfp(\rho \circ f) \leq \rho(lfp(f))$ holds, from which $\rho(lfp(f)) = lfp(\rho \circ f)$ follows.

(ii) $\Leftrightarrow$ (iii) The "$\Rightarrow$" direction trivially holds. For the "$\Leftarrow$" direction, let $f \in F$ and consider $y = \rho(lfp(f)) \in \rho$. Hence, by hypothesis, $\rho(f(\rho(lfp(f)))) \leq \rho(lfp(f))$, while the reverse inequality always holds.

Let $\eta = \{y \in C \mid \forall f \in F. \rho(lfp(f)) \leq y \Rightarrow \rho(f(\rho(lfp(f)))) \leq y\}$ and let us prove that $\mathcal{M}(\eta) = \eta$. Let $Y \subseteq \eta$. If, for all $f \in F$, $\rho(lfp(f)) \not\leq \wedge Y$, then trivially $\wedge Y \in \eta$. Thus, assume that, for some $f \in F$, $\rho(lfp(f)) \leq \wedge Y$. Therefore, for all $y \in Y$, $\rho(lfp(f)) \leq y$, and thus, by definition of η, $\rho(f(\rho(lfp(f)))) \leq y$. As a consequence, $\rho(f(\rho(lfp(f)))) \leq \wedge Y$, that is, $\wedge Y \in \eta$, which concludes the proof. □

Let us remark that, by the equivalence (i) $\Leftrightarrow$ (ii), in order to check if some ρ is fixpoint complete for a monotone function f it is enough to check if ρ is complete on the single point $lfp(f)$: In fact, since $\rho(f(lfp(f))) = \rho(lfp(f))$, ρ is fixpoint complete iff $\rho(f(lfp(f))) = \rho(f(\rho(lfp(f))))$.

We exploit Theorem 4.12 in order to show that *glb*'s of fixpoint complete abstract domains are still fixpoint complete.

PROPOSITION 4.13. *Let* $F \subseteq C \overset{m}{\rightarrow} C$ *and* $\{\rho_i\}_{i \in I} \subseteq \Delta(C, F)$. *Then,* $\sqcap_{i \in I} \rho_i \in \Delta(C, F)$.

PROOF. By Theorem 4.12, we know that $\sqcap_{i \in I} \rho^i \in \Delta(C, F)$ iff for all $f \in F$, $(\sqcap_{i \in I} \rho_i)(lfp(f)) = (\sqcap_{i \in I} \rho_i)(f((\sqcap_{i \in I} \rho_i)(lfp(f))))$. Then, since the *glb* $\sqcap$ is pointwise (cf. Section 2), for all $f \in F$ and $j \in I$, we have that:

$$\begin{aligned}
&(\sqcap_{i \in I} \rho_i)(f((\sqcap_{i \in I} \rho_i)(lfp(f)))) \leq \\
&\quad (\sqcap_{i \in I} \rho_i)(f(\rho_j(lfp(f)))) \leq \\
&\qquad \rho_j(f(\rho_j(lfp(f)))) = \text{(by Theorem 4.12, since } \rho_j \in \Delta(C, F)) \\
&\qquad \rho_j(lfp(f)),
\end{aligned}$$

and therefore $(\sqcap_{i \in I} \rho_i)(f((\sqcap_{i \in I} \rho_i)(lfp(f)))) \leq \wedge_{j \in I} \rho_j(lfp(f)) = (\sqcap_{j \in I} \rho_j)(lfp(f))$. Since the reverse inequality always holds, this closes the proof. □

5. *Making Abstract Interpretations Complete*

Since completeness and fixpoint completeness are properties that only depend on the underlying abstract domains, a natural question arises: Can we transform the abstract domains in order to achieve (fixpoint) completeness? Let us observe that Proposition 3.5(i) provides a first naïve answer to this question. In order to achieve completeness, replace the input abstraction with the corresponding whole concrete domain and/or the output abstraction with the trivial least informative abstraction, and similarly for fixpoint completeness. Of course, these trivial solutions are useless and unsatisfactory. Actually, this also means that the above question is badly stated. Instead, one should try to solve the following key problem: Can we *minimally* transform abstract domains in order to achieve (fixpoint) completeness? The interesting goal is here that of adding to or removing from an abstract domain the least amount of elements so that the resulting domain induces a (fixpoint) complete abstract interpretation. By exploiting the constructive characterization of completeness given in the previous section, we show that, whenever the semantic operations are continuous, the problems concerning completeness always admit solutions, and, remarkably, we provide explicit constructive characterizations for them. As far as fixpoint completeness is concerned, we prove that the problem of minimally restricting an abstract domain in order to achieve fixpoint completeness always admits solutions, and moreover we characterize them, while the existence of least domain extensions with respect to fixpoint completeness, in general, cannot be ensured even under very strong hypotheses.

5.1. RELATIVE COMPLETE CORES AND SHELLS. Let us first introduce the following operators transforming abstract domains—as usual, we follow the standard conventions $\sqcap\varnothing = \top$ and $\sqcup\varnothing = \bot$.

Definition 5.1. Given $F \subseteq C^n \overset{\text{m}}{\rightarrow} D$, $\rho \in uco(C)$ and $\eta \in uco(D)$, define

—$\mathscr{C}^{\rho}_{F}$: $uco(D) \rightarrow uco(D)$ where

$$\mathscr{C}^{\rho}_{F}(\mu) \overset{\text{def}}{=} \sqcap \{\beta \in uco(D) \mid \mu \sqsubseteq \beta, \langle \rho, \beta\rangle \in \Gamma(C,D,F)\};$$

—$\mathscr{S}^{\rho}_{F}$: $uco(D) \rightarrow uco(D)$ where

$$\mathscr{S}^{\rho}_{F}(\mu) \overset{\text{def}}{=} \sqcup \{\beta \in uco(D) \mid \beta \sqsubseteq \mu, \langle \rho, \beta\rangle \in \Gamma(C,D,F)\};$$

—$\mathscr{C}^{\eta}_{F}$: $uco(C) \rightarrow uco(C)$ where

$$\mathscr{C}^{\eta}_{F}(\varphi) \overset{\text{def}}{=} \sqcap \{\delta \in uco(C) \mid \varphi \sqsubseteq \delta, \langle \delta, \eta\rangle \in \Gamma(C,D,F)\};$$

—$\mathscr{S}^{\eta}_{F}$: $uco(C) \rightarrow uco(C)$ where

$$\mathscr{S}^{\eta}_{F}(\varphi) \overset{\text{def}}{=} \sqcup \{\delta \in uco(C) \mid \delta \sqsubseteq \varphi, \langle \delta, \eta\rangle \in \Gamma(C,D,F)\}.$$

Let $\langle \rho, \eta\rangle \in uco(C) \times uco(D)$ be a pair of abstract domains, and let us consider, for example, $\mathscr{S}^{\eta}_{F}$. Then, the output abstraction η is kept fixed, and, for any $\varphi \in uco(C)$, $\mathscr{S}^{\eta}_{F}(\varphi)$ yields the least common abstraction of all the domains $\delta \in uco(C)$ extending (i.e., more precise than) φ and such that $\langle \delta, \eta\rangle$ is complete for F. Similar explanations hold for the remaining operators. However, as consequences of Proposition 3.5 (vii), one gets the following two remarks:

(i) If $\langle \mathscr{C}^{\eta}_{F}(\varphi), \eta\rangle \in \Gamma(C, D, F)$, then $\mathscr{C}^{\eta}_{F}(\varphi) = \varphi$: By Proposition 3.5 (vii), since $\varphi \sqsubseteq \mathscr{C}^{\eta}_{F}(\varphi)$, we have that $\langle \varphi, \eta\rangle \in \Gamma(C, D, F)$, and therefore $\mathscr{C}^{\eta}_{F}(\varphi) = \varphi$.
(ii) If $\langle \rho, \mathscr{S}^{\rho}_{F}(\mu)\rangle \in \Gamma(C, D, F)$, then $\mathscr{S}^{\rho}_{F}(\mu) = \mu$: Similarly, this follows from Proposition 3.5 (vii).

Thus, one can draw the following important consequence: It does not make sense to wonder whether the greatest restriction φ^{g} of φ such that $\langle \varphi^{g}, \eta\rangle$ is complete for F exists, and whether the least extension μ^{l} of μ such that $\langle \rho, \mu^{l}\rangle$ is complete for F exists. This is because either they coincide with their arguments or they do not exist. This explains why we introduce just the following notions.

Definition 5.2. Let $F \subseteq C^n \overset{\text{m}}{\rightarrow} D$, $\rho \in uco(C)$ and $\eta \in uco(D)$.

(i) If $\langle \rho, \mathscr{C}^{\rho}_{F})\rangle \in \Gamma(C, D, F)$, then $\mathscr{C}^{\rho}_{F}(\eta)$ is called the *complete core* of η (for F) relative to ρ.
(ii) If $\langle \mathscr{S}^{\eta}_{F}(\rho), \eta\rangle \in \Gamma(C, D, F)$, then $\mathscr{S}^{\eta}_{F}(\rho)$ is called the *complete shell* of ρ (for F) relative to η.

By exploiting the constructive characterization of complete abstract interpretations given in Section 4, we provide a key characterization result for relative complete cores and shells. Under the hypothesis of continuity for the semantic operations, the following result explicitly states what one must add to ρ in order to get its complete shell relative to η and, dually, what one must subtract from η in order to get its complete core relative to ρ.

THEOREM 5.3. *Let* $F \subseteq C^n \xrightarrow{c} D$, $\rho \in uco(C)$, *and* $\eta \in uco(D)$.

(*i*) $\eta \sqcup L_F(\rho) = \eta \cap \{y \in D \mid \cup_{f\in F, i\in[1,n], \vec{x}\in C^n} max((f^i_{\vec{x}})^{-1}(\downarrow y)) \subseteq \rho\}$ *is the complete core of* η *relative to* ρ*;*
(*ii*) $\rho \sqcap R_F(\eta) = \mathscr{M}(\rho \cup (\cup_{f\in F, i\in[1,n], \vec{x}\in C^n, y\in\eta} max((f^i_{\vec{x}})^{-1}(\downarrow y))))$ *is the complete shell of* ρ *relative to* η*.*

PROOF. Corollary 4.7 gives the following characterizations for the operators $\mathscr{C}^{\rho}_{F}$: $uco(D) \to uco(D)$ and $\mathscr{S}^{\eta}_{F}$: $uco(C) \to uco(C)$ of Definition 5.1:

—$\mathscr{C}^{\rho}_{F}(\mu) = \sqcap\{\beta \in uco(D) \mid \mu, L_F(\rho) \sqsubseteq \beta\} = \mu \sqcup L_F(\rho)$;
—$\mathscr{S}^{\eta}_{F}(\varphi) = \sqcup\{\delta \in uco(C) \mid \delta \sqsubseteq \varphi, R_F(\eta)\} = \varphi \sqcap R_F(\eta)$.

Hence, since $L_F(\rho) \sqsubseteq \eta \sqcup L_F(\rho) = \mathscr{C}^{\rho}_{F}(\eta)$ and $\mathscr{S}^{\eta}_{F}(\rho) = \rho \sqcap R_F(\eta) \sqsubseteq R_F(\eta)$ trivially hold, by Corollary 4.7, we obtain that $\langle \rho, \mathscr{C}^{\rho}_{F}(\eta)\rangle$, $\langle \mathscr{S}^{\eta}_{F}(\rho), \eta\rangle \in \Gamma(C, D, F)$. Thus, according to Definition 5.2, $\eta \sqcup L_F(\rho)$ is the complete core of η relative to ρ, and $\rho \sqcap R_F(\eta)$ is the complete shell of ρ relative to η. □

It is yet worthwhile to stress the constructive flavor of the above result: It not only states that, for continuous semantic operations, both relative complete cores and shells exist, but it also states how to get them. Furthermore, it is important to remark that continuity for the semantic operations is definitely a reasonable and sufficiently weak hypothesis in order that the above theorem be widely applicable. Let us see how to use this theorem to derive the relative complete shell for the example on strictness analysis sketched in the introduction.

Example 5.4. Following Burn et al. [1986, Section 4], the concrete domains are given by the Hoare powerdomains $\mathbf{P}(\mathbb{N}_\perp \times \mathbb{N}_\perp)$ and $\mathbf{P}(Bool_\perp)$, ordered by set inclusion, where, for any directed-complete partial order (CPO) P with least element, $\mathbf{P}(P) \stackrel{\text{def}}{=} \{S \subseteq P \mid S \neq \varnothing, \downarrow S = S, (T \subseteq S \text{ and } T \text{ directed}) \Rightarrow \vee_P T \in S\}$—$\mathbb{N}_\perp$ and $Bool_\perp$ are considered as flat CPOs, and recall that if P is a CPO with least element then $\langle \mathbf{P}(P), \subseteq\rangle$ is a complete lattice [Abramsky and Jung 1994, Theorem 6.2.13]. Let $\rho \in uco(\mathbf{P}(\mathbb{N}_\perp \times \mathbb{N}_\perp))$ be the closure associated to the input abstract domain $S \times S$, and $\eta \in uco(\mathbf{P}(Bool_\perp))$ be the closure associated to the output abstract domain S. Hence, $\rho \stackrel{\text{def}}{=} \{\{\langle\perp, \perp\rangle\}, \{\perp\} \times \mathbb{N}_\perp, \mathbb{N}_\perp \times \{\perp\}, \mathbb{N}_\perp \times \mathbb{N}_\perp\}$ and $\eta \stackrel{\text{def}}{=} \{\{\perp\}, Bool_\perp\}$. The semantic function f: $\mathbf{P}(\mathbb{N}_\perp \times \mathbb{N}_\perp) \to \mathbf{P}(Bool_\perp)$ is clearly continuous and therefore, by Theorem 5.3 (i), the complete shell of ρ for f relative to η does exist, and it is given by the reduced product $\rho \sqcap R_f(\eta)$. Thus, for $Y \in \eta$, let us compute $\max(f^{-1}(\downarrow Y)) = \max(\{Z \in \mathbf{P}(\mathbb{N}_\perp \times \mathbb{N}_\perp) \mid f(Z) \subseteq Y\})$. We have that:

—$\max(f^{-1}(\downarrow Bool_\perp)) = \{\mathbb{N}_\perp \times \mathbb{N}_\perp\}$;
—$\max(f^{-1}(\downarrow \{\perp\})) = \max(\{Z \in \mathbf{P}(\mathbb{N}_\perp \times \mathbb{N}_\perp) \mid f(Z) \subseteq \{\perp\}\})$
$= \max(\{Z \in \mathbf{P}(\mathbb{N}_\perp \times \mathbb{N}_\perp) \mid \langle 3, 3\rangle \notin Z\})$
$= \{(\mathbb{N}_\perp \times \mathbb{N}_\perp)\setminus\{\langle 3, 3\rangle\}\}$.

Hence,

$$\rho \sqcap R_f(\eta) = \mathcal{M}(\rho \cup \{(\mathbb{N}_\perp \times \mathbb{N}_\perp)\setminus\{\langle 3, 3\rangle\}\}) = \rho \cup \{(\mathbb{N}_\perp \times \mathbb{N}_\perp)\setminus\{\langle 3, 3\rangle\}\}.$$

Thus, as announced in Section 1, and as one naturally expects, this shows that the complete shell of $S \times S$ relative to S can be obtained by adding a point $\langle \neq, \neq\rangle$ with concrete meaning $(\mathbb{N}_\perp \times \mathbb{N}_\perp)\setminus\{\langle 3, 3\rangle\}$, that is, denoting that the first and second components are not simultaneously equal to the value 3. In this way, we get a refined input abstract domain inducing a complete abstract interpretation for f. □

As far as complete cores are concerned, it is not hard to show that they exist even for merely monotone semantic operations, although, in this more general case, no explicit characterization can be given.

PROPOSITION 5.5. *If $F \subseteq C^n \overset{\mathrm{m}}{\to} D$, then there exists the complete core of any $\eta \in uco(D)$ relative to any $\rho \in uco(C)$.*

PROOF. Consider $\mathscr{C}_F^\rho(\eta) = \sqcap\{\beta \in uco(D) \mid \eta \sqsubseteq \beta, \langle \rho, \beta\rangle \in \Gamma(C, D, F)\}$. Then, by Proposition 4.10, $\langle \rho, \mathscr{C}_F^\rho(\eta)\rangle \in \Gamma(C, D, F)$, that is, there exists the complete core $\mathscr{C}_F^\rho(\eta)$ of η for F relative to ρ. □

Example 5.6. Let us consider Example 4.4. As observed there, the closure $\rho = \{\omega, \omega + 1\}$ on C is not complete for f. Notice that f is monotone but not continuous. Thus, by Proposition 5.5, the complete core of ρ relative to ρ does exist. It is trivially given by the greatest closure $\{\omega + 1\}$ in $uco(C)$: In fact, $\{\omega + 1\}$ is the unique proper abstraction of ρ, and since, by Proposition 3.5 (i), $\langle \rho, \{\omega + 1\}\rangle \in \Gamma(C, C, f)$, $\{\omega + 1\}$ actually is the complete core of ρ relative to ρ. On the other hand, as a consequence of Example 4.9, it is not hard to observe that the complete shell of ρ relative to ρ does not exist.

