Optimal Domains for Disjunctive Abstract Interpretation

Roberto Giacobazzi [a] Francesco Ranzato [b]

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

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

Abstract

In the context of standard abstract interpretation theory, we define the inverse operation to the disjunctive completion of abstract domains, introducing the notion of least disjunctive basis of an abstract domain D. This is the most abstract domain inducing the same disjunctive completion as D. We show that the least disjunctive basis exists in most cases, and study its properties, also in relation with reduced product and complementation of abstract domains. The resulting framework provides advanced algebraic methodologies for abstract domain manipulation and optimization. These notions are applied to well-known abstract domains for static analysis of functional and logic programming languages.

Keywords: abstract interpretation, closure operator, disjunctive abstract domain, least disjunctive basis, functional and logic program analysis.

1 Introduction

Abstract interpretation ([12,13]) is a widely established theory for programming language semantics approximation. Different semantics, at different levels of abstraction, can be derived by abstract interpretation of a given concrete semantics ([8,11,15]). In particular, frameworks for program analysis, type inference and debugging can be specified and derived by abstract interpretation as approximations of the concrete semantics (see [8–10]).

A key rôle in this process of semantics approximation by abstract interpretation is played by *abstract domains*. Abstract domains provide a domain-like presentation for those semantic properties of the concrete domain of computation that are approximated by abstract interpretation. An abstract domain is assumed to be a

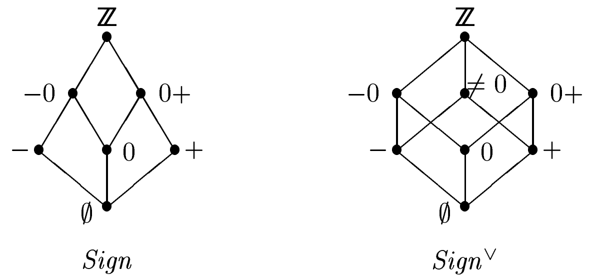


Fig. 1. The abstract domain *Sign* and its disjunctive completion $Sign^{\vee}$.

complete lattice, where the ordering relation describes the relative precision of domain objects – the top element representing no information. This is the case of the simple domain *Sign*, depicted in Fig. 1, abstracting sets of integer numbers (i.e. $\wp(\mathbb{Z})$), and which can be used for sign analysis of integer variables (we present a simple example later on). The interpretation of the elements of *Sign* is the obvious one: For example, the element -0 denotes the set of nonpositive integers. On the other hand, the concrete value $\{-5, -2\} \in \wp(\mathbb{Z})$ is abstracted into the element $-$ of *Sign*, which is the least element in *Sign* containing all negative numbers. More technically, the relationship between the concrete and the abstract domain is given by Galois connections (or, equivalently, by closure operators). However, for the rest of this section, the reader can simply consider a domain D more abstract than C if C contains all the information of D, i.e. $D \subseteq C$.

Since the very beginning of abstract interpretation, Cousot and Cousot ([13]) observed the importance of incrementally designing abstract domains. More expressive domains can be obtained by combining simpler ones, or by lifting them by adding new information. The first kind of operators are known as *domain combinators*, while the latter ones are known as *domain completions*. Examples of domain combinators are *reduced product* ([13]), *reduced power* ([13]), and *tensor product* ([35]), while domain completions include for instance *disjunctive completion* ([13]) and *Moore-set completion* ([23]). Both kinds of operators are devoted to enhance the expressive power of abstract domains, and have been called *domain refinements* (cf. [18,25]). All these domain operators provide high-level facilities to tune program analysis in accuracy and cost.

Disjunctive completion was originally introduced to exploit disjunctive program properties, notably to prove that merge-over-all-paths data-flow analysis can always be expressed in fixpoint form (cf. [13, Theorem 9.2.0.4]). Disjunctive completion was also considered in Nielson's approach to abstract interpretation using domain theory ([35]), and applied in program analysis of functional and logic languages, e.g., in Jensen's strictness logic for functional languages ([28]), in Cousot and Cousot's comportment analysis of functional languages ([16]), and in ground-dependency analysis of logic languages ([19]). Some forms of disjunctive completion are also included as tools for automatically refining abstract domains, in modern systems for program analysis, like, for instance, in System Z ([39]).

The basic idea of disjunctive completion is simple: Abstract domains are enhanced by adding the information corresponding to the concrete disjunction of their elements, i.e., by adding a denotation for the *lub* in the concrete domain of the concretization of the abstract values. For instance, in the example above, the concrete disjunction is union of sets of integers. Hence, it is immediate to see that the domain $Sign^{\vee}$ depicted in Fig. 1 corresponds exactly to the disjunctive completion of $Sign$ in $\langle \wp(\mathbb{Z}), \subseteq \rangle$. In this example, a unique further element is added to $Sign$, namely $\neq 0$. The symbol $\neq 0$ denotes the concrete disjunction of the $Sign$-elements $-$ and $+$ (viz., $\neq 0$ represents $\{x \mid x \neq 0\} = \{x \mid x < 0\} \cup \{x \mid x > 0\}$). In this sense, the abstract value of a program variable x is $\neq 0$ if x is known to be either a positive or a negative integer.

As all domain refinements, the disjunctive completion of abstract domains allows to improve the precision of program analysis. As a very simple example, let us consider the sign analysis of the integer variables in the following simple program fragment:

C_1: **if** B **then** $x := -5$ **else** $x := 5$ **fi**;
C_2: $y := 3$ **div** x

Assume that the analysis is defined by sequentially following the data flow, and by ignoring the conditional test B. This is specified by the abstract transition rules presented in Fig. 2, where the transitions occur among abstract states of the form $\langle (x, \sigma_x), (y, \sigma_y), \ldots \rangle$, where $x, y, \ldots$ are the variables into consideration, while $\sigma_x, \sigma_y \in Sign$ are their corresponding abstract values. Here, $\alpha : \wp(\mathbb{Z}) \to Sign$ is the abstraction function, approximating any set of integers into the sign of its elements, as represented in $Sign$, whereas $\vee$ is the *lub* in $Sign$. The semantics of the conditional command corresponds to a form of merge-over-all-paths data flow analysis: Since the conditional expression is not evaluated, the analysis considers both branches as possible computations. For the program fragment $C_1; C_2$, supposing that x and y are undefined before the execution of C_1, it is clear that the analysis (with the abstract domain $Sign$) does not supply any information on the sign of the variables x and y: [1]

$$\langle (x, \emptyset), (y, \emptyset) \rangle \stackrel{C_1}{\to} \langle (x, \mathbb{Z}), (y, \emptyset) \rangle \stackrel{C_2}{\to} \langle (x, \mathbb{Z}), (y, \mathbb{Z}) \rangle.$$

On the other hand, suppose that the same analysis is performed with the refined domain $Sign^{\vee}$. Then, we get a more precise result saying that x and y are both different from zero:

$$\langle (x, \emptyset), (y, \emptyset) \rangle \stackrel{C_1}{\to} \langle (x, \neq 0), (y, \emptyset) \rangle \stackrel{C_2}{\to} \langle (x, \neq 0), (y, \neq 0) \rangle.$$

In particular, we can correctly deduce that no error can occur during the execution of C_2, provided that x be defined after the execution of C_1. This information is not available from the former analysis, because, even if x is defined, the state $\langle (x, \mathbb{Z}), \ldots \rangle$ may include the case $x = 0$ for which 3 **div** x is undefined.

[1] Obviously, we implicitly use the most natural approximation $\mathbf{div}^a$ in the abstract domain $Sign$ of the operation of division of integers (where, e.g., $+ \ \mathbf{div}^a \ \mathbb{Z} = \mathbb{Z}$).

$$\overline{\langle (x,\sigma_x),(y,\sigma_y),\ldots\rangle \overset{x:=n}{\rightarrow} \langle (x,\alpha(n)),(y,\sigma_y),\ldots\rangle}$$

$$\frac{\langle \ldots,(x,\sigma_x),\ldots\rangle \overset{C_1}{\rightarrow} \langle \ldots,(x,\sigma_x^1),\ldots\rangle \quad \langle \ldots,(x,\sigma_x^1),\ldots\rangle \overset{C_2}{\rightarrow} \langle \ldots,(x,\sigma_x^2),\ldots\rangle}{\langle \ldots,(x,\sigma_x),\ldots\rangle \overset{C_1;C_2}{\rightarrow} \langle \ldots,(x,\sigma_x^2),\ldots\rangle}$$

$$C \equiv \textbf{if } B \textbf{ then } C_1 \textbf{ else } C_2 \textbf{ fi}$$

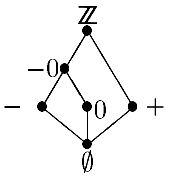
$$\frac{\langle \ldots,(x,\sigma_x),\ldots\rangle \overset{C_1}{\rightarrow} \langle \ldots,(x,\sigma_x^1),\ldots\rangle \quad \langle \ldots,(x,\sigma_x),\ldots\rangle \overset{C_2}{\rightarrow} \langle \ldots,(x,\sigma_x^2),\ldots\rangle}{\langle \ldots,(x,\sigma_x),\ldots\rangle \overset{C}{\rightarrow} \langle \ldots,(x,\sigma_x^1 \vee \sigma_x^2),\ldots\rangle}$$

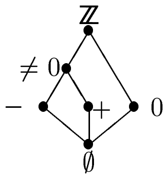
Fig. 2. Abstract transition rules.

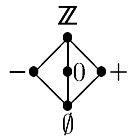
The disjunctive completion of abstract domains is not only essential to achieve precision in modeling alternative computations. It plays also an important rôle in the design of *relational* program analyses ([29,35]). Cousot and Cousot suggested in [14] that a relational analysis can be induced by combining reduced product (denoted by $\sqcap$) and disjunctive completion (denoted by $\mho$) of abstract domains, in the same way as the space of relations can be obtained in standard algebra by combining cartesian product and powerset. Thus, if D_1 and D_2 are abstract domains, a corresponding domain for relational analysis can be defined as $\mho(D_1 \sqcap D_2)$. In this construction, reduced product is attribute independent (viz. the information obtainable from the combination of analyses is essentially the same as the one obtainable by performing the analyses separately), while disjunctive completion introduces relational information by exploiting sets of attribute independent abstract values.

The Problem: Simplifying Abstract Domains for Disjunctive Analysis. A natural question is: "Can we invert a process of abstract domain refinement"? Namely, can we define the "least basis" (that is, the most abstract domain) which induces a given domain by composition or completion? This question has been technically formulated in a very general setting in [25]. Recently, [5] solved positively the problem of inverting the reduced product domain refinement, introducing the notion of domain complementation in abstract interpretation.

In this work, we consider the inverse for the operator of disjunctive completion, namely an operator, denoted by Ω, that, given any domain D (abstracting C), provides the most abstract domain (if it exists) whose disjunctive completion (in C) coincides with that of D, viz. $\mho(\Omega(D)) = \mho(D)$. Intuitively, Ω and $\mho$ should work as *compression* and *uncompression* operations (w.r.t. disjunction) on abstract domains. In the example above, for instance, it is easy to see that in addition to *Sign* the domains depicted below all satisfy this condition, namely their disjunctive completion (in $\wp(\mathbb{Z})$) is $\mathit{Sign}^\vee$. In this sense, they can all be candidates to be the "compression" of $\mathit{Sign}^\vee$ with respect to disjunction.

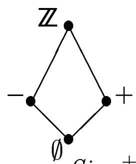




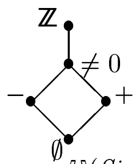


Sign^b

It is simple to note that Sign^b is contained in all these domains, as well as in any other abstraction of $\mathit{Sign}^\vee$ which, by disjunctive completion, returns $\mathit{Sign}^\vee$ back. Hence, there is no abstract domain that is more abstract than Sign^b, and at the same time, induces the same disjunctive completion as Sign^b. For instance, the disjunctive completion of $\mathit{Sign}^\pm$, that is a proper abstraction of Sign^b, does not coincide with $\mathit{Sign}^\vee$, i.e. $\mho(\mathit{Sign}^\pm) = \mathit{Sign}^{\neq 0} \neq \mathit{Sign}^\vee$, as depicted below.



$\mathit{Sign}^\pm$



$\mho(\mathit{Sign}^\pm) = \mathit{Sign}^{\neq 0}$

In this sense, Sign^b is the most abstract domain which, once refined by disjunctive completion, gives $\mathit{Sign}^\vee$ back.

The problem of finding a canonical subset of objects which, by means of a given operation, represent a given algebraic structure, is typical in lattice theory, and in particular in representation theory ([4,26]). We restrict here this investigation to specific algebraic structures, namely abstract domains for program analysis, and we introduce the notion of *least disjunctive basis* for them. The interest in this operation is twofold. Theoretically, least disjunctive bases contain the least amount of information which characterizes a given disjunctive property. This information must surely be included in any domain which is intended to meet a given disjunctive completion. On the practical side, least disjunctive bases are minimal (viz. nonredundant), providing useful space saving techniques to implement disjunctive completion of abstract domains, for instance for relational analyses. In particular, the disjunctive completion of the least disjunctive basis involves the least number of reduction tests in domain implementation (e.g., by a powerset construction), as most of the redundant information has been already removed from the source. Further, the least disjunctive basis operator can also be combined with complementation, in order to characterize optimal (viz. most abstract) decompositions for complex relational abstract domains. The resulting framework is powerful enough to be applied to arbitrary abstract domains for analysis, providing advanced algebraic methodologies for domain manipulation and optimization.

