ELSEVIER Theoretical Computer Science 222 (1999) 77–111

Theoretical Computer Science

www.elsevier.com/locate/tcs

The powerset operator on abstract interpretations

Gilberto Filé, Francesco Ranzato [*]

Dipartimento di Matematica Pura ed Applicata, Università di Padova, Via Belzoni 7, 35131 Padova, Italy

Communicated by G. Levi

Abstract

In the context of the standard Cousot and Cousot framework, refinement operators that systematically produce more precise abstract interpretations from simpler ones are useful. We present a theoretical study of one such operator: the powerset. For any given abstract interpretation, i.e. an abstract domain equipped with corresponding abstract operations, the powerset operator yields a new abstract interpretation, where the abstract domain is (very close to) the powerset of the original one and the operations are accordingly extended. It turns out that the refined powerset domain is able to represent in the best possible way the concrete disjunction. Conditions that guarantee the correctness of the powerset operator are given, and the relationship, with respect to the precision, between any abstract interpretation and its powerset is studied. The general theory is applied to the well-known abstract interpretation *POS*, typically used for ground-dependency analysis of logic languages. We show that the powerset *P*(*POS*) is strictly more precise than *POS* both at the domain and operations level. Furthermore, the standard bottom-up abstract semantics of logic programs based on *POS* and *P*(*POS*) are compared by exhibiting a completeness relationship between them, i.e. the first semantics can be obtained by abstracting back the second one. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Abstract interpretation; Powerset operator; Logic program ground-dependency analysis

1. Introduction

Abstract interpretation [13, 14] is a widely known methodology for programming language semantics approximation, which is primarily used for specifying static program analysis frameworks. Its basic idea is as follows. Let L be any programming language and *SEM* be a semantic description of L parameterized w.r.t. the domain of computation and the semantic operations. Several semantics of L can be specified by giving different interpretations of *SEM*. The standard or concrete semantics is obtained

[*] Corresponding author. E-mail: franz@math.unipd.it.

by specifying the concrete interpretation $\mathscr{C} = \langle C, o_C^1, \ldots, o_C^k \rangle$, where C is the actual domain of computation of the programs, and $o_C^1, \ldots, o_C^k$ are the operations that can be evaluated during the execution. In this setting, an approximate semantics is specified by giving a nonstandard or abstract interpretation $\mathscr{D} = \langle D, o_1, \ldots, o_k \rangle$ of *SEM*, where D represents the approximate properties of C, and $o_1, \ldots, o_k$ mimic on D the operations $o_C^1, \ldots, o_C^k$. Following the standard terminology, C and $o_C^1, \ldots, o_C^k$ are called the concrete domain and operations, respectively, while D and $o_1, \ldots, o_k$ are, respectively, the abstract domain and operations. The concrete and abstract domains are equipped with partial ordering relations describing the relative precision of domain values (where the top element gives no information). The correctness of the approximation is guaranteed when $\mathscr{D}$ satisfies some conditions relating it to $\mathscr{C}$ (in this case, we say that $\mathscr{D}$ abstracts $\mathscr{C}$). In particular, the correspondence between the domains C and D must be given by a Galois connection (γ, D, C, α), where $\alpha(c) \leqslant_D d$ (or, equivalently, $c \leqslant_C \gamma(d)$) holds if d is an abstract approximation of the concrete value c. Here, $\alpha(c)$ is the most precise abstract (in D) approximation of c, while $\gamma(d)$ is the concrete meaning of d. The main advantages of abstract interpretation over "ad hoc" dataflow analysis methods are its generality and the fact that it supports the correctness proof of the analysis.

Clearly, the accuracy of a semantics approximation depends on the expressiveness of the chosen abstract interpretation. Thus, it is very interesting to define refinement operators that systematically produce new and more precise abstract interpretations from simpler ones (see [21] for a general treatment of refinement operators at the abstract domain level). Examples of well-known refinement operators on abstract domains are Cousot and Cousot's reduced product [14] and Nielson's tensor product [35]. The *powerset operator* (also called *disjunctive completion*) on abstract domains was originally introduced by Cousot and Cousot in their seminal work [14], where it has been exploited in order to demonstrate that a merge-over-all-paths dataflow analysis can be expressed in least fixpoint form. It has been further studied in [15, 17], and applied, e.g., for the definition of comportment analysis for higher-order functional languages in [17], and in Jensen's disjunctive strictness logic [29] for functional languages.

The basic idea of the powerset operator is simple. Given an abstract domain D, where the concrete domain is C, any subset S of D is considered as a denotation for the concrete disjunction of its elements, namely for the lub in C of the meaning of the values in S. Let us consider a classical [14] and very simple example. Assume that the concrete domain is the powerset $\wp(\mathbb{Z})$ of the set of integers, equipped with the set-inclusion ordering. The abstract domain is *Sign* depicted in Fig. 1, which enjoys an obvious Galois connection with $\wp(\mathbb{Z})$ (for instance, $\gamma(+) = \{x \in \mathbb{Z} : x > 0\}$ and $\alpha(\{-1, -3\}) = -$). In this case, the concrete disjunction is given by the union of sets of integers. It is simple to verify that the powerset abstract domain $P(\mathit{Sign})$ is that depicted in Fig. 1. $P(\mathit{Sign})$ contains three new elements with respect to *Sign*: For example, $[+, -]$ represents precisely the disjunction of $+$ and $-$, and therefore $[+, -]$ is a denotation for the set of nonzero integers. Also notice that $[\top] = [+, 0, -]$, since $\mathbb{Z} = \gamma(\top) = \gamma(+) \cup \gamma(0) \cup \gamma(-)$.

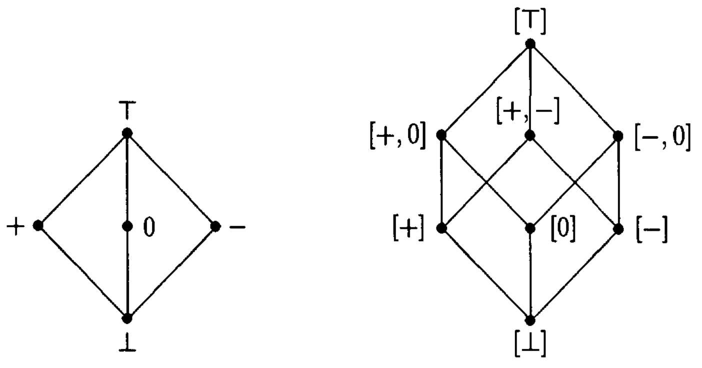
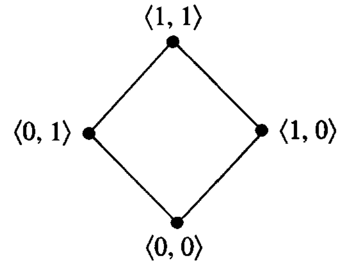


Fig. 1. The abstract domain *Sign* and its poweset *P*(*Sign*).

It should be clear that the powerset operator can generate very expressive abstract interpretations that allow to improve the precision of a program analysis. Let us consider the example of Mycroft's strictness analysis for functional languages [4, 33]. A function f is said to be strict if it is undefined when applied to undefined arguments. If $\bot$ denotes a generic undefined value, then f is strict if $f(\bot) = \bot$. For example, the abstract domain for the basic type Int of integers $\wp(\mathbb{Z}_\bot)$ (where $\bot$ denotes undefinedness) is the two-point domain $str = \{0, 1\}$ (with $0 < 1$), where $\gamma(0) = \{\bot\}$ and $\gamma(1) = \mathbb{Z}_\bot$. In this case, whenever the strictness analysis of some function f of type Int $\rightarrow$ Int is able to detect that the semantic abstraction f^a over str of f is such that $f^a(0) = 0$, then one can infer that f is strict. As observed in [29], the standard abstract domains used for strictness analysis are not able to model precisely the logical disjunction. The abstract domain for the product type Int $\times$ Int is given by the cartesian product abstract domain $str \times str$ depicted below.



The meaning of the abstract values of $str \times str$ is the most natural: For instance, $\langle 0, 1\rangle$ represents the set of pairs of integers, whose first component is undefined, whereas $\langle 1, 1\rangle$ represents the whole set of pairs of integers (defined or not). It is clear that the lub $\langle 1, 1\rangle = \langle 0, 1\rangle \vee \langle 1, 0\rangle$ is not precise:

$$\gamma(\langle 0, 1\rangle) \cup \gamma(\langle 1, 0\rangle) = \{\langle z_1, z_2\rangle \in \mathbb{Z}_\bot \times \mathbb{Z}_\bot : z_1 = \bot \text{ or } z_2 = \bot\}$$
$$\subset \mathbb{Z}_\bot \times \mathbb{Z}_\bot = \gamma(\langle 1, 1\rangle),$$

and therefore $str \times str$ is less precise than its powerset. Consider now the following functional program suggested by Jensen [29], where **sum** performs the addition of a pair of integers:

$$f(x) = \textbf{if } B \textbf{ then } \langle x,3\rangle \textbf{ else } \langle 3,x\rangle$$
$$g(x) = \textbf{sum}(f(x))$$

Suppose that the value of the Boolean expression B cannot be statically determined, and that we already know that **sum** is strict in both its components. It is clear that g is a strict function. However, by using the abstract domain $str \times str$ we are not able to detect the strictness of f. In fact, the best approximation of applying f to an undefined argument is given by $\alpha(\{\langle\bot,3\rangle\}) \vee \alpha(\{\langle 3,\bot\rangle\})$, which in $str \times str$ is $\langle 1,1\rangle$. But evaluating **sum** for such an abstract value, we cannot detect the strictness of g. On the other hand, if one uses the powerset abstract domain $P(str \times str)$ then $\alpha(\{\langle\bot,3\rangle\}) \vee \alpha(\{\langle 3,\bot\rangle\})$ is given by $[\langle 0,1\rangle,\langle 1,0\rangle]$. Then, we get $[\textbf{sum}(\langle 0,1\rangle) = 0, \textbf{sum}(\langle 1,0\rangle) = 0] = [0]$, and therefore we detect that g is strict.

This paper contains a general study of the powerset operator on abstract interpretations. For any abstract interpretation $\mathscr{D} = \langle D, o_1, \ldots, o_k\rangle$ abstracting $\mathscr{C} = \langle C, o_C^1, \ldots, o_C^k\rangle$, its powerset $P(\mathscr{D}) = \langle P(D), o_1^*, \ldots, o_k^*\rangle$ is systematically defined. Thus, our approach considers full abstract interpretations, namely abstract domains equipped with corresponding abstract operations. Conditions on the concrete interpretation $\mathscr{C}$ that assure the correctness of the powerset $P(\mathscr{D})$ are stated: The concrete domain C must be a completely distributive lattice, and any concrete operation o_C^i must satisfy a restricted form of additivity. We show that the powerset operator actually is a refinement operator in the sense of [21], and therefore $P(\mathscr{D})$ (when it is correct) is an abstract interpretation that is always better than $\mathscr{D}$. Also, we demonstrate that under certain conditions it is possible to sharp the definition of the systematic lifting to the powerset of an abstract operation o_i, so that when o_i is complete (i.e., $\alpha \circ o_C^i = o_i \circ \alpha$ holds), its lifting to the powerset is complete too. This will be the case of an abstract operation considered in our application to logic program analysis.

As recalled above, the powerset operator was first introduced by Cousot and Cousot in [14], and then successively generalized in [15, 17]. However, in those papers the powerset is applied to abstract domains only, whereas the corresponding new abstract operations are only defined implicitly as the best correct approximations of the concrete ones. In contrast, our approach allows to derive correct abstract operations for the powerset domain that are directly based on the definition of those of the original domain. Moreover, if the latter are finitely computable (and the abstract domain is finite) then so are the former. As far as the abstract domain is concerned, we show that our definition of the powerset is equivalent to those of [15, 17], although they differ from a syntactic perspective. In particular, our approach defines a simple and natural ordering relation on the powerset abstract domain that is based on the obvious set-inclusion relation.

The powerset operator is applied to the well-known abstract interpretation POS [1, 10, 31, 32], typically used for ground-dependency analysis of logic languages. The abstract domain *Pos* consists of positive (i.e., true under the unitary truth-assignment) propositional formulae, while the abstract operations are logical conjunction (that simulates concrete unification), disjunction, and existential quantification. We show that the powerset abstract interpretation *P*(*POS*) is strictly better than *POS*. This result is somehow against the intuition, given that the abstract domain *Pos* is already closed under logical disjunction. In order to clarify this phenomenon, we characterize precisely the subsets of formulae of *Pos* for which the concretization map preserves their logical disjunction. Furthermore, we show that this is the case for every subset consisting only of monotone formulae, which are exactly the abstraction in *POS* of all the sets of ground substitutions. We also study the relationship between the standard bottom-up abstract semantics (cf. [3, 5, 32]) instantiated to the abstract interpretations *POS* and *P*(*POS*). We prove that the semantics using *POS* is complete with respect to that using *P*(*POS*), in the sense that the former semantics can be obtained by abstracting the latter back to *Pos*. From this result it follows that, using *P*(*POS*) instead of *POS* for analysing logic programs, one cannot gain plain ground-dependency *Pos*-information, but possibly only disjunctive ground-dependency information, i.e. information that the base abstract domain *Pos* is not able to represent with no loss of precision.

The rest of the paper is organized as follows. Section 2 contains the basic notations and notions on abstract interpretation used throughout the paper. The powerset operator is defined and studied in Sections 3 and 4. Section 3 is concerned with the powerset abstract domain, while in Section 4 the abstract operations for the powerset domain are defined and studied. The application of the powerset operator to the abstract interpretation *POS* is described in Section 5. Finally, Section 6 contains some concluding remarks. A preliminary short version of this paper appeared as [22].

2. Preliminaries

In this section, we briefly introduce some notation used throughout the paper and summarize some well-known notions concerning abstract interpretation. For more details on Galois connections see e.g. [25], while for abstract interpretation see the survey [15].

2.1. Galois connections

Throughout the paper, we will use the following basic notation and terminology. If f is a function defined on the set X and $A \subseteq X$ then $f(A) = \{f(a) : a \in A\}$. We use $\circ$ for function composition. If X and Y are sets, we write $X \setminus Y$ to denote the set difference between X and Y, and $Y \subset X$ to denote that Y is a proper subset of X. If $\leqslant$ is a partial ordering then $a < b$ stands for $a \leqslant b$ and $a \neq b$. If P is a partially ordered set (poset) and $Y \subseteq P$, then $max(Y) = \{y \in Y : \forall z \in Y.\ y \leqslant_P z \Rightarrow y = z\}$ denotes the set of maximal elements of Y, while $\downarrow Y = \{x \in P : \exists y \in Y.\ x \leqslant_P y\}$ denotes the downward closure of Y. A subset $Y \subseteq P$ is downward closed if $Y = \downarrow Y$. A function $f : L \to M$ between complete lattices is additive if for any $X \subseteq L$, $f(\bigvee_L X) =$

$\bigvee_M f(X)$ (co-additivity is dually defined). A complete lattice L is join-generated by a subset $S \subseteq L$ if for all $z \in L$ there exists $X \subseteq S$ such that $z = \bigvee_L X$. An element $x \in L$ of a complete lattice is (completely) join-irreducible if for all $X \subseteq L$, $x = \bigvee_L X$ implies $x \in X$; the set of join-irreducible elements of L is denoted by $JI(L)$.

We recall the definitions of Galois connection and insertion. If C and D are two posets and $\alpha : C \to D$, $\gamma : D \to C$ are monotonic functions such that $\forall c \in C.\ c \leqslant_C \gamma(\alpha(c))$ and $\forall d \in D.\ \alpha(\gamma(d)) \leqslant_D d$, then the quadruple (γ, D, C, α) is a *Galois connection* (G.c. for short) between D and C. If in addition $\forall d \in D.\ \alpha(\gamma(d)) = d$, then (γ, D, C, α) is a *Galois insertion* (G.i. for short) of D in C. We also recall that the above definition of G.c. is equivalent to that of adjunction: (γ, D, C, α) is an *adjunction* if $\forall c \in C. \forall d \in D.\ \alpha(c) \leqslant_D d \Leftrightarrow c \leqslant_C \gamma(d)$. The map α (γ) is called the *left* (*right*) *adjoint* to γ (α).