5.2. ABSOLUTE COMPLETE CORES AND SHELLS. Let us now turn to the case where the concrete semantic operations have type $F \subseteq C^n \to C$. Let us point out explicitly that, given some abstraction $\rho \in uco(C)$, if $\langle \mathscr{S}_F^\rho(\rho), \rho\rangle \in \Gamma(C, C, F)$, then $\mathscr{S}_F^\rho(\rho)$ is the complete shall of ρ for F relative to ρ itself, and therefore ρ as output abstraction is considered fixed. Instead, in this section, we solve the question (ii) as given in Section 1, and hence the aim is that of *simultaneously* and minimally extending or restricting both the input and output abstract domains in order to get completeness. Thus, let us introduce the following abstract domain transformers.

Definition 5.7. Given $F \subseteq C^n \overset{\mathrm{m}}{\to} C$, define:

—$\mathscr{C}_F$: $uco(C) \to uco(C)$ where

$$\mathscr{C}_F(\rho) \overset{\text{def}}{=} \sqcap\{\varphi \in uco(C) \mid \rho \sqsubseteq \varphi, \langle \varphi, \varphi\rangle \in \Gamma(C,C,F)\};$$

—$\mathscr{S}_F$: $uco(C) \to uco(C)$ where

$$\mathscr{S}_F(\rho) \overset{\text{def}}{=} \sqcup\{\varphi \in uco(C) \mid \varphi \sqsubseteq \rho, \langle \varphi, \varphi\rangle \in \Gamma(C,C,F)\}.$$

Then, question (ii) of Section 1 is formally stated as follows:

Definition 5.8. Let $F \subseteq C^n \overset{\mathrm{m}}{\to} C$ and $\rho \in uco(C)$.

(i) If $\langle C_F(\rho), \mathscr{C}_F(\rho)\rangle \in \Gamma(C, C, F)$, then $\mathscr{C}_F(\rho)$ is called the *absolute complete core* of ρ for F;

(ii) If $\langle \mathscr{S}_F(\rho), \mathscr{S}_F(\rho)\rangle \in \Gamma(C, C, F)$, then $\mathscr{S}_F(\rho)$ is called the *absolute complete shell* of ρ for F.

The following operators on abstract domains, which are based on the mappings L_F: $uco(C) \to uco(C)$ and R_F: $uco(C) \to uco(C)$ of Definition 4.6 instantiated to the type $C^n \to C$, will be essential for our constructive result of existence for absolute complete cores and shells.

Definition 5.9. Given $F \subseteq C^n \overset{\mathrm{m}}{\to} C$ and $\rho \in uco(C)$, define

—$\mathscr{R}_F^\rho$: $uco(C) \to uco(C)$ where $\mathscr{R}_F^\rho(\varphi) \overset{\mathrm{def}}{=} \rho \sqcap R_F(\varphi)$;
—$\mathscr{L}_{\mathrm{F}}^\rho$: $uco(C) \to uco(C)$ where $\mathscr{L}_F^\rho(\varphi) \overset{\mathrm{def}}{=} \rho \sqcup L_F(\varphi)$.

Hence, by Theorem 5.3, if each function in F is continuous, then $\mathscr{R}_F^\rho(\varphi)$ is the complete shell of ρ relative to φ and $\mathscr{L}_F^\rho(\varphi)$ is the complete core of ρ relative to φ. Let us consider, for example, $\mathscr{R}_F^\rho$ whenever F consists of continuous operations. If $\mathscr{R}_F^\rho(\varphi) = \varphi$, then $\varphi \sqsubseteq \rho$ and $\varphi \sqsubseteq R_F(\varphi)$. Hence, φ extends ρ, and additionally, by Corollary 4.7, $\langle\varphi, \varphi\rangle$ is complete for F. Since, for any $\rho \in uco(C)$, $\mathscr{R}_F^\rho$ is trivially a monotone operator on the complete lattice $\langle uco(C), \sqsubseteq\rangle$, and therefore, as recalled in Section 2, by Knaster–Tarski's theorem, it admits the greatest fixpoint, then it should be clear that this greatest fixpoint actually is the absolute complete shell of ρ. Of course, dual argumentations hold for $\mathscr{L}_F^\rho$. Thus, drawing on this reasoning, we get the following constructive result of existence for absolute complete shells and cores.

THEOREM 5.10. *Let $F \subseteq C^n \overset{\mathrm{c}}{\to} C$ and $\rho \in uco(C)$. Then $gfp(\mathscr{R}_F^\rho)$ and $lfp(\mathscr{L}_F^\rho)$ are, respectively, the absolute complete shell and core of ρ for F.*

PROOF. By Theorem 4.3 and Corollary 4.7, for any $\varphi \in uco(C)$, we have that:

—$\varphi \sqsubseteq \mathscr{R}_F^\rho(\varphi) \Leftrightarrow (\varphi \sqsubseteq \rho$ and $\varphi \sqsubseteq R_F(\varphi)) \Leftrightarrow (\varphi \sqsubseteq \rho$ and $\langle\varphi, \varphi\rangle \in \Gamma(C, C, F))$;
—$\mathscr{L}_F^\rho(\varphi) \sqsubseteq \varphi \Leftrightarrow (\rho \sqsubseteq \varphi$ and $L_F(\varphi) \sqsubseteq \varphi) \Leftrightarrow (\rho \sqsubseteq \varphi$ and $\langle\varphi, \varphi\rangle \in \Gamma(C, C, F))$.

Thus, the operators of Definition 5.7 are defined as follows:

—$\mathscr{S}_F(\rho) = \sqcup\{\varphi \in uco(C) \mid \varphi \sqsubseteq \mathscr{R}_F^\rho(\varphi)\}$;
—$\mathscr{C}_F(\rho) = \sqcap\{\varphi \in uco(C) \mid \mathscr{L}_F^\rho(\varphi) \sqsubseteq \varphi\}$.

Hence, since both $\mathscr{R}_F^\rho$ and $\mathscr{L}_F^\rho$ are monotone, by the fixpoint characterizations recalled in Section 2, we get that $\mathscr{S}_F(\rho) = gfp(\mathscr{R}_F^\rho)$ and $\mathscr{C}_F(\rho) = lfp(\mathscr{L}_F^\rho)$. Moreover, by Corollary 4.8, $\langle \mathscr{S}_F(\rho), \mathscr{S}_F(\rho)\rangle \in \Gamma(C, C, F)$, and by Proposition 4.10, $\langle \mathscr{C}_F(\rho), \mathscr{C}_F(\rho)\rangle \in \Gamma(C, C, F)$, and this closes the proof. □

Furthermore, the following easy observation shows that the above operators $\mathscr{R}_F^\rho$ and $\mathscr{L}_F^\rho$ are, respectively, co-continuous and continuous.

LEMMA 5.11. *Let $F \subseteq C^n \overset{\mathrm{c}}{\to} C$ and $\rho \in uco(C)$. Then, $\mathscr{R}_F^\rho$ is co-continuous and $\mathscr{L}_F^\rho$ is continuous.*

PROOF. Recall that $\mathcal{R}_F^\rho = \lambda\varphi.\rho \sqcap R_F(\varphi)$ and $\mathcal{L}_F^\rho = \lambda\varphi.\rho \sqcup L_F(\varphi)$. By Corollary 4.7, R_F and L_F are, respectively, co-additive and additive, and therefore this implies the thesis. □

Thus, in an analogous sense to Theorem 5.3, Theorem 5.10 above turns out to be a constructive existence result for absolute complete shell and cores. In fact, as recalled in Section 2, these can be obtained as limits of lower and upper Kleene's iteration sequences for $\mathcal{R}_F^\rho$ and $\mathcal{L}_F^\rho$, which, by Lemma 5.11, surely converge in at most ω steps. Let us also point out that $gfp(\mathcal{R}_F^\rho)$ and $lfp(\mathcal{L}_F^\rho)$ could be also understood as, respectively, the greatest fixpoint of R_F below (i.e., containing) ρ, and the least fixpoint of L_F above (i.e., contained in) ρ. Let us illustrate how to apply "by hand" these methodologies on the simple example of the rule of signs.

Example 5.12. Let us consider Example 3.4. It is easy to check that the abstraction $\rho_{10} = \{\mathbb{Z}, -0, \varnothing\}$ is not complete for the square operation sq: In fact, $\rho_{10}(sq(\{0\})) = \rho_{10}(\{0\}) = \{x \mid x \leq 0\}$, while $\rho_{10}(sq(\rho_{10}(\{0\}))) = \rho_{10}(sq(\{x \mid x \leq 0\})) = \rho_{10}(\{x \mid x \geq 0\}) = \mathbb{Z}$. Since sq is clearly continuous (but not additive) on $\langle \wp(\mathbb{Z}), \subseteq\rangle$, let us exploit Theorem 5.10 in order to compute the absolute complete core and shell of ρ_{10} for sq.

We have that, for any $\varphi \in uco(\wp(\mathbb{Z}))$, $\mathcal{L}_{sq}^{\rho_{10}}(\varphi) = \rho_{10} \sqcup L_{sq}(\varphi) = \{\mathbb{Z}, -0, \varnothing\} \cap \{Y \in \wp(\mathbb{Z}) \mid \max(\{X \in \wp(\mathbb{Z}) \mid sq(X) \subseteq Y\}) \subseteq \varphi\}$. Let us compute the least fixpoint of $\mathcal{L}_{sq}^{\rho_{10}}$, through the upper Kleene's iteration sequence $\bot_{uco(\wp(\mathbb{Z}))}$, $\mathcal{L}_{sq}^{\rho_{10}}(\bot_{uco(\wp(\mathbb{Z}))}), \ldots$

—$\bot_{uco(\wp(\mathbb{Z}))} = \wp(\mathbb{Z})$;

—$\mathcal{L}_{sq}^{\rho_{10}}(\wp(\mathbb{Z}))) = \rho_{10} \cap \{Y \in \wp(\mathbb{Z}) \mid \max(\{X \in \wp(\mathbb{Z}) \mid sq(X) \subseteq Y\}) \subseteq \wp(\mathbb{Z})\}$
$= \rho_{10} \cap \wp(\mathbb{Z}) = \rho_{10}$;

—$\mathcal{L}_{sq}^{\rho_{10}}(\rho_{10}) = \rho_{10} \cap \{Y \in \wp(\mathbb{Z}) \mid \max(\{X \in \wp(\mathbb{Z}) \mid sq(X) \subseteq Y\}) \subseteq \rho_{10}\}$
We have that:
$\mathbb{Z} \in \{Y \in \wp(\mathbb{Z}) \mid \max(\{X \in \wp(\mathbb{Z}) \mid sq(X) \subseteq Y\}) \subseteq \rho_{10}\}$,
$-0 \notin \{Y \in \wp(\mathbb{Z}) \mid \max(\{X \in \wp(\mathbb{Z}) \mid sq(X) \subseteq Y\}) \subseteq \rho_{10}\}$
(because $\max(\{X \in \wp(\mathbb{Z}) \mid sq(X) \subseteq -0\}) = \{\{0\}\}$),
$\varnothing \in \{Y \in \wp(\mathbb{Z}) \mid \max(\{X \in \wp(\mathbb{Z}) \mid sq(X) \subseteq Y\}) \subseteq \rho_{10}\}$,
and hence:
$= \{\mathbb{Z}, \varnothing\}$;

—$\mathcal{L}_{sq}^{\rho_{10}}(\{\mathbb{Z}, \varnothing\}) = \{\mathbb{Z}, \varnothing\}$.

Thus, $lfp(\mathcal{L}_{sq}^{\rho_{10}}) = \{\mathbb{Z}, \varnothing\}$. Hence, our algorithm determined that incompleteness was due to the point $-0 \in \rho_{10}$, and that by removing that point one gets the absolute complete core of ρ_{10} for sq, as it would not have been hard to check in a direct way.

Let us turn to the absolute complete shell. Then, for any $\varphi \in uco(\wp(\mathbb{Z}))$, $\mathcal{R}_{sq}^{\rho_{10}}(\varphi) = \rho_{10} \sqcap R_{sq}(\varphi) = \mathcal{M}(\{\mathbb{Z}, -0, \varnothing\} \cup (\bigcup_{Y\in\varphi} \max(\{X \in \wp(\mathbb{Z}) \mid sq(X) \subseteq Y\})))$. The greatest fixpoint of $\mathcal{R}_{sq}^{\rho_{10}}$ is obtained as limit of the lower Kleene's iteration sequence $\top_{uco(\wp(\mathbb{Z}))}$, $\mathcal{R}_{sq}^{\rho_{10}}(\top_{uco(\wp(\mathbb{Z}))}), \ldots$

—$\top_{uco(\wp(\mathbb{Z}))} = \{\mathbb{Z}\}$;

—$\mathcal{R}_{sq}^{\rho_{10}}(\{\mathbb{Z}\}) = \mathcal{M}(\rho_{10} \cup \max(\{X \in \wp(\mathbb{Z}) \mid sq(X) \subseteq \mathbb{Z}\}))$
$= \mathcal{M}(\rho_{10} \cup \{\mathbb{Z}\}) = \rho_{10}$;

—$\mathcal{R}_{sq}^{\rho_{10}}(\rho_{10}) = \mathcal{M}(\rho_{10} \cup (\bigcup_{Y \in \rho_{10}} \max(\{X \in \wp(\mathbb{Z}) \mid sq(X) \subseteq Y\})))$
$= \mathcal{M}(\rho_{10} \cup \{\mathbb{Z}, \{0\}, \varnothing\}) = \{\mathbb{Z}, -0, 0, \varnothing\};$

—$\mathcal{R}_{sq}^{\rho_{10}}(\{\mathbb{Z}, -0, 0, \varnothing\}) = \{\mathbb{Z}, -0, 0, \varnothing\}$
(because $\max(\{X \in \wp(\mathbb{Z}) \mid sq(X) \subseteq \{0\}\}) = \{0\}$).

Thus, $gfp(\mathcal{R}_{sq}^{\rho_{10}}) = \{\mathbb{Z}, -0, 0, \varnothing\}$. Hence, the procedure of Theorem 5.10 automatically determined that by adding the point 0 to ρ_{10}, we get its absolute complete shell for sq.

As a consequence of Theorem 5.10, under the hypothesis of continuity, absolute complete core and shell operators give rise to an adjunction on $uco(C)$.

LEMMA 5.13. *If $F \subseteq C^n \xrightarrow{c} C$, then $(\mathscr{C}_F, uco(C), uco(C), \mathscr{S}_F)$ is a GC.*

PROOF. By Theorem 5.10, for any $\rho \in uco(C)$, $\langle \mathscr{C}_F(\rho), \mathscr{C}_F(\rho)\rangle$, $\langle \mathscr{S}_F(\rho), \mathscr{S}_F(\rho)\rangle \in \Gamma(C, C, F)$. Thus, for any $\rho \in uco(C)$, $\rho \sqsubseteq \mathscr{C}_F(\rho) = \mathscr{S}_F(\mathscr{C}_F(\rho))$ and $\mathscr{C}_F(\mathscr{S}_F(\rho)) = \mathscr{S}_F(\rho) \sqsubseteq \rho$. Hence, since $\mathscr{C}_F$ and $\mathscr{S}_F$ are both monotone, they form a GC. □

The above result admits interesting consequences. Since $\mathscr{C}_F$ and $\mathscr{S}_F$ are, respectively, left- and right-adjoint maps on a complete lattice, they are, respectively, additive and co-additive. This fact is especially interesting as far as absolute complete shells are concerned. In fact, to be co-additive for $\mathscr{S}_F$ means that absolute complete shells can be modularly computed by reduced product: If $\rho = \varphi_1 \sqcap \varphi_2$ then $\mathscr{S}_F(\rho) = \mathscr{S}_F(\varphi_1) \sqcap \mathscr{S}_F(\varphi_2)$. Thus, whenever some abstract domain ρ has been incrementally designed by reduced product, this property permits to incrementally compute by reduced product the absolute complete shell of ρ by applying Theorem 5.10 to the components of its decomposition, and this may sometimes simplify the task. We will see an example of this technique in Section 6.1 concerning the reduced product *Sign* $\sqcap$ *Parity* of Example 4.11.

5.3. FIXPOINT COMPLETE CORES.

Let us now face the problem of making abstract interpretations fixpoint complete. Analogously to the completeness case, we first introduce the following two basic definitions, where Definition 5.15 below formalizes the question (iii) of Section 1.

Definition 5.14. Given $G \subseteq C \xrightarrow{m} C$, define:

—$\mathbb{C}_G$: $uco(C) \to uco(C)$ where

$$\mathbb{C}_G(\rho) \overset{\text{def}}{=} \sqcap\{\varphi \in uco(C) \mid \rho \sqsubseteq \varphi, \varphi \in \Delta(C, G)\};$$

—$\mathbb{S}_G$: $uco(C) \to uco(C)$ where

$$\mathbb{S}_G(\rho) \overset{\text{def}}{=} \sqcup\{\varphi \in uco(C) \mid \varphi \sqsubseteq \rho, \varphi \in \Delta(C, G)\}.$$

Definition 5.15. Let $G \subseteq C \xrightarrow{m} C$ and $\rho \in uco(C)$.

(i) If $\mathbb{C}_G(\rho) \in \Delta(C, G)$ then $\mathbb{C}_G(\rho)$ is called the *fixpoint complete core* of ρ for G.
(ii) If $\mathbb{S}_G(\rho) \in \Delta(C, G)$, then $\mathbb{S}_G(\rho)$ is called the *fixpoint complete shell* of ρ for G.

