Complementation of Abstract Domains made Easy

Gilberto Filé **Francesco Ranzato**
Dipartimento di Matematica Pura ed Applicata
Università di Padova
Via Belzoni 7, 35131 Padova, Italy
{gilberto,franz}@math.unipd.it

Abstract

In standard abstract interpretation theory, the inverse of the reduced product of abstract domains was recently defined and called complementation. Given two domains C and D such that D abstracts C, the complement $C \sim D$ is the most abstract domain whose reduced product with D gives C back. We show that, when C is a continuous complete lattice, there is a particularly simple method for computing $C \sim D$. Since most domains for abstract interpretation are (complete and) continuous, this method is widely applicable. In order to demonstrate its relevance, we apply this result and some of its consequences to Cousot and Cousot's domain for integer interval analysis of imperative programs, and to several well-known domains for the static analysis of logic languages, viz., *Pos*, *Def* and *Sharing*. In particular, we decompose *Sharing* in three more abstract domains whose reduced product gives back *Sharing*, and such that each component corresponds to one of the three properties that coexist in the elements of *Sharing*: ground-dependency, pair-sharing (or equivalently variable independence) and set-sharing. Using our theory, we minimize each component of this decomposition obtaining in some case domains that are surprisingly simpler than the corresponding original components.

1 Introduction

The standard Cousot and Cousot theory of abstract interpretation ([8, 9]) naturally supports the enhancement of data-approximations, providing some basic operators acting on abstract domains. Thus, abstract domains for analysis can be incrementally designed by successive abstractions, and more precise approximations can be obtained by combining domains or by lifting them by suitable property-completions (e.g. the completions in [10, 12, 14]).

The *reduced product* ([9]) is probably the most common and known operation for composing abstract domains. Its practical interest in the field of logic program analysis has been shown in [3].

In [4], it has been shown that the inverse of the reduced product, called *complementation*, exists in most cases of interest. Examples of complementation for different programming paradigms were presented there.

In the present paper, we show that if a domain C is *meet-generated by its meet-irreducible elements* then two facts hold: (i) there exists the complement $C \sim D$ of every abstraction D of C; (ii) there is a particularly simple and clean way to compute $C \sim D$. The importance of this method for computing complements can be fully appreciated when contrasting its simplicity with the rather technical proofs used in [4, 5] for the same task. This simplicity stems from the fact that, in order to compute

$C \sim D$, it suffices to consider the behaviour of the meet-irreducible elements of C under the abstraction D. This method becomes even simpler whenever the domain C is *dual-atomistic*, because in this case the meet-irreducible elements coincide with the *dual-atoms* which are easy to identify. Since all *continuous* (and hence *algebraic*) complete lattices are meet-generated by their meet-irreducible elements, our result is widely applicable. In fact, it can be used for finding complements for all abstract domains for static analysis that we know of .

In order to illustrate the relevance of these theoretical results, we apply them to the abstract domain for integer interval analysis of imperative programs with integer variables ([7, 8]), and to several well-known domains for aliasing analysis of logic languages. Namely, the domain *Pos* ([1, 19]) of positive propositional formulae that is used for ground-dependency analysis, the domain *Def* ([1, 19]) that is the abstraction of *Pos* consisting of the conjunctions of definite propositional clauses, and the domain *Sharing* ([17]) for ground-dependency and variable independence.

Since *Pos* and *Sharing* are dual-atomistic, the computation of complements for these domains is really very easy. The main results shown are as follows.

1. We characterize $Pos \sim Def$. As expected, $Pos \sim Def$ consists of the conjunction of non definite clauses, i.e., containing disjunctions with at least two variables in their conclusion.

2. We show that *Sharing* can be decomposed into three components, whose reduced product gives *Sharing* back, each representing one of the elementary properties that coexist in the elements of *Sharing*, and that are as follows: (i) the ground-dependency information; (ii) the pair-sharing information, or equivalently variable independence; (iii) the set-sharing information without variable independence and ground-dependency.

There are two general phenomena, illustrated by the decomposition of *Sharing*, that it is worth pointing out.

- The abstract domain, called PS, that expresses most naturally pair-sharing is such that $Sharing \sim PS = Sharing$. Thus, in general, there may be a domain D that abstracts C and expresses a sensible property contained in C together with other properties, but which is "too abstract" to be "extracted" from C through complementation. This happens when D does not represent exactly any meet-irreducible element of C.

- Starting from a decomposition of *Sharing* in three components, each expressing only one of the above properties, our theory allows us to minimize it finding for each component the most abstract domain whose reduced product with the remaining components gives *Sharing* back. The minimal components found in this way can be way more abstract than the corresponding original ones. This phenomenon is particularly striking for the component expressing pair-sharing: the original domain for pair-sharing has (at least) exponential size (in the number of variables of interest), whereas the corresponding minimal domain has only two elements! This shows that domain decomposition can lead to great gains in the size of the representation.

2 Abstract Interpretation Basics

Throughout the paper, we will assume familiarity with the standard notions of lattice theory (e.g. see [16]) and abstract interpretation ([8, 9]). Now, we briefly

introduce some notation and recall some well-known notions.

If C and D are posets and $\alpha : C \to D$, $\gamma : D \to C$ are functions such that $\forall x \in C. \forall y \in D.\ \alpha(x) \leq_D y \Leftrightarrow x \leq_C \gamma(y)$, then the quadruple (C, α, D, γ) is a *Galois connection* (G.c.) between C and D. If in addition γ is 1-1, or, equivalently, α is onto then (C, α, D, γ) is a *Galois insertion* (G.i.) of D in C. In the setting of abstract interpretation, 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 the *concretization* maps, respectively. Also, D is called an *abstraction* (or *abstract interpretation*) of C, and C a *concretization* of D. If (C, α, D, γ) is a Galois insertion, each value of the abstract domain D is useful in the representation of the concrete domain C as all the elements of D represent distinct members of C. Moreover, 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* on the poset C is an operator $\rho : C \to C$ monotonic, idempotent and extensive, viz. for all $x \in C$, $x \leq \rho(x)$. If $\langle C, \leq, \wedge, \vee, \top, \bot \rangle$ is a complete lattice then each closure operator ρ is uniquely determined by the set of its fixpoints, which is its image, i.e. $\rho(C) = \{x \in C \mid \rho(x) = x\}$. A subset $X \subseteq C$ is the set of fixpoints of an upper closure operator iff X is a *Moore-set*, i.e. $\top \in X$ and X is meet-closed, viz. for any $Y \subseteq X$, with $Y \neq \emptyset$, $\wedge Y \in X$. For any $X \subseteq C$, we denote by $\mathcal{M}(X)$ the *Moore-closure* of X, i.e. the least subset of C containing X which is a Moore-set of C. Furthermore, if ρ is an upper closure then the set of fixpoints $\rho(C)$ is a complete lattice with respect to the order of C, but, in general, it is not a complete sub-lattice of C, since the *lub* in $\rho(C)$ might be different from that in C. In the following, we will often denote a closure operator by the set of its fixpoints. We denote by $\langle uco(C), \sqsubseteq, \sqcap, \sqcup, \lambda x.\top, \lambda x.x \rangle$ the complete lattice of all upper closures on the complete lattice C, where for every $\rho, \eta \in uco(C)$, $\{\rho_i\}_{i \in I} \subseteq uco(C)$ and $x \in C$: (i) $\rho \sqsubseteq \eta$ iff $\forall x \in C.\ \rho(x) \leq \eta(x)$, or, equivalently, $\rho \sqsubseteq \eta$ iff $\eta(C) \subseteq \rho(C)$; (ii) $(\sqcup_{i \in I} \rho_i)(x) = x \Leftrightarrow \forall i \in I.\ \rho_i(x) = x$; (iii) $(\sqcap_{i \in I} \rho_i)(x) = \wedge_{i \in I} \rho_i(x)$; (iv) $\lambda x.\top$ is the top element, whereas $\lambda x.x$ is the bottom element. More details on closure operators can be found in [16].