The following are some well-known properties of Galois connections and insertions that will be useful later on.

(i) If (γ, D, C, α) is a G.i., then γ and α are 1–1 and onto, respectively. Also, γ is an embedding, i.e. $d \leqslant_D d' \Leftrightarrow \gamma(d) \leqslant_C \gamma(d')$.

(ii) If (γ, D, C, α) is a G.c. between the posets D and C, then α preserves lub's (i.e., if for some $S \subseteq C$ the lub $\bigvee_C S$ exists then the lub $\bigvee_D \alpha(S)$ exists, and $\alpha(\bigvee_C S) = \bigvee_D \alpha(S)$), and γ preserves glb's.

(iii) If (γ, D, C, α) is a G.i. of the poset D in the complete lattice C, then D is actually a complete lattice.

(iv) Let C and D be posets, and suppose that $\gamma : D \to C$ preserves glb's; in addition, for all $c \in C$ assume that $\bigwedge_D \{d \in D : c \leqslant_C \gamma(d)\}$ exists. If we define $\alpha : C \to D$ as $\alpha(c) = \bigwedge_D \{d \in D : c \leqslant_C \gamma(d)\}$, then (γ, D, C, α) is a G.c. between D and C. Moreover, if γ is 1–1 then it is a G.i. of D in C.

(v) In any G.c., one of the two functions uniquely determines the other.

Whenever C and D are complete lattices, property (ii) says that α and γ are, respectively, additive and co-additive. By property (v), the function α as defined in (iv) will be the only mapping such that (γ, D, C, α) is a G.c. (it is "the" left adjoint to γ). Moreover, starting with $\alpha : C \to D$ it is possible to state the dual version of (iv).

A G.c. (γ, A, C, α) is the composition of two G.c.'s $(\gamma_{A,D}, A, D, \alpha_{D,A})$ and $(\gamma_{D,C}, D, C, \alpha_{C,D})$ if $\alpha = \alpha_{D,A} \circ \alpha_{C,D}$ (or, equivalently, $\gamma = \gamma_{D,C} \circ \gamma_{A,D}$).

2.2. *Abstract interpretation basics*

As recalled in the introduction, in the setting of abstract interpretation, the concrete and abstract domains, C and D, are related by a Galois connection (γ, D, C, α). Following the standard terminology, α and γ are called the *abstraction* and *concretization* maps, respectively, and D is also called an *abstraction* of C. The intuition is that the abstract domain is a representation of some approximate properties of the values of the concrete domain. Both on the concrete and on the abstract domain, a partial order relation describing the relative precision of the values is defined: $x \leqslant y$ means that x is more precise than y. The concretization map gives the concrete value corresponding to an abstract denotation (i.e. its semantics), whereas for a concrete value the abstraction map gives its best (w.r.t. the ordering of D) abstract approximation (cf. property (iv) in Section 2.1). Thus, an abstract value $y \in D$ approximates a concrete value $x \in C$ if $x \leqslant_C \gamma(y)$, or equivalently (by adjunction), if $\alpha(x) \leqslant_D y$. When $c = \gamma(d)$, we will say that the abstract denotation d represents precisely (i.e. with no approximation) the concrete value c. If (γ, D, C, α) is G.i., each value of the abstract domain D is useful in the representation of the concrete domain C, because all the elements of D represent distinct members of C. In practice, C and D are very often complete lattices; however, for the sake of generality, we will assume that they are mere posets, unless otherwise specified.

If $o_C^1, \ldots, o_C^k$ are the operations defined on the concrete domain C, that are involved in the standard semantic definition, then $\mathscr{C} = \langle C, o_C^1, \ldots, o_C^k \rangle$ is called the *concrete interpretation*. For a G.c. (γ, D, C, α), the *abstract operations* over D must correctly simulate the behavior of the concrete operations on the properties represented by D. Let us assume that $o_C : C \times X \rightarrow C$ is a concrete operation of $\mathscr{C}$, where X is any set of possible auxiliary parameters, also mathematically unstructured. [1] Then, following the standard Cousot and Cousot approach, a corresponding abstract operation $o_D : D \times X \rightarrow D$ is a (*correct*) *approximation* of (or (*correctly*) *approximates*) o_C if

$$\forall c \in C. \forall d \in D. \forall x \in X.\ \alpha(c) \leqslant_D d \Rightarrow \alpha(o_C(c,x)) \leqslant_D o_D(d,x).$$

The intuition for this definition should be clear: If d is an approximate description of c, then the concrete computation of $o_C(c,x)$ is approximated at the abstract level (on D) by $o_D(d,x)$. It is possible to state this notion of approximation by several equivalent formulations (cf. [13, 14]). In fact, it is easily shown that o_D approximates o_C iff $\forall c \in C. \forall x \in X.\ \alpha(o_C(c,x)) \leqslant_D o_D(\alpha(c),x)$ iff $\forall d \in D. \forall x \in X.\ o_C(\gamma(d),x) \leqslant_C \gamma(o_D(d,x))$. If each o_C^i is approximated by the corresponding o_i, then we will say that $\mathscr{D} = \langle D, o_1, \ldots, o_k \rangle$ *abstracts* (or is an *abstract interpretation* of) $\mathscr{C} = \langle C, o_C^1, \ldots, o_C^k \rangle$.

Assume that o_1 and o_2 are two abstract operations (for D) both approximating a common concrete operation o_C. Following the standard Cousot and Cousot definition [14], we say that o_1 is *more precise* than o_2 if for any $d \in D$ and $x \in X$, $o_1(d,x) \leqslant_D o_2(d,x)$ (namely, if o_1 is lower than o_2 with respect to the standard functional pointwise ordering). Cousot and Cousot showed in [14] that it is always possible to define the *best* (*correct*) *approximation* of a concrete operation o_C: This is defined as $o_D^{best}(d,x) = \alpha(o_C(\gamma(d),x))$, for any $d \in D$ and $x \in X$. Actually, it is easy to verify that this o_D^{best} is more precise than every approximation of o_C. The notion of completeness for an abstract operation is also well-known [15, 34]. We say that o_D is *complete* for o_C if for any $c \in C$ and $x \in X$, $o_D(\alpha(c),x) = \alpha(o_C(c,x))$. Notice that if o_D is complete for o_C then o_D is the best approximation of o_C, while the converse is in general not true. If C and D are complete lattices and we consider the concrete and abstract lub's as operations, then we always have that $\bigvee_D$ is complete for $\bigvee_C$: In

[1] The extension to the more general case $o_C : C^n \times X \rightarrow C^m \times Y$ is straightforward.

fact, by (ii) in Section 2.1, for any $S \subseteq C$, we have that $\bigvee_D \alpha(S) = \alpha(\bigvee_C S)$, hence showing the completeness. Completeness is a quite strong property for abstract operations (see [34]). It also implies that if the least fixpoints $lfp(o_D)$ and $lfp(o_C)$ exist, then $\alpha(lfp(o_C)) = lfp(o_D)$ (see [15]).

2.3. Comparing abstract interpretations

As far as precision of representation is concerned, the standard criterion for comparing abstract domains, introduced by Cousot and Cousot in [14], is as follows. Let $G_1 = (\gamma_1, D_1, C, \alpha_1)$ and $G_2 = (\gamma_2, D_2, C, \alpha_2)$ be two G.c.'s. Then, D_2 is *better* than D_1 whenever $\gamma_1(D_1) \subseteq \gamma_2(D_2)$, while D_2 is *strictly better* than D_1 if D_2 is better than D_1 and D_1 is not better than D_2, i.e. if $\gamma_1(D_1) \subset \gamma_2(D_2)$. Also, D_1 and D_2 are *equivalent* when D_2 is better than D_1 and D_1 is better than D_2, i.e. when $\gamma_1(D_1) = \gamma_2(D_2)$. Thus, intuitively, D_2 is better than D_1 when D_2 is able to represent precisely at least all the concrete elements that are represented precisely by D_1.

Lemma 2.1. (i) *If G_2 is a G.i., then D_2 is better than D_1 iff $G_{1,2} = (\alpha_2 \circ \gamma_1, D_1, D_2, \alpha_1 \circ \gamma_2)$ is a G.c.*

(ii) *If G_1, G_2 are G.i.'s, then D_2 is better than D_1 iff $G_{1,2}$ is a G.i.* [9].

Proof. We prove (i) ($\Rightarrow$) Monotonicity of $\alpha_2 \circ \gamma_1$ and $\alpha_1 \circ \gamma_2$ follows from that of their components. Thus, it is enough to show the following two points.

- $\forall d_1 \in D_1.\ \alpha_1(\gamma_2(\alpha_2(\gamma_1(d_1)))) \leqslant_{D_1} d_1$: By hypothesis, there exists $d_2 \in D_2$ such that $\gamma_1(d_1) = \gamma_2(d_2)$; hence, $\alpha_1(\gamma_2(\alpha_2(\gamma_1(d_1)))) = \alpha_1(\gamma_2(\alpha_2(\gamma_2(d_2)))) = \alpha_1(\gamma_2(d_2)) = \alpha_1(\gamma_1(d_1)) \leqslant_{D_1} d_1$.
- $\forall d_2 \in D_2.\ d_2 \leqslant_{D_2} \alpha_2(\gamma_1(\alpha_1(\gamma_2(d_2))))$: Since $\gamma_2(d_2) \leqslant_C \gamma_1(\alpha_1(\gamma_2(d_2)))$, by monotonicity of α_2, we get $d_2 = \alpha_2(\gamma_2(d_2)) \leqslant_{D_2} \alpha_2(\gamma_1(\alpha_1(\gamma_2(d_2))))$.

($\Leftarrow$) It suffices to prove that $\forall d_1 \in D_1.\ \gamma_1(d_1) = \gamma_2(\alpha_2(\gamma_1(d_1)))$. Observe that, since $G_{1,2}$ is a G.c., $\alpha_1(\gamma_2(\alpha_2(\gamma_1(d_1)))) \leqslant_{D_1} d_1$. Hence, $\gamma_2(\alpha_2(\gamma_1(d_1))) \leqslant_C \gamma_1(d_1)$, that joint with $\gamma_1(d_1) \leqslant_C \gamma_2(\alpha_2(\gamma_1(d_1)))$ concludes the proof. □

It is also worth noting that if D_2 is better than D_1 (and G_2 is a G.i.) then G_1 is the composition of $G_{1,2}$ and G_2, i.e. for all $d_1 \in D_1$, $\gamma_1(d_1) = \gamma_2(\alpha_2(\gamma_1(d_1)))$ (this is a direct consequence of the fact that if $\gamma_1(d_1) = \gamma_2(d_2)$ then $d_2 = \alpha_2(\gamma_1(d_1))$).

We extend in the most natural way the notion of being better to full abstract interpretations. Let us assume that $o_1 : D_1 \times X \to D_1$ and $o_2 : D_2 \times X \to D_2$ are approximations of the same concrete operation o_C. Following the standard criterion of comparison of Cousot and Cousot (cf. [13]), we say that the abstract operation o_2 is *better* than the corresponding o_1 if $\forall c \in C. \forall x \in X.\ \gamma_2(o_2(\alpha_2(c), x)) \leqslant_C \gamma_1(o_1(\alpha_1(c), x))$. Furthermore, o_2 is *strictly better* than o_1 if o_2 is better than o_1 and o_1 is not better than o_2, i.e. there exist $c \in C$ and $x \in X$ such that $\gamma_2(o_2(\alpha_2(c), x)) <_C \gamma_1(o_1(\alpha_1(c), x))$. Assume now that $\mathscr{D}_1 = \langle D_1, o_{D_1}^1, \ldots, o_{D_1}^k \rangle$ and $\mathscr{D}_2 = \langle D_2, o_{D_2}^1, \ldots, o_{D_2}^k \rangle$ are two abstract interpretations of a common interpretation $\mathscr{C} = \langle C, o_C^1, \ldots, o_C^k \rangle$. $\mathscr{D}_2$ is *better* than $\mathscr{D}_1$ if the abstract domain D_2 is better than D_1, and each operation $o^i_{D_2}$ is better than the corresponding operation $o^i_{D_1}$. Moreover, $\mathscr{D}_2$ is *strictly better* than $\mathscr{D}_1$ when it is better, the abstract domain D_2 is strictly better than D_1, and at least one operation $o^j_{D_2}$ is strictly better than the corresponding $o^j_{D_1}$. Thus, according to this definition, an abstract interpretation is strictly better than another one when its domain is strictly more expressive and when at least one of its operations takes advantage of this extra expressivity in order to be more precise than the corresponding operation of the other domain.

3. The powerset abstract domain

In the following, we assume that the concrete domain C enjoys the standard generalized form of infinite distributivity, i.e. that C is a completely distributive lattice. This means that C is a complete lattice such that for each subset $\{c^i_j\}^{i\in I}_{j\in J(i)} \subseteq C$, where I and, for any $i \in I$, $J(i)$ are sets of indices, $\bigwedge_{i\in I} \bigvee_{j\in J(i)} c^i_j = \bigvee_{\varphi\in J^I} \bigwedge_{i\in I} c^i_{\varphi(i)}$, where J^I is the set of all the functions $\varphi : I \to \bigcup_{i\in I} J(i)$ such that for any $i \in I$, $\varphi(i) \in J(i)$. The dual condition, where meet and join are exchanged, is equivalent (cf. [25]). This condition of complete distributivity is satisfied by any complete ring of sets, i.e. any (complete lattice isomorphic to a) subset of a powerset, ordered by the subset or superset relation and closed under arbitrary unions and intersections (see [2]). In particular, the powerset of any set, ordered with the subset or superset relation, is completely distributive, and therefore this class comprises the concrete domains used in collecting semantics for analysis (cf. [17]). The concrete domain C is related to the abstract domain D by a Galois connection (γ, D, C, α). Notice that if we assume a G.i. of D in C, then, by property (ii) in Section 2.1, the fact that C is a complete lattice implies that so is D.

In the powerset construction, any subset S of the abstract domain D is intended to represent the concrete disjunction of its elements, namely, its concrete meaning is given by the lub $\bigvee_C \gamma(S)$. Thus, we define the following equivalence relation between subsets of D:

$$\text{if } S_1, S_2 \subseteq D \text{ then } S_1 \equiv_\gamma S_2 \Leftrightarrow \bigvee_C \gamma(S_1) = \bigvee_C \gamma(S_2),$$

where, as usual, $\bigvee_C \emptyset = \bot_C$. If $S \subseteq D$ then we denote its equivalence class for $\equiv_\gamma$ by $[S] = \{Z \subseteq D : S \equiv_\gamma Z\}$. In order to simplify the notation, we will often denote the equivalence class of a finite subset $\{d_1, \ldots, d_k\} \subseteq D$ simply by $[d_1, \ldots, d_k]$. The powerset domain of D, denoted by $P(D)$, is defined as the quotient with $\equiv_\gamma$ of the set $\wp(D)$ of all the subsets of D:

$$P(D) \stackrel{\text{def}}{=} \wp(D)_{/\equiv_\gamma} = \{[S] : S \subseteq D\}.$$

For each class $[S] \in P(D)$ it will be useful to have a canonical representative. The *fat* of $[S]$ is defined as $\uplus[S] = \cup\{Z \subseteq D : Z \in [S]\}$. We now show that $\uplus[S] \equiv_\gamma S$, and therefore $\uplus[S]$ turns out to be the greatest element in $[S]$.

Lemma 3.1. *For* $[S] \in P(D)$,
(i) $\uplus[S] \equiv_\gamma S$;
(ii) $\uplus[S] = \{d \in D : \gamma(d) \leqslant_C \vee_C \gamma(S)\}$;
(iii) $\uplus[S] = \downarrow \uplus[S]$.

