Improving Abstract Interpretations by Systematic Lifting to the Powerset

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

Abstract

Operators that systematically produce more precise abstract interpretations from simpler ones are interesting. In this paper we present a formal study of one such operator: the powerset.

The main achievements of the paper are described below:

- A formal definition of the powerset operator is given. For any given abstract interpretation $\mathcal{D} = \langle D, o_1, \ldots, o_k \rangle$, where D is the abstract domain and $o_1, \ldots, o_k$ are the abstract operations, this operator provides a new abstract interpretation $P(\mathcal{D}) = \langle P(D), o_1^\star, \ldots, o_k^\star \rangle$. Thus, the powerset concerns also the abstract operations $o_i^\star$, that are constructively defined from the o_i's.
- A necessary and sufficient condition guaranteeing that $P(D)$ is strictly better than D is given.
- The general theory is applied to the well-known abstract interpretation $\mathcal{PROP}$ for ground-dependence analysis of logic programs. It is shown that $P(\mathcal{PROP})$ is strictly better than $\mathcal{PROP}$.

1 Introduction

It is widely acknowledged that the accuracy of a data-flow analysis depends on the expressiveness of the abstract interpretation chosen. Thus, it becomes very interesting to have operators that systematically produce new and possibly more precise abstract interpretations from simpler ones. In this paper we study one such operator: the powerset. The main motivation for studying the powerset is obvious: it can generate very expressive interpretations. In fact, the powerset abstract domain gains the capability of expressing the logical disjunction of the properties represented by the original domain. Moreover, it is also worth noticing that a "hidden" powerset operator is employed when defining analyses or abstract semantics that collect sets of abstract values ([BGL93, CDG93]).

The powerset operator has also been studied by Cousot and Cousot in [CC79] (together with the cartesian product, the disjoint sum and the function space), and later in [CC92]. In these papers the Cousots were concerned only with lifting abstract domains to their powerset. Abstract operations for the new domain could be defined as the usual optimal approximations of concrete ones. However, it is also very important to be able to derive correct (not necessarily optimal) operations directly from those of the original domain, as if these latter are finitely computable

(and the abstract domain is finite) then the former are too. In the paper we characterize such a method of derivation. As far as the abstract domains are concerned, in [FR94] it is shown that our definition is equivalent to that of [CC92].

Given any interpretation $\mathcal{D} = \langle D, o_1, \ldots, o_k \rangle$ abstracting the concrete one $\mathcal{C} = \langle C, o_1^C, \ldots, o_k^C \rangle$, we describe how to systematically construct its powerset $P(\mathcal{D}) = \langle P(D), o_1^\star, \ldots, o_k^\star \rangle$. The main questions about this construction are: Does $P(\mathcal{D})$ also abstract $\mathcal{C}$? Is the abstract domain $P(D)$ always better than D? The theory developed in this paper answers these questions as follows.

1. We show that $P(\mathcal{D})$ abstracts $\mathcal{C}$ when the following two conditions hold:
   - the concrete domain C is a complete meet-distributive lattice (this is true for every lattice $\wp(X)$ ordered with set-theoretic inclusion);
   - every concrete operation o_i^C is join complete (more precisely, it only suffices join completeness on certain subsets of C).
2. As expectable, the abstract domain $P(D)$ is never worse than D. $P(D)$ is strictly better than D iff the concretization map of the Galois connection between C and D is not join complete. Since this is a quite unusual condition, $P(D)$ is very often strictly better than D. Furthermore, the powerset is idempotent, i.e. $P(P(D))$ is equivalent to $P(D)$.

The powerset operator defined in this paper can be used to lift any abstract interpretation. In order to verify its usefulness, we apply it to the well-known abstract interpretation $\mathcal{PROP}$ ([MS89, CFW91]), whose abstract domain consists of (classes wrt semantic equivalence of) propositional formulae. $\mathcal{PROP}$ is used for the groundness and equivalence analysis of logic programs. We show that $P(\mathcal{PROP})$ is strictly better than $\mathcal{PROP}$. This result is somewhat surprising. In fact, one would expect $\mathcal{PROP}$ to be able to "simulate" any element $\{f_1, \ldots, f_n\}$ of $P(\mathcal{PROP})$ with the formula $f_1 \vee \ldots \vee f_n$. However, we show that this is not the case.

The rest of the paper is organized as follows. Section 2 contains useful definitions and results about abstract interpretation theory; particularly important are the notions concerning the comparison of abstract interpretations wrt their precision. The powerset operator is defined in Sections 3 and 4: Section 3 is concerned with lifting the abstract domain, whereas in Section 4 the corresponding abstract operations are defined. The results above mentioned in the points 1 and 2 are shown in these two sections. The application of the powerset to $\mathcal{PROP}$ is described in Section 5. Finally, Section 6 contains some concluding remarks.

The results given here are described and shown in detail in the full version of the paper [FR94].

2 Preliminaries

We start by recalling the definitions of Galois connection and insertion.[1]

Definition 2.1 If C and D are posets and $\alpha : C \longrightarrow D$, $\gamma : D \longrightarrow C$ 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.

[1] For the notions of lattice theory the reader is referred to the classical textbook [Bir67].

We also recall that the above definition of G.c. is equivalent to that of adjunction: (γ, D, C, α) is an *adjunction* if $\forall c \in C. \forall d \in D.\ \alpha(c) \leq_D d \Leftrightarrow c \leq_C \gamma(d)$. α (γ) is called a *left* (*right*) *adjoint* to γ (α).

In the setting of abstract interpretation, C and D are called, respectively, the *concrete* and the *abstract domain*, whereas α and γ are called the *abstraction* and the *concretization* map, respectively. The abstract domain consists of approximated representations of properties of the values of the concrete domain. Both on the concrete and on the abstract domain, a partial order relation describing the relative precision of the values is defined: $x \leq y$ means that x is more precise than y. The concretization function gives the concrete value corresponding to an abstract property, whereas for a concrete value the abstraction function gives its best (wrt the ordering of D) abstract approximation. If (γ, D, C, α) is a G.i., then it is possible to verify that the concretization and abstraction mappings, γ and α, are 1-1 and onto, respectively. Thus in this case, 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.

We next recall some well-known results concerning Galois connections and insertions (see [CC79, GHK$^+$80, MSS86]).

Proposition 2.2

(i) *If (γ, D, C, α) is a G.c. between the posets C and D, then α preserves lub's and γ preserves glb's.*

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

(iii) *Let C and D be posets, and let $\gamma : D \longrightarrow C$ that preserves glb's; in addition, for all $c \in C$ assume that $\sqcap_D\{d \in D : c \leq_C \gamma(d)\}$ exists.*
If we define $\alpha : C \longrightarrow D$ as $\alpha(c) = \sqcap_D\{d \in D : c \leq_C \gamma(d)\}$, then (γ, D, C, α) is a G.c. between C and D.
Moreover, if γ is 1-1 then we obtain a G.i. of D in C.

In case C and D are complete lattices, *(i)* says us that α and γ are complete join and complete meet morphisms, respectively. In a G.c., and thence in an adjunction as well, one mapping uniquely determines the other (see e.g. [GHK$^+$80]). Then α defined in *(iii)* will be the only mapping such that (γ, D, C, α) is a G.c. (it is "the" left adjoint to γ). Moreover, starting with $\alpha : C \longrightarrow D$ it is possible to state the dual version of *(iii)*. We will assume from now on that both concrete and abstract domains C, D are posets, unless otherwise asserted.

