Abstract domains for sharing analysis by optimal semantics

Francesca Scozzari*

Dipartimento di Informatica
Università di Pisa
Corso Italia 40, 56125 Pisa, Italy
scozzari@di.unipi.it

Abstract. We propose a new technique for transforming abstract domains for logic program analysis in the theory of abstract interpretation. The basic idea is to exploit the notion of optimal semantics in order to improve the precision of a given analysis, with respect to a fixed property of interest. We show an application of our technique to the analysis of variable sharing. We propose a new domain for detecting pairs of independent variables which is obtained by transforming the Jacobs and Langen's domain for sharing analysis. The new domain has the advantage of being strictly more powerful than the original domain in detecting pair-sharing information and, at the same time, smaller in size.

1 Introduction

This work presents a new technique for transforming abstract domains for program analysis, in the theory of abstract interpretation [7]. Abstract interpretation is a general theory for describing both approximated semantics and program analyses. On the semantics side, it allows us to compare semantics at different levels of abstraction and to order them in a suitable hierarchy, which reflects the approximation order. From the analysis point of view, abstract domains proved to be a powerful and elegant method to formally describe and compare static analyses for a variety of programming languages.

In this paper we face with the problem of improving the precision of abstract domains for static analysis of logic programs. We propose a new methodology for transforming an abstract domain in order to improve its precision in computing some fixed property of interest. The basic idea is to exploit the notion of *optimal semantics* [10, 11] as a guideline for the definition of the transformation. An optimal semantics can be thought of as a semantics which is neither too concrete nor too abstract for characterizing a given property of interest. We consider as a starting point the semantics of computed answers substitutions, the so-called s-semantics [9]. As shown in [11], given any property of interest π (e.g., groundness, sharing), it always exists the most abstract semantics σ^π where the property π is

* Part of this work was carried out while the author was at the *Laboratoire d'Informatique, École Polytechnique*, Palaiseau, France.

observable without approximation errors. In abstract interpretation, it is often the case that the actual domain used for performing the analysis is not π, but a more complex domain, which contains π and, in addition, some other information which improves the precision of the analysis. For instance, this is the case for the basic property of groundness, where the more complex domains Def [1] and Pos [16, 5, 1] are used instead of the basic domain G by Jones and Søndergaard ([14]) which merely describes the property of groundness. Another example is the property of variable sharing, where the abstract domain Sh by Jacobs and Langen [12, 13], and even more complex domains which keep track of linearity and freeness information, are used for sharing analyses.

Our idea is to exploit the optimal semantics of a given property π for enriching any abstract domain for the analysis of π. Let A be an abstract domain for detecting the property π, i.e., which contains π. The idea is to substitute each abstract object $a \in A$ with the object $\sigma^{\pi}(a)$, where σ^{π} is the closure corresponding to the optimal semantics for π, and to consider the most abstract domain B which contains all these objects, for all $a \in A$. The new domain that we obtain is therefore the most abstract domain which contains the set $\{\sigma^{\pi}(a) \mid a \in A\}$. As a remarkable point of this construction, the new domain B turns out to be, at least, as precise as A in computing the property π and, when B is different from A, it is strictly more precise than A. It is worth noting that the new domain we set up can be unrelated to the original domain A, when the common order for comparing abstract domains is used. In other words, B can be neither an abstraction nor a concretization of A. Therefore, the method we propose is not a refinement, nor a widening operator on the domain A.

As an application, we consider the pair-sharing property ([2]), whose aim is to detect which pairs of variables are definitely independent. One of the most commonly used domain for pair-sharing analysis is the domain Sh [12, 13]. We fix the pair-sharing property and we apply our transformation technique to the abstract domain Sh. The result is quite surprising, since the domain we obtain is strictly more powerful than Sh in detecting pair-sharing information and, at the same time, is smaller in size. It is worth noting that the domain we obtain is incomparable to Sh, i.e., it is neither an abstraction nor a concretization of Sh. Moreover, it clarifies how groundness information helps in pair-sharing analysis.