Structure of the Paper. In Section 2, we introduce the basic notations and notions of abstract interpretation (and, in particular, on closure operators) used throughout the paper. The disjunctive completion operator on abstract domains is presented and discussed in Section 3, together with a number of related results. The disjunctive completion is first given in its most general form, i.e. when the concrete domain is an arbitrary complete lattice. Then, under the standard hypothesis of distributivity of the concrete domain, we give the usual powerset-like characterization of the disjunctive completion. In the latter case, additional properties of the disjunctive completion operator are proved. In Section 4, we introduce the notion of least disjunctive basis of a domain D, which is the most abstract domain inducing the same disjunctive completion as D, and we prove that, under certain reasonable hypotheses, least disjunctive bases exist. Particular emphasis is given to the least disjunctive basis of domains abstracting distributive lattices: Whenever the disjunctive completion of an abstract domain D can be expressed in the powerset-like form, its least disjunctive basis can be explicitly defined by using the *join-irreducible* elements of D. We study in Section 5 the algebraic properties of disjunctive completion and least disjunctive basis of abstract domains, also in relation with reduced product and complementation. In the same section, we also show that least disjunctive bases distribute compositionally with respect to the reduced product, and we give a domain-theoretic characterization of the *redundant disjunctive information* of an abstract domain, as the information which is not in its least disjunctive basis. These properties are all proved in the most general setting, where no hypothesis on the structure of the involved domains is assumed.

We apply the above results to abstract domains for analysis of functional (Section 6) and logic (Section 7) programming languages. The main achievements are as follows.

– The *comportment domain* has been introduced by Cousot and Cousot ([16]) for the analysis of higher-order functional programs. This abstract domain has been defined as disjunctive completion of a more simple *basic comportment domain*. We show that this definition might be sharpened, since we prove that the least disjunctive basis of the comportment domain is strictly more abstract than the basic comportment domain. Moreover, by combining domain complementation and least disjunctive basis, we introduce in Section 6 the notion of *local optimization* of abstract domains. This is obtained by removing from abstract domains only portions of their redundant disjunctive information. This is applied to the comportment domain in order to design new abstract domains for strictness and termination analysis.
– We show that the Marriott and Søndergaard domain *Def* (viz. the lattice of positive Boolean functions whose models are closed under intersection, cf. [30]) is the least disjunctive basis inducing the domain for disjunctive ground-dependency analysis of logic programs ([19]). This shows that *Def*, which is strictly more ab-

stract, and therefore less expensive than *Pos* (viz. the lattice of positive Boolean functions, cf. [6,30]), always induces the same disjunctive ground-dependency analysis as *Pos* does, and in particular $\Omega(Pos) = Def$.

The paper ends by addressing related literature and future work in Section 8.

2 Closure Operators and Abstract Interpretation

After introducing the mathematical notation used in the paper, in this section we present a brief overview of the basic notions on closure operators and abstract interpretation (in particular, complementation of abstract domains, cf. [5]). For more details about lattice theory, and in particular closure operators, the reader is referred to [4,26,31,38], while for abstract interpretation to [12,13].

2.1 Mathematical Notation

Let C and D be sets. The set-difference between C and D is denoted by $C \setminus D$, while $C \subset D$ denotes proper inclusion. The powerset of D is denoted by $\wp(D)$, and its cardinality by $|D|$. If f is a function defined on C and $D \subseteq C$ then $f(D) = \{f(x) \mid x \in D\}$. By $g \circ f$ we denote the composition of the functions f and g, i.e. $\forall x. (g \circ f)(x) = g(f(x))$. The set D equipped with a partial order $\leq$ is denoted by $\langle D, \leq \rangle$. By $x < y$ we denote strict ordering. If D is a poset, we usually denote by $\leq_D$ the corresponding partial order. If D is a poset and $I \subseteq D$ then $\downarrow I = \{x \in D \mid \exists y \in I.\ x \leq_D y\}$. For $x \in D$, $\downarrow x$ is a shorthand for $\downarrow\{x\}$. By D^{op} we denote the dual-poset of D. A complete lattice D with partial ordering $\leq$, greatest lower bound (*glb*) $\wedge$, least upper bound (*lub*) $\vee$, greatest element $\top = \wedge\emptyset = \vee D$, and least element $\bot = \vee\emptyset = \wedge D$, is denoted $\langle D, \leq, \wedge, \vee, \top, \bot \rangle$. In a complete lattice D, a subset $S \subseteq D$ is *directed* if every finite subset of S has an upper bound in S, while S is *co-directed* if it is directed in D^{op}. When D is a lattice, $\wedge_D$, $\vee_D$, $\top_D$ and $\bot_D$ denote the corresponding basic operators and elements. We use $C \cong D$ to denote that the ordered structures C and D are isomorphic. In the following, we will often abuse notation by denoting lattices with their poset notation.

2.2 Galois Connections and Closure Operators

If C and D are posets and $\alpha : C \to D$, $\gamma : D \to C$ are monotonic functions such that $\forall c \in C.\ c \leq_C \gamma(\alpha(c))$ and $\forall d \in D.\ \alpha(\gamma(d)) \leq_D d$, then we call the quadruple (γ, D, C, α) a *Galois connection* (G.c.) between C and D. If in addition $\forall d \in D.\ \alpha(\gamma(d)) = d$, then we call (γ, D, C, α) a *Galois insertion* (G.i.) 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 $\forall c \in C. \forall d \in D.\ \alpha(c) \leq_D d \Leftrightarrow c \leq_C \gamma(d)$. Abstract interpretations are traditionally specified in terms of Galois insertions: C and D are called, respectively, the *concrete* and the *abstract domain*, and they are assumed to be complete lattices, whereas α and γ are called the *abstraction* and *concretization* maps, respectively. Also, D is called an *abstraction* (or *abstract interpretation*) of C, and C a *concretization* of D. Furthermore, if C is not an abstraction of D, then we say that D is a *strict* abstraction of C. Galois insertions characterize "perfect" abstractions, as any abstract object is the abstraction of a concrete one. In this case, the concretization and abstraction mappings are 1-1 and onto, respectively. Any G.c. may be lifted to a G.i. identifying in an equivalence class those values of the abstract domain with the same concrete meaning. This process is known as reduction of the abstract domain.

An (*upper*) *closure operator*, or simply a *closure*, on a poset $\langle L, \leq\rangle$ is an operator $\rho : L \to L$ monotonic, idempotent and extensive (viz. $\forall x \in L.\ x \leq \rho(x)$). The set of all closure operators on L is denoted by $uco(L)$. If $\langle L, \leq, \wedge, \vee, \top, \bot\rangle$ is a complete lattice then each closure operator ρ is uniquely determined by the set of its fixpoints, which is its image $\rho(L)$. A set $X \subseteq L$ is the set of fixpoints of a closure operator iff X is a *Moore-family* of L, i.e. X is meet-closed (viz. for any $Y \subseteq X$, $\wedge Y \in X$, where, in particular, $\top = \wedge\emptyset \in X$). In this case, $\rho_X = \wedge\{x \in X \mid y \leq x\}$ will denote the corresponding closure operator on L. For any $X \subseteq L$, $\mathcal{M}(X) = \{\wedge S \mid S \subseteq X\}$ is the *Moore-closure* of X in L, i.e. the least subset of L containing X which is a Moore-family of L. The set of fixpoints $\rho(L)$ is a complete lattice with respect to the order of L, but, in general, it is not a complete sublattice of L, since the *lub* in $\rho(L)$ might be different from that in L. Indeed, $\rho(L)$ is a complete sublattice of L iff ρ is additive, i.e. for all $X \subseteq L$, $\rho(\vee X) = \vee\rho(X)$. The subset of $uco(L)$ given by the additive closures will be denoted by $uco^a(L)$. In view of the above equivalence, in the following a closure operator ρ will often denote the set of its fixpoints $\rho(L)$. Being a set, the set of fixpoints of a closure is often denoted with capital Latin letters. Denoting closures by sets will be particularly convenient when closure operators will denote abstract domains. However, viewing closures as functions is also important in abstract interpretation, because they define the abstraction function. Hence, in the following, we will keep this soft ambiguity by using both notations, and leave to the reader to distinguish their use as functions or sets, according to the context. If L is a complete lattice then $\langle uco(L), \sqsubseteq, \sqcap, \sqcup, \lambda x.\top, \lambda x.x\rangle$ is a complete lattice, where for every $\rho, \eta \in uco(L)$, $\{\rho_i\}_{i\in I} \subseteq uco(L)$ and $x \in L$:

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

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

(i) $\rho(\wedge\rho(Y)) = \wedge\rho(Y)$;
(ii) $\rho(\vee Y) = \rho(\vee\rho(Y))$.

A *lower closure operator* $\varphi : L \to L$ is monotonic, idempotent and reductive (viz. $\forall x \in L.\ \varphi(x) \leq x$). The complete lattice of all lower closure operators on the complete lattice L is denoted by $lco(L)$. Its lattice-theoretic properties can all be derived by duality from those above for $uco(L)$.

2.3 The Lattice of Abstract Interpretations

A key point in Cousot and Cousot's abstract interpretation theory is the equivalence between the Galois insertion and closure operator approach to the design of abstract domains (cf. [13]). Actually, an abstract domain is just a "computer representation" of its logical meaning, namely its image in the concrete domain. In fact, using a different but lattice-theoretic isomorphic domain changes nothing in the abstract reasoning. The logical meaning of an abstract domain is exactly captured by the associated closure operator on the concrete domain. More formally, on the one hand, if (γ, D, C, α) is a G.i. then the closure associated with D is the operator $\rho_D = \gamma \circ \alpha$ on C. On the other hand, if ρ is a closure on C and $\iota : \rho(C) \to D$ is an isomorphism of complete lattices (with inverse ι^{-1}) then $(\iota^{-1}, D, C, \iota \circ \rho)$ is a G.i. It turns out that these two characterizations are one the inverse of the other. The complete lattice of all abstract interpretations (identified up to isomorphism) of a domain C is therefore isomorphic to $uco(C)$. By the above equivalence, it is not restrictive to use the closure operator approach to reason about abstract properties up to isomorphic representations of abstract domains. Thus, in the rest of the paper, we will feel free to use this approach, and whenever we will say that D is an abstraction of C, we will mean that D is isomorphic to $\rho_D(C)$ ($D \cong \rho_D(C)$), for some closure $\rho_D \in uco(C)$. In this approach, the order relation on $uco(C)$ corresponds to the order by means of which abstract domains are compared with regard to their precision. More formally, if $\rho_i \in uco(C)$ and $D_i \cong \rho_i(C)$ ($i = 1, 2$), D_1 is *more precise* than D_2 iff $\rho_1 \sqsubseteq \rho_2$ (i.e. $\rho_2(C) \subseteq \rho_1(C)$). Therefore, to compare domains with regard to their precision, we will only speak about abstractions between them, and use $\sqsubseteq$ to relate both closure operators and domains ($\sqsubset$ denotes strict ordering). Further, we will often use the equality symbol $=$ instead of $\cong$. In view of this equivalence, the *lub* and *glb* on $uco(C)$ get a clear meaning. Suppose $\{\rho_i\}_{i\in I} \subseteq uco(C)$ and $D_i \cong \rho_i(C)$ for each $i \in I$. Any domain D isomorphic to the *lub* $(\sqcup_{i\in I}\rho_i)(C)$ is the most concrete among the domains which are abstractions of all the D_i's. The interpretation of the *glb* operation on $uco(C)$ is twofold. Firstly, any domain D isomorphic to the *glb* $(\sqcap_{i\in I}\rho_i)(C)$ is (isomorphic to) the well known *reduced product* ([13]) of all the domains D_i. Also, the *glb* D, and hence the reduced product, is the most abstract among the domains (abstracting C) which are more concrete than every D_i. Thus, we will denote the reduced product of abstract domains by the *glb* symbol $\sqcap$.

2.4 Complementation in Abstract Interpretation

The notion of domain *complementation* in abstract interpretation has been introduced in [5], and further studied in [21]. This operation provides a systematic method for decomposing abstract domains into simpler factors.

Complementation corresponds to the *inverse of the reduced product* (cf. [18]), that is, an operation which, starting from any two domains $C \sqsubseteq D$, gives as result the most abstract domain $C \sim D$, whose reduced product with D is exactly C (i.e., $(C \sim D) \sqcap D = C$). By the above equivalence between closure operators and abstract domains, this notion of complementation corresponds precisely to *pseudocomplementation* for ρ_D in $uco(C)$. We recall the well known lattice-theoretic notion of pseudocomplementation ([4]).

Definition 2.1 Let L be a meet-semilattice with bottom. The *pseudocomplement* of $x \in L$, if it exists, is the (unique) element $x^* \in L$ such that $x \wedge x^* = \bot$ and $\forall y \in L.\ (x \wedge y = \bot) \Rightarrow (y \leq x^*)$. If every $x \in L$ has the pseudocomplement, L is called *pseudocomplemented.* □

In a complete lattice L, if the pseudocomplement x^* exists then

$$x^* = \vee\{y \in L \mid x \wedge y = \bot\}.$$

Giacobazzi et al. solved positively in [22] the problem of pseudocomplementation for closure operators. Recall that a complete lattice C is *meet-continuous* if for any chain $Y \subseteq C$ and $x \in C$, $x \wedge (\vee Y) = \vee_{y \in Y}(x \wedge y)$ ([4,26]). The following result is recalled from [22].

Theorem 2.2 ([22]) *If C is meet-continuous then $uco(C)$ is pseudocomplemented.*

This result has been first applied in abstract interpretation in [5]. In particular, whenever C is meet-continuous, the complement $C \sim D$ of a domain D with respect to C exists, and it is defined as:

$$C \sim D = \sqcup\{E \in uco(C) \mid D \sqcap E = C\}.$$

Meet-continuity is a sufficiently weak hypothesis to include most domains used in practice. For instance, it includes (see [26]) complete Heyting algebras, completely distributive, algebraic, complete Boolean and continuous lattices. Cortesi et al. observed in [5] that most of the abstract domains for program analysis are meet-continuous. This is also the case of domains for denotational semantics, as proved in [24], where complementation has been fruitfully applied to "decompose" semantics of programming languages.