If $o_1^C, \ldots, o_k^C$ are all the operations defined on the concrete domain that express the semantics, then $\mathcal{C} = \langle C, o_1^C, \ldots, o_k^C \rangle$ is called the *concrete interpretation*. Thus if (γ, D, C, α) is a G.c., it is necessary to define the *abstract operations* over the abstract domain D, simulating the behavior of the concrete operations on the properties represented by D. Therefore for each concrete operation $o_i^C : C^n \times X \longrightarrow C$, there must be defined a corresponding abstract operation $o_i : D^n \times X \longrightarrow D$, that approximates it.[2] We will call $\mathcal{D} = \langle D, o_1, \ldots, o_k \rangle$ an *abstract interpretation*. Sometimes, with a slight abuse of terminology, we will call (γ, D, C, α) an (abstract) interpretation, disregarding the abstract operations. For both concrete and abstract operations we require, as usual, the monotonicity. In the rest of the paper,

[2]Both at the concrete and at the abstract level, X is any possible set of auxiliary parameters also mathematically unstructured.

for simplicity of notation, we will consider both the concrete and the abstract operations working on a single argument, viz., $o_C : C \times X \longrightarrow C$, and correspondingly, $o_D : D \times X \longrightarrow D$. The extension to the product domain is notationally tedious, but conceptually straightforward. Furthermore, we will always refer to properties of the concrete and abstract operations (e.g. monotonicity and join completeness) wrt their first argument.

Definition 2.3 ([CC77]) Let (γ, D, C, α) be a G.c..

(i) Let $o_C : C \times X \longrightarrow C$ be a concrete operation, and let $o_D : D \times X \longrightarrow D$ be a corresponding abstract operation.
We say that o_D *approximates* (or is an *approximation* of) o_C if $\forall d \in D. \forall x \in X.$ $o_C(\gamma(d), x) \leq_C \gamma(o_D(d, x))$.

(ii) If $o_1^C, \ldots, o_k^C$ are the concrete operations respectively approximated by $o_1, \ldots, o_k$, then we say that $\mathcal{D} = \langle D, o_1, \ldots, o_k \rangle$ *abstracts* $\mathcal{C} = \langle C, o_1^C, \ldots, o_k^C \rangle$.

The next definition compares the precision of representation of two different domains that abstract the same concrete domain.

Definition 2.4 ([CFW92]) Let $(\gamma_1, D_1, C, \alpha_1)$ and $(\gamma_2, D_2, C, \alpha_2)$ be G.i.'s. Define the following maps:

$$\begin{array}{cc} f : D_1 \longrightarrow D_2 & g : D_2 \longrightarrow D_1 \\ f(d_1) = \alpha_2(\gamma_1(d_1)) & g(d_2) = \alpha_1(\gamma_2(d_2)). \end{array}$$

We say that D_2 *C-abstracts* D_1 if (g, D_2, D_1, f) is a G.i..
Furthermore, we say that D_2 *properly* C-abstracts D_1, if D_2 C-abstracts D_1 and if D_1 does not C-abstract D_2 (i.e. (f, D_1, D_2, g) is not a G.i.).

The fact that D_2 C-abstracts D_1 intuitively means that D_1 is "less abstract" than D_2 wrt C, and thus D_1's quality of representation is not worse; the quality is surely better in the case of proper abstraction. This intuition is made more precise by the following result.[3]

Proposition 2.5 ([CFW92]) *Let $(\gamma_1, D_1, C, \alpha_1)$ and $(\gamma_2, D_2, C, \alpha_2)$ be G.i.'s.*
D_2 (properly) C-abstracts D_1 iff $\gamma_2(D_2) \subseteq \gamma_1(D_1)$ ($\gamma_2(D_2) \subset \gamma_1(D_1)$).

Notice that if D_2 properly C-abstracts D_1 then there exists $d_1 \in D_1$ such that $\bar{c} = \gamma_1(d_1) \notin \gamma_2(D_2)$, or equivalently $\bar{c} = \gamma_1(\alpha_1(\bar{c}))$ but $\bar{c} <_C \gamma_2(\alpha_2(\bar{c}))$, i.e. there exists a concrete value $\bar{c}$ that can be represented in D_1 without loss of information, whereas this is not possible in D_2.

The next definition formalizes a classical criterion of comparison between abstract interpretations ([CC79]). It is concerned only with the precision of the analyses, whereas it completely neglects their complexity.

Definition 2.6 If $\mathcal{D}_1 = \langle D_1, o_1^{D_1}, \ldots, o_k^{D_1} \rangle$ and $\mathcal{D}_2 = \langle D_2, o_1^{D_2}, \ldots, o_k^{D_2} \rangle$ both abstract the concrete interpretation $\mathcal{C} = \langle C, o_1^C, \ldots, o_k^C \rangle$, then we say that $\mathcal{D}_1$ is *better* than $\mathcal{D}_2$ if:

- D_2 C-abstracts D_1;

[3] In the paper we write $S \subset T$ if S is a proper subset of T, and, as usual, if $\leq$ is a partial ordering then $a < b$ stands for $a \leq b$ and $a \neq b$.

- every abstract operation $o_i^{D_1}$ is (a) *better* (*approximation* of o_i^C) than the corresponding operation $o_i^{D_2}$,
  i.e. $\forall c \in C. \forall x \in X.\ \gamma_1(o_i^{D_1}(\alpha_1(c), x)) \leq_C \gamma_2(o_i^{D_2}(\alpha_2(c), x))$.

Moreover, we say that $\mathcal{D}_1$ is *strictly* better than $\mathcal{D}_2$ if:

- $\mathcal{D}_1$ is better than $\mathcal{D}_2$;
- D_2 properly C-abstracts D_1;
- there exists at least one operation $o_j^{D_1}$ *strictly* better than the corresponding $o_j^{D_2}$, i.e. $o_j^{D_1}$ is better than $o_j^{D_2}$ and $o_j^{D_2}$ is not better than $o_j^{D_1}$ (thence, there exist $c \in C$ and $x \in X$ such that $\gamma_1(o_j^{D_1}(\alpha_1(c), x)) <_C \gamma_2(o_j^{D_2}(\alpha_2(c), x))$).

Lemma 2.7 *If $o_j^{D_1}$ is better than $o_j^{D_2}$, and $(\gamma_1, D_1, C, \alpha_1)$ is a G.i., then $o_j^{D_1}$ is strictly better than $o_j^{D_2}$ iff $\exists d_1 \in D_1. \exists x \in X. \gamma_1(o_j^{D_1}(d_1, x)) <_C \gamma_2(o_j^{D_2}(\alpha_2(\gamma_1(d_1)), x))$.*

3 The Powerset Abstract Domain $P(D)$

In this section we describe the powerset operator on abstract domains. The construction relies on [CFW93]. The extension to the abstract operations is treated in the next section.