A key point in Cousot and Cousot abstract interpretation theory is the equivalence between the Galois insertion and closure operator approach to the design of abstract domains. 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 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.. 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 most of the times this approach, and whenever we will say that D is an abstraction of C, we will mean that D is isomorphic to $\rho_D(C)$ (denoted by $D \cong \rho_D(C)$), 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* ([9]) 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 $\sqcap$.

3 Complementation and Decompositions

Complementation ([4]) corresponds to the *inverse of the reduced product*, namely 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 equivalence between closure operators and abstract interpretation, the above notion of complementation corresponds precisely to *pseudo-complementation* for ρ_D in $uco(C)$.[1] The following result is recalled from [13].[2]

Theorem 3.1 ([13]) *If C is meet-continuous then $uco(C)$ is pseudo-complemented.*

Cortesi *et al.* ([4]) applied this result for the first time in abstract interpretation, using the above notion of complementation, which is more precisely formulated as follows: whenever C is meet-continuous, the complement $C \sim D$ exists, and is defined as follows: $C \sim D = \sqcup\{\rho \in uco(C) \mid (\rho_D \sqcap \rho)(C) = C\}$. In particular, [4] observed that meet-continuity is satisfied in most domains for abstract interpretation and analysis. Let $C \sqsubseteq D, E$ and $\top$ be the most abstract interpretation of C (which is the closure $\lambda x.\top$). The following are some basic algebraic properties of complementation ([4]):

$$\begin{array}{ll} (a) & D \sqsubseteq C \sim (C \sim D); \\ (b) & (D \sqsubseteq E) \Rightarrow (C \sim E) \sqsubseteq (C \sim D); \\ (c) & (C \sim D) = C \sim (C \sim (C \sim D)); \\ (d) & C \sim \top = C \text{ and } C \sim C = \top. \end{array}$$

Complementation is essential 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 decompositions is twofold: (1) it provides more compact representations for complex domains, enhancing space saving techniques, and (2) it simplifies verification problems for complex domains, like correctness, by decomposing them into simpler problems for the corresponding factors. It is natural to give the following definition of minimal decomposition.

[1] If L is a meet-semilattice then the *pseudo-complement* of $x \in L$, if it exists, is the (unique) element $x^* \in L$ such that $x \wedge x^* = \bot$ and $\forall y \in L.\ (x \wedge y = \bot) \Rightarrow (y \leq x^*)$. In a complete lattice L, if the pseudo-complement x^* exists then $x^* = \vee\{y \in L \mid x \wedge y = \bot\}$. If every $x \in L$ has the pseudo-complement, L is *pseudo-complemented*.

[2] A complete lattice C is *meet-continuous* if for any chain $Y \subseteq C$ and $x \in C$, $x \wedge (\vee Y) = \vee_{y\in Y}(x \wedge y)$.

Definition 3.2 If $\langle D_i \rangle_{i \in I}$ is a decomposition for C, i.e. $C = \sqcap_{i \in I} D_i$, then it is a *minimal decomposition* if for all $k \in I$, $(D_k \sqsubset E_k) \Rightarrow (C \sqsubset (\sqcap_{i \in I \setminus \{k\}} D_i) \sqcap E_k)$. □

As the following lemma states, complementation naturally induces minimal decompositions.

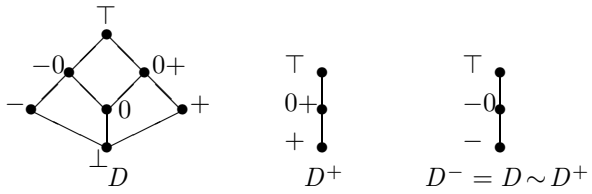
Lemma 3.3 *Assume that $uco(C)$ is pseudo-complemented. $\langle D_i \rangle_{i \in I}$ is a decomposition for C such that for any $k \in I$, $C \sim (\sqcap_{i \in I \setminus \{k\}} D_i) = D_k$ iff it is minimal.*

Proof. Straightforward, from the definition of complement. ■

In particular, note that if $C \sqsubseteq D$ then, using the above lemma and point (c), $\langle C \sim D, C \sim (C \sim D) \rangle$ always yields a minimal binary decomposition for C.

We show how complementation actually works by a simple example.

Example 3.4 Consider the usual lattice D for sign analysis of an integer variable ([9]), depicted below. The concrete domain is $\langle \wp(\mathbb{Z}), \subseteq \rangle$, and concretization and abstraction maps are the most natural. The (more abstract) lattice of positive values D^+ can be "subtracted" from D by complementation (viz., $D \sim D^+$), yielding the lattice of negative values D^-.



$\langle D^+, D^- \rangle$ is therefore a decomposition for D (i.e., $D = D^+ \sqcap D^-$). It is also immediate to observe that $\langle D^+, D^- \rangle$ is actually a minimal decomposition. In particular, 0 and $\bot$, which are not in $D^+ \cup D^-$, can be both reconstructed by reduced product from D^+ and D^-. □

4 Complementation and Meet-Irreducibility

In this section, we give a new theoretical result assuring that the complete lattice of closure operators on a complete lattice is pseudo-complemented. This result will also give a very useful tool for the computation of the pseudo-complement in most practical cases.