Proof. (i) $\bigvee_C \gamma(\uplus[S]) = \bigvee_C \gamma(\cup\{Z \subseteq D : Z \in [S]\}) = \bigvee_C \{\bigvee_C \gamma(Z) \in C : Z \in [S]\} = \bigvee_C \gamma(S)$.
(ii) On the one hand, if $Z \in [S]$ and $d \in Z$, then $\gamma(d) \leqslant_C \bigvee_C \gamma(Z) = \bigvee_C \gamma(S)$. On the other hand, if $\gamma(d) \leqslant_C \bigvee_C \gamma(S)$ then $\bigvee_C \gamma(S \cup \{d\}) = \bigvee_C \gamma(S)$, from which $d \in \uplus[S]$.
(iii) If $d' \leqslant_D d$ for some $d \in \uplus[S]$, then $\gamma(d') \leqslant_C \gamma(d) \leqslant_C \bigvee_C \gamma(S)$. Consequently, $d' \in \uplus[S]$ because $\bigvee_C \gamma(S \cup \{d'\}) = \bigvee_C \gamma(S)$. □

It is well-known (see, e.g., [19]) that a poset satisfies the ascending chain condition (ACC for short, namely it does not contain infinite strictly increasing chains) iff for every nonempty subset S of the poset, $max(S) \neq \emptyset$. As a consequence, whenever an abstract domain D satisfies the ACC, any equivalence class $[S] \in P(D)$ contains the set of maximal elements of S, i.e. $S \equiv_\gamma max(S)$. However, it is worth noting that $max(S)$ could not be taken as the canonical representative of an equivalence class $[S]$. For instance, considering the domain *Sign* in Section 1, we have that $[+,0,-] = [\top]$, whereas $max(\{+,0,-\}) = \{+,0,-\} \neq \{\top\} = max(\{\top\})$.

We exploit the fat sets in order to give the ordering relation on $P(D)$:

$$\text{if } [S],[T] \in P(D) \text{ then } [S] \sqsubseteq [T] \Leftrightarrow \uplus[S] \subseteq \uplus[T].$$

Notice that $\sqsubseteq$ is a partial order on $P(D)$ (antisymmetry is a consequence of Lemma 3.1 (i)). For this partial order, $P(D)$ has top and bottom elements: By ordering definition, $\top_{P(D)} = [D]$, and, by Lemma 3.1 (ii), it is easy to verify that $\bot_{P(D)} = [\emptyset]$. If D has the top element $\top_D$ (as observed above, this always holds if we start from a G.i. rather than a G.c.), then $\top_{P(D)} = [\top_D]$.

Proposition 3.2. *$P(D)$ is a complete lattice, where* $\bigsqcup_{i\in I}[S_i] = [\bigcup_{i\in I} \uplus[S_i]]$ *and* $\bigsqcap_{i\in I}[S_i] = [\bigcap_{i\in I} \uplus[S_i]]$, *for all* $\{[S_i]\}_{i\in I} \subseteq P(D)$.

Proof. We first show that if $S \subseteq T$ then $[S] \sqsubseteq [T]$: If $d \in \uplus[S]$ then $\gamma(d) \leqslant_C \bigvee_C \gamma(S) \leqslant_C \bigvee_C \gamma(T)$, and therefore, since $\bigvee_C \gamma(T \cup \{d\}) = \bigvee_C \gamma(T)$, we obtain that $d \in \uplus[T]$. We only prove the existence of glb's. For lub's the proof is dual. By Lemma 3.1 (i) and the above claim, $[\bigcap_{i\in I} \uplus[S_i]] \sqsubseteq [\uplus[S_i]] = [S_i]$, for each $i \in I$. Suppose now that $[T] \sqsubseteq [S_i]$ for each $i \in I$. Then, $\uplus[T] \subseteq \bigcap_{i\in I} \uplus[S_i]$, and again by Lemma 3.1 (i) and the previous claim, $[T] = [\uplus[T]] \sqsubseteq [\bigcap_{i\in I} \uplus[S_i]]$. From this it follows that $[\bigcap_{i\in I} \uplus[S_i]]$ is the glb. □

As it is natural to expect, the powerset abstract domain is always completely distributive.

Proposition 3.3. *$P(D)$ is completely distributive.*

Proof. Complete distributivity of $P(D)$ is a simple consequence of the definition of lub and glb of Proposition 3.2, since each powerset of some set is obviously completely distributive. □

The concretization map for the powerset abstract domain $\gamma^* : P(D) \to C$ is the obvious one, namely it provides the concrete disjunction of an abstract subset: $\gamma^*([S]) = \bigvee_C \gamma(S)$. The hypothesis of complete distributivity of the concrete domain is central in the proof of the next basic lemma.

Lemma 3.4. *γ^* is 1–1 and co-additive.*

Proof. That γ^* is 1–1 follows from the definition of $P(D)$. In order to show that it is co-additive, consider any subset $\mathscr{S} = \{[S_i] \in P(D) : i \in I\}$. For each $i \in I$, we pick out a set of indices $J(i)$ such that $\uplus[S_i] = \{d_j^i \in D : j \in J(i)\}$. Thus, we have

$$\begin{aligned}
\gamma^*(\sqcap\mathscr{S}) &= \bigvee_C \gamma\left(\bigcap_{i\in I}\{d_j^i : j \in J(i)\}\right), \\
\wedge_C\gamma^*(\mathscr{S}) &= \bigwedge_C \left\{\bigvee_C \gamma(\uplus[S_i]) : i \in I\right\} && \text{(by Lemma 3.1 (i))} \\
&= \bigwedge_C \left\{\bigvee_C\{\gamma(d_j^i) : j \in J(i)\} : i \in I\right\} \\
&= \bigvee_C \left\{\bigwedge_C\{\gamma(d_{\varphi(i)}^i) : i \in I\} : \varphi \in J^I\right\} && \text{(by complete distributivity of } C) \\
&= \bigvee_C \left\{\gamma\left(\bigwedge_D\{d_{\varphi(i)}^i : i \in I\}\right) : \varphi \in J^I\right\} && \text{(by co-additivity of } \gamma).
\end{aligned}$$

It suffices now to verify that $\bigcap_{i\in I}\{d_j^i : j \in J(i)\} = \{\bigwedge_D\{d_{\varphi(i)}^i : i \in I\} : \varphi \in J^I\}$.

($\subseteq$) If $d \in \bigcap_{i\in I}\{d_j^i : j \in J(i)\}$ then there exists $\varphi \in J^I$ such that $d = \bigwedge_D\{d_{\varphi(i)}^i : i \in I\} = \bigwedge_D\{d\} = d$.

($\supseteq$) Let $\varphi \in J^I$ and $d = \bigwedge_D\{d_{\varphi(i)}^i : i \in I\}$. Then, $d \leqslant_D d_{\varphi(i)}^i$ for all $i \in I$, and therefore, by Lemma 3.1 (iii), $d \in \uplus[S_i]$ for all $i \in I$. Thus, $d \in \bigcap_{i\in I} \uplus[S_i]$. □

Thus, the above lemma implies that we have correctly defined an abstract domain. By property (iv) in Section 2.1, the abstraction map $\alpha^* : C \to P(D)$ can be defined as the left adjoint of γ^*: $\alpha^*(c) = \sqcap\{[S] \in P(D) : c \leqslant_C \gamma^*([S])\}$. Notice that $\alpha^*(c)$ is well-defined, since $P(D)$ is a complete lattice. We have therefore shown the following basic result.

Theorem 3.5. *Let C be a completely distributive lattice. If (γ, D, C, α) is a Galois connection then $(\gamma^*, P(D), C, \alpha^*)$ is a Galois insertion.*

We also exploit Lemma 3.4 in order to characterize the lub and glb in the powerset abstract domain as follows.

Proposition 3.6. *For any subset* $\{[S_i]\}_{i\in I} \subseteq P(D)$, $\bigsqcup_{i\in I}[S_i] = [\bigcup_{i\in I} S_i]$ *and* $\bigsqcap_{i\in I}[S_i] = [\{\bigwedge_{i\in I} x_i : \forall i \in I.\, x_i \in S_i\}]$.

Proof. By Proposition 3.2, $\bigsqcup_{i\in I}[S_i] = [\bigcup_{i\in I} \uplus[S_i]]$. Further, $\bigvee_C \gamma(\cup_{i\in I} S_i) = \bigvee_C\{\bigvee_C \gamma(S_i) : i \in I\} = \bigvee_C\{\bigvee_C \gamma(\uplus[S_i]) : i \in I\} = \bigvee_C \gamma(\bigcup_{i\in I} \uplus[S_i])$, and therefore $\bigsqcup_{i\in I}[S_i] = [\bigcup_{i\in I} S_i]$. Consider now, for any $i \in I$, a suitable set of indices $J(i)$ such that $S_i = \{x_j^i : j \in J(i)\}$. Thus, notice that $[\{\bigwedge_{i\in I} x_i : \forall i \in I.\ x_i \in S_i\}] = [\{\bigwedge_{i\in I} x^i_{\varphi(i)} : \varphi \in J^I\}]$. Using co-additivity of γ and complete distributivity of C, we have that $\gamma^*([\{\bigwedge_{i\in I} x^i_{\varphi(i)} : \varphi \in J^I\}]) = \bigvee_{\varphi\in J^I} \gamma(\bigwedge_{i\in I} x^i_{\varphi(i)}) = \bigvee_{\varphi\in J^I} \bigwedge_{i\in I} \gamma(x^i_{\varphi(i)}) = \bigwedge_{i\in I} \bigvee_{j\in J(i)} \gamma(x^i_j) = \bigwedge_{i\in I} \vee\gamma(S_i)$. It is now clear that for any $i \in I$, $\gamma^*([\{\bigwedge_{i\in I} x_i : \forall i \in I.\ x_i \in S_i\}]) \leqslant_C \gamma^*([S_i])$, and hence, since γ^* is an embedding, $[\{\bigwedge_{i\in I} x_i : \forall i \in I.\ x_i \in S_i\}]$ is a lower bound. On the other hand, if $[T] \in P(D)$ is a lower bound, then $\gamma^*([T]) \leqslant_C \bigwedge_{i\in I} \vee\gamma(S_i)$, and therefore, $[T] \sqsubseteq [\{\bigwedge_{i\in I} x_i : \forall i \in I.\ x_i \in S_i\}]$. This concludes the proof. □

In order to clarify the above characterization of the glb on the powerset, consider the example of *Sign* and *P*(*Sign*) in Fig. 1: Then, for instance, we have that $[+,-] \sqcap [0] = [+\wedge 0, -\wedge 0] = [\bot]$.

When the concrete domain C is join-generated by its join-irreducible elements, i.e. for any $c \in C$ there exists $S \subseteq JI(C)$ such that $c = \bigvee_C S$, we can give a very useful characterization for the abstraction map α^*. Notice that any collecting concrete domain, i.e. $\wp(Z)$ (ordered by the subset or superset relation) for some set Z, is join-generated by its join-irreducible elements (e.g., $JI(\langle \wp(Z), \subseteq\rangle) = \{\{z\} \in \wp(Z) : z \in Z\}$). First, we need a standard lattice-theoretic lemma (see, e.g., [2, p. 244]).

Lemma 3.7. (Balbes and Dwinger [2, p. 244]). *Let C be a completely distributive lattice. Then, $x \in JI(C)$ iff for any $S \subseteq C$, $x \leqslant \vee S$ implies $x \leqslant s$ for some $s \in S$.*

Proposition 3.8. *If C is join-generated by $JI(C)$, then, for any $c \in C, \alpha^*(c) = [\{\alpha(x) : x \in JI(C), x \leqslant c\}]$.*

Proof. First, let us show that if $x \in JI(C)$ then $\alpha^*(x) = [\alpha(x)]$. On the one hand, by observing that $x \leqslant_C \gamma(\alpha(x)) = \gamma^*([\alpha(x)])$, we get $\alpha^*(x) \sqsubseteq [\alpha(x)]$. On the other, for all $A \subseteq D$ such that $x \leqslant_C \gamma^*([A]) = \bigvee_C\{\gamma(a) : a \in A\}$, by Lemma 3.7, there exists $a \in A$ such that $x \leqslant_C \gamma(a)$, and therefore such that $\alpha(x) \leqslant_D a$. Then, $\gamma^*([\alpha(x)]) = \gamma(\alpha(x)) \leqslant_C \gamma(a) \leqslant_C \gamma^*([A])$, from which, since γ^* is an embedding, $[\alpha(x)] \sqsubseteq [A]$. Hence, $[\alpha(x)] \sqsubseteq \alpha^*(x)$, and therefore, $\alpha^*(x) = [\alpha(x)]$. Consider now any $c \in C$. By hypothesis, $c = \bigvee_C\{x \in C : x \in JI(C), x \leqslant c\}$. Thus, $\alpha^*(c) =$ (by additivity of α^*) $= \sqcup\{\alpha^*(x) : x \in JI(C), x \leqslant c\} = \sqcup\{[\alpha(x)] : x \in JI(C), x \leqslant c\} =$ (by Proposition 3.6) $= [\{\alpha(x) : x \in JI(C), x \leqslant c\}]$. □

From this result, in particular, we get that for any $c \in JI(C)$, $\alpha^*(c) = [\alpha(c)]$. As a further consequence of the above characterization, for a collecting concrete domain $\langle \wp(Z), \subseteq\rangle$, for some set Z, we have that for any $S \in \wp(Z)$, $\alpha^*(S) = [\{\alpha(\{s\}) : s \in S\}]$.

Filé et al. introduced in [21] the notion of abstract domain *refinement*, formalizing the idea of systematic operators devoted to enhance the precision of representation of abstract domains. A refinement is an operator on abstract domains that improves the precision of abstract domains, and that is monotonic and idempotent (namely, it refines domains all at once). The powerset operator defined above turns out to be a refinement of abstract domains. This fact is precisely stated by the following result.

Proposition 3.9. *If D,E are two abstractions of C then:*
(i) *$P(D)$ is better than D;*
(ii) *If D is better than E then $P(D)$ is better than $P(E)$;*
(iii) *$P(P(D))$ is equivalent to $P(D)$.*

Proof. (i) It is sufficient to verify that $\gamma(D)\subseteq\gamma^*(P(D))$: by definition of γ^*, if $d\in D$ then $\gamma(d)=\gamma^*([d])$.
(ii) By hypothesis, $\gamma_E(E)\subseteq\gamma_D(D)$. Thus, if $[S]\in P(E)$ then there exists $T\subseteq D$ such that $\bigvee_C \gamma_E(S)=\bigvee_C \gamma_D(T)$. Then, $\gamma_E^*([S])=\gamma_D^*([T])$, proving that $\gamma_E^*(P(E))\subseteq\gamma_D^*(P(E))$.
(iii) We have to prove that $\gamma^*(P(D))=\gamma^{**}(P(P(D)))$.
($\subseteq$) For $[S]_\gamma\in P(D)$ and $[\{[S]_\gamma\}]_{\gamma^*}\in P(P(D)), \gamma^{**}([\{[S]_\gamma\}]_{\gamma^*})=\gamma^*([S]_\gamma)$.
($\supseteq$) For $[A]_{\gamma^*}\in P(P(D))$, define $\cup A=\cup\{\uplus[S]_\gamma\subseteq D : [S]_\gamma\in A\}\subseteq D$, and consider $[\cup A]_\gamma\in P(D)$. Thus, $\gamma^*([\cup A]_\gamma)=\bigvee_C\{\bigvee_C \gamma(\uplus[S]_\gamma) : [S]_\gamma\in A\}=$ (by Lemma 3.1 (i)) $=\bigvee_C \gamma^*(A)=\gamma^{**}([A]_{\gamma^*})$. □

By (i) above, Lemma 2.1 and Theorem 3.5, we get that $(\alpha^*\circ\gamma,D,P(D),\alpha\circ\gamma^*)$ is a G.c., and if in addition (γ,D,C,α) is a G.i., then $(\alpha^*\circ\gamma,D,P(D),\alpha\circ\gamma^*)$ is a G.i. We can be more precise about this latter G.i.