In what follows we assume that C is the concrete domain, D is an abstract domain and (γ, D, C, α) is a G.c.. For the following definitions the assumption of treating a G.c. between C and D suffices. Only at the moment of comparing $P(D)$ and D we will need to assume that there is a G.i. of D in C. We also assume that the concrete domain is a completely meet-distributive lattice, i.e. C must be a complete lattice such that for each family $\{c_j^i\}_{j \in J(i)}^{i \in I} \subseteq C$, $\sqcap_{i \in I} \sqcup_{j \in J(i)} c_j^i = \sqcup_{\varphi \in \mathcal{F}_{I \to J}} \sqcap_{i \in I} c_{\varphi(i)}^i$, where $\mathcal{F}_{I \to J}$ is the set of the functions φ defined over I and with values $\varphi(i) \in J(i)$. This condition on the concrete domain C is not very restrictive, as it is satisfied by the powerset of any set, ordered with the set-theoretic inclusion, and most concrete domains drawn in literature are of this sort. Notice that if we assume a G.i. of D in C, then, by Proposition 2.2 *(ii)*, the fact that C is a complete lattice implies that so is D.

We define the following relation between subsets of D:

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

where as usual $\sqcup_C \emptyset = \bot_C$. Obviously $\equiv_\gamma$ is an equivalence relation on $\wp(D)$, and if $S \subseteq D$ we denote its equivalence class by $[S]_\gamma = \{Z \subseteq D : S \equiv_\gamma Z\}$. By this definition, two sets of abstract values are equivalent if the disjunctions (i.e. lub's) of their concrete meanings coincide. The abstract domain that we obtain by applying the powerset operator to D is defined as

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

$P(D)$ is defined as the quotient of $\wp(D)$ modulo $\equiv_\gamma$ in order to identify the equivalent subsets. An element $[S]_\gamma$ of the new abstract domain $P(D)$ assumes the intuitive meaning of logical disjunction of the abstract properties represented by the elements in S.

We define the *fat* of $[S]_\gamma \in P(D)$ as $\uplus[S]_\gamma = \cup\{Z \subseteq D : Z \in [S]_\gamma\}$. We will now see that $\uplus[S]_\gamma \equiv_\gamma S$, and thus $[\uplus[S]_\gamma]_\gamma = [S]_\gamma$. Since $\uplus[S]_\gamma$ is a superset of each member of $[S]_\gamma$, we will use it as the canonical representative of the class.

Lemma 3.1 *For* $[S]_\gamma \in P(D)$,

(i) $\uplus[S]_\gamma \equiv_\gamma S$;

(ii) $\uplus[S]_\gamma = \{d \in D : \gamma(d) \leq_C \sqcup_C \gamma(S)\}$.

We now define the following relation on $P(D)$:

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

Notice that $\sqsubseteq$ is a partial order on $P(D)$ (antisymmetry follows by Lemma 3.1 *(i)*). In $P(D)$ there are the top and the bottom elements: indeed by the ordering definition $\top_{P(D)} = [D]_\gamma$, and using Lemma 3.1 *(ii)* it is easy to verify that $\bot_{P(D)} = [\emptyset]_\gamma$. If, in addition, in D there is the top $\top_D$ (it is possible to show that this holds if we start from a G.i. rather than a G.c.), then $\top_{P(D)} = [\{\top_D\}]_\gamma$.

Proposition 3.2 *$P(D)$ is a complete lattice,*
where if $\mathcal{S} = \{[S_i]_\gamma \in P(D) : i \in I\}$ then $\sqcup_{P(D)}\mathcal{S} = [\cup_{i\in I} \uplus [S_i]_\gamma]_\gamma$ and $\sqcap_{P(D)}\mathcal{S} = [\cap_{i\in I} \uplus [S_i]_\gamma]_\gamma$.
Indeed, the lub can be simplified as follows: $\sqcup_{P(D)}\mathcal{S} = [\cup_{i\in I} S_i]_\gamma$.

Next we are going to define the concretization map from the new powerset domain $P(D)$ in C:

$$\gamma^\star : P(D) \longrightarrow C$$
$$\gamma^\star([S]_\gamma) = \sqcup_C \gamma(S).$$

Obviously the concretization is well-defined, and its definition is very natural (recall the disjunctive meaning of each $[S]_\gamma$, above mentioned).

We exploit the hypothesis of complete meet-distributivity of the concrete domain C in the proof of the next lemma.

Lemma 3.3 *$\gamma^\star$ is a 1-1 complete meet morphism.*

The above lemma and Proposition 2.2 *(iii)* suggest to define the abstraction map as the left adjoint to $\gamma^\star$ in the usual way:

$$\alpha^\star : C \longrightarrow P(D)$$
$$\alpha^\star(c) = \sqcap_{P(D)}\{[S]_\gamma \in P(D) : c \leq_C \gamma^\star([S]_\gamma)\}.$$

Therefore, we have shown the following result.

Proposition 3.4 *If (γ, D, C, α) is a G.c., where C is a completely meet-distributive lattice, then $(\gamma^\star, P(D), C, \alpha^\star)$ is a G.i..*

Let us now turn to the comparison of $P(D)$ with D. Of course the precision of $P(D)$ is not worse than that of D.

Proposition 3.5 *D C-abstracts $P(D)$.*

Recalling Definition 2.4, the above proposition tells us that $(\alpha^\star \circ \gamma, D, P(D), \alpha \circ \gamma^\star)$ is a G.i.. In order to find a stronger relationship between D and $P(D)$, we need to assume, in addition to the complete meet-distributivity of C, also that (γ, D, C, α) is a G.i. (hence, as above observed, also D is a complete lattice).

Proposition 3.6 *If (γ, D, C, α) is a G.i., then for every $[S]_\gamma \in P(D)$ and $d \in D$, $\alpha \circ \gamma^\star([S]_\gamma) = \sqcup_D S$ and $\alpha^\star \circ \gamma(d) = [\{d\}]_\gamma$.*

The above proposition and the fact that for all $d \in D$, $\gamma^\star([\{d\}]_\gamma) = \gamma(d)$, emphasize that the expressive capabilities of D is maintained in $P(D)$ by the elements $[\{d\}]_\gamma$, as one would expect.

We now establish a necessary and sufficient condition for having that D properly C-abstracts $P(D)$.

Theorem 3.7 *Let (γ, D, C, α) be a G.i. (where C is a completely meet-distributive lattice). Then D properly C-abstracts $P(D)$ iff γ is not join complete.*

Corollary 3.8 *(γ, D, C, α) and $(\gamma^\star, P(D), C, \alpha^\star)$ are equivalent interpretations iff γ is a complete join morphism.*

In Section 5 we will show that the concretization map of the interpretation for the analysis of logic programs with abstract domain *Prop* satisfies this condition, viz. is not join complete.

The following are alternative ways of expressing the join completeness condition of the concretization.

Proposition 3.9 *In the hypotheses of Theorem 3.7 the following are equivalent:*[4]

(i) γ *is a complete join morphism;*

(ii) $\gamma(D)$ *is closed under lub's, i.e.* $\forall T \subseteq \gamma(D).\ \sqcup_C T \in \gamma(D)$*;*

(iii) $\gamma(D)$ *is a complete sublattice of* C*;*

(iv) $\forall [S]_\gamma \in P(D).\ \sqcup_D S \in \uplus [S]_\gamma$.