2 Basic notions

Abstract interpretations. In the abstract interpretation theory [7, 8], abstract domains can be equivalently specified either by Galois connections, i.e., adjunctions, or by upper closure operators (uco). In the first case, the concrete domain C and the abstract domain A (both assumed to be complete lattices) are related by a pair of adjoint functions of a Galois connection (α, C, A, γ). Also, it is generally assumed that (α, C, A, γ) is a Galois insertion, i.e., α is onto or, equivalently, γ is 1–1. In the latter case, an abstract domain is specified as an uco on the concrete domain C, i.e., a monotone, idempotent and extensive operators on C. These two approaches are equivalent, modulo isomorphic representation of domain's objects. When $\langle C, \leq, \vee, \wedge, \top, \bot \rangle$ is a complete lattice, the poset of all uco's $\langle uco(C), \sqsubseteq, \sqcup, \sqcap, \lambda x.\top, \lambda x.x \rangle$ is a complete lattice as well. Hence, we will identify $uco(C)$ with the lattice of abstract interpretations of C, i.e., the complete lattice of all possible abstract domains of the concrete domain C. For an abstract domain A of C, $\alpha_A : C \mapsto A$ and $\gamma_A : A \mapsto C$ will denote, respectively, the abstraction and concretization functions. When A is specified by a Galois connection, then $\rho_A = \gamma_A \circ \alpha_A \in uco(C)$ denotes the corresponding uco. Let us recall that each $\rho \in uco(C)$ is uniquely determined by the set of its fixpoints, which is its image, i.e., $\rho(C) = \{x \in C \mid \rho(x) = x\}$, and that, for $\rho, \eta \in uco(C)$, $\rho \sqsubseteq \eta$ iff $\eta(C) \subseteq \rho(C)$. A subset $X \subseteq C$ is the set of fixpoints of an uco iff X is meet-closed, i.e., $X = \curlywedge(X) = \{\wedge Y \mid Y \subseteq X\}$. Hence, often, we will identify closures with their sets of fixpoints. This does not give rise to ambiguity, since one can distinguish their use as functions or sets according to the context. The ordering on $uco(C)$ corresponds to the standard order used to compare abstract domains: A_1 is *more concrete* than A_2 (or, equivalently, A_2 is *more abstract* than A_1) iff $A_1 \sqsubseteq A_2$ in $uco(C)$. The *lub* and *glb* on $uco(C)$ have therefore the following meaning as operators on domains. Suppose $\{A_i\}_{i \in I} \subseteq uco(C)$: (i) $\sqcup_{i \in I} A_i$ is the most concrete among the domains which are abstractions of all the A_i's, i.e., it is their least common abstraction; (ii) $\sqcap_{i \in I} A_i$ is the most abstract among the domains (abstracting C) which are more concrete than every A_i; this domain is also known as *reduced product* of all the A_i's.

Logic programming. Let $\mathcal{V}$ be an infinite, recursively enumerable (r.e.) set of variables. *Term* denotes the set of terms with variables in $\mathcal{V}$. A *substitution* is a finite mapping from $\mathcal{V}$ to *Term*. If s is any syntactic object and σ and θ are substitutions, then $\sigma(s)$ denotes the application of σ to s, $\sigma \circ \theta$ denotes the standard composition of σ and θ (i.e., $\sigma \circ \theta = \lambda x.\sigma(\theta(x))$), $vars(s)$ denotes the set of variables occurring in s. A term t is ground if $vars(t) = \emptyset$. The set of *idempotent* substitutions modulo renaming (i.e., $\theta \sim \sigma$ if and only if there exist β and δ such that $\theta = \beta \circ \sigma$ and $\sigma = \delta \circ \theta$) is denoted by *Sub*. *Sub* is partially ordered by instantiation, denoted by $\preceq$, i.e., $\sigma \preceq \theta$ iff $\exists \delta \in Sub.\ \sigma = \delta \circ \theta$. By adding to *Sub* an extra object τ as least element, one gets a complete lattice $\langle Sub^{\tau}, \preceq, \vee, \wedge, \epsilon, \tau \rangle$, where $\vee$ is the least general anti-instance, $\wedge$ is the standard unification and ϵ is the empty substitution (see [18, 15] for more details). Our basic semantic structure is the algebra $\langle \wp(Sub), \subseteq, \otimes \rangle$, where $\langle \wp(Sub), \subseteq \rangle$ is a complete lattice, $\otimes : \wp(Sub) \times \wp(Sub) \mapsto \wp(Sub)$ is the obvious lifting of unification $\wedge$ to sets of substitutions, defined as:

$$X \otimes Y \stackrel{\text{def}}{=} \{x \wedge y \mid x \wedge y \neq \tau, x \in X,\ y \in Y\}.$$

In the following, we will slightly abuse the notation by applying the operation $\otimes$ also to substitutions. Since $\otimes$ is additive on both arguments, commutative and associative, given any $c \in \wp(Sub)$, the unary function $\lambda x.c \otimes x$ is still additive. Thus it admits a right adjoint given by $\lambda x. \bigcup \{d \in \wp(Sub) \mid c \otimes d \subseteq x\}$. In the following, we denote by $y \multimap x$ the object $\bigcup \{d \in \wp(Sub) \mid y \otimes d \subseteq x\}$.

3 Optimal semantics

An optimal semantics can be thought of as a semantics which is neither too concrete nor too abstract for characterizing a given property of interest. We briefly recall from [11] the definition and the main results concerning optimal semantics for logic programs. Given a program P, let $T_P^s : \wp(Sub) \mapsto \wp(Sub)$ be the immediate consequences operator for computed answer substitutions [9], and let us define the semantics of a program P as the least fixpoint of T_P^s, i.e., $[\![P]\!]^s = \mathit{lfp}(T_P^s)$. Let $\pi \in \mathit{uco}(\wp(Sub))$ be the abstract domain describing the property of interest. We say that π is *decidable* when each set $a \in \pi$ is a r.e. set. In [11] it is proved that, for decidable domains, there exists the most abstract domain $\sigma^\pi \in \mathit{uco}(\wp(Sub))$ such that π is still observable in σ^π without introducing approximation errors, as shown in the next theorem.