Proposition 3.10. *If (γ,D,C,α) is a G.i., then for all $[S]\in P(D)$ and $d\in D$, $\alpha(\gamma^*([S]))=\bigvee_D S$ and $\alpha^*(\gamma(d))=[d]$.*

Proof. By additivity of α, we get $\alpha(\gamma^*([S]))=\alpha(\bigvee_C \gamma(S))=\bigvee_D \alpha(\gamma(S))=\bigvee_D S$. Moreover, $\alpha^*(\gamma(d))=\sqcap\{[T]\in P(D) : \gamma(d)\leqslant_C\gamma^*([T])\}$. Observe that $\gamma(d)\leqslant_C\gamma^*([T])$ is equivalent to $[d]\sqsubseteq[T]$ (as γ^* is an embedding), from which $\alpha^*(\gamma(d))=[d]$. □

In particular, by the observation following Lemma 2.1, we get that (γ,D,C,α) is the composition of $(\lambda d.[d],D,P(D),\lambda[S].\bigvee_D S)$ and $(\gamma^*,P(D),C,\alpha^*)$.

It should be clear that whenever the concretization map of (γ,D,C,α) is additive, the abstract domain D is already able to represent precisely, by means of its lub, the concrete disjunction of its elements. Thus, in this case, the powerset of D is equivalent to D. The next result shows that absence of additivity of the concretization map is a necessary and sufficient condition in order to get a strict improvement of precision by powerset.

Proposition 3.11. *Assume that (γ,D,C,α) is a G.i. Then, $P(D)$ is strictly better than D iff γ is not additive.*

Proof. By definition, $P(D)$ is strictly better than $D \Leftrightarrow \gamma(D) \subset \gamma^*(P(D)) \Leftrightarrow \exists [S] \in P(D). \gamma^*([S]) \notin \gamma(D) \Leftrightarrow$ (as γ is an embedding) $\exists S \subseteq D. \bigvee_C \gamma(S) <_C \gamma(\vee_D S) \Leftrightarrow$ (as γ is monotonic) γ is not additive. □

Corollary 3.12. *If (γ, D, C, α) is a G.i., then D and $P(D)$ are equivalent iff γ is additive.*

The following are alternative ways of expressing the condition of additivity for the concretization map.

Proposition 3.13. *If (γ, D, C, α) is a G.i. then the following are equivalent:*
(i) *γ is additive,*
(ii) *$\gamma(D)$ is a complete sublattice of C,*
(iii) *For any $[S] \in P(D), \vee_D S \in \uplus[S]$.*

Proof. (i) $\Rightarrow$ (ii): Assume $T = \gamma(S)$ for some $S \subseteq D$. Thus, $\bigvee_C T = \bigvee_C \gamma(S) = \gamma(\bigvee_D S) \in \gamma(D)$. Thus, $\gamma(D)$ is a complete sublattice of C, since $\gamma(D)$ is always closed under glb's.
(ii) $\Rightarrow$ (iii): For any $[S] \in P(D)$, there exists $d \in D$ such that $\bigvee_C \gamma(S) = \gamma(d)$. Hence, $d \in \uplus[S]$, and, by additivity of α, we get $\bigvee_D S = \alpha(\bigvee_C \gamma(S)) = d \in \uplus[S]$.
(iii) $\Rightarrow$ (i): If $S \subseteq D$, then, by using Lemma 3.1 (ii), $\gamma(\bigvee_D S) \geqslant_C \bigvee_C \gamma(S) \geqslant_C \gamma(\bigvee_D S)$. □

We say that a value in a powerset $P(D)$ is *new*, if it is able to represent precisely a concrete denotation otherwise not representable precisely in D. Hence, $[S] \in P(D)$ is new iff $\bigvee_C \gamma(S) <_C \gamma(\bigvee_D S)$.

Following their seminal work [14], Cousot and Cousot first proposed in [15] a general definition of the powerset operator in a generic setting where abstract domains are specified by Galois connections. Successively, and independently of the conference version [22] of this paper, Cousot and Cousot introduced in [17] various disjunctive completions of an abstract domain, all defined by some form of powerset. They exploited their definitions in order to present the new powerful comportment analysis for higher-order functional languages. More specifically, they introduced the standard disjunctive completion, the order ideal completion and the anti-chain completion (this latter completion is isomorphic to a further Scott closed ideal completion, cf. [17]). These definitions can be quickly summarized as follows. For subsets of a domain D abstracting a completely distributive lattice C, Cousot and Cousot consider the standard lower powerdomain pre-ordering relation (cf. [26]): If $X, Y \subseteq D$, then $X \sqsubseteq^{\vee} Y \Leftrightarrow \forall x \in X. \exists y \in Y. x \leqslant_D y$. Then, one can define the equivalence relation $X \cong^{\vee} Y \Leftrightarrow X \sqsubseteq^{\vee} Y \;\&\; Y \sqsubseteq^{\vee} X$.

– The disjunctive completion of D is defined as the quotient of $\wp(D)$ with respect to the equivalence relation $\cong^{\vee}$. It is a complete lattice with respect to the lower powerdomain ordering: $[X]_{\cong^{\vee}} \preceq [Y]_{\cong^{\vee}} \Leftrightarrow X \sqsubseteq^{\vee} Y$.

- The order ideal completion of D consists of all the downward closed subsets of D. It is a complete lattice with respect to the lower powerdomain ordering $\subseteq^{\vee}$ above.
- The anti-chain completion of D consists of all subsets S of D such that $S = max(S)$. Also in this case, the complete ordering relation is given by $\subseteq^{\vee}$.

The concretization function (for all the definitions above) is obviously the same as that given in this paper: For any element S of the completion, the concretization is given by $\bigvee_C \gamma(S)$. Cousot and Cousot showed that the disjunctive and order-ideal completions are equivalent, and for abstract domains satisfying the ACC, they are in turn equivalent to the anti-chain completion. It is easy to verify that also our powerset domain is equivalent to those above.

Proposition 3.14. *$P(D)$ is equivalent to each of the above completions.*

Proof. It is enough to observe that $\{\bigvee_C \gamma(S) : S \subseteq D\}$ is the image by the concretization map of each completion, and it coincides with $\gamma^*(P(D))$. $\square$

Although from a semantic viewpoint our powerset definition is equivalent to those in [17], they differ from a synctatic perspective. In particular, our definition of the ordering relation on the powerset abstract domain is more natural, since it directly relies on the set-inclusion relation. Moreover, as we will see in the next section, our use of fat sets as canonical representatives permits to lift, by an explicit direct definition, the abstract operations to the powerset domain. This is particularly relevant as far as implementation details are concerned. It is also worth mentioning that all these definitions of the powerset abstract domain have been generalized by Giacobazzi and Ranzato in [24], who give a general, but implicit, definition of the disjunctive completion by using closure operators, which only requires that the concrete domain is a mere complete lattice. On the other hand, Giacobazzi and Ranzato show how to supply an explicit powerset-like definition of the disjunctive completion, under the hypothesis of complete distributivity for the concrete domain.

4. Lifting abstract operations to the powerset

Let us suppose that (γ, D, C, α) is a G.c., where C is a completely distributive lattice, and that $o_C : C \times X \to C$ is a concrete operation approximated by $o_D : D \times X \to D$. We want to define an operation on the powerset abstract domain $P(D)$, such that it extends o_D and that still approximates o_C. Let us consider the following definition:

$$o_D^* : P(D) \times X \to P(D)$$

$$o_D^*([S], x) = [\{o_D(d, x) \in D : d \in \uplus[S]\}].$$

The definition of this operation is the most natural one since it directly relies on that of o_D: It consists in first applying o_D to the elements of the canonical representative, and then take the equivalence class of the obtained subset of D. It is worthwhile to observe that, whenever D is finite and o_D is finitely computable, o_D^* is also finitely computable.

Proposition 4.1. *o_D^* is monotonic.*

Proof. If $[S] \sqsubseteq [T]$ and $x \in X$ then, by definition of $\sqsubseteq$, we have $\{o_D(d,x) : d \in \uplus[S]\} \subseteq \{o_D(d,x) : d \in \uplus[T]\}$, and therefore $o_D^*([S],x) \sqsubseteq o_D^*([T],x)$. □

In general, it is not true that o_D^* approximates o_C. An example showing this fact is given below.

Example 4.2. Let us consider the concrete domain $\langle \wp(\mathbb{Z}), \subseteq \rangle$, the abstract domain *Sign* of Fig. 1, and its powerset *P*(*Sign*), also in Fig. 1. Consider this concrete operation $sq : \wp(\mathbb{Z}) \to \wp(\mathbb{Z})$:

$$sq(A) = \begin{cases} \{a^2 : a \in A\} & \text{if } 0 \notin A \text{ or } A = \{0\} \\ \mathbb{Z} & \text{otherwise.} \end{cases}$$

Thus, sq is the square operation on the sets of integers not containing zero and on $\{0\}$, while maps the remaining sets to the top $\mathbb{Z}$. It is simple to check that sq is monotonic, and that it is abstractly approximated by the (monotonic) operation sq_a on *Sign* defined as follows:

$$sq_a = \{\bot \mapsto \bot,\ + \mapsto +,\ 0 \mapsto 0,\ - \mapsto +,\ \top \mapsto \top\}.$$

The operation sq_a^* on the powerset domain *P*(*Sign*) is therefore defined as follows:

$$sq_a^*([\bot]) = [\bot],\ sq_a^*([+]) = sq_a^*([-]) = sq_a^*([+,-]) = [+],$$

$$sq_a^*([0]) = [0],\ sq_a^*([+,0]) = sq_a^*([-,0]) = [+,0],\ sq_a^*([\top]) = [\top].$$

In spite of the fact that sq_a is an approximation of sq, its extension sq_a^* to the powerset does not approximate sq. In fact, considering $[-,0] \in P(Sign)$, we get

$$\begin{aligned} sq(\gamma^*([-,0])) &= sq(\{a \in \mathbb{Z} : a \leqslant 0\}) = \mathbb{Z} \\ &\not\subseteq \gamma^*(sq_a^*([-,0])) = \gamma^*([+,0]) = \{a \in \mathbb{Z} : a \geqslant 0\}. \end{aligned}$$ □

The following condition guarantees the correctness of o_D^*.

Proposition 4.3. *If o_C preserves lub's of the subsets $\gamma(S)$, for all $S \subseteq D$, i.e. for any $x \in X$, $o_C(\bigvee_C\{\gamma(d) : d \in S\},x) = \bigvee_C\{o_C(\gamma(d),x) : d \in S\}$, then o_D^* is an approximation of o_C.*

Proof. Let $[S] \in P(D)$ and $x \in X$. Then,

$$\begin{aligned}
o_C(\gamma^*([S]),x) &= o_C(\gamma^*([\uplus[S]]),x) && \text{(by Lemma 3.1 (i))}\\
&= \textstyle\bigvee_C\{o_C(\gamma(d),x) : d \in \uplus[S]\} && \text{(by hypothesis on } o_C\text{)}\\
&\leqslant \textstyle\bigvee_C\{\gamma(o_D(d,x)) : d \in \uplus[S]\} && \text{(by correctness of } o_D\text{)}\\
&= \gamma^*(o_D^*([S],x)). \quad \square
\end{aligned}$$

Clearly, if o_C is (fully) additive then it also verifies the condition in Proposition 4.3, and thus, in such a case, o_D^* approximates o_C. Even though this condition of additivity may seem restrictive, we will show later that it is trivially satisfied by the standard concrete operations used in a typical logic program abstract interpretation framework. Additivity of abstract operations is also considered and discussed in [36], with particular emphasis to additive abstract operations used in functional program analysis.

As an important example, let us consider the systematic lifting of the glb operation from D to $P(D)$. Notice that this is possible since the hypothesis of complete distributivity of the concrete domain implies that the concrete glb is additive. As expected, it turns out that the glb of $P(D)$, that we explicitly defined in the previous section, coincides with this systematically defined operation.

Proposition 4.4. *The lifted glb $\wedge^*$ on $P(D)$ actually is the glb $\sqcap$.*

Proof. Assume that $\{[S_i]\}_{i\in I} \subseteq P(D)$. Also, for any $i \in I$, we consider a suitable set of indices $K(i)$ such that $\uplus[S_i] = \{x_k^i : k \in K(i)\}$. Observe that, by definition, $\bigwedge_{i\in I}^*[S_i] = [\{\bigwedge_{i\in I} y_{\varphi(i)}^i : \varphi \in K^I\}]$. Therefore, by co-additivity of γ and complete distributivity of C, we have that $\gamma^*(\bigwedge_{i\in I}^*[S_i]) = \bigvee_{\varphi\in K^I} \gamma(\bigwedge_{i\in I} y_{\varphi(i)}^i) = \bigvee_{\varphi\in K^I} \bigwedge_{i\in I} \gamma(y_{\varphi(i)}^i) = \bigwedge_{i\in I} \bigvee_{k\in K(i)} \gamma(y_k^i) = \bigwedge_{i\in I} \vee\{\gamma(s) : s \in \uplus[S_i]\}$. Thus, by Lemma 3.1(i) and co-additivity of γ^*, $\gamma^*(\bigwedge_{i\in I}^*[S_i]) = \bigwedge_{i\in I} \gamma^*([S_i]) = \gamma^*(\bigsqcap_{i\in I}[S_i])$. This, because γ^* is 1–1, concludes the proof. $\square$

We can be more precise about the relationship between the glb of $P(D)$ and that of D. In fact, whenever D is completely distributive, the glb of D results to be complete with respect to the glb of $P(D)$, where the Galois insertion $(\lambda d.[d], D, P(D), \lambda[S].\bigvee_D S)$ of D in $P(D)$ is that given by Proposition 3.10.

Proposition 4.5. *If D is completely distributive then the glb $\wedge$ on D is complete for the glb $\sqcap$ on $P(D)$ (w.r.t. the G.i. $(\lambda d.[d], D, P(D), \lambda[S].\vee_D S)$).*

Proof. Consider any $\{[S_i]\}_{i\in I} \subseteq P(D)$. We have to show that $\vee(\bigsqcap_{i\in I}[S_i]) = \bigwedge_{i\in I}(\vee S_i)$. For any $i \in I$, we consider a suitable set of indices $J(i)$ such that $S_i = \{x_j^i : j \in J(i)\}$. Note that, by Proposition 3.6, we then have $\bigsqcap_{i\in I}[S_i] = [\{\bigwedge_{i\in I} x_{\varphi(i)}^i : \varphi \in J^I\}]$. Thus, by complete distributivity, we have that $\bigwedge_{i\in I}(\vee S_i) = \bigvee_{\varphi\in J^I} \bigwedge_{i\in I} x_{\varphi(i)}^i$, and therefore this concludes the proof. $\square$

Notice that in the above proposition, if the glb is considered as a finitary operation, then it suffices that D is (finitely) distributive. Further, it is worth noting that, as observed in Section 2.2, an analogous result for the lub trivially holds.

We next investigate the relationship existing between o_D and o_D^*. We already know that $P(D)$ is better than D. Therefore, it should not come as a surprise that o_D^* is an extension of o_D and that it is better than o_D.

Proposition 4.6. *Assume that (γ, D, C, α) is a G.i. (and that the hypotheses of Proposition* 4.3 *hold). Then,*

(i) *o_D^* extends o_D, i.e. $\forall d \in D. \forall x \in X.\ o_D^*([d],x) = [o_D(d,x)]$;*

(ii) *o_D^* is better than o_D.*

Proof. (i) The following chain of equalities holds:

$$\begin{aligned}
\gamma^*(o_D^*([d],x)) &= \textstyle\bigvee_C\{\gamma(o_D(d',x)) : d' \in \uplus[\{d\}]\} \\
&= \textstyle\bigvee_C\{\gamma(o_D(d',x)) : \gamma(d') \leqslant_C \gamma(d)\} && \text{(by Lemma 3.1 (ii))} \\
&= \textstyle\bigvee_C\{\gamma(o_D(d',x)) : d' \leqslant_D d\} && \text{(as } \gamma \text{ is an embedding)} \\
&= \gamma^*([o_D(d,x)]) && \text{(by monotonicity of } o_D \text{ and } \gamma\text{).}
\end{aligned}$$

Since γ^* is 1–1, the thesis follows.