It is easy to verify that $\gamma^\star(P(D)) \setminus \gamma(D) = \gamma^\star(P(D) \setminus \{[\{d\}]_\gamma : d \in D\})$ (see [FR94]). This observation suggests us to call *new* each element of the power domain in $P(D) \setminus \{[\{d\}]_\gamma : d \in D\}$. Thence, $[S]_\gamma \in P(D)$ will be new iff $\sqcup_C \gamma(S) <_C \gamma(\sqcup_D S)$.

As it is expectable, the powerset operator is idempotent: if we say that two G.i.'s are equivalent when one abstract domain C-abstracts the other one and vice versa (viz., by Proposition 2.5, the images of the domains via the concretization mappings coincide), then the following holds.

Proposition 3.10 *Let (γ, D, C, α) be a G.c., where C is a completely meet-distributive lattice. Then $(\gamma^\star, P(D), C, \alpha^\star)$ and $(\gamma^{\star\star}, P(P(D)), C, \alpha^{\star\star})$ are equivalent interpretations.*

4 Operations over $P(D)$

Let us suppose that the sufficient hypotheses for having that $(\gamma^\star, P(D), C, \alpha^\star)$ is a G.i. hold, viz. (γ, D, C, α) is a G.c. in which C is a completely meet-distributive lattice. Moreover, let us suppose that $o_C : C \times X \longrightarrow C$ is a concrete operation approximated by $o_D : D \times X \longrightarrow D$. We want to define an operation over the abstract domain $P(D)$, such that it extends o_D and that still approximates o_C.

Let us consider this operation over $P(D)$:

[4] The equivalence *(i)* $\Leftrightarrow$ *(iii)* was already stated in [CC79].

$$o^\star_D : P(D) \times X \longrightarrow P(D)$$
$$o^\star_D([S]_\gamma, x) = [\{o_D(d, x) \in D : d \in \uplus[S]_\gamma\}]_\gamma.$$

This mapping is well-defined, and its definition is fairly natural since it consists in applying o_D to the elements of the canonical representative (the fat) of the argument, and then to take the equivalence class of the obtained set. Furthermore, since the definition of $o^\star_D$ is based on o_D, in the case that the initial abstract domain D is finite and o_D is finitely computable (usually verified hypotheses), this operation is also finitely computable (as each $\uplus[S]_\gamma$ is finite). The monotonicity condition of $o^\star_D$ also holds.

Proposition 4.1 *$o^\star_D$ is monotonic.*

Unfortunately, in general, it is not true that $o^\star_D$ approximates o_C, i.e. it is not necessarily true that if $[S]_\gamma \in P(D)$ and $x \in X$ then $o_C(\gamma^\star([S]_\gamma), x) \leq_C \gamma^\star(o^\star_D([S]_\gamma, x))$. An example showing this fact is given in [FR94]. However, we describe below a sufficient condition for the correctness of the operation $o^\star_D$.

Proposition 4.2 *If o_C is join complete on the concretization of fat sets, i.e. for $\uplus[S]_\gamma \in \wp(D)$ and $x \in X$, $o_C(\sqcup_C\{\gamma(d) : d \in \uplus[S]_\gamma\}, x) = \sqcup_C\{o_C(\gamma(d), x) : d \in \uplus[S]_\gamma\}$, then $o^\star_D$ is an approximation of o_C.*

Clearly, if o_C is join complete it also verifies the condition of the above proposition, and thus, in such a case, $o^\star_D$ approximates o_C. Even though this condition may seem restrictive, in Section 5 we will show that it is satisfied by the concrete interpretation usually adopted for the analysis of logic programs. The concrete domain consists of sets of idempotent substitutions, and the concrete operation is the unification. This operation is join complete.

We next investigate the relationship existing between o_D and $o^\star_D$. We already know that D C-abstracts $P(D)$. Therefore, it is not surprising that $o^\star_D$ is an extension of o_D. Furthermore, $o^\star_D$ is a better approximation of o_C than o_D.

Proposition 4.3 *If o_C is join complete on the concretization of fat sets then:*

(i) $o^\star_D$ extends o_D, i.e. $\forall d \in D. \forall x \in X.\ o^\star_D([\{d\}]_\gamma, x) = [\{o_D(d, x)\}]_\gamma$;

(ii) $o^\star_D$ is better than o_D (cf. Definition 2.6).

It is straightforward to verify that *(ii)* of the above proposition also implies that o_D approximates $o^\star_D$ wrt the G.i. $(\alpha^\star \circ \gamma, D, P(D), \alpha \circ \gamma^\star)$.

If $\mathcal{D} = \langle D, o_1, \ldots, o_k\rangle$ is an abstract interpretation, then we will denote by $P(\mathcal{D}) = \langle P(D), o^\star_1, \ldots, o^\star_k\rangle$ that obtained applying the powerset operator to $\mathcal{D}$. The following theorem summarizes some of the results so far achieved.

Theorem 4.4 *Let $\mathcal{C} = \langle C, o^C_1, \ldots, o^C_k\rangle$ be a concrete interpretation abstracted by $\mathcal{D} = \langle D, o_1, \ldots, o_k\rangle$, and let $P(\mathcal{D}) = \langle P(D), o^\star_1, \ldots, o^\star_k\rangle$ be the powerset of $\mathcal{D}$. If C is a completely meet-distributive lattice and o^C_i is join complete on the concretization of fat sets for all i, then:*

(i) $P(\mathcal{D})$ abstracts $\mathcal{C}$;

(ii) $P(\mathcal{D})$ is better than $\mathcal{D}$.

5 Powerset of the Abstract Interpretation $\mathcal{PROP}$

5.1 The Concrete Interpretation $\mathcal{LP}$

We briefly recall, from the framework of [Cor92], the concrete interpretation for the analysis of logic programs.

Let us consider a finite set of variables. In order to fix the notation, we will consider the set $\mathcal{V}_n = \{x_i : 1 \leq i \leq n\}$, for some $n \in \mathbb{N}$ that is assumed to be large enough for the analysis of any program. Moreover, we will consider an alphabet $\mathcal{F}$ of function symbols. Function symbols (of arity ≥ 1) will be denoted by g, h, and constant symbols by a, b. If E is any syntactic object then $Var(E)$ will denote the set of variables occurring in E. The set of the idempotent substitutions over $\mathcal{V}_n$ and over the set of terms built on $\mathcal{V}_n$ and $\mathcal{F}$ will be denoted by $Subst_n$. Substitutions will be denoted by $\sigma, \vartheta, \delta, \eta$, their domain of definition by $Dom(\sigma)$, and their composition by $\circ$. The empty substitution will be denoted by ϵ. If σ is a substitution and E is any syntactic entity, then $\sigma(E)$ will stand for the result of applying σ to E. Observe that $Subst_n$ is not closed under composition (e.g., $\{x_1/x_2\} \circ \{x_2/g(x_1)\} = \{x_1/g(x_1), x_2/g(x_1)\}$ is not idempotent). We also recall the well-known correspondence between idempotent substitutions and sets of syntactic equations in solved form ([LMM88]). Thus, if σ is an idempotent substitution, by $Eqn(\sigma)$ we will denote the corresponding set of term equations in solved form. Over $Subst_n$ it is defined the usual relation of "more general than": if $\sigma_1, \sigma_2 \in Subst_n$ then $\sigma_1 \trianglelefteq \sigma_2 \Leftrightarrow \exists \vartheta \in Subst_n.\ \sigma_1 = \sigma_2 \circ \vartheta$.
Notice that $\trianglelefteq$ is a pre-order but not a partial order over $Subst_n$. For any set E of term equations, $mgu(E)$ is as follows: if E is not unifiable then $mgu(E) = \emptyset$, otherwise $mgu(E) = \{\delta\}$ for any idempotent mgu δ of E (recall from [LMM88] that all the idempotent mgu's of E are equal up to renaming).