Assume that $C \sqsubseteq D, E$ and let $\top$ be the most abstract interpretation of C (i.e. the top closure $\lambda x.\top$). The following are some basic algebraic properties of complementation ([5]):

(a) $D \sqsubseteq C \sim (C \sim D)$;

(b) $(D \sqsubseteq E) \Rightarrow (C \sim E) \sqsubseteq (C \sim D)$;

(c) $(C \sim D) = C \sim (C \sim (C \sim D))$;

(d) $C \sim \top = C$ and $C \sim C = \top$.

Complementation is important for *abstract domain decompositions*. If $C \sqsubseteq D$ then $\langle C \sim D, D \rangle$ is a (conjunctive binary) decomposition for C, namely C can be reconstructed by reduced product of its factors. The advantage of domain decomposition is twofold: (1) it provides more compact representations for complex domains, enhancing space saving techniques, and (2) it simplifies verification problems for complex domains, by decomposing them into simpler problems for their factors. We show how complementation actually works by a simple example.

Example 2.3 Consider the lattice *Sign* for sign analysis of an integer variable presented in Fig. 1 of Section 1. Recall that the concrete domain is $\langle \wp(\mathbb{Z}), \subseteq \rangle$, while concretization and abstraction maps are the most natural. The (more abstract) lattice of positive values $Sign^+$ can be "subtracted" from *Sign* by complementation (viz., $Sign \sim Sign^+$), yielding the lattice of negative values $Sign^-$.

$Sign^+$ $\qquad$ $Sign^- = Sign \sim Sign^+$

$\langle Sign^+, Sign^- \rangle$ is therefore a decomposition for *Sign* (i.e., $Sign = Sign^+ \sqcap Sign^-$). In particular, 0 and $\emptyset$, which are not in $Sign^+ \cup Sign^-$, can be both reconstructed by reduced product from $Sign^+$ and $Sign^-$. □

3 Disjunctive Completion by Closures

In this section, we reformulate using closure operators the standard Cousot and Cousot definition of disjunctive completion of an abstract domain ([13]), and study some of its algebraic properties, that will be useful later to characterize the inverse of disjunctive completion. Closure operators provide here the right high-level setting, where properties of abstract domains and operators for domain refinement can be studied independently from the representation of domain objects.

In view of their peculiar structure as Moore-families (cf. Sections 2.2 and 2.3), abstract domains support a *precise* interpretation for the *glb* operation, which abusively will be also called *conjunction* in the following. This is not in general the case for the dual operation of *lub*, also called *disjunction*. In technical terms, this means that if C is a complete lattice (the concrete domain), and $\rho \in uco(C)$ (the abstract domain), then for any $X \subseteq \rho(C)$, $\rho(\wedge_C X) = \wedge_C X$ (by property (i) in Section 2.2), whilst $\vee_C X \leq \rho(\vee_C X)$. Hence, while abstract conjunction (i.e., $\lambda X.\rho(\wedge_C X)$) always coincides (up to object names) with concrete conjunction (i.e., $\lambda X.\wedge_C X$), abstract disjunction (i.e., $\lambda X.\rho(\vee_C X)$) may cause a loss of information in the abstract domain. Disjunctive completion is therefore intended as a domain enhancement, which modifies a given abstract domain so that disjunction becomes precise.

Lemma 3.1 *If* $\rho \in uco(C)$ *then* $(\forall X \subseteq \rho(C).\ \rho(\vee_C X) = \vee_C X) \Leftrightarrow \rho \in uco^a(C)$.

Proof. $(\Rightarrow)$ Consider any $Y \subseteq C$. Then, we have that $\rho(\vee_C Y) =$ (by property (ii) in Section 2.2) $= \rho(\vee_C \rho(Y)) =$ (by hypothesis) $= \vee_C \rho(Y)$.
$(\Leftarrow)$ If $X \subseteq \rho(C)$ then, by hypothesis, $\rho(\vee_C X) = \vee_C \rho(X)$, but since $\rho(C)$ is the set of fixpoints of ρ, we get $\rho(\vee_C X) = \vee_C X$. $\square$

This lemma says that disjunction in an abstract domain is precise (in the sense above, as conjunction is) iff its associated closure operator is additive. Moreover, as recalled in Section 2.2, this is equivalent to the fact that the image of the closure is a complete sublattice of the concrete domain. These remarks justify the following technical definition of disjunctive completion by closure operators.

Let C be any complete lattice, and consider its lattice of abstract interpretations $uco(C)$. An *additive extension* of a closure $\rho \in uco(C)$ is any element belonging to the set $\{\eta \in uco^a(C) \mid \eta \sqsubseteq \rho\}$.

Lemma 3.2 *If* $\{\rho_i\}_{i \in I} \subseteq uco^a(C)$ *then* $\sqcup_{i \in I} \rho_i \in uco^a(C)$.

Proof. Using the results recalled in Section 2.2, it is sufficient to verify that $\cap_{i \in I} \rho_i(C)$ is a complete sublattice of C. But this is evidently true, since, by hypothesis, each $\rho_i(C)$ is a complete sublattice of C. $\square$

By this result, it is then natural to give the following definition.

Definition 3.3 The *disjunctive completion operator* is $\mho_C : uco(C) \to uco(C)$ defined as: $\mho_C(\rho) = \sqcup\{\eta \in uco^a(C) \mid \eta \sqsubseteq \rho\}$, for any $\rho \in uco(C)$. $\square$

Thus, the disjunctive completion operator maps any given abstract domain D, to the least common abstraction of the domains more precise than D and for which disjunction is precise. The correctness of this definition is therefore immediate by Lemma 3.2.

Proposition 3.4 *For any* $\rho \in uco(C)$, $\mho_C(\rho) \in uco^a(C)$.

$\mho_C(\rho)$ is called the *disjunctive completion* (*in* C) of $\rho \in uco(C)$. Since $\mho_C(\rho)$ is the greatest additive closure operator which is less than or equal to ρ, it is clear that our Definition 3.3 is equivalent to the original Cousot and Cousot definition in [13, Lemma 9.2.0.3]. It is worth noting that for any $\rho \in uco(C)$, $\bot_C \in \mho_C(\rho)$, because any additive closure is strict, namely $\rho(\bot_C) = \rho(\vee_C \emptyset) = \vee_C \emptyset = \bot_C$. In the following, we will apply the operator $\mho$ both to Moore-families and closure operators, being these two notions equivalent in view of our applications (cf. Sections 2.2 and 2.3). For the operator $\mho_C$, the following obvious result holds.

Proposition 3.5 $\mho_C \in lco(uco(C))$.

The meaning of the above proposition is clear: The disjunctive completion is a *domain refinement*, as defined in [18]. This means that $\mho_C$ is a monotonic and reductive operator on $uco(C)$, and no refinement can be obtained by disjunctive completion of a domain which is already disjunctively completed (viz., $\mho_C$ is idempotent). Being a lower closure operator, $\mho_C$ is uniquely determined by its set of fixpoints, namely its image $\mho_C(uco(C)) = uco^a(C)$, which is precisely the set of all *disjunctive abstract interpretations* of C. The following result is a straight consequence of Proposition 3.5, and characterizes the compositionality of the disjunctive completion with respect to the reduced product of abstract domains.

Proposition 3.6 *If* $C \sqsubseteq D, E$ *then* $\mho_C(D \sqcap E) = \mho_C(\mho_C(D) \sqcap \mho_C(E))$.

Proof. This follows by the dual of property (ii) in Section 2.2, since, by Proposition 3.5, $\mho_C$ is a lower closure operator on $uco(C)$. $\square$

It is worth noting that the disjunctive completion of an abstract domain depends on the fixed concrete domain. If $C \sqsubseteq D \sqsubseteq E$, then $\mho_C(E)$ is in general different from $\mho_D(E)$. Indeed, they coincide whenever D is disjunctive. In order to show this fact, we need the following technical lemma on closure operators, that can be found, e.g., in [7, Theorem 4.2.0.4.7].

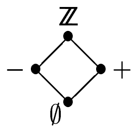
Lemma 3.7 ([7]) *Let C be a complete lattice and $\eta \in uco(C)$. Then,* $uco(\eta(C)) \cong \{\rho \in uco(C) \mid \eta \sqsubseteq \rho\}$.

Proposition 3.8 *If* $C \sqsubseteq D \sqsubseteq E$ *then* $\mho_C(E) \sqsubseteq \mho_D(E)$*, and if, in addition,* $\mho_C(D) = D$ *then* $\mho_C(E) = \mho_D(E)$.

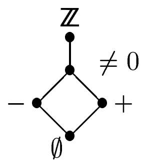
Proof. We reason on the associated closure operators; thus, the assumption is that $\rho_D, \rho_E \in uco(C)$ and $\rho_D \sqsubseteq \rho_E$. By the isomorphism of Lemma 3.7, ρ_E can also be viewed as a closure operator on D. Therefore, $\mho_C(E) = \mho_C(\rho_E) = \sqcup\{\eta \in uco^a(C) \mid \eta \sqsubseteq \rho_E\}$, while $\mho_D(E) = \mho_D(\rho_E) = \sqcup\{\mu \in uco^a(D) \mid \mu \sqsubseteq \rho_E\}$. Moreover, by Lemma 3.7, $\mho_D(\rho_E) = \sqcup\{\mu \in uco(C) \mid \rho_D \sqsubseteq \mu \sqsubseteq \rho_E,\ \mu \text{ is additive on } D\}$. Let us use the following notation: $S_C^E = \{\eta \in uco^a(C) \mid \eta \sqsubseteq \rho_E\}$ and $S_D^E = \{\mu \in uco(C) \mid \rho_D \sqsubseteq \mu \sqsubseteq \rho_E,\ \mu \text{ is additive on } D\}$. Now, if $\mu \in S_D^E$ then $\mho_C(\mu) \in S_C^E$, because $\mho_C(\mu) \sqsubseteq \mu \sqsubseteq \rho_E$ and $\mho_C(\mu)$ is additive on C. Therefore, we get that $\mho_C(E) \sqsubseteq \mho_D(E)$.
Suppose now that D is disjunctive (on C), i.e. $\mho_C(D) = D$. If $\mu \in uco(C)$ is such that $\rho_D \sqsubseteq \mu$ and μ is additive on D, then μ is additive on C: In fact, $\mu(C) \subseteq \rho_D(C) = D$, and since, by hypothesis, D is a complete sublattice of C, also $\mu(C)$ is a complete sublattice of C. Therefore, we have that $S_D^E \subseteq S_C^E$, i.e. $\mho_C(E) = \mho_D(E)$. $\square$

The following simple example shows this fact.

Example 3.9 Consider the lattice *Sign* of Example 2.3, and the lattice $\mathit{Sign}^{\pm}$ depicted below. Evidently, $\mathit{Sign}^{\pm}$ is an abstraction of *Sign*.



$\mathit{Sign}^{\pm} = \mho_{\mathit{Sign}}(\mathit{Sign}^{\pm})$



$\mho_C(\mathit{Sign}^{\pm})$

Clearly, *Sign* is not a disjunctive abstract interpretation of the concrete domain $C = \wp(\mathbb{Z})$, since the *lub* of $-$ and $+$ is not precise. In this case, $\mho_C(\mathit{Sign}^{\pm})$ does not coincide with $\mho_{\mathit{Sign}}(\mathit{Sign}^{\pm})$. In fact, the disjunctive completion of $\mathit{Sign}^{\pm}$ with respect to *Sign*, viz. $\mho_{\mathit{Sign}}(\mathit{Sign}^{\pm})$, is $\mathit{Sign}^{\pm}$ itself, while the disjunctive completion with respect to C, viz. $\mho_C(\mathit{Sign}^{\pm})$, is the lattice depicted above. □

3.1 Powerset-Like Construction of the Disjunctive Completion

Definition 3.3 above gives an implicit account of the disjunctive completion, by defining it, for a domain D abstracting C, as the most abstract domain which is both more concrete than D, and, at the same time, a complete sublattice of C. In this section, we specialize the discussion to the case of concrete domains satisfying some hypothesis of distributivity. This gives the possibility to introduce an explicit definition of the disjunctive completion of an abstract domain using a powerset-like construction, analogous to the standard ones (cf. [14,16,19]). We recall below the standard definitions involving the finite and infinite distributivity laws for lattices (see e.g. [4,26]).

Definition 3.10 A complete lattice C is *distributive* (d), or *completely distributive* (cd), if for any $\{x_{i,k} \mid i \in I, k \in K(i)\} \subseteq C$ and $x, y, z \in C$, the following identities respectively hold:

$$\text{(d)} \qquad x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z),$$

$$\text{(cd)} \qquad \bigwedge_{i\in I} \bigvee_{k\in K(i)} x_{i,k} = \bigvee_{f\in K^I} \bigwedge_{i\in I} x_{i,f(i)},$$

where for any $i \in I$, $K(i)$ is a set of indices, and K^I is the set of all functions f from I to $\cup_{i\in I} K(i)$ such that $\forall i \in I.\, f(i) \in K(i)$. □

Obviously, (cd) $\Rightarrow$ (d). For concrete domains that are completely distributive, the following result yields a powerset-like characterization of the disjunctive completion of an abstract domain.