First, we need some standard definitions of lattice theory.[3] In the following, assume that $\langle C, \leq, \wedge, \vee, \top, \bot \rangle$ is a complete lattice. If $X \subseteq C$ then C is *meet-generated* by X if $C = \mathcal{M}(X)$. Note that for any $X \subseteq C$, $\top \in \mathcal{M}(X)$. An element $x \in C$ is *meet-irreducible* if $\forall y, z \in C.\ (x = y \wedge z) \Rightarrow (x = y \text{ or } x = z)$. We denote by MI_C the set of meet-irreducible elements of C. Note that $\top \in MI_C$.

The announced result on the pseudo-complement in $uco(C)$ is stated below.

Theorem 4.1 *If C is meet-generated by MI_C then $uco(C)$ is pseudo-complemented, and for any $\rho \in uco(C)$,*

$$\rho^* = \mathcal{M}(MI_C \setminus \rho(C)).$$

[3] By $x < y$ we mean $x \leq y$ and $x \neq y$, while if X and Y are sets then $X \setminus Y$ denotes their set-difference.

Proof. Assume that $\rho \in uco(C)$. Observe that $MI_C \subseteq \rho(C) \cup (MI_C \setminus \rho(C))$, from which follows $MI_C \subseteq \rho \sqcap \mathcal{M}(MI_C \setminus \rho(C))$, which in turn implies $C = \mathcal{M}(MI_C) \subseteq \rho \sqcap \mathcal{M}(MI_C \setminus \rho(C))$, i.e. $C = \rho \sqcap \mathcal{M}(MI_C \setminus \rho(C))$. Moreover, if $C = \rho \sqcap \psi$, for some $\psi \in uco(C)$, then $MI_C \setminus \rho(C) \subseteq \psi(C)$, because, otherwise, there exists $y \in (MI_C \setminus \rho(C)) \setminus \psi(C)$ for which $y = \rho(y) \wedge \psi(y)$ with $y \neq \rho(y)$ and $y \neq \psi(y)$, which is a contradiction, since $y \in MI_C$. Hence, $\psi \sqsubseteq \mathcal{M}(MI_C \setminus \rho(C))$, i.e. $\rho^* = \mathcal{M}(MI_C \setminus \rho(C))$. ■

Observe that for any $\rho \in uco(C)$, under the above hypothesis, we can also write $\rho^* = \mathcal{M}(MI_C \setminus \rho(C)) = \mathcal{M}(\{x \in MI_C \mid x < \rho(x)\}$. With respect to the corresponding Theorem 3.1 of [13], the above theorem provides a different condition on the domain of reference C in order that $uco(C)$ is pseudo-complemented. Our result has the advantages of a simpler and shorter proof, and, more importantly, provides a natural lattice-theoretic characterization of the pseudo-complement, which is particularly useful for the practical cases, as it will be shown in Section 5.

The following immediate consequence of the above result is useful in order to recognize if the operation of pseudo-complementation performed twice acts as the identity.

Corollary 4.2 *If C is meet-generated by MI_C then for any $\rho \in uco(C)$,*

$$\rho^{**} = \mathcal{M}(MI_C \cap \rho(C)).$$

In other terms, the above corollary says that the double pseudo-complementation on a given closure is the identity iff (the set of fixpoints of) this closure is equal to the Moore-closure of its meet-irreducible elements. A further consequence of this result and the above theorem allows to simplify the task of checking if a given decomposition is minimal.

Corollary 4.3 *Assume that C is meet-generated by MI_C and $\langle D_i \rangle_{i \in I}$ is a decomposition for C. If $\{MI_C \cap \rho_{D_i}\}_{i \in I}$ is a partition of MI_C and for all $i \in I$, $D_i = C \sim (C \sim D_i)$, then $\langle D_i \rangle_{i \in I}$ is minimal.*

Proof. By Lemma 3.3, it is sufficient to verify that $C \sim (\sqcap_{i \in I \setminus \{k\}} D_i) = D_k$, for all $k \in I$. By Theorem 4.1 and Corollary 4.2, we have: $C \sim (\sqcap_{i \in I \setminus \{k\}} D_i) = \mathcal{M}(MI_C \setminus \mathcal{M}(\cup_{i \in I \setminus \{k\}} \rho_{D_i}(C))) = \mathcal{M}(MI_C \setminus (\cup_{i \in I \setminus \{k\}} \rho_{D_i}(C))) = \mathcal{M}(MI_C \cap \rho_{D_k}(C)) = C \sim (C \sim D_k) = D_k$. ■

A standard well-known result of representation in lattice theory ([16]) says that if C is a continuous complete lattice then C is meet-generated by MI_C. We recall here the notion of continuity for a complete lattice C. If $x, y \in C$ then $x \ll y$ (x is *way-below* y) if $\forall S \subseteq C.\ (y \leq \vee S) \Rightarrow (\exists T \subseteq S.\ T$ finite $\&\ x \leq \vee T)$. C is *continuous* if for any $x \in C$, $x = \vee\{z \in C \mid z \ll x\}$. It is well-known that if C satisfies the ascending chain condition then C is *algebraic*[4], and algebraicity implies continuity ([16]). It is also well-known that every continuous complete lattice is meet-continuous ([16]), and hence, by Theorem 3.1 of [13], we know that if C is continuous then $uco(C)$ is pseudo-complemented. On the other hand, this latter result is also an immediate consequence of Theorem 4.1

Corollary 4.4 *If C is a continuous complete lattice then $uco(C)$ is pseudo-complemented and for any $\rho \in uco(C)$, $\rho^* = \mathcal{M}(MI_C \setminus \rho(C))$.*

[4] C is *algebraic* if $\forall x \in C.\ x = \vee\{z \in C \mid z \leq x,\ z \ll z\}$. Note that $\{z \in C \mid z \ll z\}$ is the standard set of the *compacts* of C.

The class of continuous lattices was introduced by Dana Scott in his pioneering work on denotational semantics of programming languages ([24]). It is worth noting that this class is wide enough for practical purposes: in practice, every abstract domain used as basis of an abstract interpretation-based static analysis satisfies the ascending chain condition (most of them are even finite domains). Furthermore, even if the abstract domain does not satisfy the ascending chain condition (this may happen whenever some widening/narrowing operators used to accelerate the convergence above least fixpoints are provided), in order to use the above corollary the property of continuity can be checked. Also, any *collecting* domain, i.e. any powerset $\wp(X)$, for some set X, ordered with the subset or superset relation, is a continuous complete lattice. This latter case includes the standard concrete domains for collecting semantics in functional and logic programming (e.g. [2, 20]). Finally, complete lattices which are meet-continuous and that satisfy the descending chain condition are algebraic, and hence continuous (cf. [23]).