The concrete domain is $\wp(Subst_n)$. The partial ordering is the set-theoretic inclusion $\subseteq$, that makes $\wp(Subst_n)$ a complete lattice.
Concrete unification is defined as follows:

$$\mathbf{u}_C : \wp(Subst_n) \times Subst_n \longrightarrow \wp(Subst_n)$$
$$\mathbf{u}_C(\Sigma, \delta) = \bigcup_{\sigma \in \Sigma} mgu(Eqn(\sigma) \cup Eqn(\delta)).$$

Notice that $\mathbf{u}_C$ is the usual unification: if σ is a calling substitution and δ is an idempotent mgu of an equation between atoms $H = B$ then $mgu(Eqn(\sigma) \cup Eqn(\delta)) = \sigma \circ mgu(H\sigma = B\sigma)$, up to renaming ([CDY91, Cor92]).
In conclusion, the concrete interpretation is $\mathcal{LP}_n = \langle \wp(Subst_n), \mathbf{u}_C \rangle$.

We now state some properties of $\mathcal{LP}_n$, that will be useful later on.

Lemma 5.1

(i) $\langle \wp(Subst_n), \subseteq \rangle$ *is a completely meet-distributive lattice.*

(ii) $\mathbf{u}_C$ *is a complete join morphism.*

5.2 The Abstract Interpretation $\mathcal{PROP}$

In this subsection the abstract interpretation $\mathcal{PROP}$ is recalled from [MSJ94].

The concrete and the abstract interpretation share the set of variables: $\mathcal{V}_n$ is both the set of program variables and the set of propositional variables. If Γ is a (non-empty) set of logical connectives (in our approach the propositional constants

F and T are not included in Γ), then the set of well-formed propositional formulae built on $\mathcal{V}_n$ and Γ will be denoted by $\Omega_n(\Gamma)$. Formulae will be denoted by f, f_i. As usual, if f, f_1, f_2 are formulae and M is a truth assignment (over $\mathcal{V}_n$), then we write $M \models f$ to indicate that M is a model of f, $f_1 \models f_2$ to indicate that f_2 is a semantic consequence of f_1, and $f_1 \equiv f_2$ to indicate that f_1 and f_2 are semantically equivalent (viz., $f_1 \models f_2$ and vice versa). Moreover, by $\mathcal{A}_n$ we will denote the set of the truth assignments, and if $M \in \mathcal{A}_n$ it will be also denoted by the n-uple $(M(x_1), \ldots, M(x_n))$.

The abstract domain consists of classes, wrt $\equiv$, of propositional formulae built with the connectives $\wedge, \vee, \leftrightarrow$. Hence,

$$Prop_n \stackrel{\text{def}}{=} \Omega_n(\{\wedge, \vee, \leftrightarrow\})_{/\equiv} \cup \{\mathsf{F}\},$$

where F denotes the class of the unsatisfiable formulae. For simplicity of notation we will write f for the class of formulae equivalent to f (this does not cause any problem, as each operation involving classes of formulae does not depend on the selected representative). $Prop_n$ is ordered by semantic consequence: $f_1 \leq f_2$ if $f_1 \models f_2$. By this ordering, $Prop_n$ assumes the (finite) complete lattice structure: the lub (glb) is the disjunction $\vee$ (conjunction $\wedge$), the top is the class of the formula $x_1 \leftrightarrow x_1$ ($\equiv \mathsf{T}$), and the bottom is F. In [CFW91], Cortesi *et al.* semantically characterized the domain $Prop_n$: they showed that $\Omega_n(\{\wedge, \vee, \leftrightarrow\})_{/\equiv}$ coincides with (the equivalence classes of) all the formulae that are satisfied by the unitary truth assignment, i.e. the assignment making true each variable of $\mathcal{V}_n$.

Next we recall the definition of the concretization map. We need to introduce an auxiliary function mapping each substitution to a truth assignment satisfying only the variables that the substitution grounds:

$$assign : Subst_n \longrightarrow \mathcal{V}_n \longrightarrow \{false, true\}$$
$$assign\ \sigma\ x_i \text{ iff } Var(\sigma(x_i)) = \emptyset.$$

The concretization γ is defined as follows:

$$\gamma : Prop_n \longrightarrow \wp(Subst_n)$$
$$\gamma(f) = \{\sigma \in Subst_n : \forall \sigma' \trianglelefteq \sigma.\ assign\ \sigma' \models f\}.$$

By this definition, γ maps any formula f to the set of substitutions σ having the property that when further instantiated to some substitution σ', the corresponding truth assignment $assign\ \sigma'$ satisfies f. In the sequel, we will also use the following notation: $\mathcal{M}_\sigma = \{assign\ \sigma' \in \mathcal{A}_n : \sigma' \in Subst_n,\ \sigma' \trianglelefteq \sigma\}$.

Exploiting Proposition 2.2 *(iii)*, the abstraction map is defined as the usual left adjoint to γ:

$$\alpha : \wp(Subst_n) \longrightarrow Prop_n$$
$$\alpha(\Sigma) = \wedge\{f \in Prop_n : \Sigma \subseteq \gamma(f)\}.$$

Proposition 5.2 ([CFW91]) *$(\gamma, Prop_n, \wp(Subst_n), \alpha)$ is a G.i..*

When α is applied to a singleton $\{\sigma\}$ we will write $\alpha(\sigma)$, for simplicity of notation.

In [CFW94], an explicit definition of the abstraction map α was given, showing that it acts in a very natural way.

Proposition 5.3 *For $\sigma \in Subst_n$, $\alpha(\sigma) = \wedge\{x \leftrightarrow \wedge Var(\sigma(x)) \in Prop_n : x \in Dom(\sigma)\}$ (as usual $\wedge\emptyset = \mathsf{T}$, and thus if $Var(\sigma(x)) = \emptyset$, viz. $\sigma(x)$ is a ground term, we simply obtain the conjunct x).*

The above proposition enables us to explicitly compute α for any $\Sigma \in \wp(Subst_n)$, since from Proposition 2.2 *(i)* it follows that $\alpha(\Sigma) = \vee\{\alpha(\sigma) \in Prop_n : \sigma \in \Sigma\}$. Also notice that from the above proposition $\alpha(\epsilon) = \mathsf{T}$ follows as well.

We next recall the abstract operation approximating the concrete unification $\mathbf{u}_C$. It consists of taking the conjunction of the formula f (that represents the current set of substitutions Σ) with the abstraction of the mgu δ:

$$\mathbf{u} : Prop_n \times Subst_n \longrightarrow Prop_n$$
$$\mathbf{u}(f, \delta) = f \wedge \alpha(\delta).$$

Obviously $\mathbf{u}$ is well-defined (if $f \equiv g$ then $\mathbf{u}(f, \delta) = \mathbf{u}(g, \delta)$), and its monotonicity is trivial to verify.