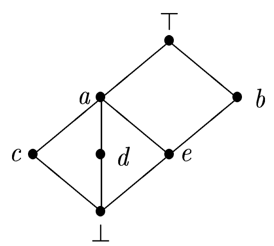
Proposition 3.11 *If C is a completely distributive lattice then, for any $\rho \in uco(C)$,*

$$\mho_C(\rho) = \{\vee S \mid S \subseteq \rho(C)\}.$$

Proof. Let us call $d_C^\rho = \{\vee S \mid S \subseteq \rho(C)\}$. In order to show that $d_C^\rho \in uco^a(C)$, we have to verify that $\mathcal{M}(d_C^\rho) = d_C^\rho$ and that $\langle d_C^\rho, \leq\rangle$ is join-closed. If $X \subseteq d_C^\rho$ then we can write $X = \{\vee S_i\}_{i\in I}$, with $S_i = \{z_{i,k}\}_{k\in K(i)} \subseteq \rho(C)$, for suitable sets of indices I and, for each $i \in I$, $K(i)$. By this expression, it is clear that, by complete distributivity of C, we have that $\wedge X \in d_C^\rho$. Moreover, it is also clear that $\vee X = \vee\{\vee_{k\in K(i)} z_{i,k} \mid i \in I\} = \vee\{z_{i,k} \mid i \in I,\ k \in K(i)\}$ belongs to d_C^ρ, proving that d_C^ρ is join-closed (i.e. a complete sublattice of C). Moreover, by definition, $\rho(C) \subseteq d_C^\rho$, hence showing that d_C^ρ is an additive extension of ρ. In order to conclude, we verify that if η is an additive extension of ρ then $d_C^\rho \subseteq \eta(C)$: If $\vee S \in d_C^\rho$, for some $S \subseteq \rho(C)$, then $S \subseteq \eta(C)$, and then $\vee S \in \eta$. $\square$

The following example shows that without the hypothesis of complete distributivity of the concrete domain, in general, the above characterization is no longer true.

Example 3.12 Consider the finite lattice C depicted below.



Consider also the closure $\rho \in uco(C)$ given by $\rho(C) = \{\top, b, c, d, \bot\}$. It is easy to verify that the least additive extension of ρ is $\mho_C(\rho) = C$, whereas $\{\vee S \mid S \subseteq \rho(C)\} = C \setminus \{e\}$, and this latter is not even a closure. $\square$

In Proposition 3.11, for finite abstract domains, we can weaken the hypothesis on the concrete domain, requiring (finite) distributivity only.

Corollary 3.13 *If C is a distributive complete lattice and $\rho \in uco(C)$ is finite (i.e., $|\rho(C)| < \omega$), then $\mho_C(\rho) = \{\vee S \mid S \subseteq \rho(C)\}$.*

Proof. Observe in the proof of Proposition 3.11 that for a finite ρ, d_C^ρ is finite too. Then, by an easy inspection of that proof, we see that finite distributivity of C is sufficient. $\square$

Proposition 3.11 does require the same hypothesis on the concrete domain as the usual standard powerset construction of the disjunctive completion ([14,19]) does. In this case, the above characterization actually gives rise to the standard powerset completion of an abstract domain written by a closure-like expression. Thus, whenever the concrete domain is completely distributive the disjunctive completion by closures (Definition 3.3) coincides with the standard powerset completion. Moreover, we have also observed in Corollary 3.13 that for finite abstract domains it is enough to consider the weaker hypothesis of (finite) distributivity for the concrete domain. However, it is worthwhile to remark that Definition 3.3 of disjunctive completion of an abstract domain is of wider applicability than the standard powerset definition. In fact, Example 3.12 shows that whenever the concrete domain is not (completely) distributive, in general, the standard powerset definition is not applicable, while, in contrast, Definition 3.3 is always applicable to any domain.

We close this section by giving an alternative, but equivalent, presentation for the disjunctive completion operator. We characterize the disjunctive completion of an abstract domain as a function, instead of a set. This may be sometimes more practical, in particular when abstract domains are defined by Galois connections, in order to identify explicitly the abstraction map corresponding to the disjunctive completion. In addition to complete distributivity, this result requires that the concrete domain is join-generated by a suitable subset of points, namely its *join-irreducible elements* (see [2,4,26]). In Section 4, join-irreducible elements will play a rôle also in characterizing the inverse operation to disjunctive completion.

Definition 3.14 Let C be a complete lattice and $x \in C$. x is (*completely*) *join-irreducible* if for any $S \subseteq C$, $x = \vee S$ implies $x \in S$. The set of join-irreducible elements of C is denoted by $JI(C)$. □

The following lemma characterizing join-irreducible elements in completely distributive lattices is standard in lattice theory (cf.[2,26]).

Lemma 3.15 ([2]) *Let C be a completely distributive lattice. Then, $x \in JI(C)$ iff for any $S \subseteq C$, $x \leq \vee S$ implies $x \leq s$ for some $s \in S$.*

Let us recall that if C is a complete lattice and $S \subseteq C$ then C is *join-generated* by S if for all $x \in C$, $x = \vee(\downarrow x \cap S)$ (cf. [4,26]). We now prove the announced explicit functional characterization of the disjunctive completion operator.

Proposition 3.16 *If C is completely distributive and join-generated by $JI(C)$, and $\rho \in uco(C)$, then*

$$\mho_C(\rho) = \lambda x. \vee \{\rho(y) \mid y \in \downarrow x \cap JI(C)\}.$$

Proof. Let us define $d_C^\rho = \lambda x. \vee \{\rho(y) \mid y \in \downarrow x \cap JI(C)\}$. We first prove that $d_C^\rho \in uco^a(C)$. Monotonicity of d_C^ρ is immediate, being ρ monotone. By hypothesis, if $x \in C$ then $x = \vee\{y \in JI(C) \mid y \leq x\}$. Hence, since ρ is extensive, we have $x \leq \vee\{\rho(y) \mid y \in \downarrow x \cap JI(C)\} = d_C^\rho(x)$, proving extensivity of d_C^ρ. We now prove that d_C^ρ is idempotent. For any $x \in C$, by Lemma 3.15, we have that

$$\begin{aligned} d_C^\rho(d_C^\rho(x)) &= \vee\{\rho(y) \mid y \in JI(C),\ y \leq \vee\{\rho(z) \mid z \in JI(C),\ z \leq x\}\} \\ &= \vee\{\rho(y) \mid y \in JI(C),\ y \leq \rho(z),\ z \in JI(C),\ z \leq x\}. \end{aligned}$$

By monotonicity and idempotency of ρ, $\rho(y) \leq \rho(z)$ for any $y \in JI(C)$ such that $y \leq \rho(z)$ for some $z \in \downarrow x \cap JI(C)$. This implies that $\vee\{\rho(y) \mid y \in JI(C),\ y \leq \rho(z),\ z \in \downarrow x \cap JI(C)\} \leq \vee\rho(\downarrow x \cap JI(C))$, namely $d_C^\rho(d_C^\rho(x)) \leq d_C^\rho(x)$. Since we have already proved that d_C^ρ is extensive, this implies that d_C^ρ is idempotent. The following equalities prove that d_C^ρ is additive. For any $S \subseteq C$:

$$\begin{aligned} d_C^\rho(\vee S) &= \vee\{\rho(y) \mid y \in JI(C),\ y \leq \vee S\} \\ &= \vee\{\rho(y) \mid y \in JI(C),\ s \in S,\ y \leq s\} \quad \text{(by Lemma 3.15)} \\ &= \textstyle\bigvee_{s\in S}(\vee\{\rho(y) \mid y \in JI(C),\ y \leq s\}) \\ &= \vee d_C^\rho(S). \end{aligned}$$

We observe that using the idempotency of ρ, for any $x \in C$, we get $d_C^\rho(\rho(x)) = \vee\{\rho(y) \mid y \leq \rho(x),\ y \in JI(C)\} \leq \rho(x)$. This clearly implies that $d_C^\rho(\rho(x)) = \rho(x)$, and therefore $d_C^\rho \sqsubseteq \rho$. Consider now any $\eta \in uco^a(C)$ such that $\eta \sqsubseteq \rho$. In order to conclude the proof, we have to demonstrate that $\eta \sqsubseteq d_C^\rho$. Being η additive, for any $x \in C$, $\eta(d_C^\rho(x)) = \vee\{\eta(\rho(y)) \mid y \in \downarrow x \cap JI(C)\}$. Because $\eta \sqsubseteq \rho$, for any $x \in C$, $\eta(\rho(x)) = \rho(x)$, and therefore $\eta(d_C^\rho(x)) = d_C^\rho(x)$. This clearly means that $\eta \sqsubseteq d_C^\rho$, as desired. □

4 Least Disjunctive Basis of Abstract Domains

Our goal is to answer to the following question:

> *Given a domain D abstracting C, under what hypotheses does exist the least abstraction of C having the same disjunctive completion of D in C?*

In the following, we positively answer this question. Firstly, we need two preliminary definitions formally stating, by closure operators, this question.

Definition 4.1 Given a complete lattice C, $\rho \in uco(C)$ is *disjunctively optimizable* if $\mho_C(\sqcup\{\eta \in uco(C) \mid \mho_C(\eta) = \mho_C(\rho)\}) = \mho_C(\rho)$. □

In the following, for any $\rho \in uco(C)$, the *lub* of all closures having the same disjunctive completion as ρ, namely $\sqcup\{\eta \in uco(C) \mid \mho_C(\eta) = \mho_C(\rho)\}$, is denoted by $\Lambda_C(\rho)$. It is worth noting that by monotonicity of $\mho_C$ (cf. Proposition 3.5), for any $\rho \in uco(C)$, $\mho_C(\rho) \sqsubseteq \mho_C(\Lambda_C(\rho))$ always holds. Hence, $\rho \in uco(C)$ is disjunctively optimizable if the converse holds.

Definition 4.2 Assume that $C \sqsubseteq D$, and the corresponding $\rho_D \in uco(C)$ is disjunctively optimizable. The *least disjunctive basis* of D in C is the complete lattice given by the set of fixpoints of $\Lambda_C(\rho_D)$ in C. $\Box$

If D is disjunctively optimizable then we also say that its least disjunctive basis exists, and in this case, we denote it by $\Omega_C(D)$. $\Omega_C(D)$ is therefore the most abstract domain such that $\mho_C(\Omega_C(D)) = \mho_C(D)$ holds. Being $\Omega_C(D) = (\Lambda_C(\rho_D))(C)$, the above definition implies that $\Omega_C(D)$ is a subset of C. Obviously, any other lattice isomorphic to $\Omega_C(D)$ can be considered in all respects as the least disjunctive basis. The following result provides an alternative characterization for the least disjunctive basis.

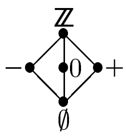
Proposition 4.3 *$\rho \in uco(C)$ is disjunctively optimizable iff there exists a unique element $\hat{\rho} \in uco(C)$ such that:*

(i) $\mho_C(\hat{\rho}) = \mho_C(\rho)$;
(ii) $\forall \eta \in uco(C).\ \mho_C(\eta) = \mho_C(\rho) \Rightarrow \eta \sqsubseteq \hat{\rho}$.

Proof. ($\Rightarrow$) Define $\hat{\rho} = \Omega_C(\rho) = \sqcup\{\eta \in uco(C) \mid \mho_C(\eta) = \mho_C(\rho)\}$. Condition (i) is satisfied by hypothesis. Moreover, if $\eta \in uco(C)$ is such that $\mho_C(\eta) = \mho_C(\rho)$, then, clearly, $\eta \sqsubseteq \hat{\rho}$, so that (ii) is satisfied. Also, $\hat{\rho}$ is the unique element in $uco(C)$ satisfying (i) and (ii): if $\psi \in uco(C)$ is another such an element different from $\hat{\rho}$ then, by (i), $\psi \sqsubseteq \hat{\rho}$, while, by (ii), $\hat{\rho} \sqsubseteq \psi$.
($\Leftarrow$) It is sufficient to verify that $\Lambda_C(\rho) = \hat{\rho}$. This is evidently true, by conditions (i) and (ii) on $\hat{\rho}$. $\Box$

The following simple example illustrates how the notion of least disjunctive basis actually works.

Example 4.4 Consider the domain *Sign* and its disjunctive completion (with respect to $C = \wp(\mathbb{Z})$) $\mho_C(\mathit{Sign}) = \mathit{Sign}^{\vee}$ both depicted in Fig. 1 of Section 1.

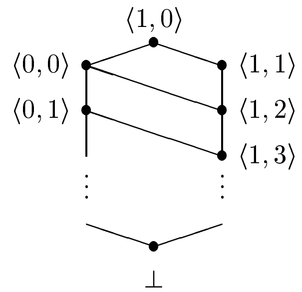


$\Lambda_C(\mathit{Sign}) = \Omega_C(\mathit{Sign})$

It is easy to verify that the least common abstraction of all the domains having as disjunctive completion $\mho_C(\mathit{Sign})$, namely $\Lambda_C(\mathit{Sign})$, is the lattice depicted above. In this case, since $\mho_C(\Lambda_C(\mathit{Sign})) = \mho_C(\mathit{Sign})$, the least disjunctive basis of *Sign* (in $C = \wp(\mathbb{Z})$) exists, and therefore it is $\Omega_C(\mathit{Sign}) = \Lambda_C(\mathit{Sign})$. $\Box$

However, the least disjunctive basis of an abstract domain does not always exist, as the following example shows.

Example 4.5 Let C be the complete lattice $\{\langle m,n\rangle \mid \; m \in \{0,1\},\ n \in \mathbb{N}\} \cup \{\bot\}$, where the ordering relation is determined by the Hasse diagram below.



For any $k \in \mathbb{N}$, consider the closure $\rho_k = \{\langle 1,n\rangle\}_{n\in\mathbb{N}} \cup \{\langle 0,n\rangle\}_{k\leq n} \cup \{\bot\}$. It is clear that for any $k \in \mathbb{N}$, $\mho_C(\rho_k) = C$, and $(\sqcup_{k\in\mathbb{N}}\rho_k)(C) = \{\langle 1,n\rangle\}_{n\in\mathbb{N}} \cup \{\bot\} = (\mho_C(\sqcup_{k\in\mathbb{N}}\rho_k))(C)$. If, by contradiction, $\Omega_C(C)$ exists, then $\sqcup_{k\in\mathbb{N}}\rho_k \sqsubseteq \Omega_C(C)$. But, since $\mho_C$ is monotonic, we should have $\mho_C(\sqcup_{k\in\mathbb{N}}\rho_k) \sqsubseteq \mho_C(\Omega_C(C)) = C$, i.e. we should obtain the contradiction $C \subseteq \{\langle 1,n\rangle\}_{n\in\mathbb{N}} \cup \{\bot\}$. □

Below, we state two theorems, one orthogonal to the other, for the existence of the least disjunctive basis. The first guarantees the existence of the least disjunctive basis for any (possibly infinite) abstract domain, provided that the concrete domain is completely distributive and co-algebraic. The second, instead, guarantees the existence of the least disjunctive basis for finite abstract domains, when the concrete domain is (finitely) distributive.