We can draw an interesting further consequence of Theorem 4.1. First, some lattice-theoretic definitions. By C^{op} we denote the dual of C, i.e. C equipped with the dual order $\geq$. An element $a \in C$ is an *atom* if a covers $\bot$, i.e., $a \neq \bot$ and $\forall x \in C.\ (\bot < x \leq a) \Rightarrow (x = a)$. $a \in C$ is a *dual-atom* if it is an atom in C^{op}. We denote by $dAtom_C$ the set of dual-atoms of C. Clearly, in general, $dAtom_C \subseteq MI_C$. C is *atomistic* if every element different from $\bot$ is the join of the atoms which precede it. C is *dual-atomistic* if C^{op} is atomistic. Observe that C is dual-atomistic iff $C = \mathcal{M}(dAtom_C)$. Also, if C is dual-atomistic then $dAtom_C = MI_C$, and therefore C is meet-generated by MI_C as well.

Corollary 4.5 *If C is dual-atomistic then $uco(C)$ is pseudo-complemented and for any $\rho \in uco(C)$, $\rho^* = \mathcal{M}(dAtom_C \setminus \rho(C))$.*

Proof. Straightforward, since $MI_C = dAtom_C$. ∎

This corollary simplifies further the task of computing a particular complement $C \sim D$ in case the domain C is dual-atomistic. In fact, in this case, it is enough to consider the behaviour of D on the dual-atoms of C, which are usually easily identifiable. It is worth remarking that the condition of dual-atomicity for an abstract domain is not frequently satisfied, and therefore the simple methodology provided by the above corollary is not always applicable. A relevant example of a non-dual-atomistic, but continuous, abstract domain is the Cousot and Cousot domain of integer intervals, which we treat in Subsection 5.1.

5 Applications

We provide three applications of the theory developed in the previous section. First, we consider the abstract domain for integer interval analysis. This domain is not dual-atomistic, thus in order to compute complements w.r.t. it, we need to identify its meet-irreducible elements. Secondly, we compute $Pos \sim Def$. This task is particularly easy since Pos is dual-atomistic. We also show that $Pos \sim (Pos \sim Def)$ is more abstract than Def. Finally, we use complementation in order to decompose in a minimal way *Sharing* (which is also dual-atomistic) into three components each expressing only one of the basic properties of *Sharing*: ground-dependency, pair-sharing and set-sharing.

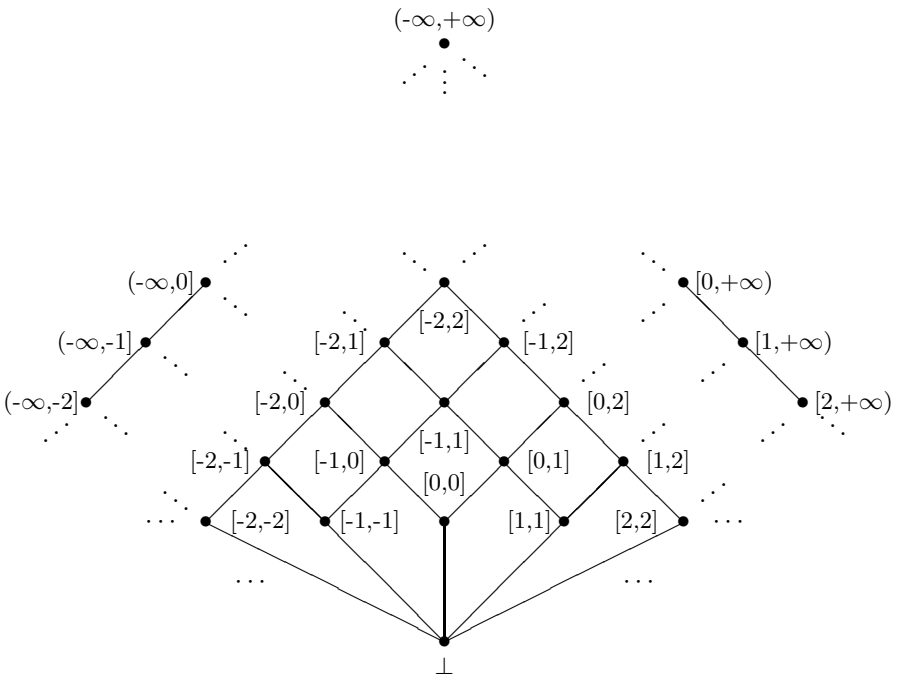


Figure 1: The abstract domain I.

5.1 Complementing the integer interval domain

As a remarkable example, we now show that the abstract lattice of intervals of integer numbers introduced in [7, 8] to analyze the values of an integer variable is continuous, although it does not satisfy the ascending chain condition, and therefore Corollary 4.4 can be applied to it. For simplicity, we consider a single integer variable to analyze (the generalization is straightforward), and therefore the domain of concrete denotations is the powerset of the integers, $\wp(\mathbb{Z})$, ordered by subset inclusion. The abstract domain I of integer intervals is depicted in Figure 1. As pointed out in [8], it turns out that I is a complete lattice. I enjoys a Galois insertion with $\wp(\mathbb{Z})_{\subseteq}$, which is the most natural: e.g., $\gamma([a,b]) = \{z \in \mathbb{Z} \mid a \leq z \leq b\}$ and $\alpha(\{-3,-1,0,2,5\}) = [-3,5]$.

Lemma 5.1 *I is a continuous complete lattice.*

Proof. We show that I is algebraic. It is immediate to see the set of the compacts of I is $K_I = \{[z_1,z_2] \mid z_1,z_2 \in \mathbb{Z},\ z_1 \leq z_2\} \cup \{\bot\}$, and each element in $I \setminus K_I$ is the join of the compacts which precedes. ■

By a simple direct inspection of the Hasse-diagram of I, it is simple to see that the set of meet-irreducible elements of I is exactly the set-theoretic complement of its compact elements, i.e. $MI_I = \{(-\infty,z] \mid z \in \mathbb{Z}\} \cup \{[z,+\infty) \mid z \in \mathbb{Z}\} \cup \{(-\infty,+\infty)\}$. Since we have shown that I is continuous, we also have that $I = \mathcal{M}(MI_I)$, as one can readily see from Figure 1.