(ii) If $c \in C$ then $\alpha^*(c) \sqsubseteq [\alpha(c)]$. Thus, applying γ^*, we get $\gamma^*(\alpha^*(c)) \leqslant_C \gamma(\alpha(c))$. We want then to show that, for any $x \in X$, $\gamma^*(o_D^*(\alpha^*(c),x)) \leqslant_C \gamma(o_D(\alpha(c),x))$:

$$\begin{aligned}
\gamma^*(o_D^*(\alpha^*(c),x)) &= \textstyle\bigvee_C\{\gamma(o_D(d,x)) : d \in \uplus\alpha^*(c)\} \\
&= \textstyle\bigvee_C\{\gamma(o_D(d,x)) : \gamma(d) \leqslant_C \gamma^*(\alpha^*(c))\} && \text{(by Lemma 3.1 (ii))} \\
&\leqslant_C \textstyle\bigvee_C\{\gamma(o_D(d,x)) : \gamma(d) \leqslant_C \gamma(\alpha(c))\} && \text{(by the above observation)} \\
&= \textstyle\bigvee_C\{\gamma(o_D(d,x)) : d \leqslant_D \alpha(c)\} && \text{(as } \gamma \text{ is an embedding)} \\
&= \gamma(o_D(\alpha(c),x)) && \text{(by monotonicity of } o_D \text{ and } \gamma\text{),}
\end{aligned}$$

and this concludes the proof. □

It is also straightforward to verify that (ii) above implies that o_D approximates o_D^* w.r.t. the G.i. $(\lambda d.[d], D, P(D), \lambda[S].\vee_D S)$.

We denote by $P(\mathscr{D}) = \langle P(D), o_1^*, \ldots, o_k^* \rangle$ the abstract interpretation obtained by applying the powerset operator to $\mathscr{D} = \langle D, o_1, \ldots, o_k \rangle$. The following theorem summarizes some of the results achieved so far.

Theorem 4.7. *Let $\mathscr{C} = \langle C, o_C^1, \ldots, o_C^k\rangle$ be a concrete interpretation, $\mathscr{D} = \langle D, o_1, \ldots, o_k\rangle$ abstracting $\mathscr{C}$, and $P(\mathscr{D}) = \langle P(D), o_1^*, \ldots, o_k^*\rangle$ be the powerset of $\mathscr{D}$. If C is completely distributive and each o_i^C preserves lub's of the subsets $\gamma(S)$, for all $S \subseteq D$, then:*
(i) *$P(\mathscr{D})$ abstracts $\mathscr{C}$;*
(ii) *If (γ, D, C, α) is a G.i., then $P(\mathscr{D})$ is better than $\mathscr{D}$.*

In general, the systematic lifting of an abstract operation to the powerset does not preserve either the property of being complete or that of being the best correct approximation, as the following example shows.

Example 4.8. Let us consider again the abstract domain *Sign* and its powerset $P(\mathit{Sign})$, depicted in Fig. 1. As concrete operation $f : \wp(\mathbb{Z}) \to \wp(\mathbb{Z})$ let us consider $f = \lambda X.\{x \cdot z \in \mathbb{Z} : x \in X,\ z \in \mathbb{Z},\ z \neq 0\}$, i.e. $f(X)$ is the pointwise multiplication of X with the set of integers different from 0. It is then easy to see that the best correct approximations f_{Sign} and $f_{P(\mathit{Sign})}$ of f on *Sign* and $P(\mathit{Sign})$ are, respectively, defined as follows:

$$f_{\mathit{Sign}} = \{\bot \mapsto \bot,\ + \mapsto \top,\ 0 \mapsto 0,\ - \mapsto \top,\ \top \mapsto \top\};$$

$$f_{P(\mathit{Sign})} = \{[\bot] \mapsto [\bot],\ [+] \mapsto [+,-],\ [0] \mapsto [0],\ [-] \mapsto [+,-],$$
$$[+,0] \mapsto [\top],\ [+,-] \mapsto [+,-],\ [-,0] \mapsto [\top],\ [\top] \mapsto [\top]\}.$$

Also, it is not too hard to verify that both f_{Sign} and $f_{P(\mathit{Sign})}$ are complete for f, i.e. for all $X \in \wp(\mathbb{Z})$, $\alpha(f(X)) = f_{\mathit{Sign}}(\alpha(X))$ and $\alpha^*(f(X)) = f_{P(\mathit{Sign})}(\alpha^*(X))$. Notice that f is additive, and therefore, by Proposition 4.3, one can correctly consider the systematic lifting f_{Sign}^* of f_{Sign} to the powerset $P(\mathit{Sign})$. Although f_{Sign} is the best correct approximation of f, it turns out that f_{Sign}^* is not. In fact, we have that $f_{\mathit{Sign}}^*([+]) = [f_{\mathit{Sign}}(+)] = [\top]$, whilst for the best correct approximation $f_{P(\mathit{Sign})}$ we get a strictly precise result, that is, $f_{P(\mathit{Sign})}([+]) = [+,-]$. Moreover, this also proves that f_{Sign}^* is not complete for f (cf. Section 2.2). □

We now show that, under certain conditions, the definition of the lifting to the powerset of an abstract operation can be suitably modified, still remaining systematic, so that this step does preserve the property of being complete.

Lemma 4.9. *If C is join-generated by $JI(C)$ and D is join-generated by $\alpha(JI(C))$, then for any $[S] \in P(D)$ there exists $S^\diamond \subseteq \alpha(JI(C))$ such that $[S] = [S^\diamond]$.*

Proof. Let $[S] \in P(D)$ and $S^- = S \setminus \alpha(JI(C))$. For any $x \in S^-$, let $A_x = \{d \in \alpha(JI(C)) : d \leqslant_D x\}$, and observe that, by hypothesis, $x = \bigvee_D A_x$. Then, let $S^\diamond = (S \cap \alpha(JI(C))) \cup (\bigcup_{x \in S^-} A_x)$. We demonstrate that $[S] = [S^\diamond]$. To show this, it is enough to verify that for any $x \in S^-$, $\bigvee_C\{\gamma(d) : d \in A_x\} = \gamma(x)$. The inequality $\bigvee_C\{\gamma(d) : d \in A_x\} \leqslant_C \gamma(x)$ trivially holds. On the other hand, note that if $z \in JI(C)$ and $z \leqslant_C \gamma(x)$, then $\alpha(z) \leqslant_D x$, and therefore $\alpha(z) \in A_x$. Hence, $z \leqslant_C \gamma(\alpha(z)) \leqslant \bigvee_C\{\gamma(d) : d \in A_x\}$. Since, by hypothesis, $\gamma(x) = \bigvee_C\{z \in JI(C) : z \leqslant_C \gamma(x)\}$, we get that $\gamma(x) \leqslant_C \bigvee_C\{\gamma(d) : d \in A_x\}$. □

Thus, under the hypotheses of the above lemma, any $[S] \in P(D)$ can be transformed in an equivalent $[S^\diamond] \in P(D)$ such that any element $x \in S^\diamond$ is the image, via the abstraction map, of a concrete join-irreducible point. For instance, the G.c. $(\gamma, \mathit{Sign}, \wp(\mathbb{Z}), \alpha)$ evidently satisfies such hypotheses: Hence, for $[\top] \in P(\mathit{Sign})$ we get that $\{\top\}^\diamond = \{+,0,-\}$, and $[\top] = [+,0,-]$, where $+$, 0, and $-$ are all the image of a singleton, i.e. a join-irreducible element, in $\wp(\mathbb{Z})$.

Given an abstract operation $o_D : D \times X \to D$, we exploit the above lemma for defining the following operation $o_D^\natural : P(D) \times X \to P(D)$ on the powerset abstract domain $P(D)$: For all $[S] \in P(D)$ and $x \in X$,

$$o_D^\natural([S],x) = [\{o_D(t,x) \in D : t \in (\uplus[S])^\diamond\}].$$

In other terms, we consider $(\uplus[S])^\diamond$ as canonical representative of an equivalence class $[S] \in P(D)$, and we apply pointwise o_D to each element of $(\uplus[S])^\diamond$. It is straightforward to verify by a simple inspection of the proof of Proposition 4.3, and under its hypotheses, that $o_D^\natural$ is a correct approximation of o_C. Here, the interesting point is that when o_D is complete for o_C and o_C preserves any join-irreducible element (i.e., if $c \in JI(C)$ then, for all $x \in X$, $o_C(c,x) \in JI(C)$), then $o_D^\natural$ on $P(D)$ is still complete for o_C, and its definition can be simplified by substituting $\uplus[S]$ with S.

Proposition 4.10. *Let C be join-generated by $JI(C)$, D be join-generated by $\alpha(JI(C))$, o_C be additive and preserving join-irreducible elements. If o_D is complete for o_C then $o_D^\natural$ is complete for o_C as well, and, for all $[S] \in P(D)$ and $x \in X$, $o_D^\natural([S],x) = [\{o_D(t,x) : t \in S^\diamond\}]$.*

Proof. Let us fix a generic parameter $x \in X$. In the proof, we will use the definition $o_D^\natural([S],x) = [\{o_D(t,x) : t \in S^\diamond\}]$, and we will demonstrate at the end of the proof that this is correct. We first show that for a join-irreducible $h \in JI(C)$, $\alpha^*(o_C(h,x)) = o_D^\natural(\alpha^*(h),x)$. By hypothesis, $o_C(h,x) \in JI(C)$, and so, by Proposition 3.8, $\alpha^*(o_C(h,x)) = [\alpha(o_C(h,x))]$. Thus, by completeness, $\alpha^*(o_C(h,x)) = [o_D(\alpha(h),x)]$. Moreover, again by Proposition 3.8, $\alpha^*(h) = [\alpha(h)]$. Therefore, since $\{\alpha(h)\}^\diamond = \{\alpha(h)\}$, $o_D^\natural(\alpha^*(h),x) = [o_D(\alpha(h),x)] = \alpha^*(o_C(h,x))$. Now, let $c \in C$. By hypothesis, $c = \bigvee_C H_c$, where $H_c = \{h \in JI(C) : h \leqslant_C c\}$. Then, by additivity of o_C and α^* and by the above observations, $\alpha^*(o_C(c,x)) = \bigsqcup_{h \in H_c} \alpha^*(o_C(h,x)) = \bigsqcup_{h \in H_c} o_D^\natural(\alpha^*(h),x) = \bigsqcup_{h \in H_c} [o_D(\alpha(h),x)]$. Thus, by Proposition 3.6, $\alpha^*(o_C(c,x)) = [\{o_D(\alpha(h),x) : h \in H_c\}]$. Observe now that, by Proposition 3.8, $\alpha^*(c) = [\{\alpha(h) : h \in H_c\}]$, and, trivially, $(\alpha(H_c))^\diamond = \alpha(H_c)$. Thus, $o_D^\natural(\alpha^*(c),c) = [\{o_D(\alpha(h),x) : h \in H_c\}] = \alpha^*(o_C(c,x))$. To conclude, let us observe that if $[R] = [S]$ then $[\{o_D(t,x) : t \in R^\diamond\}] = [\{o_D(t,x) : t \in S^\diamond\}] = [\{o_D(t,x) : t \in (\uplus[S])^\diamond\}]$: In fact, there exists some $c \in C$ such that $\alpha^*(c) = [R]$, and therefore, by what just proved above, these three equivalence classes are all equal to $\alpha^*(o_C(c,x))$. $\square$

We will exemplify this result in the next section, by applying it for the abstract operation of existential quantification used in ground-dependency analysis of logic languages over an abstract domain of propositional formulae.

5. Powerset of the logic program abstract interpretation *POS*

POS (cf. [1, 10, 31, 32]) is a well-known abstract interpretation for ground-dependency analysis of logic languages, whose underlying abstract domain *Pos* consists of certain propositional formulae. The groundness information results to be useful to a Prolog compiler for a number of relevant optimizations (see, e.g., [27, 37]). Minor variants of this abstract interpretation have been also used for a variety of other analyses, such as suspension analysis of concurrent logic programs [6]. In this section, we apply the powerset operator to *POS*. We show that *P(POS)* is a strictly better interpretation than *POS*, although by abstracting back to *Pos* the outcome of an analysis performed with *P(POS)* one gets the corresponding analysis for *POS*.

5.1. *The concrete interpretation LP*

We briefly recall the standard concrete domain and operations used in an abstract interpretation framework for logic program analysis (for more details, see e.g. [10]).

Assume that Var is an infinite set of variables, and $IVar \subseteq Var$ is an infinite denumerable subset of variables of interest. For any syntactic object o, $var(o)$ denotes the set of variables occurring in o. A substitution σ (over Var and a fixed alphabet of constant and function symbols) is denoted by its set of nontrivial bindings, i.e. $\sigma = \{x/\sigma(x) : \sigma(x) \neq x\}$. The domain of definition of σ is $dom(\sigma) = \{x \in Var : \sigma(x) \neq x\}$, while its variable range is $rng(\sigma) = \cup\{var(\sigma(x)) : x \in dom(\sigma)\}$. The composition of substitutions σ and θ is denoted by $\sigma\theta$. The empty substitution is denoted by ε. If $W \subseteq IVar$ is any subset of variables of interest and σ is a substitution, then $\sigma_{/W}$ is any substitution obtained rom σ by projecting variables of σ over W (i.e., by renaming variables in $dom(\sigma) \cup rng(\sigma)$ belonging to $IVar \setminus W$ with variables of noninterest). If σ is a substitution and E is any syntactic entity, then $E\sigma$ stands for the result of applying σ to E. The set of idempotent substitutions over Var is denoted by Sub. If $\sigma \in Sub$ then $eqn(\sigma)$ denotes the corresponding set of term equations in solved form (the correspondence between idempotent substitutions and sets of syntactic equations in solved form is well-known, cf. [30]). Over Sub the usual relation $\trianglelefteq$ of instantiation is defined as follows: If $\sigma_1, \sigma_2 \in Sub$ then $\sigma_1 \trianglelefteq \sigma_2$ iff there exists a (possibly nonidempotent) substitution θ such that $\sigma_1 = \sigma_2\theta$. For any set E of term equations, $mgu(E)$ is defined as follows: If E is not unifiable then $mgu(E) = \emptyset$, otherwise $mgu(E) = \{\sigma\}$, for an arbitrary idempotent most general unifier $\sigma \in Sub$ of E (recall that all the idempotent mgu's of E are equivalent up to renaming, cf. [30]).

The concrete domain of interpretation is given by the standard collecting domain $\langle \wp(Sub), \subseteq \rangle$. We follow [10] in defining the following concrete operation of unification:

$$\mathbf{u} : \wp(Sub) \times \wp(Sub) \longrightarrow \wp(Sub)$$

$$\mathbf{u}(\Sigma, \Theta) = \bigcup_{\sigma \in \Sigma, \theta \in \Theta} mgu(eqn(\sigma) \cup eqn(\theta)).$$

This operation of unification **u** is general enough to subsume other forms of unification and composition of substitutions used in most logic program analysis frameworks (see [10] for more details and examples). For instance, it is simple to observe that if $\sigma \in Sub$ is a calling substitution and θ is an idempotent mgu of an equation between atoms $H = B$ then $mgu(eqn(\sigma) \cup eqn(\theta)) = \{\sigma\psi : \psi \in mgu(H\sigma = B\sigma)\}$, up to renaming. The operation of projection over a set of variables of interest is defined as follows:

$$\pi : \wp(Sub) \times \wp(IVar) \to \wp(Sub)$$

$$\pi(\Sigma, W) = \{\sigma_{/W} : \sigma \in \Sigma\}.$$

The last trivial concrete operation is given by the union of sets of idempotent substitutions,[2] $\cup : \wp(Sub) \times \wp(Sub) \to \wp(Sub)$, namely the lub operation of the concrete domain. The concrete interpretation for logic programs is then given by $LP = \langle \wp(Sub), \mathbf{u}, \cup, \pi \rangle$. It is immediate to note that all the concrete operations of LP are additive functions, due to the collecting nature of the concrete domain. We will make use of this obvious observation later on.

5.2. *The abstract interpretation POS*

Let us succinctly recall the definitions of the abstract domains *Mon*, *Def* and *Pos*, and of their abstract operations. For more details see [1, 10, 31].