As announced, even when the concrete domain and the semantic operators satisfy very strong hypotheses, in general, it cannot be guaranteed that fixpoint complete shells exist, as shown by the following examples:

Example 5.16. In Example 3.4, we have shown that $\rho_5 = \{\mathbb{Z}, -0\}$ is not fixpoint complete for the square operation sq on $\wp(\mathbb{Z})$. Moreover, $\rho_9, \rho_{10} \in \Delta(\wp(\mathbb{Z}), sq)$ and it holds $\rho_9 \sqcup \rho_{10} = \rho_5$. Thus, $\rho_9, \rho_{10} \sqsubseteq \mathbb{S}_{sq}(\rho_5)$. Hence, since $\mathbb{S}_{sq}(\rho_5) \sqsubseteq \rho_5$, we get $\mathbb{S}_{sq}(\rho_5) = \rho_5$. This means that the fixpoint complete shell of ρ_5 for sq does not exist.

Let us consider a finite chain of five points $C \overset{\text{def}}{=} \{0 < 1 < 2 < 3 < 4\}$ and the function g: $C \to C$ defined as $g \overset{\text{def}}{=} \{0 \mapsto 0, 1 \mapsto 0, 2 \mapsto 0, 3 \mapsto 4, 4 \mapsto 4\}$. Note that g is monotone and hence it is both additive and co-additive, while $lfp(g) = 0$. Next, consider the closures $\varphi_1, \varphi_2 \in uco(C)$ given by $\varphi_1 \overset{\text{def}}{=} \{1, 3, 4\}$ and $\varphi_2 \overset{\text{def}}{=} \{2, 3, 4\}$. It is not difficult to verify that $\varphi_1, \varphi_2 \in \Delta(C, g)$: Indeed, $\varphi_k(lfp(g)) = k = lfp(\varphi_k \circ g)$, for $k \in \{1, 2\}$. It turns out that their *lub* $\varphi_1 \sqcup \varphi_2 = \{3, 4\}$ does not belong to $\Delta(C, g)$. In fact, $(\varphi_1 \sqcup \varphi_2)(lfp(g)) = 3$, while $(\varphi_1 \sqcup \varphi_2)(g((\varphi_1 \sqcup \varphi_2)(lfp(g)))) = (\varphi_1 \sqcup \varphi_2)(g(3)) = 4$, and therefore, by Theorem 4.12, $\varphi_1 \sqcup \varphi_2 \notin \Delta(C, g)$. Thus, by reasoning as above, we can conclude that the fixpoint complete shell of $\varphi_1 \sqcup \varphi_2$ for g does not exist. □

Let us draw more in detail the consequences of the above examples. Example 5.16 shows that even when the concrete domain is either an atomic complete Boolean algebra or a finite chain, and the concrete operator is both additive and co-additive, fixpoint complete shells do not necessarily exist. Thus, this observation precludes us the possibility of finding some reasonable, sufficient conditions on the concrete domain and/or on the semantic operators ensuring us the existence of fixpoint complete shells of abstract domains. Nevertheless, it is worth remarking that a fixpoint complete shell of a given non fixpoint complete abstract domain may well exist, that is, it may well happen that for some $\rho \in uco(C)$, $\mathbb{S}_G(\rho) \in \Delta(C, G)$.

In contrast, fixpoint complete cores always exist, and they can be constructively characterized. In fact, Theorem 4.12 implicitly specifies which elements of an abstract domain have to be removed in order to achieve fixpoint completeness. Then, similarly to what has been done in Section 5.2 for absolute complete cores, let us introduce the following operator transforming abstract domains accordingly to Theorem 4.12.

Definition 5.17. Let $G \subseteq C \overset{\text{m}}{\to} C$ and $\rho \in uco(C)$. $\mathscr{F}^{\rho}_{G}$: $uco(C) \to uco(C)$ is defined by: $\mathscr{F}^{\rho}_{G}(\varphi) \overset{\text{def}}{=} \rho \sqcup \{y \in C \mid \forall g \in G.\ \varphi(lfp(g)) \le y \Rightarrow \varphi(g(\varphi(lfp(g)))) \le y\}$.

Let us observe that $\mathscr{F}_G$ is a monotone operator on $uco(C)$, and therefore it admits the least fixpoint.

THEOREM 5.18. *Let $G \subseteq C \overset{\text{m}}{\to} C$ and $\rho \in uco(C)$. Then, $lfp(\mathscr{F}^{\rho}_{G})$ is the fixpoint complete core of ρ for G.*

PROOF. For any $\varphi \in uco(C)$, by Theorem 4.12, we have that:

$\mathcal{F}^{\rho}_{G}(\varphi) \sqsubseteq \varphi$ $\Leftrightarrow$
$\rho \sqsubseteq \varphi$ and $\varphi \sqsupseteq \{y \in C \mid \forall g \in G.\varphi(lfp(g)) \leq y \Rightarrow \varphi(g(\varphi(lfp(g)))) \leq y\}$ $\Leftrightarrow$
$\rho \sqsubseteq \varphi$ and $\varphi \in \Delta(C, G)$.

Thus, the operator $\mathbb{C}_G$ of Definition 5.14 can be rewritten as follows:

$$\mathbb{C}_G(\rho) = \sqcap\{\varphi \in uco(C) \mid \mathcal{F}^{\rho}_{G}(\varphi) \sqsubseteq \varphi\}.$$

Hence, by the characterization of least fixpoints recalled in Section 2, we get that $\mathbb{C}_G(\rho) = lfp(\mathcal{F}^{\rho}_{G})$. Moreover, by Proposition 4.13, $\mathbb{C}_G(\rho) \in \Delta(C, G)$, and therefore $lfp(\mathcal{F}^{\rho}_{G})$ is the fixpoint complete core of ρ for G. □

Thus, analogously to absolute complete cores, the above result characterizes fixpoint complete cores as least fixpoints of a suitable operator on the lattice of abstract interpretations. Notice that $lfp(\mathcal{F}^{\rho}_{G})$ can be also understood as the least fixpoint of the operator $\lambda\varphi.\{y \in C \mid \forall g \in G.\varphi(lfp(g)) \leq y \Rightarrow \varphi(g(\varphi(lfp(g)))) \leq y\}$ above (i.e., contained in) ρ. Let us see a simple example.

Example 5.19. Let us consider Example 3.4. We have already observed that $\rho_5 = \{\mathbb{Z}, -0\}$ is not fixpoint complete for sq. Moreover, in Example 5.16 we have shown that the fixpoint complete shell of ρ_5 for sq does not exist. On the other hand, by Theorem 5.18, the fixpoint complete core of ρ_5 for sq does exist. Since the greatest closure $\{\mathbb{Z}\}$ of $uco(\wp(\mathbb{Z}))$ is the unique proper abstraction of ρ_5, trivially $\{\mathbb{Z}\}$ is the fixpoint complete core of ρ_5 for sq. Let us see how the characterization of Theorem 5.18 allows us to remove the point -0 from ρ_5 in order to get $\{\mathbb{Z}\}$. In fact, $lfp(\mathcal{F}^{\rho_5}_{sq}) = \rho_5 \cap \{X \in \wp(\mathbb{Z}) \mid \rho_5(lfp(sq)) \subseteq X \Rightarrow \rho_5(sq(\rho_5(lfp(sq)))) \subseteq X\}$. Since $lfp(sq) = \varnothing$, and $-0 = \rho_5(lfp(sq)) \subseteq -0$ while $\mathbb{Z} = \rho_5(sq(\rho_5(lfp(sq)))) \nsubseteq -0$, this actually means that -0 must be removed from ρ_5, and therefore $lfp(\mathcal{F}^{\rho_5}_{sq}) = \{\mathbb{Z}\}$.

5.4. On Constructivity Issues.

In Sections 5.1, 5.2, and 5.3, we positively solved the problem of making abstract interpretations complete by constructively showing: (i) under the reasonable hypothesis of dealing with continuous semantic operations, one can minimally extend or restrict any abstract domain in order to achieve completeness; (ii) for merely monotone semantic operators, one can minimally restrict any abstract domain in order to achieve fixpoint completeness. This means that Theorems 5.3, 5.10, and 5.18 yield practical methods to compute "by hand" complete cores and shells of domains, both relative and absolute, as well as fixpoint complete cores.

The termination of these methods cannot be ensured in general for any possible input of concrete and abstract interpretations. For instance, termination could be prevented by decidability issues, as sketched by the following (simplicistic) scenario. Let us assume that an undecidable program semantic property $\mathcal{P}$—for example, strictness for functional programs or groundness for logic programs—is represented by a sufficiently rich least fixpoint based concrete interpretation $\langle C, \{T_P\}_{P\in Program}\rangle$, and that a decidable approximation is given by an abstract interpretation $\langle A, \{T^{\sharp}_P\}_{P\in Program}\rangle$ over a finite abstract domain A. If A is rich enough to encode the semantic property $\mathcal{P}$—that is, for any program P, $lfp(T_P)$ tells that P satisfies $\mathcal{P}$ iff $\alpha_{C,A}(lfp(T_P))$ tells that P satisfies $\mathcal{P}$—then, by undecidability of $\mathcal{P}$, A cannot be complete: Otherwise, since completeness implies fixpoint completeness, one could decide $\mathcal{P}$ over the finite abstract domain A by means of $lfp(T^{\sharp}_{P})$. Likewise, the absolute complete shell of A cannot be a finite abstract domain, otherwise it would allow to decide $\mathcal{P}$. Thus, the absolute complete shell of the finite abstract domain A must be infinite, and therefore cannot be obtained in a fully automatic way.

In spite of that, it is anyhow worthwhile to remark that in finite settings our constructive theorems actually give rise to terminating algorithms which fully automatize the process of making abstract interpretations complete. As a notable example, in view of the relativity of the concept of being an "abstract" interpretation, such finite scenarios arise when reasoning between static program analyses. For instance, we will discuss in Section 6.2 how complete shells may play a relevant role in designing suitable rich abstract domains for program analysis, and therefore how such design strategies may be computer assisted.

5.5. A Simplifying Property.

Concrete domains are not always the best place where abstract domain completeness and fixpoint completeness should be verified. In fact, in the following, we show that given a concrete interpretation $\mathcal{C}$ and a corresponding (fixpoint) complete abstract interpretation $\mathcal{I}$, if $\mathcal{J}$ is a further abstraction of $\mathcal{I}$, then $\mathcal{J}$ is (fixpoint) complete for $\mathcal{C}$ iff $\mathcal{J}$ is (fixpoint) complete for $\mathcal{I}$. This property allows us to perform the computation of (relative, absolute and fixpoint) complete cores and shells with respect to (fixpoint) complete abstract interpretations. We will argue that this could make easier the accomplishment of these tasks since, roughly, a complete abstract interpretation may well be simpler than its reference concrete interpretation.

Recall from Section 3.2 that given a concrete semantic operation $f\colon C^n \xrightarrow{\text{m}} D$ and a pair of abstract domains $\langle \varphi, \mu \rangle \in uco(C) \times uco(D)$, the best correct approximation $f^{b_{\varphi,\mu}}$ of f for φ and μ is $f^{b_{\varphi,\mu}} = \mu \circ f\colon \varphi(C)^n \xrightarrow{\text{m}} \mu(D)$—if F is a set of functions, then $F^{b_{\varphi,\mu}}$ denotes the corresponding set of best correct approximations. Note that $F^{b_{\varphi,\mu}}$ preserves monotonicity. Given a further pair $\langle \rho, \eta \rangle \in uco(C) \times uco(D)$ of abstractions of $\langle \varphi, \mu \rangle$, that is, such that $\langle \varphi, \mu \rangle \sqsubseteq \langle \rho, \eta \rangle$, we can reason on the completeness of $\langle \rho, \eta \rangle$ relatively to the interpretation $\langle \varphi(C), \mu(D), F^{b_{\varphi,\mu}} \rangle$. In fact, as a consequence of the following observation (whose detailed proof can be found, for example, in Cousot [1978, Theorem 4.2.0.4.7]), $\rho \in uco(\varphi(C))$ and $\eta \in uco(\mu(D))$, and therefore it makes sense to ask whether $\langle \rho, \eta \rangle$ is complete with respect to $\langle \varphi(C), \mu(D), F^{b_{\varphi,\mu}} \rangle$. Analogous considerations hold for fixpoint completeness.

LEMMA 5.20. *Assume C be a complete lattice and $\varphi \in uco(C)$, then*:

$$\langle uco(\langle \varphi(C), \leq_C \rangle), \sqsubseteq \rangle \cong \langle \{\rho \in uco(C) \mid \varphi \sqsubseteq \rho\}, \sqsubseteq \rangle.$$

PROOF SKETCH. Both structures are complete lattices. The isomorphism is given by the identity mapping on sets of fixpoints of closures. If $\rho \in uco(\varphi(C))$ then its set of fixpoints $\rho(\varphi(C))$ is a Moore-family of $\langle \varphi(C), \leq_C \rangle$, and because $\varphi(C)$ is a Moore-family of C, it turns out that $\rho(\varphi(C))$ is a Moore-family of C, too. On the reverse direction, if $\rho \in uco(C)$ and $\varphi \sqsubseteq \rho$, then $\rho(C) \subseteq \varphi(C)$, and therefore $\rho(C)$ is a Moore-family of $\langle \varphi(C), \leq_C \rangle$. It is then simple to check that this gives rise to an isomorphism. $\square$

Thus, the next result shows that given $\langle \varphi, \mu \rangle$ complete for F, $\langle \rho, \eta \rangle$ is complete for F iff it is complete relatively to $\langle \varphi(C), \mu(D), F^{b_{\varphi,\mu}} \rangle$, and similarly for fixpoint completeness.

THEOREM 5.21. *Let $F \subseteq C^n \xrightarrow{\mathrm{m}} D$, $G \subseteq C \xrightarrow{\mathrm{m}} C$, $\varphi \in uco(C)$ and $\mu \in uco(D)$.*

(i) *Let $\langle \varphi, \mu \rangle \in \Gamma(C, D, F)$ and $\langle \rho, \eta \rangle \in uco(C) \times uco(D)$ such that $\langle \varphi, \mu \rangle \sqsubseteq \langle \rho, \eta \rangle$. Then, $\langle \rho, \eta \rangle \in \Gamma(\varphi(C), \mu(D), F^{b_{\varphi,\mu}})$ iff $\langle \rho, \eta \rangle \in \Gamma(C, D, F)$.*

(ii) *Let $\varphi \in \Delta(C, G)$ and $\rho \in uco(C)$ such that $\varphi \sqsubseteq \rho$. Then, $\rho \in \Delta(\varphi(C), G^{b_{\varphi,\varphi}})$ iff $\rho \in \Delta(C, G)$.*

PROOF.
(i) This is proved by the following equalities:

$$\begin{array}{lr}
\Gamma(\varphi(C), \mu(D), F^{b_{\varphi,\mu}}) & = \\
\{\langle \rho, \eta \rangle \in uco(\varphi(C)) \times uco(\mu(D)) \mid \forall f^{b_{\varphi,\mu}} \in F^{b_{\varphi,\mu}}.\eta \circ f^{b_{\varphi,\mu}} = \eta \circ f^{b_{\varphi,\mu}} \circ \vec{\rho}\} & = \\
 & (\text{since } f^{b_{\varphi,\mu}} = \mu \circ f\colon \varphi(C)^n \to \mu(D)) \\
\{\langle \rho, \eta \rangle \in uco(\varphi(C)) \times uco(\mu(D)) \mid \forall f \in F.\eta \circ \mu \circ f \circ \vec{\varphi} = \eta \circ \mu \circ f \circ \vec{\varphi} \circ \vec{\rho}\} & = \\
 & (\text{by Lemma 5.20}) \\
\{\langle \rho, \eta \rangle \in uco(C) \times uco(D) \mid \langle \varphi, \mu \rangle \sqsubseteq \langle \rho, \eta \rangle, \forall f \in F.\eta \circ \mu \circ f \circ \vec{\varphi} = \eta \circ \mu \circ f \circ \vec{\varphi} \circ \vec{\rho}\} & = \\
 & (\text{since } \langle \varphi, \mu \rangle \in \Gamma(C, D, F)) \\
\{\langle \rho, \eta \rangle \in uco(C) \times uco(D) \mid \langle \varphi, \mu \rangle \sqsubseteq \langle \rho, \eta \rangle, \forall f \in F.\eta \circ \mu \circ f = \eta \circ \mu \circ f \circ \vec{\rho}\} & = \\
 & (\text{since, by (iii) in Section 2, } \eta \circ \mu = \eta) \\
\{\langle \rho, \eta \rangle \in uco(C) \times uco(D) \mid \langle \varphi, \mu \rangle \sqsubseteq \langle \rho, \eta \rangle, \forall f \in F.\eta \circ f = \eta \circ f \circ \vec{\rho}\} & = \\
\{\langle \rho, \eta \rangle \in uco(C) \times uco(D) \mid \langle \varphi, \mu \rangle \sqsubseteq \langle \rho, \eta \rangle, \langle \rho, \eta \rangle \in \Gamma(C, D, F)\}. &
\end{array}$$

(ii) This is proved by the following equalities:

$$\begin{array}{lr}
\Delta(\varphi(C), G^{b_{\varphi,\varphi}}) & = \\
\{\rho \in uco(\varphi(C)) \mid \forall g^{b_{\varphi,\varphi}} \in G^{b_{\varphi,\varphi}}.\rho(lfp(g^{b_{\varphi,\varphi}})) = lfp(\rho \circ g^{b_{\varphi,\varphi}})\} & = \\
 & (\text{by Theorem 4.12}) \\
\{\rho \in uco(\varphi(C)) \mid \forall g^{b_{\varphi,\varphi}} \in G^{b_{\varphi,\varphi}}.\rho(lfp(g^{b_{\varphi,\varphi}})) = \rho(g^{b_{\varphi,\varphi}}(\rho(lfp(g^{b_{\varphi,\varphi}}))))\} & = \\
 & (\text{since } g^{b_{\varphi,\varphi}} = \varphi \circ g\colon \varphi(C) \to \varphi(C)) \\
\{\rho \in uco(\varphi(C)) \mid \forall g \in G.\rho(lfp(\varphi \circ g \circ \varphi)) = \rho \circ \varphi \circ g \circ \varphi \circ \rho(lfp(\varphi \circ g \circ \varphi))\} & = \\
 & (\text{by Lemma 5.20}) \\
\{\rho \in uco(C) \mid \varphi \sqsubseteq \rho, \forall g \in G.\rho(lfp(\varphi \circ g \circ \varphi)) = \rho \circ \varphi \circ g \circ \varphi \circ \rho(lfp(\varphi \circ g \circ \varphi))\} & = \\
 & (\text{since } \varphi \in \Delta(C, G), \text{ by Lemma 3.3 (ii)}) \\
\{\rho \in uco(C) \mid \varphi \sqsubseteq \rho, \forall g \in G.\rho(\varphi(lfp(g))) = \rho \circ \varphi \circ g \circ \varphi \circ \rho(\varphi(lfp(g)))\} & = \\
 & (\text{since, by (iii) in Section 2, } \rho \circ \varphi = \varphi \circ \rho = \rho) \\
\{\rho \in uco(C) \mid \varphi \sqsubseteq \rho, \forall g \in G.\rho(lfp(g)) = \rho \circ g \circ \rho(lfp(g))\} & = \\
 & (\text{by Theorem 4.12}) \\
\{\rho \in uco(C) \mid \varphi \sqsubseteq \rho, \rho \in \Delta(C, G)\}. \quad \square &
\end{array}$$

As a consequence, we get the following characterization result.