An abstraction of the intervals I is given by the domain CP depicted in Figure 2. This domain is obtained by identifying any integer number z by the interval $[z,z]$, and the element $\top$ by (-∞,+∞). It is clear that CP is an abstraction of I since it corresponds to a Moore-set of elements of I. This domain CP is the standard

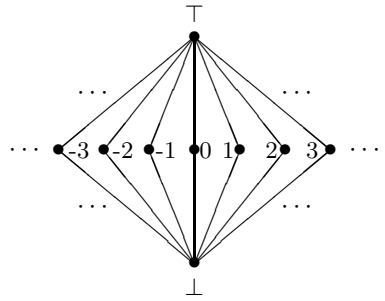


Figure 2: The abstract domain *CP*.

lattice for *constant propagation* analysis ([18]), namely an analysis that identifies program expressions computing the same value on all executions of the program.

The complement $I \sim CP$ is somehow surprising: in fact, it turns out that this complement is precisely the lattice of intervals itself.

Proposition 5.2 $I \sim CP = I$.

Proof. Applying Corollary 4.4: $I \sim CP = \mathcal{M}(MI_I \setminus CP) = \mathcal{M}(\{(-\infty, z] \mid z \in \mathbb{Z}\} \cup \{[z, +\infty) \mid z \in \mathbb{Z}\}) = I$. ■

It should be noted that this complement has been already computed in [5] by using a direct proof method more complicated than that here proposed.

5.2 $Pos \sim Def$

Both *Pos* and *Def* are domains that are used for inferring ground-dependency information in logic program analysis ([19]). Clearly, *Pos* infers more precise information than *Def*, since *Def* is a proper abstraction of *Pos*. This has been experimentally evaluated in [1]. Now we want to characterize, with the help of the theory developed in Section 4, what is in *Pos* that is not in *Def*, i.e. $Pos \sim Def$.

We briefly recall the definitions of *Pos* and *Def*. Let *Var* be a countable set of variables, and let *VI* be any (non-empty) finite subset of *Var* containing the variables of interest. 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. *Pos* is the finite lattice of *positive* Boolean functions on *VI*, where a Boolean function f is positive if $f(true, \ldots, true) = true$. Obviously, the order of *Pos* is given by logical consequence, and, *lub* and *glb* on *Pos* are given by logical disjunction and conjunction, respectively. *Def* is the finite lattice of positive Boolean functions on *VI* whose models are closed under intersection. Formulae in *Def* are called *definite*. Here, we do not add to *Pos* and *Def* the bottom Boolean function *false* representing the empty set of substitutions. Evidently, this is not a severe restriction, and it allows to reason on *Pos* and *Def* more uniformly, avoiding to consider the particular straightforward case of the element *false*. For more details about *Pos* and *Def* see [1]. It is well-known that Boolean functions can be represented by means of propositional formulae. Thus, in the following, we will use propositional formulae over *VI* to represent Boolean functions in *Pos* and *Def*. In Figure 3, *Pos* and *Def* are depicted for $VI = \{x, y\}$.

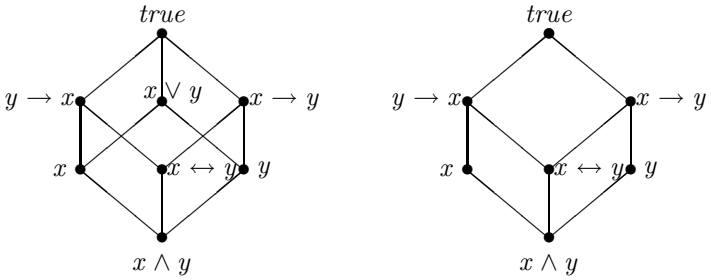


Figure 3: The domains *Pos* and *Def* for $VI=\{x, y\}$.

As observed in [1], *Def* is a meet-sublattice of *Pos*. Further, the top Boolean function *true* is in *Def*. Hence, *Def* is a Moore-set of *Pos*, namely, 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 [19]. 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,[5] $\sigma_1 = \{x/a, y/b, z/c\}$ and $\sigma_2 = \{x/a, y/w, z/w, v/u\}$ satisfy this property. Thus, $\{\sigma_1, \sigma_2\} \subseteq \gamma(x \wedge (y \leftrightarrow z))$.

By viewing each positive formula as the set of its models, it is immediate to observe that *Pos* is a Boolean lattice, and therefore it is dual-atomistic, where the dual-atoms are the positive formulae that are satisfied by all but one truth assignments different from the unitary truth assignment VI. All the other formulae of *Pos* can be obtained by logical conjunction (i.e., intersection of their sets of models) of some of these dual-atoms.[6] Let us identify which of the dual-atoms represent definite formulae, namely which of their corresponding sets of models is closed under model intersection. Thus, consider a dual-atom $A \in Pos$, i.e. a set of models $A = \wp(VI) \setminus \{t\}$, for some truth assignment $t \in \wp(VI) \setminus \{VI\}$.

Lemma 5.3 *A is closed under model intersection iff $t = VI \setminus \{x\}$, for some $x \in VI$.*

Proof. If $t = VI \setminus \{x\}$, for some $x \in VI$, then clearly $A = \wp(VI) \setminus \{t\}$ is closed under model intersection. Consider now a truth assignment t that maps to *false* at least two different variables of VI, i.e. $|t| \leq |VI| - 2$. Then $A = \wp(VI) \setminus \{t\}$ contains truth assignments $t_x = t \cup \{x\}$ for any $x \in VI \setminus t$. Obviously, the intersection of these t_x's gives t, and therefore, A is not closed under intersection. ■

From the above Lemma 5.3, each formula in $dAtom_{Def}$ is equivalent to an implication $\wedge(VI \setminus \{x\}) \to x$, for some $x \in VI$, whereas each formula in $dAtom_{Pos} \setminus dAtom_{Def}$ is equivalent to an implication of the shape $\wedge(VI \setminus B) \to \vee B$, for some $B \subseteq VI$ which contains at least two variables. This has an interesting consequence on the set of concrete substitutions that such formulae approximate: for the formulae in $dAtom_{Def}$, among the approximated substitutions, there are some whose abstraction in *Def* (and *Pos*) is exactly that formula (cf. [4]), whereas for

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

[6] If *false* is considered as an element of *Pos*, then *Pos* is meet-generated by its dual-atoms plus its bottom *false*.

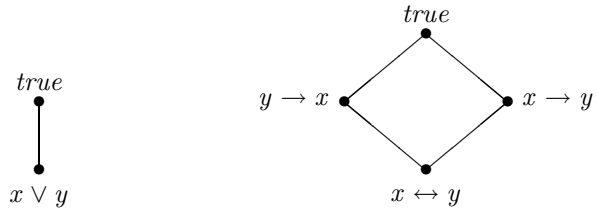


Figure 4: The domains $Pos \sim Def$ and $Pos \sim (Pos \sim Def)$ for $VI = \{x, y\}$.

the formulae in $dAtom_{Pos} \setminus dAtom_{Def}$ this is not true. Thus, the latter formulae express *possible ground-dependency* (by means of the disjunction $\vee B$) which is not expressible with Def.