Let $VI \subseteq IVar$ be a finite (nonempty) subset of variables of interest. In order to fix the notation, suppose that $VI = \{x_1, \dots, x_n\}$. Assume also that $B = \{false, true\}$ is the two point lattice. A Boolean function on VI is any function $f : B^n \to B$ (where the i-th component of B^n corresponds to the variable x_i). Obviously, Boolean functions can be represented by means of propositional formulae: If F is the propositional formula over VI representing the Boolean function f, then $f(b_1, \dots, b_n) = true$ iff $\{x_i \in VI : b_i = true\}$ is a truth-assignment (namely, the set of true logical variables) which is a model of F. On the other hand, each class of logically equivalent propositional formulae over VI determines a Boolean function. Thus, in the following, we will use without distinction Boolean functions and propositional formulae as equivalent concepts. Any subset $M \subseteq VI$ is considered as a truth-assignment for a Boolean function f, and $mod(f)$ denotes the set of truth-assignments which are models of f (it is then a subset of $\wp(VI)$). The set *Bool* of Boolean functions on VI forms a (finite) Boolean lattice for the standard ordering: $f_1 \preceq f_2$ iff $mod(f_1) \subseteq mod(f_2)$. The lub and glb are given by logical disjunction and conjunction: For two Boolean functions f_1 and f_2, they are denoted by $f_1 \vee f_2$ and $f_1 \wedge f_2$, respectively (obviously, for the corresponding propositional formulae, this corresponds to consider their syntactic disjunction and conjunction). With a slight abuse of notation, *false* and *true* denote the bottom and top elements of *Bool*. A Boolean function $f \in Bool$ is *monotone* if $mod(f)$ is upward closed (that is,

[2] Obviously, considering union as a binary operator is sufficient to get finite union.

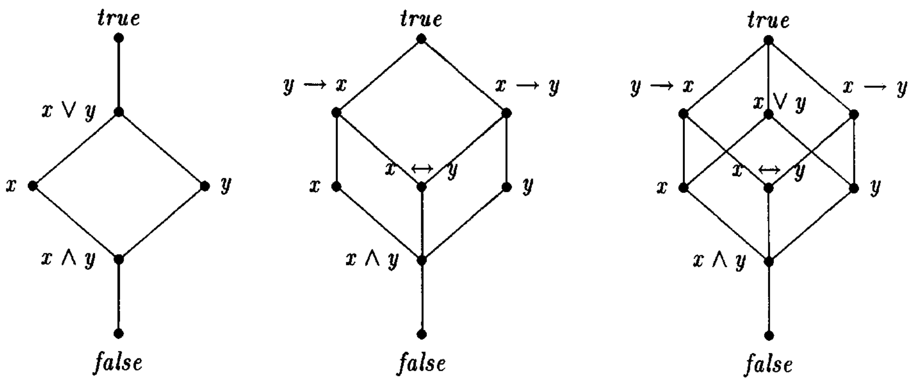


Fig. 2. *Mon*, *Def* and *Pos* for $VI = \{x, y\}$.

for any $M \subseteq N \subseteq VI$, if $M \in mod(f)$ then $N \in mod(f)$). We denote by *Mon* the set of monotone Boolean functions. It is well-known that a propositional formula $f \in Bool$ is monotone iff f is equivalent either to *false*, or to *true*, or to a formula built using only the connectives $\vee$ and $\wedge$ (see, e.g., [38]). It is immediate to observe that $\langle Mon, \preceq \rangle$ is a sublattice of $\langle Bool, \preceq \rangle$. A Boolean function $f \in Bool$ is *positive* if $VI \in mod(f)$ (or, equivalently, if $f(true, \ldots, true) = true$). The set of positive Boolean functions is denoted by *Pos*. It is shown in [8, Corollary 1, p. 325] that a propositional formula $f \in Bool$ is positive iff f is equivalent to a formula built using only the connectives $\wedge$, $\vee$ and $\leftrightarrow$; successively, this synctatic characterization has been sharpened in [1, Theorem 3.1], where it is shown that $f \in Bool$ is positive iff f is equivalent to a formula built using only the connectives $\wedge$ and $\rightarrow$. It is easy to note that *Pos* is a sublattice of *Bool*, and that *Pos* is distributive. A Boolean function $f \in Bool$ is *definite* if $f \in Pos$ and $mod(f)$ is closed under intersection (i.e., if $M, N \in mod(f)$ then $M \cap N \in mod(f)$). As observed by Armstrong et al. [1], from [18, Proposition 3.1] one gets that a propositional formula f is definite iff f is equivalent to a conjunction of formulae having the shape $x_{i_1} \wedge \cdots \wedge x_{i_k} \rightarrow x_j$, where k may be 0 (in this case, the formula is simply x_j). The subset of *Pos* given by the definite formulae is denoted by *Def*. It turns out that *Def* is merely a meet subsemilattice of *Pos* (and hence of *Bool*). This means that the glb of *Def* is given by logical conjunction, while its lub can be defined in the usual way in terms of the glb: $f_1 \vee_{Def} f_2 = \wedge\{f \in Def : f_1 \preceq f,\ f_2 \preceq f\}$. Note that *Mon* is not a subset of *Def* and *Pos*, because $false \notin Pos$. However, since the element *false* is useful in order to represent precisely the empty set of substitutions (and therefore to represent the information of reachability), as it is common practice, we add *false* both to *Def* and *Pos*, and we still use *Def* and *Pos* to denote these lattices. The lattices *Mon*, *Def* and *Pos* are depicted in Fig. 2 for the case of two variables of interest $VI = \{x, y\}$.

Let us now recall the Galois insertion of *Pos* into the concrete domain $\wp(Sub)$. For a substitution $\sigma \in Sub$, the truth-assignment specifying which variables of interest are bound by σ to ground terms and which are not, is given by $gr_\sigma = \{x \in VI : var(\sigma(x)) =$

$\emptyset\}$. On the other hand, the propositional formula expressing the ground-dependencies in σ of the variables of interest is given by[3] $dep_\sigma = \exists_{\overline{VI}}. \wedge \{x \leftrightarrow \wedge var(\sigma(x)) : x \in dom(\sigma)\}$. Note that $dep_\varepsilon = true$. The abstraction $\alpha : \wp(Sub) \to Pos$ and concretization $\gamma : Pos \to \wp(Sub)$ mappings are then defined as follows:

$$\alpha(\Sigma) = \vee\{dep_\sigma : \sigma \in \Sigma\},$$
$$\gamma(f) = \{\sigma \in Sub : \forall \sigma' \trianglelefteq \sigma.\ gr_{\sigma'} \in mod(f)\}.$$

Note that $\alpha(\emptyset) = false$, for any $\sigma \in Sub$, $\alpha(\{\sigma\}) = dep_\sigma \in Def$, and $\gamma(false) = \emptyset$. These two mappings form the Galois insertion $(\gamma, Pos, \wp(Sub), \alpha)$. The restriction of γ to Mon and Def also yields the right adjoint of a Galois insertion.

Example 5.1. Assume that $VI = \{x, y, z, v\}$. The formula $x \wedge (y \leftrightarrow z)$ is an element of Pos (and Def) representing all the substitutions σ such that for any instance σ' of σ the following conditions hold: (i) $\sigma'(x)$ is ground, (ii) $\sigma'(y)$ is ground iff $\sigma'(z)$ is ground. Thus,[4] $\sigma = \{x/a, y/b, z/c, w/d\}$ and $\theta = \{x/a, y/w, z/w\}$ satisfy these conditions, and therefore $\{\sigma, \theta\} \subseteq \gamma(x \wedge (y \leftrightarrow z))$. On the other hand, we have that $\alpha(\{\sigma\}) = x \wedge y \wedge z$ and $\alpha(\{\theta\}) = x \wedge (y \leftrightarrow z)$. $\square$

Let us now recall the abstract operations defined on Pos. Abstract unification is simply given by logical conjunction $\wedge : Pos \times Pos \to Pos$. On the other hand, logical disjunction $\vee : Pos \times Pos \to Pos$ is the obvious abstract operation corresponding to the concrete union of sets of substitutions. Finally, the concrete operation of projection is simulated in Pos by existential quantification as follows:

$$\pi_P : Pos \times \wp(IVar) \to Pos$$

$$\pi_P(f, W) = \exists_{VI \setminus W}.f.$$

Hence, $\exists_{VI \setminus W}.f$ projects away the variables of interest which are not in W. As noted by Armstrong et al. [1, Theorem 3.2], this operation is well-defined (i.e., $\exists_{VI \setminus W}.f \in Pos$).

Thus, the full abstract interpretation is given by $POS = \langle Pos, \wedge, \vee, \pi_P \rangle$. These abstract operations of Pos are correct approximations of the corresponding concrete operations. Actually, as it is shown in [10], we can be more precise, since it turns out that $\wedge$ is the best correct approximation of the concrete unification $\mathbf{u}$ [10, Theorem 5.7], π_P is complete for π [10, Lemma 6.3], and $\vee$ is trivially complete for the union of sets of substitutions (see Section 2.2).

5.3. The powerset abstract domain P(Pos)

Since the concrete domain $\wp(Sub)$ is collecting (and therefore completely distributive), we can correctly apply the construction in Section 3 of the powerset abstract

[3] $\exists_{\overline{VI}}$ is the existential quantification over the variables of noninterest (i.e. those in $\overline{VI} = Var \setminus VI$).

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

domain to *Pos*. By Proposition 3.5, we get the Galois insertion $(\gamma^*, P(Pos), \wp(Sub), \alpha^*)$. Note that, by Proposition 3.8, the abstraction in $P(Pos)$ of a set of substitutions $\Sigma \in \wp(Sub)$ is simply given by the set of the abstractions in *Pos* of every substitution in Σ, i.e. $\alpha^*(\Sigma) = [\{dep_\sigma : \sigma \in \Sigma\}]$. It is also worth noting that by Proposition 3.3, the lifting of *Pos* to $P(Pos)$ preserves the property of distributivity of the domain.

Obviously, by Proposition 3.9 (i), we know that $P(Pos)$ is better than *Pos*. More precisely, we know that $(\lambda f.[f], Pos, P(Pos), \lambda[S].\vee S)$ is a G.i. On the other hand, we now show that $P(Pos)$ actually is strictly better than *Pos*. To this end, by Proposition 3.11, it suffices to find two formulae of *Pos* such that γ does not preserve their logical disjunction. The following example shows this phenomenon.

Example 5.2. Assume that the set *VI* of variables of interest contains at least two variables x, y, and consider $x \vee (x \rightarrow y) \in Pos$. We want to verify that $\gamma(x) \cup \gamma(x \rightarrow y) \subset \gamma(x \vee (x \rightarrow y))$. Let $\sigma = \{x/v\}$, where v is any other variable (possibly not in *VI*). Obviously, $\sigma \notin \gamma(x)$, because x is not ground for σ, and $\sigma \notin \gamma(x \rightarrow y)$, because considering the instance $\sigma' = \{x/a, v/a\}$ of σ we have that $gr_{\sigma'} \notin mod(x \rightarrow y)$. On the other hand, note that the logical disjunction $x \vee (x \rightarrow y)$ is logically equivalent to *true*. Then, $\gamma(x \vee (x \rightarrow y)) = Sub$, and therefore $\sigma \in \gamma(x \vee (x \rightarrow y))$. □

As an immediate consequence of the above example, we get the following result.

Theorem 5.3. *$P(Pos)$ is strictly better than Pos.*

This result is somehow in opposition to the intuition. In fact, since the lub of *Pos* is logical disjunction, it is natural to think that the meaning of a disjunction $f \vee g$ of two formulae of *Pos* is exactly given by the logical disjunction of the meanings of f and g, namely by the union of their concretizations. However, the above result shows that this is not the case. We now give a characterization of the subsets of formulae of *Pos* whose logical disjunction is preserved by the concretization. This characterization makes use of the definite formulae of *Def*. First, we need a preliminary lemma.

Lemma 5.4. *For any $f \in Def$, there exists $\sigma_f \in Sub$ such that $dep_{\sigma_f} = f$.*

Proof. We prove indirectly this fact, by using other known results. In [7, Theorem 8.4.2], it is shown that there exists a Galois insertion of *Def* into the well-known Jacobs and Langen abstract domain *Sharing* [28]. Thus, for any $f \in Def$ there exists $a_f \in Sharing$ such that $\alpha_{Sharing,Def}(a_f) = f$. On the other hand, as observed in [28], for any $a \in Sharing$ there exists $\sigma_a \in Sub$ such that $\alpha_{\wp(Sub),Sharing}(\{\sigma_a\}) = a$. Thus, consider the substitution $\sigma_{a_f} \in Sub$. In [7, Theorem 8.4.5], it is shown that for any $\sigma \in Sub$, $dep_\sigma = \alpha_{Sharing,Def}(\alpha_{\wp(Sub),Sharing}(\{\sigma\}))$. Thus, $dep_{\sigma_{a_f}} = f$. □

Theorem 5.5. *Let $\Phi \subseteq Pos$ with $\Phi \neq \emptyset$. Then, $\cup\{\gamma(f) : f \in \Phi\} = \gamma(\vee\Phi)$ if and only if $\forall g \in Def. (g \preceq \vee\Phi) \Rightarrow (\exists f \in \Phi. g \preceq f)$.*

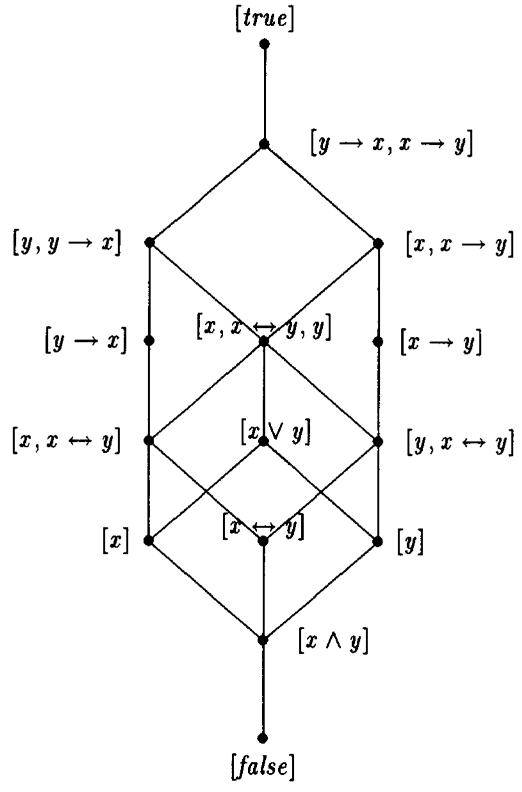


Fig. 3. $P(Pos)$ for $VI = \{x, y\}$.

Proof. (*if*) By monotonicity of γ, it is sufficient to verify that $\gamma(\vee\Phi) \subseteq \cup\{\gamma(f) : f \in \Phi\}$. Consider $\sigma \in \gamma(\vee\Phi)$. Thus, $\alpha(\{\sigma\}) = dep_\sigma \preceq \vee\Phi$ and $dep_\sigma \in Def$. By hypothesis, there exists $f \in \Phi$ such that $dep_\sigma \preceq f$. Thus, $\gamma(dep_\sigma) \subseteq \gamma(f)$, and, since $\sigma \in \gamma(dep_\sigma)$, we get $\sigma \in \gamma(f)$.

(*only if*) Let $g \in Def$ such that $g \preceq \vee\Phi$. By Lemma 5.4, there exists $\sigma_g \in Sub$ such that $\alpha(\{\sigma_g\}) = g$. Therefore, $\alpha(\{\sigma_g\}) \preceq \vee\Phi$, implies $\sigma_g \in \gamma(\vee\Phi) = \cup\{\gamma(f) \mid f \in \Phi\}$. Thus, we get that there exists $f \in \Phi$ such that $\sigma_g \in \gamma(f)$. We conclude by observing that $\alpha(\{\sigma_g\}) = g \preceq f$. $\square$

By using this characterization of Theorem 5.5, it is simple to verify that the powerset domain $P(Pos)$ for $VI = \{x, y\}$ is the lattice depicted in Fig. 3. In fact, it is easy to detect directly from the Hasse diagrams of *Def* and *Pos* in Fig. 2, that $[x, x \leftrightarrow y], [y, x \leftrightarrow y], [x, x \leftrightarrow y, y], [y, y \to x], [x, x \to y], [y \to x, x \to y]$ are all and only the new elements of $P(Pos)$. For instance, by Theorem 5.5, we have that $\gamma(x) \cup \gamma(x \to y) \subset \gamma(x \vee (x \to y)) = \gamma(true)$, because $y \to x \preceq true$ but $y \to x \not\preceq x$ and $y \to x \not\preceq x \to y$.