Thus, the abstract interpretation is $\mathcal{PROP}_n = \langle Prop_n, \mathbf{u}\rangle$. The following soundness result holds.

Proposition 5.4 ([MSJ94]) *$\mathcal{PROP}_n$ abstracts $\mathcal{LP}_n$.*

5.3 The Abstract Interpretation $P(\mathcal{PROP})$

In this subsection the powerset operator is applied to $\mathcal{PROP}_n$, and it is shown that $P(\mathcal{PROP}_n)$ is strictly better than $\mathcal{PROP}_n$.

Let $P(Prop_n)$ be the abstract domain obtained applying to $Prop_n$ the powerset operator, as described in Section 3. Moreover, let $\alpha^\star$ and $\gamma^\star$ be the mappings defined in the same section.

Proposition 5.5 *$(\gamma^\star, P(Prop_n), \wp(Subst_n), \alpha^\star)$ is a G.i..*

Proof. By Propositions 3.4 and 5.2, and Lemma 5.1 *(i)*. ∎

Since $Prop_n$ is a finite lattice all its subsets are finite too, and thus, to simplify the notation, we will denote the generic element of $P(Prop_n)$ by $[f_1, \ldots, f_k]_\gamma$. Notice that, since $\gamma(\mathsf{F}) = \emptyset$, $\bot_{P(Prop_n)} = [\mathsf{F}]_\gamma$.

Observe that, by Proposition 3.5, we already know that $Prop_n$ $\wp(Subst_n)$-abstracts $P(Prop_n)$, namely $(\alpha^\star \circ \gamma, Prop_n, P(Prop_n), \alpha \circ \gamma^\star)$ is a G.i.. Our aim is now to show that indeed $P(Prop_n)$ is strictly $\wp(Subst_n)$-abstracted by $Prop_n$. To this end, by Theorem 3.7, it suffices to find two formulae of $Prop_n$ for which γ does not preserve the lub. The reason for this is very simple. Let f_1 and f_2 be formulae of $Prop_n$, and let $\Sigma_1 = \gamma(f_1)$, $\Sigma_2 = \gamma(f_2)$ be the corresponding sets of substitutions. Since $\alpha(\Sigma_1 \cup \Sigma_2) = f_1 \vee f_2$, this formula is the best approximation of $\Sigma_1 \cup \Sigma_2$ in $Prop_n$, but it may also represent more substitutions. In fact, it is not true, in general, that $\gamma(f_1 \vee f_2) = \gamma(f_1) \cup \gamma(f_2) = \Sigma_1 \cup \Sigma_2$ (only $\supseteq$ holds). At the contrary, $[f_1, f_2]_\gamma \in P(Prop_n)$ exactly represents $\Sigma_1 \cup \Sigma_2$ (by definition of $\gamma^\star$). Two formulae showing this phenomenon are given in the example below.

Example 5.6 *Let x_1, $x_1 \leftrightarrow x_2 \in Prop_n$ $(n \geq 3)$.*
We want to verify that $\gamma(x_1) \cup \gamma(x_1 \leftrightarrow x_2) \subset \gamma(x_1 \vee (x_1 \leftrightarrow x_2))$. Hence, it suffices to find a substitution $\sigma \in Subst_n$ such that $\sigma \in \gamma(x_1 \vee (x_1 \leftrightarrow x_2))$, $\sigma \notin \gamma(x_1) \cup \gamma(x_1 \leftrightarrow x_2)$. Let $\sigma = \{x_2/g(x_1, x_3, \ldots, x_3)\}$, where g is a function symbol of arity ≥ 2.
Recall that $\sigma \in \gamma(f) \Leftrightarrow \forall M \in \mathcal{M}_\sigma.\ M \models f$. Since

$$\begin{aligned}\mathcal{M}_\sigma \;=\; & \{(\mathsf{F},\mathsf{F},\mathsf{F},v_4,\ldots,v_n) : v_i \in \{\mathsf{F},\mathsf{T}\}\} \cup \{(\mathsf{T},\mathsf{F},\mathsf{F},v_4,\ldots,v_n) : v_i \in \{\mathsf{F},\mathsf{T}\}\} \cup \\ & \{(\mathsf{F},\mathsf{F},\mathsf{T},v_4,\ldots,v_n) : v_i \in \{\mathsf{F},\mathsf{T}\}\} \cup \{(\mathsf{T},\mathsf{T},\mathsf{T},v_4,\ldots,v_n) : v_i \in \{\mathsf{F},\mathsf{T}\}\},\end{aligned}$$

we have that $\sigma \in \gamma(x_1 \vee (x_1 \leftrightarrow x_2))$.
On the other hand, assign $\sigma \not\models x_1$ *and* $\mathcal{M}_\sigma \ni (\mathsf{T}, \mathsf{F}, \mathsf{F}, v_4, \ldots, v_n) \not\models x_1 \leftrightarrow x_2$, *i.e.* $\sigma \notin \gamma(x_1) \cup \gamma(x_1 \leftrightarrow x_2)$.

Theorem 5.7 *If the alphabet* $\mathcal{F}$ *contains at least a function symbol of arity* ≥ 2, *then* $Prop_n$ *properly* $\wp(Subst_n)$*-abstracts* $P(Prop_n)$, *for all* $n \geq 3$.

Proof. Example 5.6 shows that γ is not join complete, and thus, by Theorem 3.7, we get the thesis. ■

The restriction of the above theorem, namely the existence of a function symbol of arity ≥ 2, is of little importance, as the cases $n = 1, 2$.

Let us dwell a moment on Example 5.6. Let $\Sigma = \{\sigma = \{x_1/a\}, \vartheta = \{x_1/x_2\}\} \in \wp(Subst_n)$, for which $\alpha(\sigma) = x_1$, $\alpha(\vartheta) = x_1 \leftrightarrow x_2$, and thence $\alpha(\Sigma) = x_1 \vee (x_1 \leftrightarrow x_2)$. The formula x_1 abstractly represents all and only the substitutions grounding $\{x_1\}$, whereas $x_1 \leftrightarrow x_2$ represents all and only the substitutions for which $\{x_1\}$ is equivalent to $\{x_2\}$. It is obvious that the formula $x_1 \vee (x_1 \leftrightarrow x_2)$ does not represent all and only the substitutions either grounding $\{x_1\}$ or for which $\{x_1\}$ and $\{x_2\}$ are equivalent, since $\eta = \{x_2/g(x_1, x_3)\} \in \gamma(x_1 \vee (x_1 \leftrightarrow x_2))$, and evidently η neither grounds $\{x_1\}$ nor $\{x_1\}$ and $\{x_2\}$ are equivalent wrt η. Indeed, $x_1 \vee (x_1 \leftrightarrow x_2)$ is semantically equivalent to $x_2 \rightarrow x_1$, and thus it represents the substitutions such that $\{x_2\}$ covers $\{x_1\}$. Therefore, correctly, $\eta \in \gamma(x_1 \vee (x_1 \leftrightarrow x_2))$. On the other hand, $P(Prop_n)$ can be more precise with the element $[x_1, x_1 \leftrightarrow x_2]_\gamma$, whose concretization is just $\gamma^\star([x_1, x_1 \leftrightarrow x_2]_\gamma) = \gamma(x_1) \cup \gamma(x_1 \leftrightarrow x_2)$.