By Corollary 4.5, $Pos \sim Def$ is the domain obtained by closing under meet (i.e. logical conjunction) the set of formulae in $dAtom_{Pos} \setminus dAtom_{Def}$ (and adding the top element *true*). In Figure 4, $Pos \sim Def$ is shown for the case $VI = \{x, y\}$. The same figure represents also the domain $Pos \sim (Pos \sim Def)$. Clearly, this domain properly abstracts Def. The reader should not be surprised by this fact in view of Corollary 4.2. In fact, Def is clearly more concrete than the domain obtained taking all possible conjunctions of $dAtom_{Def}$. In order to see this, it suffices to observe that $dAtom_{Def}$ contains only n formulae, where $n = |VI|$, and therefore $Pos \sim (Pos \sim Def)$ has 2^n elements, whereas Def has many more elements as shown in Figure 3 for $n = 2$.

Thus, by Lemma 3.3, we have characterized the minimal binary decomposition $\langle Pos \sim Def, Pos \sim (Pos \sim Def)\rangle$ for Pos, where the second component expresses conjunctions of definite clauses and the first one conjunctions of non definite clauses.

5.3 Decomposing *Sharing*

Since, for simplicity, we have considered Pos and Def without an additional bottom element representing the empty set of substitutions, we proceed analogously for *Sharing*. It is evident that an eventual adaptation will be straightforward. Thus, we consider $Sharing = \{S \subseteq \wp(VI) \mid \emptyset \in S\}$ ([17]). *Sharing* is a finite distributive lattice with respect to the partial order given by set-theoretic inclusion. *Sharing* enjoys a well-known Galois insertion into the concrete domain $\wp(Sub)$ (for details see [17]). For instance, assuming $VI = \{x, y, z, u\}$, the element $\{\emptyset, \{y, z\}, \{y, z, u\}\}$ is an element of *Sharing* representing substitutions with respect to which x is ground, and z and y may share, and so may y and u, and z and u. In particular, $\sigma_1 = \{x/a, y/b, z/c\}$ and $\sigma_2 = \{x/b, y/v, z/v, u/v\}$ satisfy these properties. Therefore, $\{\sigma_1, \sigma_2\} \subseteq \gamma(\{\emptyset, \{y, z\}, \{y, z, u\}\})$.

Def abstracts *Sharing* and, in fact, it represents the ground-dependency information of *Sharing* ([4, 6]). The abstraction function which maps an element $S \in Sharing$ into a formula of *Def* capturing its ground-dependency information is defined as follows:

$$\mathcal{C}(S) = \wedge\{\wedge W \rightarrow x \mid W, \{x\} \subseteq VI\ ,\ \forall A \in S.\, (x \in A) \Rightarrow (W \cap A \neq \emptyset)\}.$$

For instance, if $VI = \{x, y, z, u\}$ and $S = \{\emptyset, \{x, y\}, \{x, z\}\}$, then $\mathcal{C}(S) = u \wedge (x \leftrightarrow (y \wedge z))$ outlines the fact that for every $\sigma \in \gamma(S)$ the variable u is ground in σ, and x is ground in σ iff also y and z are.

In [4], $\mathit{Sharing} \sim \mathit{Def}$ has been characterized as the closure $\rho^+ \in \mathit{uco}(\mathit{Sharing})$ defined as follows: $\forall S \in \mathit{Sharing}.\ \rho^+(S) = S \cup \{\{x\} \mid x \in \mathit{VI}\}$. The set of fixpoints of ρ^+ is called $\mathit{Sharing}^+$. Using the new theory developed above, the proof of $\mathit{Sharing}^+ = \mathit{Sharing} \sim \mathit{Def}$ becomes very simple with respect to that proposed in [5]. Therefore, we give this proof here.

Clearly, *Sharing* is dual-atomistic, and its dual-atoms are $\wp(\mathit{VI}) \setminus \{S\}$, for any $S \in \wp(\mathit{VI}) \setminus \{\emptyset\}$.[7] Let us characterize which of these dual-atoms are fixpoints of the abstraction *Def*.

Lemma 5.4 *Assume that* $A \in \mathit{dAtom}_{\mathit{Sharing}}$. *$A$ is a fixpoint of Def iff* $\exists x \in \mathit{VI}.\ \{x\} \notin A$.

Proof. Assume that A is a fixpoint of *Def* and that it contains all singletons. Evidently, $\mathcal{C}(A) = \mathit{true}$. Since $\gamma_{\mathit{Def},\mathit{Sharing}}(\mathit{true}) = \top_{\mathit{Sharing}}$, we have a contradiction. Consider now A such that it does not contain a singleton $\{x\}$. It is not difficult to see that $\mathcal{C}(A) = \wedge(\mathit{VI} \setminus \{x\}) \rightarrow x \neq \mathit{true}$. Hence, we get $A \subseteq \gamma_{\mathit{Def},\mathit{Sharing}}(\mathcal{C}(A)) = \gamma_{\mathit{Def},\mathit{Sharing}}(\wedge(\mathit{VI} \setminus \{x\}) \rightarrow x) \subset \top_{\mathit{Sharing}}$. Since A is a dual-atom, it follows that $A = \gamma_{\mathit{Def},\mathit{Sharing}}(\mathcal{C}(A))$. ■

Theorem 5.5 $\mathit{Sharing}^+ = \mathit{Sharing} \sim \mathit{Def}$.

Proof. From Lemma 5.4, the dual-atoms of *Sharing* that are not fixpoints of *Def* are exactly those that contain any singleton $\{x\}$, for $x \in \mathit{VI}$, and, by Corollary 4.4, $\mathit{Sharing} \sim \mathit{Def}$ is obtained closing under meet (i.e., intersection) these elements. It is clear that this operation produces $\mathit{Sharing}^+$. ■

Recall from the proof of Lemma 5.4 that for any dual-atom A of *Sharing* which is a fixpoint for *Def*, $\mathcal{C}(A) = \wedge(\mathit{VI} \setminus \{x\}) \rightarrow x$, for some $x \in \mathit{VI}$. Recall also from Subsection 5.2 that these are exactly the formulae corresponding to the dual-atoms of *Pos* that are fixpoints of *Def*. Hence, the following result is immediate.

Theorem 5.6 $\mathit{Pos} \sim (\mathit{Pos} \sim \mathit{Def}) = \mathit{Sharing} \sim (\mathit{Sharing} \sim \mathit{Def})$.