The following result proves that γ preserves the logical disjunction of monotone formulae.

Proposition 5.6. *If $\Phi \subseteq Mon$ then* $\sqcup\{\gamma(f) : f \in \Phi\} = \gamma(\vee\Phi)$.

Proof. Obvious if $\Phi = \emptyset$. Thus, assume that $\Phi \neq \emptyset$. Let us suppose by contradiction that $\sqcup\{\gamma(f) : f \in \Phi\} \subset \gamma(\vee\Phi)$. Then, by Theorem 5.5, there exists $g \in Def$ such that $mod(g) \subseteq mod(\vee\Phi) = \bigcup_{f\in\Phi} mod(f)$ and for any $f \in \Phi$, $mod(g) \nsubseteq mod(f)$. Thus, for any $f \in \Phi$, there exists $M_f \in mod(g) \setminus mod(f)$. Since Φ is nonempty, by definiteness of g, we get that $\bigcap_{f\in\Phi} M_f \in mod(g)$. Hence, there exists $h \in \Phi$ such that $\bigcap_{f\in\Phi} M_f \in mod(h)$. Observe now that, since h is monotone, $\bigcap_{f\in\Phi} M_f \subseteq M_h$ implies that $M_h \in mod(h)$, which is a contradiction. □

The converse of the above proposition does not hold. It is enough to consider the formulae x, which is monotone, and $y \to x$, which is not: since $x \preceq y \to x$, γ trivially preserves their disjunction. The following result shows that the abstraction in *Pos* of all the ground substitutions yields precisely all the monotone formulae. In this sense, monotone formulae are only able to express plain groundness and no information of ground-dependency between variables. This fact also provides an intuitive justification to Proposition 5.6.

Proposition 5.7. *If $Sub^{GR} = \{\sigma \in Sub : rng(\sigma) = \emptyset\}$ is the subset of Sub of the ground substitutions, then $\alpha(\wp(Sub^{GR})) = Mon$.*

Proof. ($\subseteq$) Note that by definition of α, for any $\sigma \in Sub^{GR}$, $\alpha(\{\sigma\}) = \wedge(dom(\sigma) \cap VI)$, and hence $\alpha(\{\sigma\}) \in Mon$ (in particular, $\alpha(\{\varepsilon\}) = true$). Thus, if $\Sigma \in \wp(Sub^{GR})$ then, by definition of α and since *Mon* is closed by logical disjunction, we get that $\alpha(\Sigma) \in Mon$.

($\supseteq$) If $f \in Mon$, then it is possible to transform f in an equivalent formula in disjunctive normal form (see, e.g., [38]). Thus, we can assume that $f = \bigvee_{j\in J}(\bigwedge_{i\in I} x_{j_i})$, for some finite J and I (where $x_{j_i} \in VI$). For every $j \in J$, consider the substitution $\sigma_j = \{x_{j_i}/a : i \in I\} \in Sub^{GR}$. Then, we have that $\alpha(\{\sigma_j : j \in J\}) = \bigvee_{j\in J} \alpha(\{\sigma_j\}) = \bigvee_{j\in J}(\bigwedge_{i\in I} x_{j_i}) = f$. □

5.4. *Abstract operations on P(Pos)*

Let us now turn to lifting abstract operations from *Pos* to *P(Pos)*. As far as the lub is concerned, we noticed in Section 2.2 that the lub of any abstract domain is always complete with respect to the concrete one. Thus, in view of this simple observation, for the powerset *P(Pos)* we will consider its lub $\sqcup$, which is characterized by Proposition 3.6, as approximating the concrete union of sets of substitutions.

Next, let us see how to lift the existential quantification π_P from *Pos* to *P(Pos)*. We recalled above that π_P is complete for the corresponding concrete operation of projection π (cf. [10, Lemma 6.3]). Thus, we can hope to apply Proposition 4.10, so that to maintain this desirable property by lifting π_P to the powerset. We show that indeed this is the case. In fact, first note that the hypotheses of Lemma 4.9 are satisfied by the G.c. of *Pos* into $\wp(Sub)$: The concrete domain $\wp(Sub)$ is collecting, and therefore it is join-generated by its join-irreducible elements, namely by the singletons $\{\sigma\}$ with $\sigma \in \mathit{Sub}$; moreover, by Lemma 5.4, $\alpha(JI(\wp(\mathit{Sub}))) = \mathit{Def}$, and therefore Pos is join-generated by $\alpha(JI(\wp(\mathit{Sub})))$. Hence, for every $[S] \in P(\mathit{Pos})$ there exists $[S^\diamond] \in P(\mathit{Pos})$ such that $S^\diamond \subseteq \mathit{Def}$ and $[S] = [S^\diamond]$. Such $S^\diamond$ can be obtained as in the proof of Lemma 4.9: If $f \in S$ and $f \in \mathit{Pos} \setminus \mathit{Def}$, then f is substituted with the set of formulae of Def less than f itself. Furthermore, the concrete projection π trivially preserves join-irreducible elements, i.e. singletons. Thus, we can apply Proposition 4.10 in defining $\pi_P^\natural$ over $P(\mathit{Pos})$ as follows: For all $[S] \in P(\mathit{Pos})$ and $W \in \wp(\mathit{IVar})$, $\pi_P^\natural([S], W) = [\{\pi_P(f, W) : f \in S^\diamond\}]$.

Finally, by Proposition 4.4, lifting the logical conjunction of Pos to $P(\mathit{Pos})$ actually gives rise to the glb $\sqcap$ of $P(\mathit{Pos})$.

The following result shows the precision of these abstract operations on $P(\mathit{Pos})$ just defined.

Proposition 5.8. *The abstract operations* $\sqcup$ *and* $\pi_P^\natural$ *are complete for the corresponding concrete operations* $\cup$ *and* π, *while* $\sqcap$ *is the best correct approximation of* $\mathbf{u}$.

Proof. As recalled above, $\sqcup$ is trivially complete for $\cup$. Also, by the observations above and by Proposition 4.10, we get that $\pi_P^\natural$ is complete for π. Let us now turn to the glb $\sqcap$. In order to show that $\sqcap$ is the best correct approximation of $\mathbf{u}$, we have to prove that for any $[R], [S] \in P(\mathit{Pos})$, $\alpha^*(\mathbf{u}(\gamma^*([R]), \gamma^*([S]))) = [R] \sqcap [S]$. Let us first show the following particular case: For all $f, g \in \mathit{Def}$, $\alpha^*(\mathbf{u}(\gamma^*([f]), \gamma^*([g]))) = [f] \sqcap [g]$, i.e. $\alpha^*(\mathbf{u}(\gamma(f), \gamma(g))) = [f \wedge g]$. To prove this, we need to note that by exploiting the proof of [10, Theorem 5.7], it is not too hard to demonstrate the following fact: If $f, g \in \mathit{Def}$ and $f \wedge g \neq \mathit{false}$ then there exist $\sigma_f \in \gamma(f)$ and $\theta_g \in \gamma(g)$ such that $\alpha(\mathbf{u}(\{\sigma_f\}, \{\theta_g\})) = f \wedge g$ (this fact strongly relies on the hypothesis that both f and g are formulae of Def). We will refer to this observation by (†). If $f \wedge g = \mathit{false}$ then either $f = \mathit{false}$ or $g = \mathit{false}$. Hence, either $\gamma(f)$ or $\gamma(g)$ is the empty set, and so $\alpha^*(\mathbf{u}(\gamma(f), \gamma(g))) = \alpha^*(\emptyset) = [\mathit{false}] = [f \wedge g]$. Thus, let us assume that $f \wedge g \neq \mathit{false}$. By Proposition 3.8, $\alpha^*(\mathbf{u}(\gamma(f), \gamma(g))) = [\{\alpha(\mathbf{u}(\{\sigma\}, \{\theta\})) : \sigma \in \gamma(f),\ \theta \in \gamma(g)\}]$. Moreover, observe that for any $\sigma \in \gamma(f)$ and $\theta \in \gamma(g)$, by monotonicity of $\mathbf{u}$ and α, $\alpha(\mathbf{u}(\{\sigma\}, \{\theta\})) \preceq \alpha(\mathbf{u}(\gamma(f), \gamma(g)))$, and therefore from $f \wedge g = \alpha(\mathbf{u}(\gamma(f), \gamma(g)))$ (viz., $\wedge$ is the best correct approximation on Pos of $\mathbf{u}$, cf. [10, Theorem 5.7]), we get that $\alpha(\mathbf{u}(\{\sigma\}, \{\theta\})) \preceq f \wedge g$. Thus, by (†), we get $\alpha^*(\mathbf{u}(\gamma(f), \gamma(g))) = [f \wedge g]$. Let us now turn to the general case. As already observed above, by Lemma 4.9, there exist $[R^\diamond], [S^\diamond] \in P(\mathit{Pos})$ such that $[R] = [R^\diamond]$ and $[S] = [S^\diamond]$, and $R^\diamond, S^\diamond \subseteq \mathit{Def}$. Thus, w.l.o.g., let $R, S \subseteq \mathit{Def}$. Hence, the following equalities hold:

$$\begin{aligned}
\alpha^*(\mathbf{u}(\gamma^*([R]), \gamma^*([S]))) &= \bigsqcup_{f \in R, g \in S} \alpha^*(\mathbf{u}(\gamma(f), \gamma(g))) && \text{(by additivity of } \mathbf{u} \text{ and } \alpha^*) \\
&= \bigsqcup_{f \in R, g \in S} [f \wedge g] && \text{(by what proved above)}
\end{aligned}$$

$$= [\{f \wedge g : f \in R,\ g \in S\}] \quad \text{(by Proposition 3.6)}$$
$$= [R] \sqcap [S] \quad \text{(by Proposition 3.6),}$$

and this closes the proof. □

To conclude, let us observe that $\sqcap$ is not complete for $\mathbf{u}$: In fact, by considering $\sigma = \{x/a\}$ and $\theta = \{x/b\}$, we get $\alpha^*(\mathbf{u}(\{\sigma\},\{\theta\})) = \alpha^*(\emptyset) = [\mathit{false}]$, whereas $\alpha^*(\{\sigma\}) \sqcap \alpha^*(\{\theta\}) = [x] \sqcap [x] = [x]$.

5.5. Comparing POS and P(POS)

Let us first show that all the operations of $P(POS) = \langle Pos, \sqcap, \sqcup, \pi_P^\natural \rangle$ are strictly better than the corresponding ones of *POS*.

Theorem 5.9. *P(POS) is strictly better than POS.*

Proof. Theorem 5.3 showed that $P(Pos)$ is strictly better than Pos. Let us consider here the abstract operations. First, consider the lub operations of Pos and $P(Pos)$. According to the definitions in Section 2.3, we have to show that there exist $\Sigma_1, \Sigma_2 \in \wp(Sub)$ such that $\gamma^*(\alpha^*(\Sigma_1) \sqcup \alpha^*(\Sigma_2)) \subset \gamma(\alpha(\Sigma_1) \vee \alpha(\Sigma_2))$. Consider $\sigma_1 = \{x/a\}$ and $\sigma_2 = \{x/f(v,w), y/v\}$, where $x, y \in VI$ and $v, w \notin VI$, such that $\alpha^*(\{\sigma_1\}) = [\alpha(\{\sigma_1\})] = [x]$ and $\alpha^*(\{\sigma_2\}) = [\alpha(\{\sigma_2\})] = [x \to y]$. In this case, Example 5.2 directly shows the thesis. Let us now turn to the glb. Consider $\sigma_3 = \{y/a\}$ and $\sigma_4 = \{x/v, y/f(v,w)\}$. Then, $\alpha^*(\{\sigma_1,\sigma_2\}) = [x, x \to y]$, $\alpha^*(\{\sigma_3,\sigma_4\}) = [y, y \to x]$, while $\alpha(\{\sigma_1,\sigma_2\}) = \mathit{true} = \alpha(\{\sigma_3,\sigma_4\})$. Moreover, by Proposition 3.6, $[x, x \to y] \sqcap [y, y \to x] = [x, x \leftrightarrow y, y]$. Then, analogous to Example 5.2, it is immediate to show the thesis. Finally, consider the abstract projections. Consider $\sigma_5 = \{x/f(y,z)\}$, where $z \in VI$. Then, $\alpha^*(\{\sigma_1,\sigma_5\}) = [x, x \leftrightarrow (y \wedge z)]$ and $\alpha(\{\sigma_1,\sigma_5\}) = x \vee (x \leftrightarrow (y \wedge z))$. Thus, $\pi_P^\natural([x, x \leftrightarrow (y \wedge z)], \{x,y\}) = [\exists_z.x, \exists_z.x \leftrightarrow (y \wedge z)] = [x, x \to y]$, whereas $\pi_P(x \vee (x \leftrightarrow (y \wedge z)), \{x,y\}) = \exists_z.x \vee (x \leftrightarrow (y \wedge z)) = \mathit{true}$. Hence, as before, we conclude that $\pi_P^\natural$ is strictly better than π_P. □

We can go more in depth about the relationship between $P(POS)$ and POS. In fact, it turns out that the operations of POS are complete for the corresponding operations of $P(POS)$, where Pos and $P(Pos)$ are related by the Galois insertion $(\lambda f.[f], Pos, P(Pos), \lambda[S].\vee S)$, as recalled at the beginning of Section 5.3.

Proposition 5.10. $\wedge, \vee$, *and* π_P *are complete, respectively, for* $\sqcap, \sqcup$ *and* $\pi_P^\natural$.

Proof. Completeness for $\wedge$ follows by Proposition 4.5, since Pos is distributive. Completeness for the lub $\vee$ always holds (see Section 2.2). As far as the projections are concerned, we have to show that for any $[S] \in P(Pos)$, and $W \subseteq IVar$, $\vee\pi_P^\natural([S], W) = \pi_P(\vee S, W)$. This is a consequence of the fact that the existential quantification is additive on all formulae of *Bool* (see, e.g., [38]): $\vee\pi_P^\natural([S], W) = \bigvee_{f \in S^\diamond} \exists_{VI \setminus W}.f = \exists_{VI \setminus W}. \vee S^\diamond = \exists_{VI \setminus W}. \vee S = \pi_P(\vee S, W)$. □

We now present an example where analysing a logic program using $P(POS)$ we get an output which is strictly better than the corresponding one for POS. We follow the approach outlined in [1, 31], where the abstract semantics of a logic program is obtained by means of a simple computation of a least fixpoint. This abstract semantics is the well-known abstraction of the declarative s-semantics (cf. [20]) characterizing the computed answer substitutions. We do not present all the details of the computation of the abstract semantics, and we refer the reader to [1, 31] for a full exposition.