From $\mathbf{u}$ defined on $Prop_n$ we obtain the abstract unification $\mathbf{u}^\star$ on $P(Prop_n)$, as described in Section 4.

$$\mathbf{u}^\star : P(Prop_n) \times Subst_n \longrightarrow P(Prop_n)$$
$$\mathbf{u}^\star([f_1, \ldots, f_k]_\gamma, \delta) = [\mathbf{u}(\uplus[f_1, \ldots, f_k]_\gamma, \delta)]_\gamma.$$

By Proposition 4.1, $\mathbf{u}^\star$ is monotone.
Furthermore, letting $P(\mathcal{PROP}_n) = \langle P(Prop_n), \mathbf{u}^\star \rangle$ be the powerset abstract interpretation the following holds.

Proposition 5.8 $P(\mathcal{PROP}_n)$ *abstracts* $\mathcal{LP}_n$, *and is better than* $\mathcal{PROP}_n$.

Proof. By Theorem 4.4, Proposition 5.4, and Lemma 5.1. ■

The definition of $\mathbf{u}^\star$ is simplified and made more natural by the result below.

Proposition 5.9 *For* $[f_1, \ldots, f_k]_\gamma \in P(Prop_n)$ *and* $\delta \in Subst_n$

$$\mathbf{u}^\star([f_1, \ldots, f_k]_\gamma, \delta) = [f_1 \wedge \alpha(\delta), \ldots, f_k \wedge \alpha(\delta)]_\gamma.$$

We now state this useful result.

Lemma 5.10 $\gamma^\star(\mathbf{u}^\star([f_1, \ldots, f_k]_\gamma, \delta)) \subset \gamma(\mathbf{u}(\alpha(\gamma^\star([f_1, \ldots, f_k]_\gamma)), \delta))$ *iff there exists* $\sigma \in \gamma(\alpha(\delta))$ *such that* $\sigma \in \gamma(f_1 \vee \ldots \vee f_k)$ *and* $\sigma \notin \gamma(f_1) \cup \ldots \cup \gamma(f_k)$.

On each new element of $P(Prop_n)$ (namely, on $P(Prop_n) \setminus \{[f]_\gamma : f \in Prop_n\}$), the unification $\mathbf{u}^\star$ provides a result strictly better than that obtainable by the corresponding computation of $\mathbf{u}$ in $Prop_n$. In fact, if $\sigma \in \gamma(f_1 \vee \ldots \vee f_k)$ and $\sigma \notin \gamma(f_1) \cup \ldots \cup \gamma(f_k)$, viz. $[f_1, \ldots, f_k]_\gamma$ is a new element of $P(Prop_n)$, then, by the

above lemma, $\gamma^\star(\mathbf{u}^\star([f_1,\ldots,f_k]_\gamma,\sigma)) \subset \gamma(\mathbf{u}(\alpha(\gamma^\star([f_1,\ldots,f_k]_\gamma)),\sigma))$. Notice that this observation does not mean that if $[f_1,\ldots,f_k]_\gamma$ is a new element of $P(Prop_n)$ then $\gamma^\star(\mathbf{u}^\star([f_1,\ldots,f_k]_\gamma,\delta)) \subset \gamma(\mathbf{u}(\alpha(\gamma^\star([f_1,\ldots,f_k]_\gamma)),\delta))$ for any (mgu) δ. The following example shows that there are some $\delta \in Subst_n$ for which this is not true.

Example 5.11 *Let $[x_1, x_1 \leftrightarrow x_2]_\gamma$ be the new element of $P(Prop_n)$ of the Example 5.6, and let $\delta = \{x_1/g(b,x_2), x_3/a\}$.*
Then, $\alpha(\delta) = (x_1 \leftrightarrow x_2) \wedge x_3$ and $\alpha(\delta) \models x_1 \leftrightarrow x_2 \models x_1 \vee (x_1 \leftrightarrow x_2)$.
The concretizations of the unifications in $\mathcal{PROP}_n$ and $P(\mathcal{PROP}_n)$ are as follows:

$$\begin{aligned}
\gamma(\mathbf{u}(\alpha(\gamma^\star([x_1, x_1 \leftrightarrow x_2]_\gamma)),\delta)) &= \gamma(\mathbf{u}(x_1 \vee (x_1 \leftrightarrow x_2),\delta)) \\
&= \gamma(x_1 \vee (x_1 \leftrightarrow x_2)) \cap \gamma(\alpha(\delta)) \text{ (γ preserves glb's)} \\
&= \gamma(\alpha(\delta)),
\end{aligned}$$

$$\begin{aligned}
\gamma^\star(\mathbf{u}^\star([x_1, x_1 \leftrightarrow x_2]_\gamma,\delta)) &= \gamma(x_1 \wedge \alpha(\delta)) \cup \gamma((x_1 \leftrightarrow x_2) \wedge \alpha(\delta)) \\
&= \gamma(x_1 \wedge \alpha(\delta)) \cup \gamma(\alpha(\delta)) \\
&= \gamma(\alpha(\delta)).
\end{aligned}$$

Therefore, $\mathbf{u}^\star([x_1, x_1 \leftrightarrow x_2]_\gamma,\delta) = [\mathbf{u}(\alpha(\gamma^\star([x_1, x_1 \leftrightarrow x_2]_\gamma)),\delta)]_\gamma$.

We then show the following key result.

Theorem 5.12 *If the alphabet $\mathcal{F}$ contains at least a function symbol of arity ≥ 2, then, for all $n \geq 3$, $P(\mathcal{PROP}_n)$ is strictly better than $\mathcal{PROP}_n$.*

Proof. By Theorem 5.7, there are new elements of $P(Prop_n)$. Thus, by Lemma 5.10, the observation following it, and Lemma 2.7, $\mathbf{u}^\star$ is strictly better than $\mathbf{u}$. Hence, cf. Definition 2.6, we obtain the thesis by Proposition 5.8 and Theorem 5.7. ■

Finally, the example below shows a case in which the analysis using $P(\mathcal{PROP}_n)$ is more precise than that using $\mathcal{PROP}_n$.