In this way, we have also characterized $\langle \mathit{Sharing}^+, \mathit{Sharing} \sim \mathit{Sharing}^+ \rangle$ as a minimal binary decomposition for *Sharing*. Moreover, the above theorem corrects the wrong claim made in [4] that $\mathit{Sharing} \sim (\mathit{Sharing} \sim \mathit{Def}) = \mathit{Def}$, since we have already observed that $\mathit{Def} \sqsubset \mathit{Pos} \sim (\mathit{Pos} \sim \mathit{Def})$.

As observed in the proof of Lemma 5.4, $\forall S \in \mathit{Sharing}^+.\ \mathcal{C}(S) = \mathit{true}$. Therefore, $\mathit{Sharing}^+$ contains no ground-dependency information anymore. However, $\mathit{Sharing}^+$ still contains two different types of information that we want to separate using complementation: (1) pair-sharing, or equivalently variable independence: the information about which pairs of variables may share, and thus which pairs are independent; (2) set-sharing: the knowledge that certain sets of variables may share a common variable.

Let us then design an abstract domain expressing pair-sharing. For a set of variables A, let $\mathit{Pairs}(A) = \{\{x, y\} \mid x, y \in A, x \neq y\}$. The obvious candidate domain for pair-sharing is $PS = \wp(\mathit{Pairs}(\mathit{VI}))$, with abstraction $\alpha_{\mathit{Sh}^+,PS} : \mathit{Sharing}^+ \rightarrow PS$ defined as follows: for any $S \in \mathit{Sharing}^+$, $\alpha_{\mathit{Sh}^+,PS}(S) = \cup\{\mathit{Pairs}(A) \mid A \in S\}$. It is easy to see that $\alpha_{\mathit{Sh}^+,PS}$ is completely additive and onto, and therefore, by standard results, together with its unique adjoint concretization map, it forms a

[7] Analogously to *Pos*, if $\emptyset$ is considered as an element of *Sharing*, then *Sharing* is meet-generated by its dual-atoms plus its bottom $\emptyset$.

Galois insertion of PS into $Sharing^+$. The adjoint concretization, for any $B \in PS$, is defined as $\gamma_{PS,Sh^+}(B) = \cup\{S \in Sharing^+ \mid \alpha_{Sh^+,PS}(S) \subseteq B\}$.

We want to compute now $Sharing^+ \sim PS$ using our theory. It is immediate to observe that $Sharing^+$ is dual-atomistic, where each dual-atom is $\wp(VI) \setminus \{A\}$, where $A \in \wp(VI)$ with $|A| > 1$. We must now characterize the dual-atoms which are fixpoints of the corresponding closure $\gamma_{PS,Sh^+} \circ \alpha_{Sh^+,PS}$. The result is pretty surprising.

Lemma 5.7 $\forall S \in dAtom_{Sh^+}.\ \gamma_{PS,Sh^+}(\alpha_{Sh^+,PS}(S)) = \top_{Sharing^+}$.

Proof. If $VI \in S$ then the thesis is obvious because $\alpha_{Sh^+,PS}(S) = Pairs(VI) = \top_{PS}$. If $VI \notin S$, then again $Pairs(VI) \subseteq S$, i.e., $\alpha_{Sh^+,PS}(S) = \top_{PS}$. ■

Thus, by Corollary 4.5, we get the following result.

Theorem 5.8 $Sharing^+ \sim PS = Sharing^+$.

Let us try to understand the reason of this phenomenon by examining closely the behaviour of $\rho_{PS} = \gamma_{PS,Sh^+} \circ \alpha_{Sh^+,PS}$ on $dAtom_{Sh^+}$. Let $\widehat{d}$ be the only element of $dAtom_{Sh^+}$ that does not contain VI. The following observation will put us on the right track. Consider any $a \in dAtom_{Sh^+} \setminus \{\widehat{d}\}$: since a contains VI it can express no interesting pair-sharing information (all pairs of variables share). Thus, it is expectable that a closure expressing pair-sharing maps all such a's to the top of $Sharing^+$, like ρ_{PS} does. The point is that $\rho_{PS}(\widehat{d}) = \top_{Sh^+}$ too. Thus, the loss of information produced by ρ_{PS} introduces the new sharing set VI that is originally absent. However, ρ_{PS} must throw away the set-sharing information as much as possible, and this is obtained precisely by adding new sharing sets as long as they do not introduce new pair-sharing. Thus, all we need to do is to modify ρ_{PS} in such a way that it adds any sharing set compatible with the original pair-sharing with the exception of VI. The following closure $\rho_{PS'}$ does this:

$$\forall S \in Sharing^+.\ \ \rho_{PS'}(S) = \rho_{PS}(S) \setminus (\{VI\} \setminus S).$$

Observe that $\rho_{PS'}$ throws away as much set-sharing information as possible: somehow, it only remembers whether VI is present or not in the original value.

Clearly, $\rho_{PS'}(Sharing^+) \cap dAtom_{Sh^+} = \{\widehat{d}\}$. Therefore, by Corollary 4.5, $Sharing^+ \sim \rho_{PS'}(Sharing^+) = \mathcal{M}(dAtom_{Sh^+} \setminus \{\widehat{d}\})$. As $Sharing^+ \sim \rho_{PS'}(Sharing^+)$ is obtained closing under intersection those dual-atoms of $Sharing^+$ that contain VI, this complement corresponds to the following closure $\rho^*_{PS'}$:

$$\forall S \in Sharing^+.\ \rho^*_{PS'}(S) = S \cup \{VI\}.$$

Obviously, $\langle \rho_{PS'}(Sharing^+), \rho^*_{PS'}(Sharing^+)\rangle$ is not a minimal decomposition of $Sharing^+$. In fact, $\rho_{PS'}(Sharing^+) \sqsubset Sharing^+ \sim \rho^*_{PS'}(Sharing^+) = \{\top_{Sh^+}, \widehat{d}\} = \mathbf{2}$. Observe that the difference between $\rho_{PS'}(Sharing^+)$ and $\mathbf{2}$ is really surprising: the former has size at least exponential (in the size of VI), whereas the latter has only two elements!

We are now in position to give the announced minimal ternary decomposition for $Sharing$. Let us define SS the abstraction of $Sharing$ corresponding to the set-sharing closure computed above, i.e., $SS = \{S \in Sharing \mid S \in Sharing^+,\ VI \in S\}$, and $Def^- = Pos \sim (Pos \sim Def)$.

Theorem 5.9 $\langle Def^-, \mathbf{2}, SS\rangle$ *is a minimal decomposition for* $Sharing$.