THEOREM 5.22. *Let $F \subseteq C^n \xrightarrow{\mathrm{m}} D$, $G \subseteq C^n \xrightarrow{\mathrm{m}} C$, $H \subseteq C \xrightarrow{\mathrm{m}} C$, $\varphi \in uco(C)$ and $\mu \in uco(D)$.*

(1) *Let $\langle \varphi, \mu \rangle \in \Gamma(C, D, F)$ and $\langle \rho, \eta \rangle \in uco(C) \times uco(D)$ such that $\langle \varphi, \mu \rangle \sqsubseteq \langle \rho, \eta \rangle$.*
   (1.1) *If there exists the complete core of η relative to ρ for the interpretation $\langle C, D, F \rangle$, then there exists the complete core of η relative to ρ for the interpretation $\langle \varphi(C), \mu(D), F^{b_{\varphi,\mu}} \rangle$. In this case, they coincide.*
   (1.2) *If there exists the complete shell of ρ relative to η for the interpretation $\langle C, D, F \rangle$, then there exists the complete shell of ρ relative to η for the interpretation $\langle \varphi(C), \mu(D), F^{b_{\varphi,\mu}} \rangle$. In this case, they coincide.*

(2) *Let $\langle \varphi, \varphi \rangle \in \Gamma(C, C, G)$ and $\rho \in uco(C)$ such that $\varphi \sqsubseteq \rho$. If there exists the absolute complete core (shell) of ρ for the interpretation $\langle C, C, G \rangle$, then there exists the absolute complete core (shell) of ρ for the interpretation $\langle \varphi(C), \varphi(C), G^{b_{\varphi,\varphi}} \rangle$. In this case, they coincide.*

(3) *Let $\varphi \in \Delta(C, H)$ and $\rho \in uco(C)$ such that $\varphi \sqsubseteq \rho$. If there exists the fixpoint complete core (shell) of ρ for the interpretation $\langle C, C, H \rangle$, then there exists the fixpoint complete core (shell) of ρ for the interpretation $\langle \varphi(C), \varphi(C), H^{b_{\varphi,\varphi}} \rangle$. In this case, they coincide.*

PROOF. The proof consists of a careful application of Theorem 5.21 combined with Lemma 5.20. We prove (1.1) only, because proofs for the other points follow an analogous pattern.

Firstly observe that, by Lemma 5.20, $\langle \rho, \eta \rangle \in uco(\varphi(C)) \times uco(\mu(D))$. The hypothesis means that

$$\langle \rho, \sqcap_{uco(D)}\{\beta \in uco(D) \mid \eta \sqsubseteq \beta, \langle \rho, \beta \rangle \in \Gamma(C, D, F)\}\rangle \in \Gamma(C, D, F).$$

Thus, since, by Lemma 5.20, $uco(\mu(D))$ is a complete lattice isomorphic to $\{\psi \in uco(C) \mid \mu \sqsubseteq \psi\}$, we have that:

$$\langle \rho, \sqcap_{uco(\mu(D))}\{\beta \in uco(\mu(D)) \mid \eta \sqsubseteq \beta, \langle \rho, \beta \rangle \in \Gamma(C, D, F)\}\rangle \in \Gamma(C, D, F).$$

Thus, by applying twice Theorem 5.21, we have $\langle \rho, \sqcap_{uco(\mu(D))}\{\beta \in uco(\mu(D)) \mid \eta \sqsubseteq \beta, \langle \rho, \beta \rangle \in \Gamma(\varphi(C), \mu(D), F^{b_{\varphi,\mu}})\}\rangle \in \Gamma(\varphi(C), \mu(D), F^{b_{\varphi,\mu}})$, that is, there exists the complete core of η relative to ρ for the interpretation $\langle \varphi(C), \mu(D), F^{b_{\varphi,\mu}} \rangle$. Furthermore, notice that above we have also shown that

$$\begin{aligned}&\sqcap_{uco(D)}\{\beta \in uco(D) \mid \eta \sqsubseteq \beta, \langle \rho, \beta \rangle \in \Gamma(C, D, F)\}\\ &\qquad = \sqcap_{uco(\mu(D))}\{\beta \in uco(\mu(D)) \mid \eta \sqsubseteq \beta, \langle \rho, \beta \rangle \in \Gamma(\varphi(C), \mu(D), F^{b_{\varphi,\mu}})\},\end{aligned}$$

where the equality symbol is intended as equality of sets of fixpoints, that is, the isomorphism of Lemma 5.20, and hence, the two relative complete cores coincide. □

Let us consider, for example, the case of absolute complete shells of domains (analogous considerations apply for the other cases). Then, point (2) of the above theorem states that given $\varphi \in uco(C)$ such that $\langle \varphi, \varphi \rangle \in \Gamma(C, C, G)$, then for any $\rho \in uco(C)$ more abstract than φ, that is, such that $\varphi \sqsubseteq \rho$, if the absolute complete shell of ρ with respect to the "real" concrete interpretation $\langle C, C, G \rangle$ does exist, then it can be equivalently computed as absolute complete shell of ρ with respect to the more abstract, but still concrete from the viewpoint of ρ, interpretation $\langle \varphi(C), \varphi(C), G^{b_{\varphi,\varphi}} \rangle$. This and the other analogous properties may introduce some simplification in the task of computing (relative, absolute, fixpoint) complete cores and shells, since a (fixpoint) complete abstract interpretation might well be less complex than its concrete counterpart. Even more, the following fact shows that best correct approximations over complete abstract domains preserve the property of continuity, and therefore this enables us to apply the constructive methodologies of Theorems 5.3 and 5.10 combined with Theorem 5.22.

LEMMA 5.23. *If $F \subseteq C^n \xrightarrow{c} D$ and $\langle\rho, \eta\rangle \in \Gamma(C, D, F)$, $F^{b_{\rho,\eta}} \subseteq \rho(C)^n \xrightarrow{c} \eta(D)$.*

PROOF. Let $f \in F$ and $Y \subseteq \rho(C)^n$ be a chain. Hence, if, for $i \in [1, n]$, $Y_i \overset{\text{def}}{=} \{z \in \rho(C) \mid \exists \vec{y} \in Y.\vec{y}_i = z\}$, then $\vee_{\rho(C)^n}Y = \langle\vee_{\rho(C)}Y_1, \ldots, \vee_{\rho(C)}Y_n\rangle$. Recall that $f^{b_{\rho,\eta}} = \eta \circ f$: $\rho(C)^n \to \eta(D)$. Then:

$$\begin{array}{lll}
f^{b_{\rho,\eta}}(\vee_{\rho(C)^n}Y) & = & \\
\eta(f(\langle\vee_{\rho(C)}Y_1, \ldots, \vee_{\rho(C)}Y_n\rangle)) & = & \text{(by definition of the } \textit{lub } \vee_{\rho(C)}\text{, cf. Section 2)} \\
\eta(f(\langle\rho(\vee_C Y_1), \ldots, \rho(\vee_C Y_n)\rangle)) & = & \text{(since } \langle\rho, \eta\rangle \in \Gamma(C, D, f)) \\
\eta(f(\langle\vee_C Y_1, \ldots, \vee_C Y_n\rangle)) & = & \\
\eta(f(\vee_{C^n}Y)) & = & \text{(since } Y \text{ is a chain in } C^n\text{, by continuity of } f) \\
\eta(\vee_D f(Y)) & = & \text{(by (i) in Section 2)} \\
\eta(\vee_D \eta(f(Y))) & = & \text{(by definition of the } \textit{lub } \vee_{\eta(D)}\text{, cf. Section 2)} \\
\vee_{\eta(D)}\eta(f(Y)) & = & \\
\vee_{\eta(D)}f^{b_{\rho,\eta}}(Y), & &
\end{array}$$

and therefore $f^{b_{\rho,\eta}}$ is continuous on $\rho(C)^n$. □

Let us see on the rule of signs example how Theorem 5.22 and Lemma 5.23 allows to remarkably simplify the task of computing complete cores and shells. Theorem 5.22 will be exploited in more complex examples in Section 6.

Example 5.24. In Example 5.12, we have computed the absolute complete shell and core of $\rho_{10} = \{\mathbb{Z}, -0, \varnothing\}$ for the concrete interpretation sq: $\wp(\mathbb{Z}) \to \wp(\mathbb{Z})$. We know that *Sign* is complete for sq, and therefore, since ρ_{10} is an abstraction of *Sign*, by Theorem 5.22, we can compute the absolute complete shell and core of ρ_{10} for sq^b: *Sign* $\to$ *Sign*, where sq^b is the best correct approximation of sq on *Sign*—hence, $sq^b \overset{\text{def}}{=} \{\mathbb{Z} \mapsto 0+, 0+ \mapsto 0+, -0 \mapsto 0+, 0 \mapsto 0, \varnothing \mapsto \varnothing\}$. Even more, sq^b is trivially continuous on *Sign*—this is also a consequence of Lemma 5.23—and thus, we can apply the constructive methodologies of Theorem 5.10. Let us remark that we move from an infinite to a finite setting, and therefore the calculations below would be fully automatizable.

As an example, let us check that, coherently with Example 5.12, the absolute complete shell of ρ_{10} turns out to be $\{\mathbb{Z}, -0, 0, \varnothing\}$.

We have that $\mathcal{R}^{\rho_{10}}_{sq^b}$: $uco(Sign) \to uco(Sign)$, and for any $\varphi \in uco(Sign)$, $\mathcal{R}^{\rho_{10}}_{sq^b}(\varphi) = \rho_{10} \sqcap R_{sq^b}(\varphi) = \mathcal{M}_{Sign}(\{\mathbb{Z}, -0, \varnothing\} \cup (\cup_{y \in \varphi}\max(\{x \in Sign \mid sq^b(x) \leq_{Sign} y\})))$. The greatest fixpoint of $\mathcal{R}^{\rho_{10}}_{sq^b}$ is obtained as limit of the lower Kleene's iteration sequence $\top_{uco(Sign)}$, $\mathcal{R}^{\rho_{10}}_{sq^b}(\top_{uco(Sign)}), \ldots$

—$\top_{uco(Sign)} = \{\mathbb{Z}\}$;

—$\mathcal{R}^{\rho_{10}}_{sq^b}(\{\mathbb{Z}\}) = \mathcal{M}_{Sign}(\rho_{10} \cup \max(\{x \in Sign \mid sq^b(x) \leq_{Sign} \mathbb{Z}\}))$
$= \mathcal{M}_{Sign}(\rho_{10} \cup \{\mathbb{Z}\}) = \rho_{10}$;

—$\mathcal{R}^{\rho_{10}}_{sq^b}(\rho_{10}) = \mathcal{M}_{Sign}(\rho_{10} \cup (\cup_{y \in \rho_{10}}\max(\{x \in Sign \mid sq^b(x) \leq_{Sign} y\})))$
(because $\max(\{x \in Sign \mid sq^b(x) \leq_{Sign} -0\}) = \{0\}$)
$= \mathcal{M}_{Sign}(\rho_{10} \cup \{\mathbb{Z}, 0, \varnothing\}) = \{\mathbb{Z}, -0, 0, \varnothing\}$;

—$\mathcal{R}^{\rho_{10}}_{sq^b}(\{\mathbb{Z}, -0, 0, \varnothing\}) = \{\mathbb{Z}, -0, 0, \varnothing\}$
(because $\max(\{x \in \mathit{Sign} \mid sq^b(x) \leq_{\mathit{Sign}} 0\}) = \{0\}$).

Thus, $\{\mathbb{Z}, -0, 0, \varnothing\}$ is the absolute complete shell of ρ_{10} for both sq and sq^b.

It is worthwhile to point out that the converses of the statements of Theorem 5.22, in general, do not hold. The following is a counterexample for fixpoint complete shells, that is, for Theorem 5.22 (3): It shows that it may exist the fixpoint complete shell of ρ for the interpretation $\langle \varphi(C), \varphi(C), H^{b_{\varphi,\varphi}} \rangle$, while the fixpoint complete shell of ρ for the interpretation $\langle C, C, H \rangle$ does not exist.

Example 5.25. In Example 5.16, we have shown that the fixpoint complete shell of $\rho_5 = \{\mathbb{Z}, -0\}$ for sq does not exist. Consider now the closure $\rho_9 = \{\mathbb{Z}, -0, 0\}$. Clearly, $\rho_9 \sqsubseteq \rho_5$ and $\rho_9 \in \Delta(\wp(\mathbb{Z}), sq)$ (see Example 3.4). However, observe from the Hasse diagram in Figure 2 that the fixpoint complete shell of ρ_5 for the interpretation $\langle \rho_9, \rho_9, sq^{b_{\rho_9,\rho_9}} \rangle$ does exist, and it is precisely ρ_9: In fact, ρ_9 is the unique concretization of ρ_5 in $\uparrow \rho_9$.

6. *Applications*

In this section, we present some applications of the general techniques developed above. It is first shown in Section 6.1 how to reconstruct the Cousot and Cousot [1976; 1977] abstract domain of integer intervals as absolute complete shell of *Sign* for the operation of addition on sets of integers. In Section 6.2, it is argued how absolute complete shells can be exploited for tuning the efficiency/precision trade-off when systematically improving the precision of an abstract interpretation-based analysis by abstract domain refinements. A case-study is presented in Section 6.3, where an intelligent efficiency-oriented disjunctive completion refinement is applied to ground-dependency analysis of logic programs.

6.1. RECONSTRUCTING THE INTEGER INTERVAL DOMAIN. The lattice of integer intervals *Int* was first used in static program analysis by Cousot and Cousot [1976; 1977] as an abstract domain for data-flow analysis of integer variables of (imperative) programs. *Int* is defined in the most natural way as follows:

$$\mathit{Int} \overset{\text{def}}{=} \{[a, b] \mid a, b \in \mathbb{Z}, a \leq b\} \\ \cup \{[-\infty, b] \mid b \in \mathbb{Z}\} \cup \{[a, +\infty] \mid a \in \mathbb{Z}\} \cup \{\mathbb{Z}, \varnothing\}.$$

Objects of *Int* having as extremities $-\infty$ or $+\infty$ represent infinite intervals. *Int* is therefore a subset of $\wp(\mathbb{Z})$, and, since it is closed under arbitrary set-intersections, it turns out that $\langle \mathit{Int}, \subseteq \rangle$ is an abstraction of $\langle \wp(\mathbb{Z}), \subseteq \rangle$. Notice that the *Pub* $\bigvee_{\mathit{Int}}$ of *Int* does not coincide with set-union. We observed in Example 3.4 that the abstract domain *Sign* (of Figure 1) is not complete for the addition $\oplus$: $\wp(\mathbb{Z})^2 \to \wp(\mathbb{Z})$ on sets of integers (also recalled in Example 3.4). Since $\oplus$ is evidently additive, by Theorem 5.10, the absolute complete shell of *Sign* for $\oplus$ does exist. In the following, we will show that *Int* is such absolute complete shell.

The following lemma proves that *Int* actually is complete for integer addition.

LEMMA 6.1. $\langle \mathit{Int}, \mathit{Int} \rangle \in \Gamma(\wp(\mathbb{Z}), \wp(\mathbb{Z}), \oplus)$.