Example 5.13 *Let us consider the following program fragment.*

$$\begin{array}{l}
\vdots \\
\mathtt{p(g(U,a),U,V)} :- \ldots \\
\mathtt{p(V,W,g(W,W))} :- \ldots \\
\vdots
\end{array}$$

Furthermore, let $G = :- \dagger\, \mathtt{p(X,Y,Z)}\, \ddagger, \ldots$ be a query ($\dagger$ and $\ddagger$ are the entry and the exit point of the first literal of G).
We consider $\{\mathtt{X},\mathtt{Y},\mathtt{Z},\mathtt{U},\mathtt{V},\mathtt{W}\} \subseteq \mathcal{V}_n$ (thus $n \geq 6$), and we assume to be in the first step of a SLD-derivation, namely $\{\epsilon\}$ is the set of calling substitutions for G at the program point $\dagger$. As usual, union of sets of substitutions is approximated by lub, and projection onto a set of variables by existential quantification (for more details see [CFW94]).
The unification of G with the head of the two clauses for $\mathtt{p}$ gives the mgu's $\delta_1 = \{\mathtt{X/g(U,a)}, \mathtt{Y/U}, \mathtt{Z/V}\}$ and $\delta_2 = \{\mathtt{X/V}, \mathtt{Y/W}, \mathtt{Z/g(W,W)}\}$.
The unification on $\mathcal{PROP}_n$ yields the values (recall that $\alpha(\epsilon) = \mathsf{T}$) $\mathbf{u}(\alpha(\epsilon),\delta_1) = (\mathtt{X} \leftrightarrow \mathtt{U}) \wedge (\mathtt{Y} \leftrightarrow \mathtt{U}) \wedge (\mathtt{Z} \leftrightarrow \mathtt{V})$ and $\mathbf{u}(\alpha(\epsilon),\delta_2) = (\mathtt{X} \leftrightarrow \mathtt{V}) \wedge (\mathtt{Y} \leftrightarrow \mathtt{W}) \wedge (\mathtt{Z} \leftrightarrow \mathtt{W})$. Thus, if we project the two formulae onto the variables of G, and then take their lub, we obtain at the program point $\ddagger$ the abstract value $(\mathtt{X} \leftrightarrow \mathtt{Y}) \vee (\mathtt{Y} \leftrightarrow \mathtt{Z})$.
A similar computation on $P(\mathcal{PROP}_n)$ leads to the value $[\mathtt{X} \leftrightarrow \mathtt{Y}, \mathtt{Y} \leftrightarrow \mathtt{Z}]_\gamma$ at $\ddagger$.
Therefore, the result of the analysis on $P(\mathcal{PROP}_n)$ is strictly better than the one on

$\mathcal{PROP}_n$. *In fact,* $\gamma^\star([\mathtt{X} \leftrightarrow \mathtt{Y}, \mathtt{Y} \leftrightarrow \mathtt{Z}]_\gamma) \subset \gamma((\mathtt{X} \leftrightarrow \mathtt{Y}) \vee (\mathtt{Y} \leftrightarrow \mathtt{Z}))$. *For instance, it is possible to verify, likewise to Example 5.6, that* $\sigma = \{\mathtt{Y}/\mathtt{g}(\mathtt{X}, \mathtt{Z}), \ldots\}$ *is a substitution in* $\gamma((\mathtt{X} \leftrightarrow \mathtt{Y}) \vee (\mathtt{Y} \leftrightarrow \mathtt{Z}))$ *but not in* $\gamma(\mathtt{X} \leftrightarrow \mathtt{Y}) \cup \gamma(\mathtt{Y} \leftrightarrow \mathtt{Z}) = \gamma^\star([\mathtt{X} \leftrightarrow \mathtt{Y}, \mathtt{Y} \leftrightarrow \mathtt{Z}]_\gamma)$.

6 Conclusion

This paper proposes a general and complete study of the powerset operator on abstract interpretations. This operator produces a new full abstract interpretation, that is both a new abstract domain and new operations. Conditions that guarantee that the new abstract interpretation is safe are given, as well as a condition that guarantee that the new abstract domain is strictly better than the original one. Moreover, the general theory is applied to the abstract interpretation $\mathcal{PROP}$ for the analysis of logic programs, whose domain *Prop* consists of classes (wrt semantic equivalence) of certain propositional formulae. We obtain a strict improvement by lifting from $\mathcal{PROP}$ to its powerset $P(\mathcal{PROP})$. In some way, this is a surprising result. In fact it would seem that *Prop* is already able to express the disjunction of the properties by means of the logical connective of disjunction $\vee$, whereas the facts contradict this insight.

We plan to carry on a similar complete study of the cartesian product of abstract interpretations. The motivation of such a study is that many abstract domains proposed in literature consist of several cooperating components (see e.g. [CDY91, MH91]). We think that a general study of the cartesian product will shed some light on the functioning of such interpretations.

Acknowledgements

We are grateful to Roberto Giacobazzi for his valuable suggestions. Thanks also to the anonymous referees for their comments.

References

[BGL93] R. Barbuti, R. Giacobazzi, and G. Levi. A general framework for semantics-based bottom-up abstract interpretation of logic programs. *ACM Transactions on Programming Languages and Systems*, 15(1):133–181, 1993.

[Bir67] G. Birkhoff. *Lattice Theory.* AMS Colloquium Publications, third edition, 1967.

[CC77] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *ACM POPL '77*, pages 238–252, 1977.

[CC79] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *ACM POPL '79*, pages 269–282, 1979.

[CC92] P. Cousot and R. Cousot. Abstract interpretation and application to logic programs. *Journal of Logic Programming*, 13(2,3):103–179, 1992.

[CDG93] M. Codish, S.K. Debray, and R. Giacobazzi. Compositional analysis of modular logic programs. In *ACM POPL '93*, pages 451–464, 1993.

[CDY91] M. Codish, D. Dams, and E. Yardeni. Derivation and safety of an abstract unification algorithm for groundness and aliasing analysis. In *ICLP '91*, pages 79–96, 1991.

[CFW91] A. Cortesi, G. Filé, and W. Winsborough. Prop revisited: propositional formulas as abstract domain for groundness analysis. In *IEEE LICS '91*, pages 322–327, 1991.

[CFW92] A. Cortesi, G. Filé, and W. Winsborough. Comparison of abstract interpretations. In *ICALP '92, LNCS Vol. 623*, pages 521–532, 1992.

[CFW93] A. Cortesi, G. Filé, and W. Winsborough. Internal note. Dip. di Mat. Pura ed Appl., Università di Padova, 1993.

[CFW94] A. Cortesi, G. Filé, and W. Winsborough. Optimal groundness analysis using propositional logic. Draft Version, 1994.

[Cor92] A. Cortesi. *Domini Astratti per l'Analisi Statica di Programmi Logici.* PhD thesis, Università di Padova, 1992. In Italian.

[FR94] G. Filé and F. Ranzato. The powerset operator on abstract interpretations. Technical Report, Dip. di Mat. Pura ed Appl., Università di Padova, 1994.

[GHK+80] G. Gierz, K.H. Hofmann, K. Keimel, J.D. Lawson, M. Mislove, and D.S. Scott. *A Compendium of Continuous Lattices.* Springer-Verlag, 1980.

[LMM88] J.L. Lassez, M.J. Maher, and K. Marriott. Unification revisited. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 587–625. Morgan Kaufmann, 1988.

[MH91] K. Muthukumar and M. Hermenegildo. Combined determination of sharing and freeness of program variables through abstract interpretation. In *ICLP '91*, pages 49–63, 1991.

[MS89] K. Marriott and H. Søndergaard. Notes for a tutorial on abstract interpretation of logic programs. In *NACLP '89*, 1989.

[MSJ94] K. Marriott, H. Søndergaard, and N.D. Jones. Denotational abstract interpretation of logic programs. *ACM Transactions on Programming Languages and Systems*, 1994. To appear.

[MSS86] A. Melton, D. A. Schmidt, and G. E. Strecker. Galois connections and computer science applications. In *Proc. of the Workshop on Category Theory and Computer Programming, LNCS Vol. 240*, pages 299–312, 1986.