Proof. Using the results above, it is simple to verify that $Def^- = Sharing \sim (\mathbf{2} \sqcap SS)$, $\mathbf{2} = Sharing \sim (Def^- \sqcap SS)$, and $SS = Sharing \sim (Def^- \sqcap \mathbf{2})$. Hence, the thesis follows by Lemma 3.3. ■

To conclude, we note that the closure associated to $Sharing \sim Sharing^+$ admits a natural lattice-theoretic generalization. In fact, the closure defining $Sharing^+$ can be viewed as an instance of the following definition: if A is any set and e is any fixed element of A, then $plus_e : \wp(A) \to \wp(A)$ is defined as $plus_e(X) = X \cup \{e\}$, for any $X \in \wp(A)$. Evidently, $plus_e \in uco(\wp(A)_\subseteq)$, and since $\wp(A)_\subseteq$ is dual-atomistic, by Corollary 4.5 its pseudo-complement exists: it is easy to verify that $plus_e^* = \lambda X.(A \setminus (\{e\} \setminus X))$. In case $A = \wp(B)_\subseteq$, for some set B, this definition specializes to that of $Sharing \sim Sharing^+$.

6 Concluding Remarks

In this paper, we have stated new sufficient conditions that guarantee the existence of complements of abstract domains and provide a practical and simple systematic method, based on standard lattice-theoretic notions, to compute complements. These conditions differ from those that were put forward, for the same purpose, in [4, 13]. The relationship between these two different conditions is, to the best of our knowledge, not yet known in the literature. Further work will be devoted to understanding this relationship.

It is also worthwhile to remark that our use of meet-irreducible elements for the computation of the complement, i.e., the inverse of the reduced product of abstract domains, has a strong analogy with the dual method of using the *completely join-irreducible* elements, introduced recently by Giacobazzi and Ranzato in [15], for the computation of the *least disjunctive basis* of an abstract domain, i.e., the inverse for the *disjunctive completion* of abstract domains. We think that such a duality deserves a further investigation, since it might shed some light on a general methodology for defining the inverse of any *abstract domain refinement* (cf. [11]).

Finally, we remark that the role of meet-irreducible elements and dual-atoms (or their dual notions) in the context of Cousot and Cousot's abstract interpretation theory was first investigated by Nielson, in his work on abstract interpretation using domain theory (cf. [21]), and on the tensor product of abstract domains (cf. [22]).

Acknowledgments. We would like to thank Roberto Giacobazzi and Harald Søndergaard for their helpful suggestions. Thanks also to the participants of the *Prima Scuola Nazionale di Dottorato in Informatica* held in Pontignano (Italy), for stimulating discussions on the subject of this work.

References

[1] T. Armstrong, K. Marriott, P. Schachte, and H. Søndergaard. Boolean functions for dependency analysis: algebraic properties and efficient representation. In *Proc. of the 1st International Static Analysis Symposium (SAS '94)*, LNCS 864, pages 266–280, 1994.

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

[3] M. Codish, A. Mulkers, M. Bruynooghe, M. García de la Banda, and M. Hermenegildo. Improving abstract interpretations by combining domains. *ACM TOPLAS*, 17(1):28–44, 1995.

[4] A. Cortesi, G. Filé, R. Giacobazzi, C. Palamidessi, and F. Ranzato. Complementation in abstract interpretation. In *Proc. of the 2nd International Static Analysis Symposium (SAS '95)*, LNCS 983, pages 100–117, 1995.

[5] A. Cortesi, G. Filé, R. Giacobazzi, C. Palamidessi, and F. Ranzato. Domain complementation in abstract interpretation. Preprint 1/96, Dept. of Mathematics, Univ. of Padova, 1996. Extended version of [4].

[6] A. Cortesi, G. Filé, and W. Winsborough. The quotient of an abstract interpretation. Tech. Rep. 12/94, Dept. of Mathematics, Univ. of Padova, 1994.

[7] P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In *Proc. of the 2nd Int. Symp. on Programming*, pages 106–130, Paris, 1976.

[8] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. of ACM POPL '77*, pages 238–252, 1977.

[9] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Proc. of ACM POPL '79*, pages 269–282, 1979.

[10] P. Cousot and R. Cousot. Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and PER analysis of functional languages). In *Proc. of IEEE ICCL '94*, pages 95–112, 1994.

[11] G. Filé, R. Giacobazzi, and F. Ranzato. A unifying view of abstract domain design. *ACM Comp. Surveys*, 28(2), Symp. on Models of Progr. Lang. and Comput., 1996.

[12] G. Filé and F. Ranzato. Improving abstract interpretations by systematic lifting to the powerset. In *Proc. of the 1994 International Logic Programming Symposium (ILPS '94)*, pages 655–669, 1994.

[13] R. Giacobazzi, C. Palamidessi, and F. Ranzato. Weak relative pseudo-complements of closure operators. *Algebra Universalis*, 1996. To appear.

[14] R. Giacobazzi and F. Ranzato. Functional dependencies and Moore-set completions of abstract interpretations and semantics. In *Proc. of the 1995 International Logic Programming Symposium (ILPS '95)*, pages 321–335, 1995.

[15] R. Giacobazzi and F. Ranzato. Compositional optimization of disjunctive abstract interpretations. In *Proc. of ESOP '96*. LNCS 1058, pages 141–155, 1996.

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

[17] D. Jacobs and A. Langen. Static analysis of logic programs for independent AND-parallelism. *Journal of Logic Programming*, 13(2,3):154–165, 1992.

[18] J.B. Kam and J.D. Ullman. Monotone data flow analysis frameworks. *Acta Informatica*, 7:305–317, 1977.

[19] K. Marriott and H. Søndergaard. Precise and efficient groundness analysis for logic programs. *ACM LOPLAS*, 2(1–4):181–196, 1993.

[20] A. Mycroft and F. Nielson. Strong abstract interpretation using power domains. In *Proc. of ICALP '83*, LNCS 154, pages 536–547, 1983.

[21] F. Nielson. *Abstract Interpretation using Domain Theory*. Ph.D. Thesis, Dept. of Computer Science, U. of Edinburgh, CST-31-84, 1984.

[22] F. Nielson. Tensor products generalize the relational data flow analysis method. In *Proc. of the 4th Hungarian Computer Science Conf.*, pp. 211–225, 1985.

[23] M. Ramalho. On upper continuous and semimodular lattices. *Algebra Universalis*, 32:330–340, 1994.

[24] D.S. Scott. Lattice theory, data types, and semantics. In *Formal Semantics of Program. Lang.*, Courant Computer Science Symp., vol. 2, pages 65–106, 1972.