4.1 The Infinite Case

In this section, we prove that every (possibly infinite) abstraction of a co-algebraic completely distributive concrete domain is disjunctively optimizable. In order to do this, we need some technical results on algebraic complete lattices, also involving the notion of join-irreducibility. Firstly, let us recall the lattice-theoretic definition of co-algebraicity (see, e.g., [26]).

Definition 4.6 Let C be a complete lattice and $x \in C$.

(i) x is *compact* if for any $S \subseteq C$, $x \leq \vee S$ implies that there exists $T \subseteq S$, with T finite, such that $x \leq \vee T$. The set of compact elements of C is denoted by $K(C)$.
(ii) The complete lattice C is *algebraic* if for any $y \in C$, $y = \vee(\downarrow y \cap K(C))$, while it is *co-algebraic* if C^{op} is algebraic. □

We recall from [26] two technical results on algebraic lattices. The first one gives a characterization of (co-)algebraic lattices (cf. [26, Theorem 4.15, pag. 89]), while the second one provides a representation for the elements of (co-)algebraic lattices using join-irreducible elements (cf. [26, Theorem 4.23, pag. 93]).

Theorem 4.7 ([26]) *Assume that C is a complete lattice. C is co-algebraic iff C is isomorphic to a subset of $\wp(X)$, for some set X, ordered by inclusion and closed under arbitrary unions and intersections of co-directed families (in $\langle\wp(X),\subseteq\rangle$) of sets.*

Theorem 4.8 ([26]) *If C is a co-algebraic complete lattice, then C is join-generated by $JI(C)$.*

In order to prove the main theorem of this section, we need the following lemma stating that the images of additive closures preserve co-algebraicity of complete lattices.

Lemma 4.9 *If $\langle C,\leq\rangle$ is a co-algebraic complete lattice and $\rho \in uco^a(C)$, then $\langle\rho(C),\leq\rangle$ is a co-algebraic complete lattice.*

Proof. By Theorem 4.7, assume that $C_\leq \cong S_\subseteq$, for some S and X such that $S \subseteq \wp(X)$, where S is closed under arbitrary unions and co-directed intersections. Hence, also $\rho(C)$ is isomorphic to a subset of $\wp(X)$. Suppose that $\{Y_i\}_{i\in I} \subseteq \rho(C)$, for some set of indices I. Since $\rho(C)$ is a Moore-family, we have that $\cap_{i\in I} Y_i \in \rho(C)$. Moreover, since ρ is an additive closure, by Lemma 3.1, we also have that $\cup_{i\in I} Y_i \in \rho(C)$. Thus, being closed under arbitrary unions and intersections, by Theorem 4.7, $\rho(C)$ is, in particular, co-algebraic. $\square$

We are now in the position to prove the main result of this section.

Theorem 4.10 *If C is a co-algebraic completely distributive lattice, then each $\rho \in uco(C)$ is disjunctively optimizable.*

Proof. As observed previously, it is enough to prove that $\mho_C(\rho) \subseteq \mho_C(\Lambda_C(\rho))$. Since, by Proposition 3.4, $\mho_C(\rho)$ is additive, we have, by Lemma 4.9, that $\mho_C(\rho)$ (intended as its set of fixpoints) is co-algebraic. Now, consider any $x \in \mho_C(\rho)$. Since $\mho_C(\rho)$ is co-algebraic, by Theorem 4.8, there exists $R \subseteq JI(\mho_C(\rho))$ such that $x = \vee R$ (note that the *lub* of $\mho_C(\rho)$ coincides with the *lub* $\vee$ of C because $\mho_C(\rho)$ is a complete sublattice of C). If $y \in R$ then, by Proposition 3.11, there exists $T_y \subseteq \rho$ such that $y = \vee T_y$. Since y is join-irreducible (in $\mho_C(\rho)$ where the *lub* is $\vee$), this implies that $y \in T_y$. Thus, we get $R \subseteq \rho$. Moreover, if we consider $\eta \in uco(C)$ such that $\mho_C(\eta) = \mho_C(\rho)$ then, by the same reasoning, we get $R \subseteq \eta$. Thus, since $\Lambda_C(\rho) = \cap\{\eta \in uco(C) \mid \mho_C(\eta) = \mho_C(\rho)\}$, we have that $R \subseteq \Lambda_C(\rho)$. Therefore, by Proposition 3.11, $x = \vee R \in \mho_C(\Lambda_C(\rho))$, which concludes the proof. $\square$

It is worthwhile to remark that any complete lattice isomorphic to a *complete ring of sets* (i.e., a subset of a powerset $\wp(X)$, for some set X, closed under arbitrary unions and intersections) is both co-algebraic and completely distributive (see [2,26]). Thus, in particular, the class of co-algebraic completely distributive lattices comprises the well known *collecting domains*, i.e. any powerset $\wp(X)$, for some set X, ordered with the subset or supset relation. Therefore, the standard concrete domains for collecting semantics of functional and logic programming (e.g. [3,34]) are included in this class (we will use these concrete domains in our examples in Sections 6 and 7).

4.2 The Finite Case

We now prove a second theorem assuring the existence of the least disjunctive basis for any finite domain abstracting a distributive complete lattice. Thus, the restriction to finite abstract domains allows us to weaken the hypothesis of complete distributivity on the concrete domain considered in Theorem 4.10.

Theorem 4.11 *If C is a distributive complete lattice and $\rho \in uco(C)$ is finite, then ρ is disjunctively optimizable.*

Proof. First, let us point out that, as usual, we do not distinguish between a closure operator and its set of fixpoints (i.e. its image). Recall the definition $\Lambda_C(\rho) = \sqcup\{\eta \in uco(C) \mid \mho_C(\eta) = \mho_C(\rho)\}$. As observed previously, it is enough to prove that $\mho_C(\rho) \subseteq \mho_C(\Lambda_C(\rho))$. Assume by contradiction, that there exists $x \in \mho_C(\rho) \setminus \mho_C(\Lambda_C(\rho))$. We prove by induction on natural numbers that for any $n \in \mathbb{N}$ we are able to build a strictly decreasing chain $\{x_j\}_{j \leq n}$ such that for any $j \leq n$, $x_j \in \mho_C(\rho) \setminus \mho_C(\Lambda_C(\rho))$.
$(n = 0)$ By defining $x_0 = x$, we have that $x_0 \in \mho_C(\rho) \setminus \mho_C(\Lambda_C(\rho))$.
$(n+1)$ By inductive hypothesis, $x_n \notin \mho_C(\Lambda_C(\rho))$, from which we get $x_n \notin \Lambda_C(\rho)$, viz. $x_n \notin \cap\{\eta \in uco(C) \mid \mho_C(\eta) = \mho_C(\rho)\}$. Therefore, there exists $\eta \in uco(C)$ such that $\mho_C(\eta) = \mho_C(\rho)$ and $x_n \notin \eta$. Hence, $x_n \in \mho_C(\eta) \setminus \eta$. By Corollary 3.13, $\mho_C(\rho) = \mho_C(\eta)$ is finite, and therefore, $\eta \subseteq \mho_C(\eta)$ is finite as well. This implies that, by Corollary 3.13, there exists $S \subseteq \eta$ such that $x_n = \vee_C S$ and $x_n \notin S$. Also, $S \not\subseteq \mho_C(\Lambda_C(\rho))$, otherwise we would have $x_n = \vee_C S \in \mho_C(\Lambda_C(\rho))$, a contradiction. Thus, there exists $y \in S \setminus \mho_C(\Lambda_C(\rho))$. Define $x_{n+1} = y$. We have that $x_{n+1} \in \eta \subseteq \mho_C(\eta) = \mho_C(\rho)$, and, by construction, $x_{n+1} \notin \mho_C(\Lambda_C(\rho))$. Moreover, $x_{n+1} \leq x_n$, and $x_{n+1} \neq x_n$, because $x_{n+1} \in S$ while $x_n \notin S$.
To conclude the proof, note that, as observed above, $\mho_C(\rho)$ is finite. This implies that if we build the above strictly decreasing chain of infinite length, we get the desired contradiction. $\square$

It is worthwhile to note that most of the abstract domains used as basis of a static program analysis are finite, and hence for them one can exploit the weaker hypothesis on the concrete domain. In Sections 6 and 7, we will determine the least disjunctive basis of some finite abstract domains used for the analysis of functional and logic languages.

It is important to remark that the least disjunctive basis of a domain D, abstracting C, may well exist even if D and C do not satisfy the hypotheses of one of the two theorems of existence given above (i.e., Theorems 4.10 and 4.11). This is shown by the following example.

Example 4.12 Consider the finite lattice C and the closure $\rho \in uco(C)$ presented in Example 3.12. It is easily seen that C is not distributive, and therefore, for any abstraction of C neither Theorem 4.10 nor Theorem 4.11 are applicable. However, the least disjunctive basis $\Omega_C(C)$ of C itself (in C) exists, and it is actually given by the closure ρ. □

4.3 Least Disjunctive Basis and Join-Irreducible Elements

In case the concrete domain is a co-algebraic completely distributive lattice, from the proof of Theorem 4.10, we can observe that join-irreducible elements play an important rôle in the computation of the least disjunctive basis. Intuitively, they represent information that cannot be further decomposed by disjunction. In particular, join-irreducible elements allow to give an explicit characterization of the least disjunctive basis of an abstract domain D which is disjunctive, as proved below.

Theorem 4.13 *Assume that $C \sqsubseteq D$. If $\mho_C(D) = D$, and C is completely distributive and co-algebraic, or C is distributive and D is finite, then $\Omega_C(D) = \mathcal{M}(JI(D))$.*

Proof. The case C completely distributive and co-algebraic is immediate, by inspection of the proof of Theorem 4.10. Assume C distributive and D finite. Let $\rho \in uco(C)$ be the closure operator associated with D, such that $\mho_C(\rho) = \rho$. We show that $\Omega_C(\rho) = \mathcal{M}(JI(\rho))$. First, observe that $\mho_C(\mathcal{M}(JI(\rho))) = \mho_C(\rho)$. Under the hypotheses of Theorems 4.11, since $\rho = \mho_C(\rho)$ is finite, then, by Theorem 4.7, $\mho_C(\rho) = \{\vee S \mid S \subseteq JI(\mho_C(\rho))\}$, i.e. $\mho_C(\rho) = \mho_C(JI(\mho_C(\rho)))$. Thus, $\mho_C(JI(\rho)) = \mho_C(\rho)$, as desired. Next, assume that $\eta \in uco(C)$ and $\mho_C(\eta) = \mho_C(\rho)$. In this case, we want to prove that $JI(\rho) \subseteq \eta$. Suppose that $x \in JI(\rho)$. Then, $x \in \rho = \mho_C(\rho) = \mho_C(\eta)$. Note that since $\mho_C(\eta) = \rho$ is finite, $\eta \subseteq \mho_C(\eta)$ is finite as well. Therefore, by Corollary 3.13, there exists $R \subseteq \eta$ such that $x = \vee R$. On the other hand, since $\eta \subseteq \mho_C(\eta) = \mho_C(\rho) = \rho$, we have that $R \subseteq \rho$. Hence, being x join-irreducible in ρ, we get $x \in R \subseteq \eta$. □

It is worth noting that the above result does not hold if the least disjunctive basis $\Omega_C(D)$ does not exist. For instance, in Example 4.5, $JI(C) = \{\langle 1, n\rangle\}_{n\in\mathbb{N}\setminus\{0\}}$, but, $\mathcal{M}(JI(C)) = \{\langle 1, n\rangle\}_{n\in\mathbb{N}} \cup \{\bot\}$ is not the least disjunctive basis for C, as shown in that example. Obviously, since $\mho_C(C) = C$, the least disjunctive basis of every concrete domain, satisfying the hypotheses above, is just the Moore-closure of its join-irreducible elements. Theorem 4.13 has another interesting consequence.

Corollary 4.14 *Assume that $C \sqsubseteq D$ and C is completely distributive and co-algebraic, or C is distributive and D is finite. Then, $\Omega_C(D) = \mathcal{M}(JI(\mho_C(D)))$.*

Proof. Since $\Omega_C(D) = \Omega_C(\mho_C(D))$ and $\mho_C(\mho_C(D)) = \mho_C(D)$. □

In other terms, whenever the hypotheses of Theorems 4.10 and 4.11 are satisfied, the least disjunctive basis of an abstract domain D is always given by the Moore-closure of the join-irreducible elements of the disjunctive completion of D.

Example 4.15 With reference to Fig. 1 in Section 1, consider the abstract domain *Sign* and its disjunctive completion (w.r.t. $C = \wp(\mathbb{Z})$) $\mho_C(\mathit{Sign}) = \mathit{Sign}^{\vee}$. It is clear that $JI(\mho_C(Sign)) = \{-, 0, +\}$. Since the concrete domain $C = \wp(\mathbb{Z})$ is a collecting domain, we can apply Corollary 4.14, obtaining the least disjunctive basis $\Omega_C(Sign) = \mathcal{M}(\{-, 0, +\})$, as already computed in Example 4.4. $\square$

Obviously, computing explicitly the whole disjunctive completion $\mho_C(D)$ in order to characterize its join-irreducible elements, although being theoretically possible, is hardly feasible, because of the (usually exponential) size of $\mho_C(D)$. Indeed, when $\mho_C(D)$ is finite and isomorphic to a powerset, $|\Omega_C(D)| = log(|\mho_C(D)|) + k$, where k is a constant.