PROOF. By Theorem 4.3, and since $\oplus$ is additive (and commutative), $\langle \mathit{Int}, \mathit{Int} \rangle \in \Gamma(\wp(\mathbb{Z}), \wp(\mathbb{Z}), \oplus)$ if and only if

$$\cup_{Y \in Int, Z \in \wp(\mathbb{Z})}(\{\cup\{X \in \wp(\mathbb{Z}) \mid Z \oplus X \subseteq Y\}\}) \subseteq Int.$$

Then, given $Y \in Int$ and $Z \in \wp(\mathbb{Z})$, let us consider the following equalities:

$$\begin{array}{rcl}
\cup\{X \in \wp(\mathbb{Z}) \mid Z \oplus X \subseteq Y\} & = & \text{(by additivity of } \oplus\text{)} \\
\cup\{X \in \wp(\mathbb{Z}) \mid \cup_{x \in X}(Z \oplus \{x\}) \subseteq Y\} & = & \\
\{x \in \mathbb{Z} \mid Z \oplus \{x\} \subseteq Y\} & = & \text{(by additivity of } \oplus\text{)} \\
\{x \in \mathbb{Z} \mid \cup_{z \in Z}(\{z\} \oplus \{x\}) \subseteq Y\} & = & \\
\cap_{z \in Z}\{x \in \mathbb{Z} \mid \{z\} \oplus \{x\} \subseteq Y\}. & &
\end{array}$$

It is easy to check by cases on Y that, for any $z \in \mathbb{Z}$, each set $\{x \in \mathbb{Z} \mid \{z\} \oplus \{x\} \subseteq Y\}$ actually is an interval: For instance, $\{x \in \mathbb{Z} \mid \{z\} \oplus \{x\} \subseteq [a, b]\} = [a - z, b - z]$. Hence, since Int is closed by arbitrary set-intersections, we get that $\cap_{z \in Z}\{x \in \mathbb{Z} \mid \{z\} \oplus \{x\} \subseteq Y\} \in Int$, and therefore this closes the proof. □

We go beyond the above expected result, and prove that *Int* is indeed the absolute complete shell of *Sign*, that is, the most abstract domain which extends *Sign* and is complete for $\oplus$.

THEOREM 6.2. *Int is the absolute complete shell of Sign for* $\oplus$.

PROOF. By Lemma 6.1, $\langle Int, Int\rangle \in \Gamma(\wp(\mathbb{Z}), \wp(\mathbb{Z}), \oplus)$, and therefore, by Theorem 5.22 (2), it suffices to prove that *Int* is the absolute complete shell of *Sign* with respect to the concrete interpretation $\langle Int, Int, \oplus^{Int}\rangle$.

By Theorem 5.10, the absolute complete shell of *Sign* with respect to $\langle Int, Int, \oplus^{Int}\rangle$ is $gfp(\lambda\varphi.Sign \sqcap R_{\oplus^{Int}}(\varphi))$, where $\lambda\varphi.Sign \sqcap R_{\oplus^{Int}}(\varphi)$: $uco(Int) \to uco(Int)$. Recall that since $\oplus^{Int}$ is additive, max's simplify to set-unions. Let us first show that

$$\{[-\infty, a] \mid a \in \mathbb{Z}\} \cup \{[a, +\infty] \mid a \in \mathbb{Z}\} \subseteq \bigcup_{x \in Sign, J \in Int}(\vee_{Int}\{\mathrm{I} \in Int \mid J \oplus^{Int} I \subseteq x\}).$$

Let $a \in \mathbb{Z}$, and let us consider the case of $[a, +\infty] \in Int$. Note that $[a, +\infty]$ is the greatest integer interval which added to $[-a, +\infty]$ gives $[0, +\infty]$, that is,

$$\vee_{Int}\{I \in Int \mid [-a, +\infty] \oplus^{Int} I \subseteq 0+\} = [a, +\infty],$$

and therefore $[a, +\infty] \in R_{\oplus^{Int}}(Sign)$. Analogously, we have that

$$\vee_{Int}\{\mathrm{I} \in Int \mid [-\infty, -a] \oplus^{Int} I \subseteq -0\} = [-\infty, a],$$

and therefore $[-\infty, a] \in R_{\oplus^{Int}}(Sign)$. Since

$$R_{\oplus^{Int}}(Sign) = \mathcal{M}_{Int}(\cup_{x \in Sign, J \in Int}(\vee_{Int}\{\mathrm{I} \in Int \mid J \oplus^{Int} I \subseteq x\})),$$

we therefore get that for all $a, b \in \mathbb{Z}$, with $a \leq b$, $[-\infty, b] \cap [a, +\infty] = [a, b] \in R_{\oplus^{Int}}(Sign)$. Hence, since trivially $\varnothing, \mathbb{Z} \in R_{\oplus^{Int}}(Sign)$, this proves that $Int \subseteq R_{\oplus^{Int}}(Sign)$, that is, $R_{\oplus^{Int}}(Sign) \sqsubseteq Int$. Hence, $gfp(\lambda\varphi.Sign \sqcap R_{\oplus^{Int}}(\varphi)) = Int$, as desired. □

We think that a remarkable point of the previous result is given by the fact that it shows how the lattice *Int* of integer intervals can be fully reconstructed by a natural abstract domain refinement from *Sign.* Thus, in a sense, *Int* is an example of an abstract domain which does not need to be designed from scratch, but instead can be systematically obtained from the obvious *Sign* by resorting to completeness.

We can also exploit the above result in order to modularly compute the absolute complete shell of the abstract domain *Sign* $\sqcap$ *Parity* for integer addition.

Example 6.3. Consider the abstract domain *Parity* defined in Example 4.11, and the reduced product *Sign* $\sqcap$ *Parity* depicted in Figure 3. By Lemma 5.13, the absolute complete shell of *Sign* $\sqcap$ *Parity* is the reduced product of the absolute complete shells of *Sign* and *Parity.* It is straightforward to check that *Parity* is already complete for $\oplus$. Thus, by Theorem 6.2, *Int* $\sqcap$ *Parity* is such absolute complete shell.

6.2. SMART REFINEMENT OF ABSTRACT DOMAINS. A general notion of abstract domain refinement has been studied by Filé et al. [1996], and subsequently extended by Giacobazzi and Ranzato [1997; 1998b], as a formalization and generalization for most of the operators devoted to enhance systematically the precision of abstract domains, like the well-known reduced product and disjunctive completion [Cousot and Cousot 1979]. An abstract domain refinement is an operator on the lattice of abstract interpretations that monotonically enhance the precision of abstract domains. Hence, for a fixed concrete domain C, a (unary) abstract domain refinement is a monotone and decreasing (with respect to $\sqsubseteq$) mapping $\Re: \mathfrak{L}_C \to \mathfrak{L}_C$, where decreasingness encodes that $\Re(A)$ is more precise than A. Moreover, most of the times, refinements are idempotent, that is, they upgrade domains all at once, and therefore this last condition evidently defines idempotent refinements as lower closure operators on $\mathfrak{L}_C$. Abstract domain refinements have been successfully applied for enhancing the precision of abstract interpretation-based static analyses. On the other hand, it is well known, also thanks to the literature on experimental implementation results, that both complexity and efficiency of abstract interpretation-based static analyses depend intrinsically on the underlying abstract domains. A typical complexity measure is in fact the maximal length of chains in an abstract domain (e.g., see Deutsch [1997]), because this yields an absolute upper bound on the number of iterations needed for calculating (least or greatest) fixpoints. A systematic step of abstract domain refinement might considerably increase the global cost of an analysis, perhaps so much that it could become unacceptable. Even worse, a refined abstract interpretation might become undecidable. For example, this could be the case for the disjunctive completion refinement. This refinement operator enhances an abstract domain by adding the necessary structure to model in a precise way the concrete disjunctive information, like the alternative computation paths in, for instance, the branches of a conditional choice. It turns out that, in general, such refinement does not preserve the ascending chain condition (ACC) property of abstract domains.

In the following, we illustrate how absolute and relative complete shells may play a helpful role in order to achieve a right balance between efficiency and precision when systematically refining abstract domains. Let us first generalize and formalize the scenario sketched above. The situation is close to that considered in Section 5.5. Let $\mathfrak{S} = \langle C, D, f \rangle$ be a concrete semantics, where f: $C^n \overset{\mathrm{m}}{\rightarrow} D$, and consider the abstractions $A \in \mathfrak{L}_C$ and $B \in \mathfrak{L}_D$, which induce the sound abstract semantics $\mathfrak{S}^{A,B} = \langle A, B, f^{b_{A,B}} \rangle$. Thus, if A' and B' are further abstractions, respectively, of C and D such that $A \sqsubseteq A'$ and $B \sqsubseteq B'$, then one can reason on the completeness relationships of $\langle A', B' \rangle$ with respect to their relative concrete semantics $\mathfrak{S}^{A,B}$—that is, by Lemma 5.20, A' and B' are thought of, respectively, as abstractions of A and B. Then, we simply say that $\langle A', B' \rangle$ is complete for (or with respect to) $\langle A, B \rangle$ to mean that $\langle A', B' \rangle \in \Gamma(A, B, f^{b_{A,B}})$. In particular, such a situation arises whenever $\mathfrak{R}_1$: $\mathfrak{L}_C \rightarrow \mathfrak{L}_C$ and $\mathfrak{R}_2$: $\mathfrak{L}_D \rightarrow \mathfrak{L}_D$ are domain refinements, and hence $C \sqsubseteq \mathfrak{R}_1(A) \sqsubseteq A$ and $D \sqsubseteq \mathfrak{R}_2(B) \sqsubseteq B$ hold. In such cases, we take into consideration the possible completeness relations of $\langle A, B \rangle$ with respect to the relative concrete semantics $\langle \mathfrak{R}_1(A), \mathfrak{R}_2(B), f^{b_{\mathfrak{R}_1(A),\mathfrak{R}_2(B)}} \rangle$. Roughly speaking, in general, an abstract domain refinement $\mathfrak{R}$ is intended to transform "analyses into analyses," that is, at least decidability should be preserved by $\mathfrak{R}$; thus, A and $\mathfrak{R}(A)$ should be "quite close," and therefore completeness should not be an exceptional event.

In what follows, we consider the case of absolute complete shells, since this will be illustrated by a case-study in Section 6.3; similar considerations apply for relative complete shells. Thus, let f: $C^n \rightarrow C$, $A \in \mathfrak{L}_C$ and $\mathfrak{R}$: $\mathfrak{L}_C \rightarrow \mathfrak{L}_C$. Let us assume that the absolute complete shell of A with respect to $\mathfrak{R}(A)$ (in precise terms, with respect to $\langle \mathfrak{R}(A), \mathfrak{R}(A), f^{b_{\mathfrak{R}(A),\mathfrak{R}(A)}} \rangle$) exists (e.g., by Theorem 5.10), and let us denote it by $\mathcal{S}_{\mathfrak{R}(A)}(A)$. Hence, in general, we have that $\mathfrak{R}(A) \sqsubseteq \mathcal{S}_{\mathfrak{R}(A)}(A) \sqsubseteq A$. We say that $\mathfrak{R}(A)$ is *too refined* whenever $\mathcal{S}_{\mathfrak{R}(A)}(A)$ is a proper abstraction of $\mathfrak{R}(A)$, that is, $\mathfrak{R}(A) \sqsubset \mathcal{S}_{\mathfrak{R}(A)}(A)$. The intuition behind this definition is as follows. If $\mathcal{I}$ and $\mathcal{I}'$ are two abstract interpretations of a common concrete semantics, and the corresponding underlying abstract domains are A and A' such that $A' \sqsubseteq A$, then whenever it happens that A is complete with respect to A', $\mathcal{I}$ may rightly be understood of as just thinly less precise than $\mathcal{I}'$. In such a case, a reasonable efficiency/precision trade-off would then suggest to prefer $\mathcal{I}$ rather than $\mathcal{I}'$, especially when $\mathcal{I}'$ is much less efficient than $\mathcal{I}$. In other terms, since, thanks to completeness, the gap of precision between $\mathcal{I}$ and $\mathcal{I}'$ is narrow, in general, one should be inclined to choose $\mathcal{I}$ rather than the more costly $\mathcal{I}'$. Thus, in the above definition, whenever $\mathcal{S}_{\mathfrak{R}(A)}(A) \neq \mathfrak{R}(A)$, one should prefer the thinly less refined domain $\mathcal{S}_{\mathfrak{R}(A)}(A)$ rather than the fully refined domain $\mathfrak{R}(A)$; in particular, if A is already complete with respect to $\mathfrak{R}(A)$ (i.e., $\mathcal{S}_{\mathfrak{R}(A)}(A) = A$), then A actually does not need to be refined by $\mathfrak{R}$. This explains why in these cases $\mathfrak{R}(A)$ is termed "too refined". An "efficiency-oriented" version $\mathfrak{R}^*$ of the refinement $\mathfrak{R}$ can be therefore defined as follows: $\mathfrak{R}^* \overset{\mathrm{def}}{=} \lambda A.\mathcal{S}_{\mathfrak{R}(A)}(A)$. This strategy may lead to significant savings. For instance, in the aforementioned 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, sometimes essential, property, while, in contrast, our strategy provides a smart disjunctive completion $\mathbb{P}^*(A)$ that still satisfies the ACC.

It is worthwhile to mention that very similar applications of relative complete shells aiming at improving on the complexity of static analyses have been already explored. We showed in Giacobazzi et al. [1998, Section 5] that Cortesi et al.'s [1998] technique for "quotienting" abstract interpretations is indeed nothing else than an instance of our relative complete shells (actually, in Giacobazzi et al.

[1998, Section 5] it is shown how our completeness-based viewpoint allows us to subsume and generalize Cortesi et al.'s [1998] approach and results). Then, Bagnara et al.'s [1997] and King et al.'s [1999] works, where quotienting is applied for lowering the complexity of some abstract unification operations for logic program analysis, can be seen as applications of our relative complete shells following the same basic idea detailed above: Whenever an analysis $\mathcal{A}$ results to be complete with respect to a more precise analysis $\mathcal{A}'$ and $\mathcal{A}$ is appreciably more efficient than $\mathcal{A}'$, then completeness suggests to trade the slightly greater precision of $\mathcal{A}'$ for the efficiency of $\mathcal{A}$. We will discuss Bagnara et al.'s [1997] and King et al.'s [1999] works more in detail in Section 7.

6.3. Smart Disjunctive Refinement of Ground-Dependency Analysis. Groundness analysis is arguably one of the most important analysis for logic-based programming languages. Groundness analysis aims to statically detect whether logical variables or predicate arguments will be bound to ground terms in run-time evaluations. This allows optimizing compilers to speed up unification [Hermenegildo et al. 1992; Van Roy and Despain 1990]—the prime computational engine of any logic language—and is of support to other analyses, such as independence or occur-check analysis. Ground-dependency analysis is a generalization taking into account groundness dependencies between logical variables or predicate arguments, that is, properties such as "if a program variable x is (becomes) ground, so is (does) program variable y". *Def* and *Pos* are two well-known, widely used and experimentally tested abstract domains for ground-dependency analysis [Armstrong et al. 1998; Cortesi et al. 1995; 1996; Marriott and Søndergaard 1993]. We deal here with the plain case of positive logic programs, but it is worth mentioning that minor variants of *Def* and *Pos* have also been used for analysing constraint and concurrent logic languages (see e.g., Codish et al. [1994]). The standard *s-semantics* by Falaschi et al. [1989], which is fully abstract for the observable given by computed answer substitutions, will be our concrete least fixpoint semantics. This is a standard choice for the so-called bottom-up approach to abstract interpretation-based logic program analysis [Barbuti et al. 1993; Codish et al. 1994; Marriott et al. 1994].

6.3.1. *The Least Fixpoint Semantics.* Let us succinctly recall the salient ingredients of s-semantics for logic programs. 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 $\#(p) \in \mathbb{N}$ denotes the arity of p. The set of atoms, that is, atomic logic formulae, and (positive) logic programs, that is, finite sets of Horn clauses, 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. An atom A is ground if $var(A) = \varnothing$.

A substitution is a mapping from *Var* to terms built from L, which acts as the identity almost everywhere. A substitution σ will be denoted by its finite set of nontrivial bindings, that is, $\sigma = \{x/\sigma(x) \mid \sigma(x) \neq x\}$. The empty substitution—viz. the identity mapping—is denoted by ϵ. If σ is a substitution and o is any syntactic object, then, as usual, $o\sigma$ stands for the result of applying σ to o. The standard composition of substitutions θ and σ is denoted by $\theta\sigma$, and is defined by $\theta\sigma \overset{\text{def}}{=} \lambda x.(x\theta)\sigma$. A substitution σ is idempotent if $\sigma\sigma = \sigma$. The set of idempotent substitutions (built from the underlying language L) is denoted by

Sub. If o and o' are two syntactic objects, then o' is an instance of o—or o is more general than o'—denoted by $o' \trianglelefteq o$, iff there exists a substitution σ such that $o' = o\sigma$. Also, o and o' are equivalent up to renaming, denoted by $o \simeq o'$, iff $o \trianglelefteq o'$ and $o' \trianglelefteq o$. Of course, the relation $\simeq$ is an equivalence, and therefore the lifting of the relation $\trianglelefteq$ to the quotient $Atom/\simeq$ gives rise to a partial order. With abuse of notation, such a poset will be still denoted by $Atom$, and any atom will denote its equivalence class by renaming. For a syntactic object o and a set of (equivalence classes up to renaming of) atoms $I \subseteq Atom$, the notation $\langle a_1, \ldots, a_n\rangle \ll_o I$, with $n \geq 0$, will denote that $a_1, \ldots, a_n$ are (representatives of) elements of I renamed apart from o and from each other (i.e., $\forall i, j \in [1, n].var(a_i) \cap var(o) = \varnothing$ and $(i \neq j \Rightarrow var(a_i) \cap var(a_j) = \varnothing)$). The concepts of unification and most general unifier are well known. Let us fix the following partial function mgu: Given two tuples of atoms $\bar{a}$ and $\bar{b}$, $mgu(\bar{a}, \bar{b}) = \theta$ means that $\bar{a}$ and $\bar{b}$ are unifiable and θ is an idempotent most general unifier of $\bar{a}$ and $\bar{b}$—it is well known that they are all equivalent up to renaming, cf. Lassez et al. [1988].