Example 5.11. Let us consider the following logic program P :

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

This program allows us to illustrate the role played by all the abstract operations in the computation of the abstract semantics. As an intermediate step, let us compute the Clark completion of P, which is as follows:

$$p(x,y) \leftrightarrow (x=y) \vee (x=a) \vee \exists z.(y=a \wedge p(x,z) \wedge p(a,x))$$

Then, one considers the mgu (if any) of each constraint appearing in the completion. Thus, by abstracting in *Pos* these mgu's (if unification fails we abstract to *false*), we get the following recursive definition for the Boolean function p :

$$\begin{aligned} p(x,y) &= (x \leftrightarrow y) \vee x \vee \pi_P(y \wedge p(x,z) \wedge p(true,x), \{x,y\}) \\ &= (y \rightarrow x) \vee \pi_P(y \wedge p(x,z) \wedge p(true,x), \{x,y\}) \end{aligned}$$

Using Kleene iteration starting from the bottom element *false*, we get the following sequence:

$$\begin{aligned} p^0(x,y) &= false \\ p^1(x,y) &= (y \rightarrow x) \vee false = y \rightarrow x \\ p^2(x,y) &= (y \rightarrow x) \vee \exists_z.(y \wedge (z \rightarrow x) \wedge (x \rightarrow true)) \\ &= (y \rightarrow x) \vee y = true \qquad \text{(least fixpoint)} \end{aligned}$$

Thus, an analyzer performing the analysis of P with the domain *Pos* yields the top propositional formula *true*. This means that we get no ground-dependency information. Notice that this lack of ground-dependency information for P is due to the fact that the two logical disjunctions $(x \leftrightarrow y) \vee x$ and $(y \rightarrow x) \vee y$ are not disjunctions in the concrete sense, namely they do not faithfully model the union of the corresponding sets of substitutions. In contrast, by using the powerset abstract domain $P(Pos)$, the analyzer is able to mimic in a precise way the union of sets of substitutions by using the lub of $P(Pos)$. In fact, for $P(Pos)$ and using its abstract operations that we defined above, we get the following recursive definition for $p(x,y) \in P(Pos)$:

$$\begin{aligned} p(x,y) &= [x \leftrightarrow y] \sqcup [x] \sqcup \pi_P^\natural([y] \sqcap p(x,z) \sqcap p(true,x), \{x,y\}) \\ &= [\{x \leftrightarrow y, x\}] \sqcup \pi_P^\natural([y] \sqcap p(x,z) \sqcap p(true,x), \{x,y\}) \end{aligned}$$

In this case, starting from the bottom element [*false*] of $P(Pos)$, the Kleene iteration is as follows.

$$\begin{aligned} p^0(x,y) &= [false] \\ p^1(x,y) &= [x \leftrightarrow y, x] \sqcup \pi_P^\natural([false], \{x,y\}) = [x \leftrightarrow y, x] \\ p^2(x,y) &= [x \leftrightarrow y, x] \sqcup [\exists_z.y \wedge (x \leftrightarrow z) \wedge (true \leftrightarrow x), \exists_z.y \wedge (x \leftrightarrow z) \wedge true, \\ &\qquad \exists_z.y \wedge x \wedge (true \leftrightarrow x), \exists_z.y \wedge x \wedge true] \\ &= [x \leftrightarrow y, x, x \wedge y, y, x \wedge y, x \wedge y] \\ &= [x \leftrightarrow y, x, y] \\ p^3(x,y) &= [x \leftrightarrow y, x] \sqcup [\exists_z.y \wedge (x \leftrightarrow z) \wedge (true \leftrightarrow x), \exists_z.y \wedge (x \leftrightarrow z) \wedge true, \\ &\qquad \exists_z.y \wedge (x \leftrightarrow z) \wedge x, \exists_z.y \wedge x \wedge (true \leftrightarrow x), \exists_z.y \wedge x \wedge true, \\ &\qquad \exists_z.y \wedge x \wedge x, \exists_z.y \wedge z \wedge (true \leftrightarrow x), \exists_z.y \wedge z \wedge true, \exists_z.y \wedge z \wedge x] \\ &= [x \leftrightarrow y, x, x \wedge y, y, x \wedge y, x \wedge y, x \wedge y, x \wedge y, x \wedge y, y, x \wedge y] \\ &= [x \leftrightarrow y, x, y] \quad \text{(least fixpoint)} \end{aligned}$$

Thus, using $P(Pos)$ we are able to infer that in each computed answer substitution for the predicate p, either its first argument is ground or its second argument is ground or they are equivalent (namely, they are bound to the same variables). □

In the previous example, we can observe that abstracting in *Pos* the abstract semantics of the program P obtained for $P(Pos)$, we get exactly the abstract semantics of P for *Pos*. In fact, the logical disjunction of the formulae in $[x \leftrightarrow y, x, y]$ coincides with *true*. In other terms, this means that there is completeness between the abstract semantics for *POS* and $P(POS)$. It turns out that this relationship of completeness always holds, whenever the abstract semantics is the standard bottom-up abstraction (cf. [3, 5, 32]) of the denotational s-semantics. Let us introduce the following notation: For any logic program P, $[\![P]\!]_D$ denotes the abstract bottom-up s-semantics of P instantiated to the abstract domain D (and corresponding abstract operations).

Proposition 5.12. *For any program P, $[\![P]\!]_{Pos} = \vee[\![P]\!]_{P(Pos)}$.*

Proof. We do not consider the details of a particular definition of an abstract bottom-up semantics, but we reason on their general pattern of definition. W.l.o.g., we can suppose that for an abstract interpretation $\mathscr{D} = \langle D, \bigwedge_D, \bigvee_D, \pi_D \rangle$ abstracting LP, for any program P, $[\![P]\!]_D = lfp(T_P^D)$, where $T_P^D : D \longrightarrow D$ is the approximation of the immediate consequence operator of s-semantics induced by $\mathscr{D}$ on D (cf. [3, 5]). Every step of T_P^D

typically consists of applying a (finite) number of the abstract operations of $\mathscr{D}$. Thus, the completeness result of Proposition 5.10 implies that T_P^{Pos} is complete for $T_P^{P(Pos)}$ (w.r.t. the usual G.i. $(\lambda f.[f], Pos, P(Pos), \lambda[S].\vee S)$. As recalled in Section 2.2, it is well-known that if f^a is complete for f then $\alpha(lfp(f)) = lfp(f^a)$. Hence, we get the thesis, namely $[\![P]\!]_{Pos} = \vee[\![P]\!]_{P(Pos)}$. $\square$

We can draw the following consequence of the above result: Using $P(POS)$ instead of POS for analysing logic programs, one cannot gain plain ground-dependency Pos-information, but possibly only disjunctive ground-dependency information, i.e. the information represented precisely by the new elements of $P(Pos)$ and that the base abstract domain Pos is not able to represent with no loss of precision.

Although the above results set a limit to what can be achieved by using the powerset abstract domain $P(Pos)$, it is worthwhile to remark that, in general, it is not possible to recover $[\![P]\!]_{P(Pos)}$ from $[\![P]\!]_{Pos}$. For instance, if for some P the analyzer yields the answer $[\![P]\!]_{Pos} = true$, for $P(Pos)$ we could have the answers $[\![P]\!]_{P(Pos)} = [x, y, x \leftrightarrow y], [z, z \rightarrow y], [w \rightarrow x, x \rightarrow w]$, etc., namely the corresponding analysis of P with $P(Pos)$ might well be strictly more precise.

6. Conclusion

This paper proposed a general study of the powerset refinement operator on abstract interpretations. This operator, given an abstract interpretation $\mathscr{D} = \langle D, o_1, \ldots, o_k \rangle$, produces a new full abstract interpretation $P(\mathscr{D}) = \langle P(D), o_1^*, \ldots, o_k^* \rangle$, i.e., it defines both a new refined powerset domain $P(D)$ which is able to represent in the best possible way the concrete disjunction, and new conveniently defined abstract operations o_i^* for it. We have given conditions guaranteeing the correctness of $P(\mathscr{D})$, and we have studied the relationship, as far as precision is concerned, between $\mathscr{D}$ and $P(\mathscr{D})$. The general theory is applied to the well-known abstract interpretation POS (whose abstract domain Pos consists of certain propositional formulae) for ground-dependency analysis of logic languages. We obtained a strict improvement by lifting POS to its powerset $P(POS)$. This is somehow an unexpected result, since the abstract domain Pos is already closed by logical disjunction of formulae. We have also characterized precisely the relationship between the abstract semantics using POS and $P(POS)$, by showing the existence of a form of completeness between them.

We have not addressed the complexity issue of the powerset operator, since our main aim was to study the powerset operator from a theoretical perspective, considering those aspects related to the precision. It is clear that a static program analysis based on a powerset abstract interpretation $P(\mathscr{D})$ will be more costly than one based on $\mathscr{D}$, being, in general, the size of the powerset domain $P(D)$ exponential with respect to the size of D. It could be interesting to study practically the trade-off between the loss in efficiency and the gain in precision for some specific abstract interpretations. However, it is worth mentioning that abstract interpretation does not apply only to program analysis, but it is also useful in many other fields. For instance, abstract interpretation is a valuable tool in comparative semantics, i.e. for studying the relationship occurring between semantics of programming languages at different levels of abstraction (cf. [11, 12, 16]). In this context, where obviously complexity issues are much less important, we believe that the powerset operator can be successfully applied in order to systematically derive many well-known collecting semantics by powerset of some base semantics. This might be particularly useful in order to simplify their semantic definitions, given that the results of [24] allow to characterize the optimal (i.e. the simplest) base semantics. A similar application for logic program semantics has been given in [23], where complementation [7], i.e. the inverse operation to reduced product, has been exploited in order to systematically derive new semantic definitions.

Acknowledgements

We are grateful to Roberto Giacobazzi and Harald Søndergaard for many valuable suggestions and discussions. This research was partly supported by *Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo* of Italian CNR under grant no. 93.01603 PF69.

References

[1] T. Armstrong, K. Marriott, P. Schachte, H. Søndergaard, Two classes of Boolean functions for dependency analysis, Sci. Comput. Program., to appear. A preliminary version appeared in Proc. SAS '94, Lecture Notes in Computer Science, vol. 864, Springer, Berlin, 1994.

[2] R. Balbes, P. Dwinger, Distributive Lattices, University of Missouri Press, Columbia, Missouri, 1974.

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

[4] G.L. Burn, C.L. Hankin, S. Abramsky, Strictness analysis for higher-order functions, Sci. Comput. Program. 7 (1986) 249–278.

[5] M. Codish, D. Dams, E. Yardeni, Bottom-up abstract interpretation of logic programs, Theoret. Comput. Sci. 124(1) (1994) 93–126.

[6] M. Codish, M. Falaschi, K. Marriott, Suspension analyses for concurrent logic programs, ACM Trans. Program. Lang. Syst. 16(3) (1994) 649–686.

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

[8] A. Cortesi, G. Filé, W. Winsborough, Prop revisited: propositional formula as abstract domain for groundness analysis, in: Proc. 6th IEEE Symp. on Logic in Computer Science (LICS '91), IEEE Computer Society Press, Los Alamitos, CA, 1991, pp. 322–327.

[9] A. Cortesi, G. Filé, W. Winsborough, Comparison of abstract interpretations, in: W. Kuich (Ed), Proc. 19th Internat. Colloq. on Automata, Languages and Programming (ICALP '92), Lecture Notes in Computer Science, vol. 623, Springer, Berlin, 1992, pp. 521–532.

[10] A. Cortesi, G. Filé, W. Winsborough, Optimal groundness analysis using propositional logic, J. Logic Program. 27(2) (1996) 137–167.

[11] P. Cousot, Abstract interpretation, ACM Comput. Surveys 28(2) (1996) 324–328.

[12] P. Cousot, Types as abstract interpretations (Invited Paper), in Conf. Record of the 24th ACM Symp. on Principles of Programming Languages (POPL '97), ACM Press, New York, 1997, pp. 316–331.

[13] P. Cousot, R. Cousot, Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints, in: Conf. Record of the 4th ACM Symp. on Principles of Programming Languages (POPL '77), ACM Press, New York, 1977, pp. 238–252.
[14] P. Cousot, R. Cousot, Systematic design of program analysis frameworks, in: Conf. Record of the 6th ACM Symp. on Principles of Programming Languages (POPL '79), ACM Press, New York, 1979, pp. 269–282.
[15] P. Cousot, R. Cousot, Abstract interpretation and application to logic programs, J. Logic Program. 13(2-3) (1992) 103–179.
[16] P. Cousot, R. Cousot, Inductive definitions, semantics and abstract interpretation, in: Conf. Record of the 19th ACM Symp. on Principles of Programming Languages (POPL '92), ACM Press, New York, 1992, pp. 83–94.
[17] P. Cousot, R. Cousot, Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and PER analysis of functional languages) (Invited Paper), in: Proc. IEEE Internat. Conf. on Computer Languages (ICCL '94), IEEE Computer Society Press, Los Alamitos, CA, 1994, pp. 95–112.
[18] P. Dart, On derived dependencies and connected databases, J. Logic Program. 11(2) (1991) 163–188.
[19] B.A. Davey, H.A. Priestley, Introduction to Lattices and Order, Cambridge University Press, Cambridge, UK, 1990.
[20] M. Falaschi, G. Levi, M. Martelli, C. Palamidessi, Declarative modeling of the operational behavior of logic languages, Theoret. Comput. Sci. 69(3) (1989) 289–318.
[21] G. Filé, R. Giacobazzi, F. Ranzato, A unifying view of abstract domain design, ACM Comput. Surveys 28(2) (1996) 333–336.
[22] G. Filé, F. Ranzato, Improving abstract interpretations by systematic lifting to the powerset, in: M. Bruynooghe (Ed.), Proc. Internat. Logic Programming Symp. (ILPS '94), The MIT Press, Cambridge, MA, 1994, 655–669.
[23] R. Giacobazzi, F. Ranzato, Complementing logic program semantics, in: M. Hanus, M. Rodríguez-Artalejo (Eds.), Proc. 5th Internat. Conf. on Algebraic and Logic Programming (ALP '96), Lecture Notes in Computer Science, vol. 1139, Springer, Berlin, 1996, pp. 238–253.
[24] R. Giacobazzi, F. Ranzato, Optimal domains for disjunctive abstract interpretation, Sci. Comput. Program., 1998, to appear.
[25] G. Gierz, K.H. Hofmann, K. Keimel, J.D. Lawson, M. Mislove, D.S. Scott, A Compendium of Continuous Lattices, Springer, Berlin, 1980.
[26] C.A. Gunter, D.S. Scott, Semantic domains, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, vol. B: Formal Models and Semantics, Elsevier, Amsterdam, and The MIT Press, Cambridge, MA, 1990, pp. 633–674.
[27] M. Hermenegildo, D.S. Warren, S.K. Debray, Global flow analysis as a practical compilation tool, J. Logic Program. 13(4) (1992) 349–366.
[28] D. Jacobs, A. Langen, Static analysis of logic programs for independent AND-parallelism, J. Logic Program. 13(2-3) (1992) 154–165.
[29] T.P. Jensen, Disjunctive strictness analysis, in: Proc. 7th IEEE Symp. on Logic in Computer Science (LICS '92), IEEE Computer Society Press, Los Alamitos, CA, 1992, pp. 174–185.
[30] J.L. Lassez, M.J. Maher, K. Marriott, Unification revisited, in: J. Minker (Ed.), Foundations of Deductive Databases and Logic Programming, Morgan Kaufmann, Los Altos, CA, 1988, pp. 587–625.
[31] K. Marriott, H. Søndergaard, Precise and efficient groundness analysis for logic programs, ACM Lett. Program. Lang. Syst. 2(1–4) (1993) 181–196.
[32] K. Marriott, H. Søndergaard, N.D. Jones, Denotational abstract interpretation of logic programs, ACM Trans. Program. Lang. Syst. 16(3) (1994) 607–648.
[33] A. Mycroft, The theory and practice of transforming call-by-need into call-by-value, in: B. Robinet (Ed.), Proc. 4th Internat. Symp. on Programming, Lecture Notes in Computer Science, vol. 83, Springer, Berlin, 1980, pp. 270–281.
[34] A. Mycroft, Completeness and predicate-based abstract interpretation, in: Proc. ACM Symp. on Partial Evaluation and Program Manipulation (PEPM '93), ACM Press, New York, 1993, pp. 179–185.
[35] F. Nielson, Tensor products generalize the relational data flow analysis method, in: M. Arató, I. Kátai, L. Varga, (Eds.), Proc. 4th Hungarian Computer Science Conf., 1985, pp. 211–225.
[36] H.R. Nielson, F. Nielson, Bounded fixed point iteration, in: Conf. Record of the 19th ACM Symp. on Principles of Programming Languages (POPL '92), ACM Press, New York, 1992, pp. 71–82.

[37] P. Van Roy, A.M. Despain, The benefits of global dataflow analysis for an optimizing Prolog compiler, in: S.K. Debray, M. Hermenegildo (Eds.), Proc. North American Conf. on Logic Programming (NACLP '90), The MIT Press, Cambridge, MA, 1990, pp. 501–515.

[38] I. Wegener, The Complexity of Boolean Functions, Wiley-Teubner Series in Computer Science, Wiley, New York, 1987.