5 Algebraic Properties and Compositionality

In this section, we study the algebraic properties of the least disjunctive basis with respect to disjunctive completion and reduced product. From now on, whenever we will speak about least disjunctive bases, we will not suppose that conditions of Theorems 4.10 and 4.11 are satisfied, but merely we will assume their existence.

In general, the least disjunctive basis operator depends on the fixed concrete domain of reference (an example will be given at the end of Section 7), unless disjunctive abstract interpretations are considered.

Proposition 5.1 *If $C \sqsubseteq D \sqsubseteq E$ and $\mho_C(D) = D$ then $\Omega_C(E) = \Omega_D(E)$.*

Proof. Let us define $S_C^E = \{A \in uco(C) \mid \mho_C(A) = \mho_C(E)\}$ and $S_D^E = \{B \in uco(D) \mid \mho_D(B) = \mho_D(E)\}$. Thus, explicitly, we have $\Omega_C(E) = \sqcup S_C^E$ and $\Omega_D(E) = \sqcup S_D^E$. By Proposition 3.5, we have that $\mho_C(D) \sqsubseteq \mho_C(E)$, and because $D = \mho_C(D)$ and $\mho_C(E) \sqsubseteq \Omega_C(E)$, we get $D \sqsubseteq \Omega_C(E)$ (i.e. $\Omega_C(E) \in uco(D)$). Moreover, by Proposition 3.8, we have $\mho_D(\Omega_C(E)) = \mho_C(\Omega_C(E)) = \mho_C(E) = \mho_D(E)$, which implies $\Omega_C(E) \in S_D^E$. Thus, we get $\Omega_C(E) \sqsubseteq \Omega_D(E)$. To prove the other inclusion, assume that $B \in S_D^E$. Then, by Proposition 3.8, $\mho_C(B) = \mho_D(B) = \mho_D(E) = \mho_C(E)$, i.e. $B \in S_C^E$. Therefore, $\Omega_D(E) \sqsubseteq \Omega_C(E)$. $\square$

Next proposition summarizes the basic algebraic properties of the least disjunctive basis.

Proposition 5.2 *Assume that $C \sqsubseteq D, E$, $\top$ is the most abstract interpretation of C (and $\Omega_C(D), \Omega_C(E)$ exist). Then,*

(a) $D \sqsubseteq \Omega_C(D)$;
(b) $\Omega_C(\Omega_C(D)) = \Omega_C(D)$;
(c) $\Omega_C(\top) = \top$;
(d) $\Omega_C(\mho_C(D)) = \Omega_C(D)$;
(e) $\mho_C(D) = \mho_C(E) \Leftrightarrow \Omega_C(D) = \Omega_C(E)$;
(f) $\mho_C(D \sqcap E) = \mho_C(\Omega_C(D) \sqcap \Omega_C(E))$.

Proof. *(a)*: Obvious from the definition.
(b): We have that $\Omega_C(\Omega_C(D)) = \sqcup\{A \in uco(C) \mid \mho_C(A) = \mho_C(\Omega_C(D))\} =$

$\sqcup\{A \in uco(C) \mid \mho_C(A) = \mho_C(D)\} = \Omega_C(D)$.
(c): By using *(a)*.
(d): Since $\mho_C$ is idempotent (Proposition 3.5).
(e): ($\Rightarrow$) By applying Ω_C and by *(d)*; ($\Leftarrow$) By applying $\mho_C$.
(f): $\mho_C(\Omega_C(D) \sqcap \Omega_C(E)) =$ (Proposition 3.6) $= \mho_C(\mho_C(\Omega_C(D)) \sqcap \mho_C(\Omega_C(E))) = \mho_C(\mho_C(D) \sqcap \mho_C(E)) =$ (Proposition 3.6) $= \mho_C(D \sqcap E)$. $\square$

It is important to remark that the least disjunctive basis operator is neither monotonic nor anti-monotonic (hence it is not a closure), as shown below.

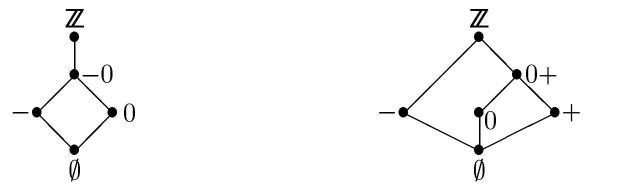
Example 5.3 Consider the abstract domains *Sign* and *Sign*$^+$ of Example 2.3, where *Sign* $\sqsubseteq$ *Sign*$^+$ (the concrete domain is as usual $C = \wp(\mathbb{Z})$). The least disjunctive basis $\Omega_C(\mathit{Sign})$ is in Example 4.4, while it is simple to check that $\Omega_C(\mathit{Sign}^+) = \mathit{Sign}^+$. This proves that the least disjunctive basis operator is neither monotonic nor anti-monotonic, since $\Omega_C(\mathit{Sign})$ and $\Omega_C(\mathit{Sign}^+)$ are uncomparable abstractions of C. $\square$

Combining points *(e)* and *(f)* of Proposition 5.2, we get an interesting form of *compositionality* of the least disjunctive basis operator with respect to the reduced product of abstract domains.

Corollary 5.4 $\Omega_C(D \sqcap E) = \Omega_C(\Omega_C(D) \sqcap \Omega_C(E))$.

Domain decomposition by complementation and least disjunctive bases can be combined to exploit this form of compositionality of the least disjunctive basis operator. Indeed, complementation provides binary decompositions of abstract domains, and therefore least disjunctive bases can be computed compositionally. The following is a simple example exploiting the above result on compositionality.

Example 5.5 Consider the abstract domains *Sign* and *Sign*$^+$ of Example 2.3, and suppose that *Sign* has been incrementally designed by reduced product of *Sign*$^+$ and of the domain *Sign*$^{-0}$ given below (the concrete domain is as usual $C = \wp(\mathbb{Z})$).



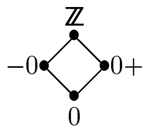
Sign^{-0} $\qquad$ $\Omega_C(\mathit{Sign}^+) \sqcap \Omega_C(\mathit{Sign}^{-0})$

Exploiting Corollary 5.4, we can compositionally compute the least disjunctive basis $\Omega_C(\mathit{Sign})$ (in Example 4.4) of *Sign* from the least disjunctive bases $\Omega_C(\mathit{Sign}^+)$ and $\Omega_C(\mathit{Sign}^{-0})$ of its factors. Indeed, the domain $\Omega_C(\mathit{Sign}^+) \sqcap \Omega_C(\mathit{Sign}^{-0})$, depicted above, is a proper abstraction of the starting domain $\mathit{Sign} = \mathit{Sign}^+ \sqcap \mathit{Sign}^{-0}$, and therefore the task of computing the least disjunctive basis of $\Omega_C(\mathit{Sign}^+) \sqcap \Omega_C(\mathit{Sign}^{-0})$ is more simple. $\square$

Characterizing Disjunctive Redundancies. The notion of least disjunctive basis allows to give a formal definition for the redundant disjunctive information of an abstract domain. Assume that D is an abstract interpretation of C and that $\Omega_C(D)$ exists. We define the *redundant disjunctive information* of the abstract domain D as the set $D \setminus \Omega_C(D)$ given by the set-theoretic difference between D and its least disjunctive basis $\Omega_C(D)$. Hence, $D \setminus \Omega_C(D)$ is precisely the subset of D containing all and only those elements of D which can be systematically constructed by disjunctive completion with basis D.

Exploiting complementation, we can introduce another related interesting concept. We define the *abstraction of disjunctive redundancy* of D as the domain $D \sim \Omega_C(D)$, namely the complement of the least disjunctive basis $\Omega_C(D)$ with respect to D itself. Thus, $D \sim \Omega_C(D)$ characterizes precisely the most abstract domain (of D, and hence of C) which composed by reduced product with the least disjunctive basis of D gives D back. Obviously, this definition makes sense whenever the involved least disjunctive basis and complement exist. Roughly speaking, the abstraction of disjunctive redundancy captures the least abstraction of D containing the redundant disjunctive information of D and which recovers the entire domain D when composed by reduced product with the least disjunctive basis of D. We illustrate these notions by our simple running example.

Example 5.6 Consider again *Sign* and $\mho_C(Sign) = Sign^\vee$ in Fig. 1. From Example 4.4 we know that $Sign \sqsubset \Omega_C(Sign)$, where $C = \wp(\mathbb{Z})$. The set of the redundant disjunctive information of *Sign* is therefore given by $\{-0, 0+\}$. In fact, both -0 and $0+$ can be reconstructed by the subsets of the least disjunctive basis $\{-, 0\}$ and $\{0, +\}$, respectively. On the other hand, it is easy to verify that the abstraction of disjunctive redundancy of *Sign* is the domain depicted below.



$Sign \sim \Omega_C(Sign)$

□

6 Functional Programming: Optimizing Comportment Analysis

We apply the theory of the least disjunctive basis to the *comportment* analysis, designed by Cousot and Cousot in [16] to generalize Mycroft's *strictness* and *termination* analysis ([32,33]), Wadler and Hughes' *projection* analysis ([37]), and Hunt's *PER* analysis ([27]). The comportment analysis applies to higher order monomorphically typed lazy functional programming languages.

To illustrate Cousot and Cousot's comportment analysis, we consider abstract interpretation of a simply typed lambda calculus with basic types β. Let denote

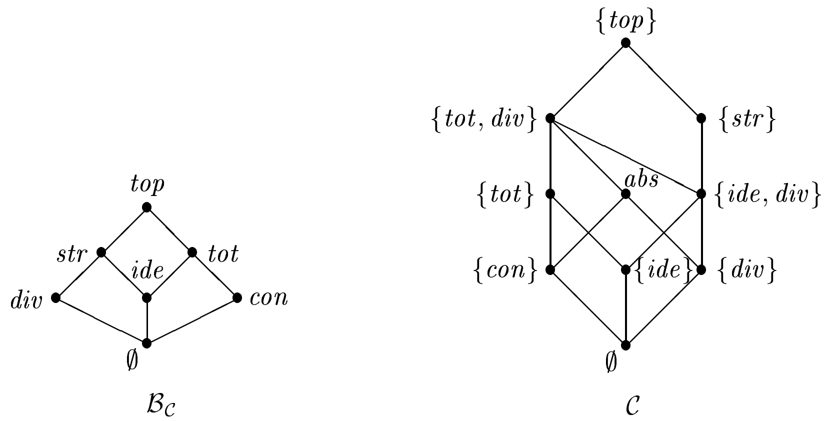


Fig. 3. The abstract domains for basic comportment $\mathcal{B}_\mathcal{C}$ and comportment $\mathcal{C}$.

truth	$\gamma^{\beta\to\beta}(top)$	$=$	$D^{\beta\to\beta}$
strictness	$\gamma^{\beta\to\beta}(str)$	$=$	$\{f \mid f(\bot)=\bot\}$
totality	$\gamma^{\beta\to\beta}(tot)$	$=$	$\{f \mid \forall x \in D^\beta \setminus \{\bot\}.\, f(x) \neq \bot\}$
identity	$\gamma^{\beta\to\beta}(ide)$	$=$	$\{f \mid \forall x \in D^\beta.\, f(x)=\bot \Leftrightarrow x=\bot\}$
divergence	$\gamma^{\beta\to\beta}(div)$	$=$	$\{f \mid \forall x \in D^\beta.\, f(x)=\bot\}$
convergence	$\gamma^{\beta\to\beta}(con)$	$=$	$\{f \mid \forall x \in D^\beta.\, f(x)\neq\bot\}$
falsity	$\gamma^{\beta\to\beta}(\emptyset)$	$=$	$\emptyset$

Fig. 4. Basic comportment analysis $\mathcal{B}_\mathcal{C}$.

D^τ the domain of values of a type τ, and $\bot$ its bottom element. For simplicity, we will consider abstractions of basic function types $\beta \to \beta$ (i.e., elements in $D^{\beta\to\beta} = D^\beta \to D^\beta$, the lattice of continuous functions from D^β to D^β ordered pointwise). The abstract domain $\mathcal{B}_\mathcal{C}$ depicted in Fig. 3 represents the lattice of *basic comportment* analysis, ordered with respect to the approximation order, for basic function types $\beta \to \beta$. The meaning of basic comportments in $\mathcal{B}_\mathcal{C}$ is given in Fig. 4, in terms of a concretization function $\gamma^{\beta\to\beta}$ mapping basic comportments into $\langle\wp(D^{\beta\to\beta}), \subseteq\rangle$, which is the concrete domain of the collecting semantics. It is immediate to observe that $\mathcal{B}_\mathcal{C}$ is precisely the reduced product of the standard Mycroft lattices for *strictness* ($div < str < top$, denoted by $\mathcal{S}$) and *termination* ($con < tot < top$, denoted by $\mathcal{T}$) analysis (more precisely, using complementation, $\mathcal{B}_\mathcal{C} \sim \mathcal{S} = \mathcal{T}$ and $\mathcal{B}_\mathcal{C} \sim \mathcal{T} = \mathcal{S}$).

As proved by Cousot and Cousot in [16], more precise comportment properties for higher-order functional languages can be characterized by disjunctive completion of the lattice $\mathcal{B}_\mathcal{C}$ of basic comportment analysis. In order to exploit sets of values, Cousot and Cousot considered a powerset completion of the abstract domain, used

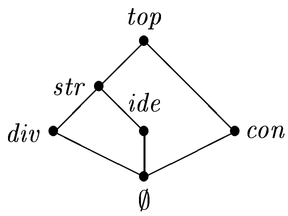


Fig. 5. The least disjunctive basis $\Omega_{\wp(D^{\beta\to\beta})}(\mathcal{B}_{\mathcal{C}})$.

to mimic the collecting semantics construction. The corresponding abstract domain for comportments is derived by reduction of the powerset completion of the basic comportment lattice $\mathcal{B}_{\mathcal{C}}$, i.e. sets of basic comportments denoting the same object in $\wp(D^{\beta\to\beta})$ are identified. Thus, the meaning of sets Ψ of basic comportments is given by a concretization $\gamma^{\wp}$ such that $\gamma^{\wp}(\Psi) = \cup\{\gamma^{\beta\to\beta}(\psi) \mid \psi \in \Psi\}$. The lattice $\mathcal{C}$ in Fig. 3, ordered by the approximation order, corresponds precisely to this (extended) comportment analysis for basic function types $\beta \to \beta$. The new element *abs* corresponds here to the set of basic comportments $\{con, div\}$ (it represents the property of absence, cf. [16]).