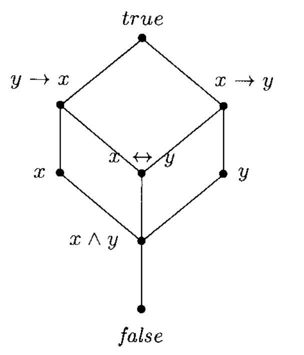
The concrete domain of the s-semantics is the set of all the so-called s-interpretations, ordered by set inclusion, that is, the complete lattice $\langle \wp(Atom), \subseteq\rangle$. The well-known immediate consequences operator T^s: $Program \rightarrow (\wp(Atom) \rightarrow (Atom))$ is defined as follows: For any $P \in Program$ and s-interpretation $I \in \wp(Atom)$,

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

It turns out that, for any program P, T^s_P is continuous, and therefore the least fixpoint s-semantics of P is defined as $lfp(T^s_P) = \cup_{n\in\mathbb{N}}(T^s_P)^n(\varnothing)$. In the following, $\mathfrak{S} \stackrel{\text{def}}{=} \langle \wp(Atom), \{T^s_P\}_{P\in Program}\rangle$ will play the role of concrete semantics.

6.3.2. *The Abstract Domains Def and Pos.* Let us briefly recall the definitions of the abstract domains *Def* and *Pos.* For more details we refer to Armstrong et al. [1998], and Marriott and Søndergaard [1993]. Both domains can be characterized as suitable sets of propositional Boolean formula, built from a (nonempty) finite set of propositional variables $VI \subseteq Var$, called variables of interest. Boolean formulas are ordered by standard logical consequence relation $\leqslant$ ($<$ denotes strict ordering). The standard logical connectives of conjunction, disjunction and implication will be denoted, respectively, by $\wedge$, $\vee$ and $\rightarrow$, while *true* and *false* will denote, respectively, the greatest and least Boolean formula. In the following, we will slightly abuse the notation by letting a synctatic propositional formula to denote its equivalence class with respect to logical equivalence. In the following, truth assignments on VI will be understood as subsets of VI, where, as usual, the standard intended meaning is that $M \subseteq VI$ contains all and only the variables mapped to true.

Pos is the set of *positive* Boolean formulas on VI, where a formula is positive if it gives *true* for the truth assignment where each variable is set to *true*—that is, f is positive if $VI \models f$. *Def* is the set of *definite* Boolean formulas on VI, where a formula f is definite if f is positive and its set of models is closed under



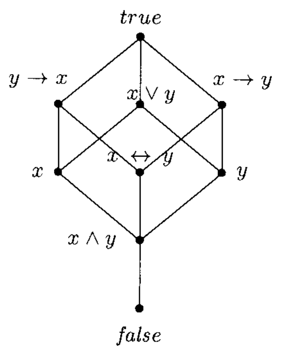


FIG. 4. *Def*$_2$ and *Pos*$_2$ for $VI = \{x, y\}$.

intersection (i.e., if $M, N \models f$ then $M \cap N \models f$). Hence, *Def* is a subset of *Pos*. Positive and definite formulas can be syntactically characterized as follows: A formula f is positive iff f is logically equivalent to a formula built using only the connectives $\wedge$ and $\rightarrow$; a formula f is definite iff f is logically equivalent to a formula which is a conjunction of clauses, that is, formulas having the shape $x \leftarrow (x_1 \wedge \ldots \wedge x_n)$, where n may be 0.

Note that the least Boolean formula *false* belongs neither to *Pos* nor to *Def*. However, since *false* is useful in order to precisely represent the empty set of substitutions (and therefore to represent the information of reachability), following a standard practice, *false* is added to both *Def* and *Pos*, and *Def* and *Pos* are still used to denote these sets of formulae. $\langle Pos, \preccurlyeq\rangle$ and $\langle Def, \preccurlyeq\rangle$ are (finite) lattices. The *glb*'s of *Pos* and *Def* are both given by logical conjunction $\wedge$, while only the *lub* of *Pos* is given by logical disjunction $\vee$; the *lub* of *Def* can be defined in the usual way in terms of logical conjunction: $f_1 \vee_{Def} f_2 = \wedge\{f \in Def \mid f_i \preccurlyeq f\ (i = 1, 2)\}$. Thus, *Def* is a Moore-family of *Pos*, and hence it is an abstraction of *Pos*. If $|VI| = n$ then we will often use the notations Pos_n and Def_n to emphasize the size of the underlying set of variables of interest. The lattices *Def*$_2$ and *Pos*$_2$ are depicted in Figure 4 for the case of two variables of interest $VI = \{x, y\}$.

Def and *Pos* are canonically viewed as abstractions of the concrete domain given by the powerset of all the idempotent substitutions $\langle \wp(Sub), \subseteq\rangle$. This is a standard nonrestrictive practice, because standard unification algorithms produce idempotent substitutions as output. The abstraction and concretization maps between *Pos*, *Def* and $\wp(Sub)$ are well known, and can be found, for example, in Armstrong et al. [1998] and Marriott and Søndergaard [1993]. The intuition is that a conjunction $x \wedge y$ abstractly represents the set of all the (idempotent) substitutions which ground both the variables x and y, and an implication $x \rightarrow y$ represents the set of all the substitutions σ such that if the term $\sigma(x)$ is ground then so is $\sigma(y)$. For example, assuming $VI = \{x, y, z, u\}$, $x \wedge (y \leftrightarrow z)$ 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,[7] $\sigma_1 = \{x/a, y/b, z/c\}$ and $\sigma_2 = \{x/a, y/w, z/w, v/u\}$ satisfy this property. Thus, if $(\alpha_{Def}, \wp(Sub), Def, \gamma_{Def})$ and

[7] $a, b, c, \ldots$ denote ground terms.

$(\alpha_{Pos}, \wp(Sub), Pos, \gamma_{Pos})$ are the corresponding GIs, $\{\sigma_1, \sigma_2\} \subseteq \gamma_{Pos}(x \wedge (y \leftrightarrow z)) = \gamma_{Def}(x \wedge (y \leftrightarrow z))$.

6.3.3. *The Disjunctive Completion Refinement.* Let us recall how the disjunctive completion refinement is defined (for more details, see Cousot and Cousot [1979], Filé and Ranzato [1999] and Giacobazzi and Ranzato [1998a]). Here, for our purposes, it is enough to 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), the following equivalence relation on $\wp(A)$ is defined: For all $S, T \subseteq A$, $S \overset{\vee}{\simeq} T \Leftrightarrow \vee_C \gamma(S) = \vee_C \gamma(T)$. The disjunctive completion $\mathbb{P}(A)$ of A is defined as the quotient of $\wp(A)$ for the equivalence relation $\overset{\vee}{\simeq}$. $\mathbb{P}(A)$ is a complete lattice for 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 [1999]). $\mathbb{P}(A)$ is related to the concrete domain C by the following natural GI $(\alpha^*, C, \mathbb{P}(A), \gamma^*)$: $\gamma^* \overset{\text{def}}{=} \lambda[S].\vee_C \gamma(S)$, while α^* is defined as the usual left-adjoint of γ^*. The *glb* and the *lub* of $\mathbb{P}(A)$ are defined as follows [Filé and Ranzato 1999, Proposition 3.6]: Given $[S], [T] \in \mathbb{P}(A)$, $[S] \wedge_{\mathbb{P}(A)} [T] = [\{s \wedge_A t \mid s \in S, t \in T\}]$ and $[S] \vee_{\mathbb{P}(A)} [T] = [S \cup T]$. It turns out that the disjunctive completion operator $\mathbb{P}$ actually is an idempotent refinement. When $\langle C, \leq_C \rangle = \langle \wp(X), \subseteq \rangle$ for some set X, it turns out that, for any $Z \in \wp(X)$, $\alpha^*(Z) = [\{\alpha(\{z\}) \mid z \in Z\}]$ (cf. [Filé and Ranzato 1999, 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 [1999] investigated the disjunctive completion of *Pos* as a suitable domain for disjunctive ground-dependency analysis and they proved that $\mathbb{P}(Pos) \sqsubset Pos$, namely there is a strict improvement in precision by lifting *Pos* to its disjunctive completion $\mathbb{P}(Pos)$ [Filé and Ranzato 1999, Theorem 5.3]. For example, by considering as variables of interest $VI = \{x, y\}$, we have that the logical disjunction $(x \rightarrow y) \vee (y \rightarrow x)$ in *Pos* does not represent the concrete disjunction of the two formulae $x \rightarrow y$ and $y \rightarrow x$, that is, the union of their concretizations. In fact, it holds $\gamma_{Pos}(x \rightarrow y) \cup \gamma_{Pos}(y \rightarrow x) \subset \gamma_{Pos}((x \rightarrow y) \vee (y \rightarrow x)) = \gamma_{Pos}(true)$. For instance, the empty substitution ϵ trivially belongs to $\gamma_{Pos}(true) = Sub$, whilst it is clear that ϵ belongs neither to $\gamma_{Pos}(x \rightarrow y)$ nor to $\cup \gamma_{Pos}(y \rightarrow x)$. Moreover, Giacobazzi and Ranzato [1998a, Theorem 7.2] showed that $\mathbb{P}(Def) = \mathbb{P}(Pos)$. Thus, a disjunctive ground-dependency analysis can be implemented using the disjunctive completion of *Def*, without losing precision and at a lower cost with respect to the disjunctive completion of *Pos*. The disjunctive completion of Def_2 (and Pos_2) is depicted in Figure 5.

6.3.4. *Abstracting the s-Semantics.* Let us see 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 $\mathfrak{S}$. We consider the case of *Def*, since the other two are completely analogous. Given $n \in \mathbb{N}$, by Def_n we denote the set of definite formulae whose underlying set of variables of interest is $\{x_1, \ldots, x_n\}$, and given a predicate $p \in \Pi$ of arity n, we implicitly refer to x_i as the ith argument of an atom whose predicate symbol is p. Note that $Def_0 = \{true, false\}$. Def^s is therefore defined as follows:

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

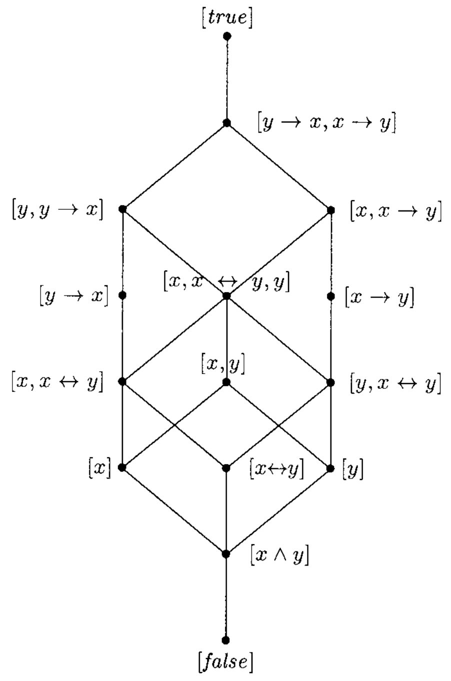


FIG. 5. The disjunctive completion $\mathbb{P}(Def_2) = \mathbb{P}(Pos_2)$ for $VI = \{x, y\}$.

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 about p. Def^s inherits componentwise the ordering from *Def*: For any F, $G \in Def^s$, $F \leq G$ iff $\forall (p, f_p) \in F$, $(p, g_p) \in G$. $f_p \preccurlyeq g_p$. Obviously, Def^s is a complete lattice satisfying the ACC (when considering a finite alphabet Π, Def^s is even finite), where *lub* and *glb* are defined componentwise from those of *Def*.

It is then straightforward to design the GI $(\alpha_{Def^s}, \wp(Atom), Def^s, \gamma_{Def^s})$: For any $\langle (p, f_p) \rangle_{p \in \Pi} \in Def^s$, define

$$\gamma_{Def^s}(\langle (p, f_p) \rangle_{p\in\Pi}) \stackrel{\text{def}}{=} \{p(t_1, \ldots, t_n) \in Atom \mid p \in \Pi, \\ \{x_1/t_1, \ldots, x_n/t_n\} \in \gamma_{Def_n}(f_p)\}$$

where it is assumed that any atom $p(t_1, \ldots, t_n)$ is renamed apart from $x_1, \ldots, x_n$. Thus, for any s-interpretation $I \in \wp(Atom)$, its abstraction in Def^s is defined as follows:

$$\alpha_{Def^s}(I) \stackrel{\text{def}}{=} \langle (p, \vee_{Def_n}\{\alpha_{Def_n}(\{x_1/t_1, \ldots, x_n/t_n\}) \mid p(t_1, \ldots, t_n) \in I\}) \rangle_{p\in\Pi},$$

Note that if I contains a constant predicate symbol p of arity zero then the corresponding component in $\alpha_{Def^s}(I)$ is always $(p, true)$, and if some predicate p is missing in I then the corresponding component in $\alpha_{Def^s}(I)$ is given by $(p, false)$, since $false = \vee_{Def_{\#(p)}}\varnothing$. The fact that $(\alpha_{Def^s}, \wp(Atom), Def^s, \gamma_{Def^s})$ is a GI is a straightforward consequence of the GI of *Def* in $\wp(Sub)$. Analogously, we also get the GIs $(\alpha_{Pos^s}, \wp(Atom), Pos^s, \gamma_{Pos^s})$ and $(\alpha_{\mathbb{P}(Def)^s}, \wp(Atom), \mathbb{P}(Def)^s,$

$\gamma_{\mathbb{P}(Def)^s}$). By a slight abuse of notation, for the sake of conciseness, in the following we omit the superscript when denoting such abstract domains. Obviously, it turns out that $\wp(Atom) \sqsubseteq \mathbb{P}(Def) \sqsubseteq Pos \sqsubseteq Def$. The following example helps in clarifying the notions introduced above.

Example 6.4. Let $\Pi = \{p/2, q/1, r/0, s/3\}$ be the alphabet of predicate symbols. Consider the following 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 check that:

—$\alpha_{Def}(I_1) = \langle (p, x_1 \vee_{Def_2} (x_1 \leftrightarrow x_2) \equiv x_2 \rightarrow x_1), (q, \mathit{true}), (r, \mathit{false}), (s, \mathit{false})\rangle$;
$\alpha_{Def}(I_2) = \langle (p, x_2 \vee_{Def_2} x_1 \equiv \mathit{true}), (q, x_1), (r, \mathit{true}), (s, \mathit{false})\rangle$.
—$\alpha_{Pos}(I_1) = \alpha_{Def}(I_1)$; $\alpha_{Pos}(I_2) = \langle (p, x_1 \vee x_2), (q, x_1), (r, \mathit{true}), (s, \mathit{false})\rangle$.
—$\alpha_{\mathbb{P}(Def)}(I_1) = \langle (p, [x_1, x_1 \leftrightarrow x_2]), (q, [\mathit{true}]), (r, [\mathit{false}]), (s, [\mathit{false}])\rangle$;
$\alpha_{\mathbb{P}(Def)}(I_2) = \langle (p, [x_1, x_2]), (q, [x_1]), (r, [\mathit{true}]), (s, [\mathit{false}])\rangle$.

From $\mathfrak{S}$ we get the following abstract semantics: $\mathfrak{S}^{Def} \overset{\text{def}}{=} \langle Def, \{T_P^{Def}\}_{P\in Program}\rangle$, $\mathfrak{S}^{Pos} \overset{\text{def}}{=} \langle Pos, \{T_P^{Pos}\}_{P\in Program}\rangle$, and $\mathfrak{S}^{\mathbb{P}(Def)} \overset{\text{def}}{=} \langle \mathbb{P}(Def), \{T_P^{\mathbb{P}(Def)}\}_{P\in Program}\rangle$, such that each abstract semantic operator is defined as the corresponding best correct approximation of T_P^s (e.g., for any P, $T_P^{\mathbb{P}(Def)} \overset{\text{def}}{=} \alpha_{\mathbb{P}(Def)} \circ T_P^s \circ \gamma_{\mathbb{P}(Def)}$).

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

Example 6.5. Consider the following program Q.

$$p(x, y) :- q(x, y), r(x, y).$$

$$q(a, x).$$

$$q(x, a).$$

$$r(x, x).$$

Let us compute the least fixpoint *Def*, *Pos* and $\mathbb{P}(Def)$ abstract s-semantics of Q. It is here understood that $\Pi = \{p/2, q/2, r/2\}$.

—$T_Q^{Def}(\bot_{Def}) = \langle (p, \mathit{false}), (q, x_1 \vee_{Def_2} x_2 \equiv \mathit{true}), (r, x_1 \leftrightarrow x_2)\rangle$;
$(T_Q^{Def})^2(\bot_{Def}) = \langle (p, \mathit{true} \wedge (x_1 \leftrightarrow x_2) \equiv x_1 \leftrightarrow x_2), (q, \mathit{true}), (r, x_1 \leftrightarrow x_2)\rangle$ (least fixpoint).
—$T_Q^{Pos}(\bot_{Pos}) = \langle (p, \mathit{false}), (q, x_1 \vee x_2), (r, x_1 \leftrightarrow x_2)\rangle$;
$(T_Q^{Pos})^2(\bot_{Pos}) = \langle (p, (x_1 \vee x_2) \wedge (x_1 \leftrightarrow x_2) \equiv x_1 \wedge x_2), (q, x_1 \vee x_2), (r, x_1 \leftrightarrow x_2)\rangle$ (least fixpoint).
—$T_Q^{\mathbb{P}(Def)}(\bot_{\mathbb{P}(Def)}) = \langle (p, [\mathit{false}]), (q, [x_1] \vee_{\mathbb{P}(Def)} [x_2] = [x_1, x_2]), (r, [x_1 \leftrightarrow x_2])\rangle$;
$(T_Q^{\mathbb{P}(Def)})^2(\bot_{\mathbb{P}(Def)}) = \langle (p, [x_1 \wedge x_2]), (q, [x_1, x_2]), (r, [x_1 \leftrightarrow x_2])\rangle$ (least fixpoint).

Thus, by using *Pos* and $\mathbb{P}(\mathit{Def})$ 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 $\alpha_{\mathbb{P}(\mathit{Def}),\mathit{Def}}(\mathit{lfp}(T_Q^{\mathbb{P}(\mathit{Def})})) = \alpha_{\mathit{Pos},\mathit{Def}}(\mathit{lfp}(T_Q^{\mathit{Pos}})) = \langle (p, x_1 \wedge x_2), (q, \mathit{true}), (r, x_1 \leftrightarrow x_2)\rangle < \mathit{lfp}(T_Q^{\mathit{Def}})$, this shows that

$$\mathit{Def} \notin \Delta(\mathit{Pos}, \{T_P^{\mathit{Pos}}\}_{P\in \mathit{Program}}) \cup \Delta(\mathbb{P}(\mathit{Def}), \{T_P^{\mathbb{P}(\mathit{Def})}\}_{P\in \mathit{Program}})$$

and hence

$$\mathit{Def} \notin \Gamma(\mathit{Pos}, \mathit{Pos}, \{T_P^{\mathit{Pos}}\}_{P\in \mathit{Program}}) \cup \Gamma(\mathbb{P}(\mathit{Def}), \mathbb{P}(\mathit{Def}), \{T_P^{\mathbb{P}(\mathit{Def})}\}_{P\in \mathit{Program}}).$$

On the other hand, Filé and Ranzato [1999, Proposition 5.13] proved that *Pos* is complete with respect to $\mathbb{P}(\mathit{Def})$, namely, translated in our notation, the following result holds.

THEOREM 6.6. [FILÉ AND RANZATO 1999]

$$\langle \mathit{Pos}, \mathit{Pos}\rangle \in \Gamma(\mathbb{P}(\mathit{Def}), \mathbb{P}(\mathit{Def}), \{T_P^{\mathbb{P}(\mathit{Def})}\}_{P\in \mathit{Program}}).$$

As a consequence, *Pos* is also fixpoint complete with respect to $\mathbb{P}(\mathit{Def})$. Let us see an example taken from Filé and Ranzato [1999, Example 5.11].

Example 6.7. Let us consider the following program Q.

$$p(x, a) :- p(x, z), p(y, x).$$

$$p(x, x).$$

$$p(a, y).$$

Here, $\Pi = \{p/2\}$. The upper Kleene's iterations for T_Q^{Pos} and $T_Q^{\mathbb{P}(\mathit{Def})}$ are as follows:

—$T_Q^{\mathit{Pos}}(\langle (p, \mathit{false})\rangle) = \langle (p, x_2 \rightarrow x_1)\rangle$;
$T_Q^{\mathit{Pos}}(\langle (p, x_2 \rightarrow x_1)\rangle) = \langle (p, \mathit{true})\rangle$ (least fixpoint).
—$T_Q^{\mathbb{P}(\mathit{Def})}(\langle (p, [\mathit{false}])\rangle) = \langle (p, [x_1 \leftrightarrow x_2, x_1])\rangle$;
$T_Q^{\mathbb{P}(\mathit{Def})}(\langle (p, [x_1 \leftrightarrow x_2, x_1])\rangle) = \langle (p, [x_1 \leftrightarrow x_2, x_1, x_2])\rangle$ (least fixpoint).

According to Theorem 6.6, we have that, for $i \in \{0, 1, 2\}$, $(T_Q^{\mathit{Pos}})^i(\langle (p, \mathit{false})\rangle) = \vee(T_Q^{\mathbb{P}(\mathit{Def})})^i(\langle (p, [\mathit{false}])\rangle)$. Thus, using $\mathbb{P}(\mathit{Def})$ 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 terms containing 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 $\mathbb{P}(\mathit{Def})$ we exactly get the output (p, true) of *Pos*, coherently with Theorem 6.6.

Since $\mathbb{P}(\mathit{Def})$ satisfies the ACC, and each $T_P^{\mathbb{P}(\mathit{Def})}$ is monotone, and hence continuous, by Theorem 5.10, the absolute complete shell of *Def* with respect to $\langle \mathbb{P}(\mathit{Def}), \{T_P^{\mathbb{P}(\mathit{Def})}\}_{P\in \mathit{Program}}\rangle$ does exist. Then, we sharp Theorem 6.6 by proving that this absolute complete shell actually coincides with *Pos.*

THEOREM 6.8. *Pos is the absolute complete shell of Def with respect to the interpretation* $\langle \mathbb{P}(Def), \{T_P^{\mathbb{P}(Def)}\}_{P\in Program}\rangle$.

PROOF. By Theorem 6.6, *Pos* is complete for $\langle \mathbb{P}(Def), \{T_P^{\mathbb{P}(Def)}\}_{P\in Program}\rangle$. Thus, by Theorem 5.22 (2), it is enough to prove that *Pos* is the absolute complete shell of *Def* with respect to $\langle Pos, \{T_P^{Pos}\}_{P\in Program}\rangle$. Let $F \overset{\text{def}}{=} \{T_P^{Pos}\}_{P\in Program}$. By Theorem 5.10, this absolute complete shell is given by $gfp(\lambda\varphi.Def \sqcap R_F(\varphi))$, where $\lambda\varphi.Def \sqcap R_F(\varphi)$: $uco(Pos) \to uco(Pos)$. Let us unfold the definition of R_F:

$$R_F(\varphi) = \mathcal{M}_{Pos}(\cup_{P\,\in\, Program, J\,\in\,\varphi} \max(\{I \in Pos \mid T_P^{Pos}(I) \leq_{Pos} J\})).$$

The first three iterations in the lower Kleene's iteration sequence of $\lambda\varphi.Def \sqcap R_F(\varphi)$ yield: $\top_{uco(Pos)}$, *Def* and $R_F(Def)$. Hence, in order to conclude, it is sufficient to show that

$$Pos \sqsubseteq \mathcal{M}_{Pos}(\cup_{P\in Program, J\,\in\, Def} \max(\{I \in Pos \mid T_P^{Pos}(I) \leq_{Pos} J\})),$$

because this implies that $R_F(Def) = Pos$.

Let $I = \langle \dots, (p, f_p), \dots\rangle$ be a generic element of *Pos*. As a straight consequence of results that appear in Armstrong et al. [1998, Section 3], let us observe that each positive formula can be represented as logical disjunction of suitable definite formulae. Hence, $f_p = g_1 \vee \cdots \vee g_n$, where $n \geq 1$ and each g_i is a definite formula. For the sake of simplicity of notation, we deal with the case $n = 2$, that is $f_p = g \vee h$ with $g, h \in Def_{\#(p)}$. The case $n = 1$, that is, $f_p \in Def_{\#(p)}$, can be easily retrieved as a particular case, while the case $n > 2$ is a straightforward extension. Note that, since definite formulas are closed under logical conjunction, we have that $g \wedge h \in Def_{\#(p)}$. Then, consider $g \leftrightarrow h \in Pos_{\#(p)}$. As above, there exist $\psi_1, \dots, \psi_k \in Def_{\#(p)}$, with $k \geq 1$, such that $g \leftrightarrow h = \psi_1 \vee \cdots \vee \psi_k$. Then, we exploit the following fact [Filé and Ranzato 1999, Lemma 5.4]: For any definite formula ψ there exists a suitable idempotent substitution $\sigma_\psi \in Sub$ such that $\alpha_{\wp(Sub),Def}(\{\sigma_\psi\}) = \psi$. Moreover, note that $\alpha_{\wp(Sub),Pos}(\{\sigma_\psi\}) = \alpha_{\wp(Sub),Def}(\{\sigma_\psi\})$. Thus, there exist $\sigma_1, \dots, \sigma_k \in Sub$ such that, for any $i \in [1, k]$, $\alpha_{\wp(Sub),Pos_{\#(p)}}(\{\sigma_i\}) = \psi_i$. Now, let us consider the following program Q (we assume that $\#(p) = n$):

$$\begin{array}{l} p(x_1, \dots, x_n)\sigma_1. \\ \vdots \\ p(x_1, \dots, x_n)\sigma_k. \end{array}$$

If J is a generic object belonging to *Def* or *Pos* and $q \in \Pi$ is a predicate symbol, then let us denote by J_q the definite or positive formula associated to q by J. Then, let us observe that for any $J \in Pos$, if $H = T_Q^{Pos}(J) \in Pos$, then:

(1) If $q \neq p$, then, since the predicate q does not appear in Q, $H_q = false$;
(2) By definition of Q, $H_p = J_p \wedge (\psi_1 \vee \cdots \vee \psi_k)$.

Consider the object $K \in Def$ defined as follows: $K_p = g \wedge h$, and if $q \neq p$, then $K_q = true$. Thus, the following equalities hold.

$$\begin{aligned}
\max(\{J \in \mathit{Pos} \mid T_Q^{\mathit{Pos}}(J) \leq_{\mathit{Pos}} K\}) &= \text{(by the observations above)}\\
\max(\{J \in \mathit{Pos} \mid J_p \wedge (\psi_1 \vee \cdots \vee \psi_k) \leq g \wedge h\}) &=\\
\max(\{J \in \mathit{Pos} \mid J_p \wedge (g \leftrightarrow h) \leq g \wedge h\}) &= \text{(by logical rules)}\\
\max(\{J \in \mathit{Pos} \mid J_p \leq (g \leftrightarrow h) \rightarrow (g \wedge h)\}) &= \text{(by logical rules)}\\
\max(\{J \in \mathit{Pos} \mid J_p \leq g \vee h\}).&
\end{aligned}$$

Hence, $\max(\{J \in \mathit{Pos} \mid T_Q^{\mathit{Pos}}(J) \leq_{\mathit{Pos}} K\}) = \{U\}$, where: $U_p = g \vee h = I_p$, and if $q \neq p$, then $U_q = \mathit{true}$. Thus,

$$\{U_p\}_{p\in\Pi} \subseteq \mathcal{M}_{\mathit{Pos}}(\cup_{P\in \mathit{Program}, J\in \mathit{Def}} \max(\{I \in \mathit{Pos} \mid T_P^{\mathit{Pos}}(I) \leq_{\mathit{Pos}} J\})).$$

Hence, because $I = \wedge_{\mathit{Pos}}\{U_p \in \mathit{Pos} \mid p \in \Pi\}$, we get that

$$I \in \mathcal{M}_{\mathit{Pos}}(\cup_{P\in \mathit{Program}, J\in \mathit{Def}} \max(\{I \in \mathit{Pos} \mid T_P^{\mathit{Pos}}(I) \leq_{\mathit{Pos}} J\})),$$

that is, $I \in R_F(\mathit{Def})$, and this closes the proof. □

Let us discuss the consequences of Theorem 6.8, in view of the general observations made in Section 6.2. 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 $\mathbb{P}(\mathit{Def})$. However, according to our definitions in Section 6.2, $\mathbb{P}(\mathit{Def})$ is too refined, since, by Theorem 6.8, *Pos*, that is, the absolute complete shell of *Def* with respect to $\mathbb{P}(\mathit{Def})$, is a proper abstraction of $\mathbb{P}(\mathit{Def})$. By contrast, according to the "efficiency-oriented" strategy of refinement, 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 would dramatically gain in efficiency: While $\mathbb{P}(\mathit{Def})$ has an exponential size with respect to *Def*, that is $|\mathbb{P}(\mathit{Def})| = O(2^{|\mathit{Def}|})$, by contrast, $|\mathit{Pos}| = O(|\mathit{Def}| + 2^{|VI|})$, yet maintaining a relationship of completeness (and hence fixpoint completeness) between *Pos* and $\mathbb{P}(\mathit{Def})$.

7. *Related Work*

Completeness issues in abstract interpretation have been first considered by Cousot and Cousot [1979, Section 7], who studied the basic properties of complete abstract interpretations and gave some examples in data-flow analysis (cf. Cousot and Cousot [1979, Examples 7.2.0.6.2 and 7.2.0.6.3]). After that seminal paper, a number of works considered complete abstract interpretations in disparate specific applications. It would be rather hard to give a full account of all these works; here, we consider the most related works and some representative works in the most traditional areas of application of abstract interpretation.

To our knowledge, few works devote particular attention to completeness in abstract interpretation by its own. Let us cite Steffen's [1989] work on optimal data-flow analysis for imperative programs (extending author's previous work [Steffen 1987]), which is loosely related to our approach. Leaving out the details, let us recall the overall picture of Steffen's approach. Steffen considers a fixed standard data-flow semantics $\mathcal{C}$, and a given observation abstract domain Ω. An abstract semantics $\mathfrak{G}$ is then understood as any semantics which is more abstract than (i.e., sound for) $\mathscr{C}$ and more concrete than Ω (i.e., Ω must be sound for $\mathfrak{S}$). An abstract semantics is called a model of Ω, and two models $\mathfrak{S}_1$ and $\mathfrak{S}_2$ of Ω are called observationally equivalent for Ω whenever they induce the same best correct approximations over Ω. Thus, Steffen [1989, Theorem 3.13] shows that, given the observation domain Ω, each class of observationally equivalent models of Ω admits a (unique) most abstract model. Hence, although Steffen focusses on the weaker relation "to be the best correct approximation of" rather than our completeness or fixpoint completeness, Steffen's goal is related to our complete shells, because they both aim to find the most abstract domain for a certain optimality concept. Moreover, Steffen [1989, Theorem 3.16] also proves that observational equivalence is preserved by complete abstract interpretations: If $\mathfrak{S}_1$ is a model of Ω and $\mathfrak{S}_2$ is a complete abstraction of $\mathfrak{S}_1$ then $\mathfrak{S}_1$ and $\mathfrak{S}_2$ are observationally equivalent. A categorical generalization of Steffen's [1989] work has been successively investigated by Steffen et al. [1992]. Mycroft's [1993] work on completeness in predicate-based abstract interpretation is certainly related to this article. 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. Mycroft [1993, Section 3.2] argues that the well-known state minimization algorithm for finite deterministic automata may be used to produce a canonical complete abstract interpretation by removing useless elements from the domains of a given abstract interpretation. Although the technical approach followed by Mycroft is different from ours, his idea of systematically defining a canonical simplest complete abstract interpretation is basically the same idea of our complete cores. This relationship needs to be deepened in order to investigate whether these two systematic methodologies actually share the same behavior. Instead, Mycroft does not consider the dual problem analogous to our complete shells. More recently, Colby [1997] isolated the phenomenon of accumulated imprecision in abstract interpretation. This is essentially the same as the lack of (fixpoint) completeness for the considered abstract domains: An error is accumulated in a 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 be used as a specification metalanguage for small-step operational semantics of programming languages. Transfer relations are able to describe precisely each single execution step of a program and a key composition operation computes precise behaviors of control paths. Colby showed that this approach allows us to overcome the accumulated imprecision problem in some relevant examples. Thus, in a sense, Colby's approach is orthogonal to ours: The whole semantic metalanguage is changed to gain precision. It could be useful instead to understand accumulated imprecision as an inherent abstract domain property, and then, precision (i.e., completeness) could be enforced in abstract domains by either minimally refining or simplifying them.

In comparative semantics of programming languages, Cousot and Cousot [1992b] first introduced the idea that many classical program semantics may be recasted and constructively derived within the uniform framework of abstract interpretation from a "ground" sufficiently rich concrete semantics. Notably, they showed that from a suitable generalization of Plotkin's structured operational semantics (SOS) for transition systems, it is possible to derive trace, (erratic, demoniac and angelic) relational and denotational program semantics as intermediate steps of an abstract interpretation-based approximation process. Successively, Cousot [2000] extended this work by considering a wider spectrum of semantics, and by demonstrating that a lot of these abstraction relationships between semantics actually are complete or fixpoint complete. Let us also cite Cousot and Cousot's [1997] work on abstract interpretation of algebraic polynomial systems: Here, a hierarchy of abstract semantics of algebraic polynomial systems, recovering and generalizing well-known results from formal language and grammar theory, is introduced, and many relationships are proved to be complete abstract interpretations. In the subfield of logic program semantics, Giacobazzi [1996] proved that the standard declarative semantics for definite logic programs form a hierarchy of complete abstract interpretations, while Comini and Levi [1994] and Comini et al. [1995] introduced an abstract interpretation hierarchy of observables of operational SLD-trees and SLD-derivations where some relationships are formalized by completeness. Amato and Levi [1997] successively studied the lattice-theoretic structure of these observables, and, as recalled in Section 4, independently stated an analogous but weaker result to our Corollary 4.8, under the hypothesis of *additive* concrete operations of type $C^n \to C$. It should be remarked that, even for this restricted case, Amato and Levi [1997] gave no constructive methodology for building their complete abstract interpretations.