The disjunctive completion of $\mathcal{B}_{\mathcal{C}}$ is the lattice $\mathcal{C}$ in Fig. 3 since:

(i) $\gamma^{\wp}(\{con, div\}) \subset \gamma^{\beta\to\beta}(con \vee div)$,
(ii) $\gamma^{\wp}(\{ide, div\}) \subset \gamma^{\beta\to\beta}(ide \vee div)$,
(iii) $\gamma^{\wp}(\{tot, div\}) \subset \gamma^{\beta\to\beta}(tot \vee div)$,

while for any other $\Psi \subseteq \mathcal{B}_{\mathcal{C}}$, $\gamma^{\wp}(\Psi) = \gamma^{\beta\to\beta}(\vee\Psi)$. For instance, for any basic type β, the identity map $\lambda x^{\beta}.x^{\beta}$ is such that $\lambda x^{\beta}.x^{\beta} \in \gamma^{\beta\to\beta}(con \vee div) = \gamma^{\beta\to\beta}(top) = D^{\beta\to\beta}$, whilst $\lambda x^{\beta}.x^{\beta} \notin \gamma^{\wp}(\{con, div\}) = \gamma^{\beta\to\beta}(con) \cup \gamma^{\beta\to\beta}(div)$.

To find out the least disjunctive basis of $\mathcal{B}_{\mathcal{C}}$ in $\wp(D^{\beta\to\beta})$, viz. the least disjunctive basis for the lattice of basic comportment analysis (which, by Theorem 4.10 or Theorem 4.11, exists since $\langle\wp(D^{\beta\to\beta}), \subseteq\rangle$ is co-algebraic and completely distributive), we simply apply Corollary 4.14, that is we compute the Moore-closure of the join-irreducible elements of the disjunctive completion $\mathcal{C}$ of $\mathcal{B}_{\mathcal{C}}$. It is straightforward to verify that the least disjunctive basis $\Omega_{\wp(D^{\beta\to\beta})}(\mathcal{B}_{\mathcal{C}})$ is the lattice depicted in Fig. 5. The least disjunctive basis $\Omega_{\wp(D^{\beta\to\beta})}(\mathcal{B}_{\mathcal{C}})$ is therefore a proper abstraction of the original basis $\mathcal{B}_{\mathcal{C}}$ of basic comportments. The redundant disjunctive information is given by totality *tot*, which can be recovered as *lub* of the elements *ide* and *con* representing identity and convergence, respectively.

Local Optimization. The least disjunctive basis can be combined with complementation in order to systematically simplify portions of abstract domains. Let C be a concrete domain and let $D \sqsubseteq E$ be abstractions of C. The domain D can be locally simplified by removing the disjunctive information which is not expressible by E. We call this method *local optimization*. The local optimization of D with respect to

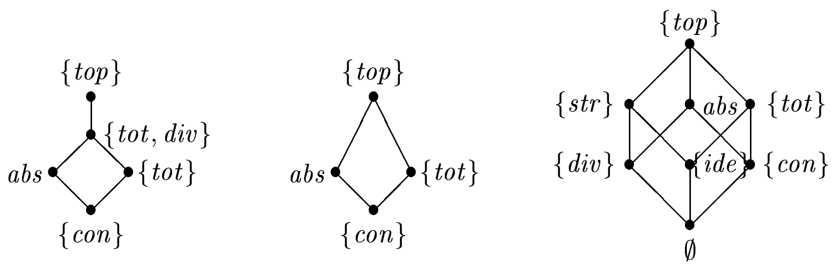
E is thus defined as $\Omega_C(D \sim E) \sqcap E$. Obviously, this definition makes sense whenever the involved least disjunctive basis and complement exist. As proved by the proposition below, the local optimization of D with respect to E is an abstraction of D. Intuitively, the only disjunctive information which is preserved when moving from D to $\Omega_C(D \sim E) \sqcap E$ is that directly depending on the information in E. Any other form of disjunctive information is removed. In this sense, all the disjunctive information in D and in E is saved, while the disjunctive information of D which does not directly depend on E is removed. The following proposition also shows that the local optimization of D with respect to E has a more concrete least disjunctive basis than that of D, while it has the same disjunctive completion as D.

Proposition 6.1 *Let $C \sqsubseteq D \sqsubseteq E$. Then,*

(a) $\mho_C(\Omega_C(D \sim E) \sqcap E) = \mho_C(D)$;
(b) $D \sqsubseteq \Omega_C(D \sim E) \sqcap E \sqsubseteq \Omega_C(D)$;

Proof. *(a)* $\mho_C(\Omega_C(D \sim E) \sqcap E) =$ (Proposition 3.6) $= \mho_C(\mho_C(\Omega_C(D \sim E)) \sqcap \mho_C(E)) = \mho_C(\mho_C(D \sim E) \sqcap \mho_C(E)) =$ (Proposition 3.6) $= \mho_C((D \sim E) \sqcap E) = \mho_C(D)$.
(b) Since $D \sqsubseteq D \sim E \sqsubseteq \Omega_C(D \sim E)$, we have that $D = D \sqcap E \sqsubseteq \Omega_C(D \sim E) \sqcap E$. On the other hand, $\Omega_C(D \sim E) \sqcap E \sqsubseteq \Omega_C(D)$ follows by *(a)*. $\square$

As an example, we apply local optimization to the comportment domain, which clearly contains too much information with respect to strictness. The lattice $\mathcal{C} \sim \mathcal{S}$ depicted below captures precisely *nonstrictness* analysis. In this case, by removing the element $\{tot, div\}$ we get the least disjunctive basis $\Omega_{\mathcal{C}}(\mathcal{C} \sim \mathcal{S})$. Then, by adding strictness via reduced product we obtain the lattice $\Omega_{\mathcal{C}}(\mathcal{C} \sim \mathcal{S}) \sqcap \mathcal{S}$, which is properly more abstract than $\mathcal{C}$ yet containing strictness, termination and basic comportment.



$\mathcal{C} \sim \mathcal{S}$ $\qquad$ $\Omega_{\mathcal{C}}(\mathcal{C} \sim \mathcal{S})$ $\qquad$ $\Omega_{\mathcal{C}}(\mathcal{C} \sim \mathcal{S}) \sqcap \mathcal{S}$

Here, $\{tot, div\}$ is removed because it can be obtained by disjunctive completion of objects that do not include strictness str: $abs \vee \{tot\} = \{tot, div\}$. The divergence information in $\{tot, div\}$ is here inherited from abs, and not directly from any of the elements in $\mathcal{S}$. It is also worth noting that $\{ide, div\}$ cannot be reconstructed by reduced product from $\mathcal{S}$ and $\Omega_{\mathcal{C}}(\mathcal{C} \sim \mathcal{S})$.

7 Logic Programming: Optimizing Disjunctive Ground-Dependency Analysis

In this section, we apply the theory of the least disjunctive basis to *Pos*, a well known relational abstract domain of propositional formulae for ground-dependency analysis of (constraint) logic programs ([1,6,30]). The disjunctive completion of *Pos* has been studied by Filé and Ranzato in [19], where it has been shown that it is strictly more precise than *Pos* itself. In a sense, this was a surprising result, since the fact that *Pos* is closed under logical disjunction should lead to an opposite conclusion. We show that the least disjunctive basis for the disjunctive completion of *Pos* is the domain *Def*, which is a proper abstraction of *Pos*. *Def* is a domain of propositional formulae introduced by Dart in [17] for groundness analysis in deductive databases, and used by Marriott and Søndergaard in [30] for ground-dependency analysis of logic programs. Armstrong et al. in [1] investigated various representations for the formulae in *Pos* and *Def*, and they experimentally compared precision and efficiency of static program analyses using these two domains. They showed that analyses using *Pos* achieve higher precision than those using *Def*, although there is a relatively small additional cost. However, this additional cost becomes relevant when lifting *Pos* and *Def* to the powerset, due to the combinatorial explosion of the disjunctive completion. In view of the work in [1], the results of this section gain an important and significant practical impact: The disjunctive ground-dependency analysis of logic programs can always be obtained by disjunctive completion of *Def*, without losing precision and at a lower cost with respect to the disjunctive completion of *Pos*. Moreover, this completes the understanding of the problem, since it is the best that one can do in this direction.

7.1 The Abstract Domains Pos and Def

Let *Var* be a countable set of variables, and let *VI* be any (nonempty) finite subset of *Var* containing the variables of interest. As usual, variables are denoted by $x, y, z, u, \ldots$ We assume that the concrete domain of computation of a given logic program is the powerset $\wp(Sub)$ of idempotent substitutions, ordered with set-theoretic inclusion. A substitution σ is typically specified by the set of its non-trivial bindings, viz. $\sigma = \{x/\sigma(x) \mid \sigma(x) \neq x\}$.

Pos is the finite lattice (indeed Boolean lattice) of *positive* Boolean functions on *VI*, where a Boolean function is positive if it gives *true* for the truth assignment where each variable is *true*. Obviously, the order of *Pos* is given by logical consequence $\models$, *glb* is given by logical conjunction, while *lub* on nonempty sets by logical disjunction (the *lub* of the empty set yields the bottom $\wedge VI$). *Def* is the finite lattice of positive Boolean functions on *VI* whose models are closed under intersection. Formulae in *Def* are called *definite*. For more details about *Pos* and *Def* we refer to [1,30]. It is well known that Boolean functions can be represented by means of propositional formulae. Thus, in the following, we will use propositional formulae

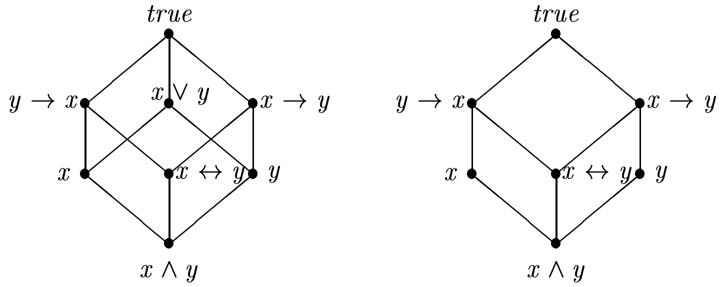


Fig. 6. The domains *Pos* and *Def* for $VI = \{x, y\}$.

over VI to represent Boolean functions in *Pos* and *Def*. In Fig. 6, *Pos* and *Def* are depicted for $VI = \{x, y\}$. As observed in [1], *Def* is a meet subsemilattice of *Pos*. Further, the top Boolean function *true* is in *Def*. Hence, *Def* is a Moore-family of *Pos*, namely, being (the set of fixpoints of) a closure operator on *Pos*, it is an abstract interpretation of *Pos*. The abstraction and concretization maps between *Pos*, *Def* and $\wp(Sub)$ are well known, and can be found, e.g., in [30]. For instance, assuming $VI = \{x, y, z, u\}$, the formula $x \wedge (y \leftrightarrow z)$ is an element of *Pos* (and *Def*) that represents the substitutions σ such that for any instance σ' of σ: (i) the term $\sigma'(x)$ is ground; (ii) $\sigma'(y)$ is ground iff also $\sigma'(z)$ is ground. In particular,[2] $\sigma_1 = \{x/a, y/b, z/c\}$ and $\sigma_2 = \{x/a, y/w, z/w, v/u\}$ satisfy this property. Thus, $\{\sigma_1, \sigma_2\} \subseteq \gamma(x \wedge (y \leftrightarrow z))$.

7.2 The Least Disjunctive Basis of Pos is Def

Filé and Ranzato showed in [19] that $\mho_{\wp(Sub)}(Pos) \sqsubset Pos$, namely there is a strict improvement in precision by lifting *Pos* to its disjunctive completion. For example, by considering as variables of interest $VI = \{x, y\}$, we have that the logical disjunction $(x \to y) \vee (y \to x)$ in *Pos* does not represent the concrete disjunction of the two formulae $x \to y$ and $y \to x$, i.e. the union of their concretizations. In fact, $\gamma(x \to y) \cup \gamma(y \to x) \subset \gamma((x \to y) \vee (y \to x)) = \gamma(true)$: $\sigma = \{x/v, y/w\}$ is such that $\sigma \in \gamma((x \to y) \vee (y \to x)) \setminus \gamma(x \to y) \cup \gamma(y \to x)$.

Sets of positive formulae for which logical (i.e., in *Pos*) and concrete (i.e., in $\wp(Sub)$) disjunctions coincide have been characterized as follows (cf. [20]).

Theorem 7.1 ([20]) *If* $\emptyset \neq \Phi \subseteq Pos$ *then*

$$\cup\{\gamma(f) \mid f \in \Phi\} = \gamma(\vee\Phi) \Leftrightarrow \forall g \in Def.\ (g \models \vee\Phi) \Rightarrow (\exists f \in \Phi.\ g \models f).$$

By Theorem 7.1, the disjunctive completion of *Pos* (after reduction), for $VI = \{x, y\}$, is the lattice depicted in Fig. 7.

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

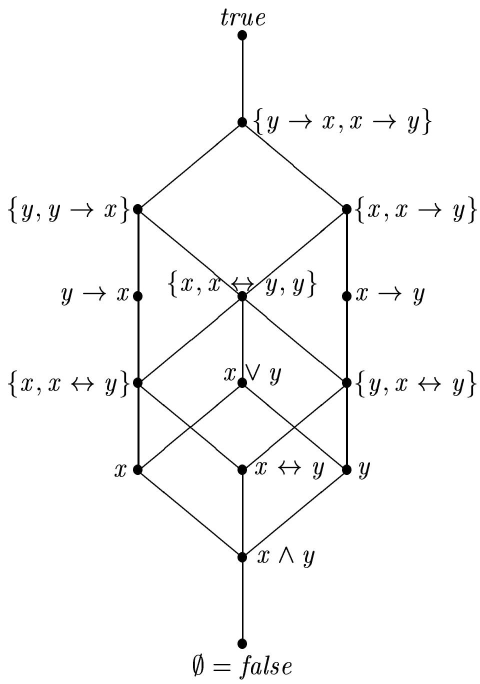


Fig. 7. $\mho_{\wp(Sub)}(Pos)$ for $VI = \{x, y\}$.

By Theorem 4.10, $\Omega_{\wp(Sub)}(Pos)$ exists, since $\langle \wp(Sub), \subseteq \rangle$ is collecting (on the other hand, Theorem 4.11 may be exploited as well). The complexity of the simple case of two variables shows that the application of the method of join-irreducible elements of Corollary 4.14 to compute the least disjunctive basis of Pos can result quite hard. The main result of this section can however be proved directly on the definition of Def and Pos.

Theorem 7.2 $\Omega_{\wp(Sub)}(Pos) = Def$.

Proof. We exploit Proposition 4.3. Therefore, let us first show that $\mho_{\wp(Sub)}(Def) = \mho_{\wp(Sub)}(Pos)$. Since $Pos \sqsubseteq Def$, and, by Proposition 3.5, $\mho_{\wp(Sub)}$ is monotonic, we have $\mho_{\wp(Sub)}(Pos) \sqsubseteq \mho_{\wp(Sub)}(Def)$. Now, consider any $f \in Pos$, and $\Phi_f = \{g \in Def \mid g \models f\} \subseteq Def$. It is known (cf. [1]) that in this case, $f = \vee\Phi_f$ holds. We want to verify that $\gamma(\vee\Phi_f) = \cup\gamma(\Phi_f)$. If $h \in Def$ and $h \models \vee\Phi_f = f$ then $h \in \Phi_f$, and therefore, by Theorem 7.1, we have that $\gamma(\vee\Phi_f) = \cup\gamma(\Phi_f)$. This means that for any $f \in Pos$ there exists $S \subseteq Def$ such that $\gamma(f) = \cup\gamma(S)$, and hence $\mho_{\wp(Sub)}(Def) \sqsubseteq Pos$. Hence, applying the idempotent and monotonic (cf. Proposition 3.5) operator $\mho_{\wp(Sub)}$, we get the other inclusion, $\mho_{\wp(Sub)}(Def) \sqsubseteq \mho_{\wp(Sub)}(Pos)$. Now, assume that there exists D abstracting $\wp(Sub)$ such that $\mho_{\wp(Sub)}(D) = \mho_{\wp(Sub)}(Pos)$. Then, we show that $D \sqsubseteq Def$. Assume that $f \in Def$. Since $Def \subseteq \mho_{\wp(Sub)}(Def) = \mho_{\wp(Sub)}(Pos) = \mho_{\wp(Sub)}(D)$, we have that $f \in \mho_{\wp(Sub)}(D)$. Hence, by Proposition 3.11, there exists $S \subseteq D$ such that $\gamma(f) = \cup\gamma(S)$. Since $D \subseteq \mho_{\wp(Sub)}(Pos)$, for any $x \in S$ there exists $F_x \subseteq Pos$ such that $\gamma(x) = \cup\gamma(F_x)$. Hence, there exists $\Phi \subseteq Pos$ such that $\gamma(f) = \cup\gamma(\Phi)$. Applying the Pos-abstraction map, and exploiting its additivity, we get $f = \vee\Phi$, i.e. $\gamma(\vee\Phi) = \cup\gamma(\Phi)$. Thus, by Theorem 7.1, we have that for any $g \in Def$, $g \models \vee\Phi$ implies that there exists $h \in \Phi$ such that $g \models h$. By using this implication for $f = \vee\Phi \in Def$, we get that there exists $h \in \Phi$ such that $\vee\Phi \models h$, i.e. $f = \vee\Phi \in \Phi$. Therefore, there exists $x \in S$ such that $f \in F_x$, and $\gamma(f) \subseteq \gamma(x)$. But, since $\gamma(f) = \cup\gamma(S)$, we have that $\gamma(x) = \gamma(f)$, and thus $f = x$, from which $f \in D$. □

Indeed, it is simple to verify on the previous diagrams for the case of two variables $VI = \{x, y\}$, that Def is the least abstraction of Pos having the same disjunctive completion. This particular case is also verifiable by applying Corollary 4.14: in fact, Def is precisely the Moore-closure of the join-irreducible elements of $\mho_{\wp(Sub)}(Pos)$. Moreover, $\Omega_{\wp(Sub)}(Pos) = Def$, while $\Omega_{Pos}(Pos) = \mathcal{M}(JI(Pos))$, and for the case $VI = \{x, y\}$, the Moore-closure of the join-irreducible elements of Pos clearly does not coincide with Def (for instance, $y \to x \in Def \setminus \mathcal{M}(JI(Pos))$). This also proves that the least disjunctive basis operator depends on the concrete domain of reference, as postulated in Section 5. Finally, note that the redundant disjunctive information of Pos is obviously given by the formula $x \vee y$, which can be reconstructed in the disjunctive completion of the least disjunctive basis Def as $\{x, y\}$.

8 Related and Further Work

To the best of our knowledge, this is the first systematic characterization of least bases for the disjunctive completion of domains in abstract interpretation. However, the use of join-irreducible elements to represent disjunctive properties is definitely not new, in particular in relation with the work of Flemming Nielson. Join-irreducible elements were first investigated in the context of abstract interpretation in [35], with the aim of giving an alternative (more concise) representation for the relational (Hoare) powerdomain in analysis of typed functional languages. We extend Nielson's idea in the definition of our notion of least disjunctive basis. Least disjunctive bases are more general in this sense, since join-irreducible elements can only represent domains which are already disjunctive, provided that the hypotheses of Theorem 4.10 or Theorem 4.11 are satisfied (cf. Theorem 4.13). In [35], Nielson investigates also the situation where the abstraction function maps join-irreducible elements to join-irreducible elements, defining a notion of expected form for an abstract interpretation, further studied in [36]. This could be a related topic, and it should provide an interesting application of least disjunctive bases. Nielson suggests the use of expected forms in order to simplify the implementation of functionals induced in abstract interpretation of denotational semantics. The aim of expected forms is therefore similar to that of least disjunctive bases, both providing sensible simplifications in abstract interpretation design. In particular, some expected forms defined on collecting semantics, i.e. on some powerset domain (e.g. for *cond* in [36]), can be viewed as functionals on the least disjunctive basis of the abstract domain.

We conclude by observing that least disjunctive bases could be also applied in semantics of programming languages. Cousot and Cousot proved in [11,15] that many semantics of programming languages can be derived by abstract interpretation from a more concrete (usually operational) semantics. However, collecting semantics are generally defined exploiting a (more abstract) standard semantics, by "collecting" sets of possible output values corresponding to a given set of possible input values (e.g. [34]). This construction can be achieved by a disjunctive completion of the standard semantics, relatively to a more concrete operational semantics. The meaning of the least disjunctive basis should therefore be evident in semantics as well as in analysis: The least disjunctive basis of a collecting semantics is precisely the most abstract (standard) semantics which induces the same collecting semantics, and our results allow to characterize this "optimal" semantics as an abstract interpretation of the operational one. A similar application to semantics has been presented in [24], where complementation, i.e. the inverse of reduced product, has been used to derive new semantics for programming languages.

Acknowledgments

We would like to thank Patrick Cousot, Gilberto Filé and Harald Søndergaard for many stimulating discussions on the subject of this paper. We are also grateful to one anonymous referee for many helpful suggestions.

References

[1] T. Armstrong, K. Marriott, P. Schachte, and H. Søndergaard. Two classes of Boolean functions for dependency analysis. To appear in *Sci. Comput. Program.* A preliminary version appeared in Proc. SAS '94, LNCS 864.

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

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

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

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

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

[7] P. Cousot. *Méthodes Itératives de Construction et d'Approximation de Points Fixes d'Opérateurs Monotones sur un Treillis, Analyse Sémantique des Programmes.* PhD thesis, Université Scientifique et Médicale de Grenoble, 1978.

[8] P. Cousot. Abstract interpretation. *ACM Comp. Surv.*, 28(2):324–328, 1996.

[9] P. Cousot. Program analysis: the abstract interpretation perspective. *ACM Comp. Surv.*, 28(4), 1996.

[10] P. Cousot. Types as abstract interpretations (invited paper). In *Conference Record of the 24th ACM Symp. on Principles of Programming Languages (POPL '97)*, pages 316–331. ACM Press, New York, 1997.

[11] P. Cousot. Design of semantics by abstract interpretation. Slides of an invited talk at *13th Int. Symp. on Mathematical Foundations of Programming Semantics (MFPS '97)*, 1997.

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

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

[14] P. Cousot and R. Cousot. Abstract interpretation and application to logic programs. *J. Logic Program.*, 13(2–3):103–179, 1992.

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

[16] P. Cousot and R. Cousot. Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and PER analysis of functional languages) (invited paper). In *Proceedings of the IEEE Int. Conf. on Computer Languages (ICCL '94)*, pages 95–112. IEEE Computer Society Press, Los Alamitos, Calif., 1994.

[17] P. Dart. On derived dependencies and connected databases. *J. Logic Program.*, 11(2):163–188, 1991.

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

[19] G. Filé and F. Ranzato. Improving abstract interpretations by systematic lifting to the powerset. In M. Bruynooghe, editor, *Proceedings of the 1994 Int. Logic Programming Symp. (ILPS '94)*, pages 655–669. The MIT Press, Cambridge, Mass., 1994.

[20] G. Filé and F. Ranzato. The powerset operator on abstract interpretations. To appear in *Theor. Comput. Sci.*, 1998.

[21] G. Filé and F. Ranzato. Complementation of abstract domains made easy. In M. Maher, editor, *Proceedings of the 1996 Joint Int. Conf. and Symp. on Logic Programming (JICSLP '96)*, pages 348–362. The MIT Press, Cambridge, Mass., 1996.

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

[23] R. Giacobazzi and F. Ranzato. Functional dependencies and Moore-set completions of abstract interpretations and semantics. In J. Lloyd, editor, *Proceedings of the 1995 Int. Logic Programming Symp. (ILPS '95)*, pages 321–335. The MIT Press, Cambridge, Mass., 1995.

[24] R. Giacobazzi and F. Ranzato. Complementing logic program semantics. In M. Hanus and M. Rodríguez Artalejo, editors, *Proceedings of the 5th Int. Conf. on Algebraic and Logic Programming (ALP '96)*, volume 1139 of *Lecture Notes in Computer Science*, pages 238–253. Springer-Verlag, Berlin, 1996.

[25] R. Giacobazzi and F. Ranzato. Refining and compressing abstract domains. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Proceedings of the 24th Int. Colloq. on Automata, Languages and Programming (ICALP '97)*, volume 1256 of *Lecture Notes in Computer Science*, pages 771–781. Springer-Verlag, Berlin, 1997.

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

[27] S. Hunt. PERs generalize projections for strictness analysis. In S. Peyton Jones, G. Hutton, and C. K. Holst, editors, *Proceedings of the 1990 Glasgow Functional Programming Workshop*, Workshops in Computing, pages 156–168. Springer-Verlag, Berlin, 1990.

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

[29] N. D. Jones and S. S. Muchnick. Complexity of flow analysis, inductive assertion synthesis and a language due to Dijkstra. In S. S. Muchnick and N. D. Jones, editors, *Program Flow Analysis: Theory and Applications*, pages 380–393. Prentice-Hall, 1981.

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

[31] J. Morgado. Some results on the closure operators of partially ordered sets. *Portug. Math.*, 19(2):101–139, 1960.

[32] A. Mycroft. The theory and practice of transforming call-by-need into call-by-value. In B. Robinet, editor, *Proceedings of the 4th Int. Symp. on Programming*, volume 83 of *Lecture Notes in Computer Science*, pages 270–281. Springer-Verlag, Berlin, 1980.

[33] A. Mycroft. *Abstract Interpretation and Optimising Transformations for Applicative Programs.* PhD thesis, CST-15-81, Univ. of Edinburgh, Scotland, 1981.

[34] A. Mycroft and F. Nielson. Strong abstract interpretation using power domains. In J. Díaz, editor, *Proceedings of the 10th Int. Colloq. on Automata, Languages and Programming (ICALP '83)*, volume 154 of *Lecture Notes in Computer Science*, pages 536–547. Springer-Verlag, Berlin, 1983.

[35] F. Nielson. *Abstract Interpretation using Domain Theory.* PhD thesis, CST-31-84, Univ. of Edinburgh, Scotland, 1984.

[36] F. Nielson. Expected forms of data flow analysis. In H. Ganzinger and N. D. Jones, editors, *Programs as Data Objects*, volume 217 of *Lecture Notes in Computer Science*, pages 192–205. Springer-Verlag, Berlin, 1986.

[37] P. Wadler and R. J. M. Hughes. Projections for strictness analysis. In G. Kahn, editor, *Proceedings of the Int. Conf. on Functional Programming Languages and Computer Architecture (FPCA '87)*, volume 274 of *Lecture Notes in Computer Science*, pages 385–407. Springer-Verlag, Berlin, 1987.

[38] M. Ward. The closure operators of a lattice. *Ann. Math.*, 43(2):191–196, 1942.

[39] K. Yi and W. L. Harrison III. Automatic generation and management of interprocedural program analyses. In *Conference Record of the 20th ACM Symp. on Principles of Programming Languages (POPL '93)*, pages 246–259. ACM Press, New York, 1993.