In static program analysis, Sekar et al. [1997] focussed on completeness of Mycroft's [1980; 1981] strictness analysis for functional programs. Their aim is fundamentally different from ours. In fact, they characterized from an operational viewpoint the greatest class $\mathbb{C}$ of functional programs (roughly, all the programs whose strictness behavior is unaffected under any variation of constants) such that the strictness analysis of a program P is complete iff $P \in \mathbb{C}$. Thus, if for any functional program P of type $\mathsf{C}^n \to \mathsf{D}$, $[\![P]\!]$: $C^n \to D$ denotes its standard collecting semantics, and $[\![P]\!]^{Strict}$: $A^n \to B$ denotes the corresponding best correct approximation of $[\![P]\!]$ over the strictness abstract domains (as recalled in Section 5.1, the details of abstract interpretation-based strictness analysis can be found in [Burn et al. 1986]), then, in our terminology, Sekar et al. [1997] have identified, for any type $\mathsf{C}^n \to \mathsf{D}$, the greatest class $\mathbb{C}_{C^n \to D}$ of programs whose collecting semantics has type $C^n \to D$, such that $[\![P]\!]^{Strict} \in \Gamma(C, D, \{[\![P]\!]: C^n \to D \mid P \in \mathbb{C}_{C^n \to D}\})$ holds. The proof of this result heavily relies on operational reduction techniques for functional programs. Reddy and Kamin [1993] generalize Sekar et al.'s approach, which first appeared in POPL 1991, from a denotational perspective, by studying completeness of strictness analysis with respect to a so-called similarity concrete denotational semantics. They obtained two 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 Reddy and Kamin's goal substantially differs from ours.

It is well known that completeness is highly desirable in abstract model checking, where it is more commonly known as strong preservation. In fact, ideally, abstract model checking should be complete for a sufficiently expressive class Ψ of temporal logic formulae: If $\mathscr{C}$ and $\mathscr{A}$ are, respectively, the concrete and abstract models, then strong preservation on Ψ arises whenever, for any $\psi \in \Psi$, $\mathscr{C} \models \psi$ iff $\mathscr{A} \models \psi$ [Cleaveland et al. 1995; Dams 1996; Dams et al. 1997]. As Dams et al. [1997, Sections 6 and 9] observe, one may get complete abstract models by refining the precision of the underlying abstract domains. However, Dams et al. [1997, Section 9] claim that abstract interpretation does not offer methodologies to accomplish this task. Accordingly to the work on systematic abstract domain refinement [Cousot and Cousot 1979; Filé et al. 1996; Giacobazzi and Ranzato 1997; 1998b], we believe that the present work might provide a valuable basis for systematically and minimally refine the abstract domains of a given abstract model checking specification in order to achieve completeness.

Completeness, or related variations thereof, can be also helpful in complexity studies on program analysis systems. As we mentioned at the end of Section 6.2, let us first cite Bagnara et al. [1997], who obtained an abstract unification algorithm of quadratic complexity for pair-sharing analysis of logic programs, by characterizing the complete shell of the pair-sharing domain relative to itself with respect to the abstract unification of standard Jacobs and Langen's [1992] set-sharing abstract domain, which suffered of being exponential. Further, King et al. [1999] showed how to get a very efficient implementation of ground-dependency logic program analysis based on the following representation for the abstract domain *Def* of Section 6.3: *Def* is represented as an abstraction of Jacobs and Langen's [1992] set-sharing domain, thanks to a result by Cortesi et al.'s [1998, Theorem 4.10], who demonstrated that *Def* is the complete shell of a basic plain groundness abstract domain relative to itself (in Cortesi et al.'s terminology, a so-called quotient) with respect to the abstract unification of Jacobs and Langen's set-sharing domain. Let us also cite Deutsch's [1997] work on Park and Goldberg's [1992] escape analysis of strongly typed functional languages. Deutsch improved, both in precision and complexity, the escape analysis by specifying a complete abstraction which is polynomial in the first-order case. This represented a strong improvement with respect to the original exponential proposal of Park and Goldberg [1992]. To conclude, it is worthwhile to cite Debray's [1995] work on logic program analysis complexity. Here, the complexity of a data-flow analysis algorithm A is defined to be the complexity of typical programs belonging to the so-called class of exactness of A. The notion of exactness is stronger than completeness: If the algorithm A computes the least fixpoint of an abstract operator $T_P^\sharp$, then the class of exactness of A consists of all the programs P such that $lfp(T_P) = \gamma(lfp(T_P^\sharp))$. Although exactness is strictly stronger than fixpoint completeness, the author proves that it still represents a reasonable absolute (i.e., not relative to other algorithms) measure of precision for practical purposes.

8. *Conclusion*

This article investigated the problem of making abstract interpretations complete from a nonrestrictive domain perspective, and then showed how, under weak hypotheses, it can be constructively solved by minimally extending or restricting the underlying domains of a given abstract interpretation. This is, to the best of our knowledge, a novel approach to the completeness problem in abstract interpretation, and it opens up a range of further research directions in comparative semantics, complexity of program analysis, and abstract model checking, as discussed in Section 7. To conclude, let us mention that a recent result by Ranzato [1999] allows to overcome the assumption made in this article of dealing with concrete domains which must be complete lattices. In fact, Ranzato [1999] demonstrated that the key lattice of abstract interpretations (cf. Section 2) actually can be considered for concrete domains which are mere CPOs. We checked out the constructive results and methodologies of Sections 4 and 5, and, although paying for heavier order-theoretic details which depend on the technical devolpment in Ranzato [1999], they could be extended in order to deal with concrete domains as CPOs, definitely a basic requirement for use in the semantics field.

REFERENCES

Abramsky, S., and Jung, A. 1994. Domain theory. In *Handbook of Logic in Computer Science*, Volume 3. S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, Ed. Clarendon Press, Oxford, U.K., pp. 1–168.

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

Apt, K. R., and Plotkin, G. D. 1986. Countable nondeterminism and random assignment. *J. ACM 33*, 4, 724–767.

Armstrong, T., Marriott, K., Schachte, P., and Søndergaard, H. 1998. Two classes of Boolean functions for dependency analysis. *Sci. Comput. Program 31*, 1, 3–45.

Bagnara, R., Hill, P. M., and Zaffanella, E. 1997. Set-sharing is redundant for pair sharing. In *Proceedings of the 4th International Static Analysis Symposium* (*SAS '97*), P. Van Hentenryck, Ed., Lecture Notes in Computer Science, vol. 1302. Springer-Verlag, New York, pp. 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.

Birkhoff, G. 1967. *Lattice Theory.* AMS Colloquium Publication, 3rd edition, AMS, Providence, R.I.

Burn, G. L., Hankin, C. L., and Abramsky, S. 1986. Strictness analysis of higher-order functions. *Sci. Comput. Program. 7*, 249–278.

Caseau, Y. 1991. Abstract interpretation of constraints on order-sorted domains. In *Proceedings of the 1991 International Logic Programming Symposium* (*ILPS '91*), V. Saraswat and K. Ueda, Eds., The MIT Press, Cambridge, Mass., pp. 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.

Cleaveland, R., Iyer, P., and Yankelevich, D. 1995. Optimality in abstractions of model checking. In *Proceedings of the 2nd International Static Analysis Symposium* (*SAS '95*), A. Mycroft, Ed. Lecture Notes in Computer Science, vol. 983. Springer-Verlag, New York, pp. 51–63.

Cleaveland, R., and Riely, J. 1994. Testing-based abstractions for value-passing systems. In *Proceedings of the 5th International Conference on Concurrency Theory* (*CONCUR '94*), B. Jonsson and J. Parrow, Eds. Lecture Notes in Computer Science, vol. 836. Springer-Verlag, New York, pp. 417–432.

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

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

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

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

Comini, M., Levi, G., and Meo, M. C. 1995. Compositionality of SLD-derivations and their abstractions. In *Proceedings of the 1995 International Logic Programming Symposium* (*ILPS '95*), J. Lloyd, Ed. The MIT Press, Cambridge, Mass., pp. 561–575.

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.

Cortesi, A., Filé, G., and Winsborough, W. 1998. The quotient of an abstract interpretation. *Theor. Comput. Sci. 202*, 1–2, 163–192.

Cortesi, A., Le Charlier, B., and Van Hentenryck, P. 1995. Evaluation of the domain Prop. *J. Logic Program. 23*, 3, 237–278.

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. dissertation. Université Scientifique et Médicale de Grenoble, Grenoble, France.

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

Cousot, P. 1996b. Program analysis: The abstract interpretation perspective. *ACM Comput. Surv. 28A*, 4es, 165-es.

Cousot, P. 1997. Types as abstract interpretations (Invited Paper). In *Conference Record of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '97)* (Paris, France, Jan. 15–17). ACM Press, New York, pp. 316–331.

Cousot, P. 2000. Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. *Theoret. Comput. Sci.,* to appear.

Cousot, P. and Cousot, R. 1976. Static determination of dynamic properties of programs. In *Proceedings of the 2nd International Symposium on Programming*. Dunod, Paris, pp. 106–130.

Cousot, P., and Cousot, R. 1997. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the 4th ACM Symposium on Principles of Programming Languages (POPL '77).* ACM Press, New York, pp. 238–252.

Cousot, P., and Cousot, R. 1979. Systematic design of program analysis frameworks. In *Conference Record of the 6th ACM Symposium on Principles of Programming Languages* (*POPL '79*). ACM Press, New York, pp. 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. Inductive definitions, semantics and abstract interpretation. In *Conference Record of the 19th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '92)* (Albuquerque, N.M., Jan. 19–22). ACM Press, New York, pp. 83–94.

Cousot, P., and Cousot, R. 1993. Galois connection based abstract interpretation for strictness analysis (Invited Paper). In *Proceedings of the International Conference 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, New York, pp. 98–127.

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 *Proceedings of the 7th International Conference on Computer Aided Verification* (*CAV '95*), P. Wolper, Ed. Lecture Notes in Computer Science, vol. 939. Springer-Verlag, New York, pp. 293–308.

Cousot, P., and Cousot, R. 1997. Abstract interpretation of algebraic polynomial systems. In *Proceedings of the 6th International Conference on Algebraic Methodology and Software Technology* (*AMAST '97*), M. Johnson, Ed. Lecture Notes in Computer Science. Springer-Verlag, New York, pp. 138–154.

Dams, D. 1996. Abstract interpretation and partition refinement for model checking. Ph.D. dissertation. Eindhoven Univ. of Technology, Eindhoven, The Netherlands.

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.

de Bakker, J. W., Meyer, J.-J. C., and Zucker, J. I. 1983. On infinite computations in denotational semantics. *Theoret. Comput. Sci. 26*, 1–2, 53–82.

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 Symposium on Principles of Programming Languages (POPL '97)* (Paris, France, Jan. 15–17). ACM Press, New York, pp. 358–371.

FALASCHI, M., LEVI, G., MARTELLI, M., AND PALAMIDESSI, C. 1989. Declarative modeling of the operational behavior of logic languages. *Theoret. 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. 1999. The powerset operator on abstract interpretations. *Theoret. Comput. Sci. 222*, 1–2, 77–111.

GIACOBAZZI, R. 1996. "Optimal" collecting semantics for analysis in a hierarchy of logic program semantics. In *Proceedings of the 13th International Symposium on Theoretical Aspects of Computer Science (STACS '96)*, C. Puech and R. Reischuk, Eds. Lecture Notes in Computer Science, vol. 1046. Springer-Verlag, New York, pp. 503–514.

GIACOBAZZI, R., AND RANZATO, F. 1997. Refining and compressing abstract domains. In *Proceedings of the 24th International Colloquium 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, New York, pp. 771–781.

GIACOBAZZI, R., AND RANZATO, F. 1998a. Optimal domains for disjunctive abstract interpretation. *Sci. Comput. Program 32*, 1–3, 177–210.

GIACOBAZZI, R., AND RANZATO, F. 1998b. Uniform closures: order-theoretically reconstructing logic program semantics and abstract domain refinements. *Info. Comput. 145*, 2, 153–190.

GIACOBAZZI, R., RANZATO, F., AND SCOZZARI, F. 1998. Complete abstract interpretations made constructive. In *Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science (MFCS '98)*, L. Brim, J. Gruska, and J. Zlatuška, Eds. Lecture Notes in Computer Science, vol. 1450. Springer-Verlag, New York, pp. 366–377.

GRAF, S. 1999. Characterization of a sequentially consistent memory and verification of a cache memory by abstraction. *Distrib. Comput. 12*, 2–3, 75–90.

HERMENEGILDO, M., WARREN, D. S., AND DEBRAY, S. K. 1992. Global flow analysis as a practical compilation tool. *J. Logic Program. 13*, 4, 349–366.

JACOBS, D., AND LANGEN, A. 1992. Static analysis of logic programs for independent AND-parallelism. *J. Logic Program. 13*, 2–3, 154–165.

KING, A., SMAUS, J., AND HILL, P. 1999. Quotienting Share for dependency analysis. In *Proceedings of the 8th European Symposium on Programming (ESOP '99)*, S. D. Swierstra, Ed. Lecture Notes in Computer Science, vol. 1576. Springer-Verlag, New York, pp. 59–73.

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., pp. 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.

MATHEMATICS OF PROGRAM CONSTRUCTION GROUP. 1995. Fixed-point calculus. *Inform. Process. Lett. 53*, 3, 131–136.

MONSUEZ, B. 1995. System F and abstract interpretation. In *Proceedings of the 2nd International Static Analysis Symposium (SAS '95)*, A. Mycroft, Ed. Lecture Notes in Computer Science, vol. 983. Springer-Verlag, New York, pp. 279–295.

MYCROFT, A. 1980. The theory and practice of transforming call-by-need into call-by-value. In *Proceedings of the 4th International Symposium on Programming*, B. Robinet, Ed. Lecture Notes in Computer Science, vol. 83. Springer-Verlag, New York, pp. 269–281.

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

MYCROFT, A. 1993. Completeness and predicate-based abstract interpretation. In *Proceedings of the ACM Symposium on Partial Evaluation and Program Manipulation (PEPM '93).* ACM, New York, pp. 179–185.

ØRBÆK, P. 1995. Can you trust your data? In *Proceedings of the 6th International Joint Conference CAAP/FASE, Theory and Practice of Software Development (TAPSOFT '95)*, P. Mosses, M. Nielsen, and M. Schwartzbach, Eds. Lecture Notes in Computer Science, vol. 915. Springer-Verlag, New York, pp. 575–589.

ØRBÆK, P., AND PALSBERG, J. 1997. Trust in the lambda-calculus. *J. Funct. Program. 7*, 6, 557–591.

PARK, Y., AND GOLDBERG, B. 1992. Escape analysis on lists. In *Proceedings of the 1992 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '92)* (San Francisco, Calif., June 17–19). ACM, New York, pp. 116–127.

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

RANZATO, F. 1999. Closures on CPOs form complete lattices. *Infor. Comput.*, *152,* 2, 236–249.

REDDY, U. S., AND KAMIN, S. N. 1993. On the power of abstract interpretation. *Comput. Lang. 19*, 2, 79–89.

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. 1987. Optimal run time optimization proved by a new look at abstract interpretation. In *Proceedings of the International Joint Conference on Theory and Practice of Software Development (TAPSOFT '87)*, H. Ehrig, R. A. Kowalski, G. Levi, and U. Montanari, Eds. Lecture Notes in Computer Science, vol. 249. Springer-Verlag, New York, pp. 52–68.

STEFFEN, B. 1989. Optimal data flow analysis via observational equivalence. In *Proceedings of the 14th International Symposium on Mathematical Foundations of Computer Science (MFCS '89)*, A. Kreczmar and G. Mirkowska, Eds. Lecture Notes in Computer Science, vol. 379. Springer-Verlag, New York, pp. 492–502.

STEFFEN, B., JAY, C. B., AND MENDLER, M. 1992. Compositional characterization of observable program properties. *RAIRO Inform. Théor. Appl. 26*, 5, 403–424.

VAN ROY, P., AND DESPAIN, A. M. 1990. The benefits of global dataflow analysis for an optimizing Prolog compiler. In *Proceedings of the 1990 North American Conference on Logic Programming (NACLP '90)* S. K. Debray and M. Hermenegildo, Eds. The MIT Press, Cambridge, Mass., pp. 501–515.

VENET, A. 1996. Abstract interpretation of the π-calculus. In *Proceedings of the 5th LOMAPS Meeting on Analysis and Verification of High-Level Concurrent Languages*, M. Dam, Ed. Lecture Notes in Computer Science, vol. 1192. Springer-Verlag, New York, pp. 51–75.

VON KARGER, B. 1998. Temporal algebra. *Math. Struct. Comput. Sci. 8*, 3, 277–320.

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

RECEIVED OCTOBER 1997; REVISED JUNE 1999; ACCEPTED SEPTEMBER 1999