Theorem 1 ([11]). *Let $\pi \in \mathit{uco}(\wp(Sub))$ be a decidable domain and let $\sigma^\pi : \wp(Sub) \mapsto \wp(Sub)$ be defined as follows, for any $x \in \wp(Sub)$:*

$$\sigma^\pi(x) \stackrel{\text{def}}{=} \bigcap \{\theta \multimap a \mid \theta \in Sub, a \in \pi, x \subseteq \theta \multimap a\}.$$

Then it holds $\pi(\mathit{lfp}(T_P^s)) = \pi(\mathit{lfp}(\sigma^\pi \circ T_P^s))$. Moreover, σ^π is the most abstract domain in $\mathit{uco}(\wp(Sub))$ which enjoys this property.

This theorem ensures us that the result of any abstract computation on the domain σ^π (i.e., $\mathit{lfp}(\sigma^\pi \circ T_P^s)$) is equal to the result on the concrete domain $\wp(Sub)$ (i.e., $\mathit{lfp}(T_P^s)$) when they are observed on the abstract domain π. Therefore, the domain σ^π can be freely used instead of the concrete domain without introducing approximation errors. Moreover, σ^π is the most abstract domain (abstracting $\wp(Sub)$) for which this property holds. Thus, σ^π can be rightfully considered as the optimal semantics for the property π. It is worth noting that being a decidable domain is a very weak notion and practically all the common domains used in practice are decidable domains. In fact, an abstract domain π is not decidable when there exists an abstract object $a \in \pi$ such that a is not r.e., i.e., one cannot decide when a substitution θ belongs to a. Therefore, the property π is not decidable even for single substitutions, and the abstraction function $\alpha_\pi : \wp(Sub) \mapsto \pi$ cannot be effectively computed. In the following, we will always assume that the property of interest is representable by a decidable domain $\pi \in \mathit{uco}(\wp(Sub))$ and we will denote by σ^π the optimal semantics for π. The abstraction σ^π can be equivalently defined as follows (see [11, 19] for more details):

- σ^π is the most abstract domain which contains π (i.e., $\sigma^\pi \sqsubseteq \pi$) and is complete for $\otimes$, i.e., for any $a, b \in \wp(Sub)$, it holds:

  $$\sigma^\pi(a \otimes b) = \sigma^\pi(\sigma^\pi(a) \otimes \sigma^\pi(b));$$

- for any $x \in \wp(Sub)$:

  $$\sigma^\pi(x) = \{\theta \in Sub \mid \forall c, d \in \wp(Sub)\ x \otimes c \subseteq \pi(d) \Rightarrow \theta \otimes c \subseteq \pi(d)\}.$$

The first characterization of σ^π shows the strict connection between the immediate consequences operator T^s_P and the unification operation $\otimes$. In fact, [11] showed that an abstract domain ρ is complete for T^s_P (i.e., $\rho \circ T^s_P \circ \rho = \rho \circ T^s_P$) if and only if ρ is complete for $\otimes$ (i.e., $\rho \circ \otimes \circ \langle \rho, \rho \rangle = \rho \circ \otimes$). According to this result, one can disregard the other operations involved in the computation of the T^s_P, namely union and variable projecting, in order to prove completeness properties for the semantics of computed answer substitutions.

The latter characterization of σ^π is particularly interesting, since it precisely describes the way optimal semantics behave. For any $x \in \wp(Sub)$, it states that a substitution θ belongs to $\sigma^\pi(x)$ if and only if the following property holds:

$$\forall c, d \in \wp(Sub)\ x \otimes c \subseteq \pi(d) \Rightarrow \theta \otimes c \subseteq \pi(d). \tag{1}$$

According to this property, the optimal semantics enriches the set x with all and only the substitutions θ such that the unification of θ with any object c cannot be distinguished from the unification of x with c, when they are observed in π. This property also explains in which sense σ^π is optimal. In fact, for any substitution θ which is not in $\sigma^\pi(x)$, there always exist two objects $c, d \in \wp(Sub)$ such that $x \otimes c \subseteq \pi(d)$ but $\theta \otimes c \not\subseteq \pi(d)$. As a consequence, we have that $\theta \otimes c \not\subseteq \pi(x \otimes c)$ and thus Property (1) can be rephrased in a more compact way as follows:

$$\forall c \in \wp(Sub)\ \ \theta \otimes c \subseteq \pi(x \otimes c).$$

This last characterization gives us a precise upper bound to the set of substitutions which can be added still preserving the property π in the unification operation.

4 Abstract domains by optimal semantics

In abstract interpretation, it is often the case that the abstract domain π, which encodes the property of interest, is not actually used for the analysis. Instead of π, some more complex domain α (which contains π) is used, in order to improve the precision of the analysis. For instance, this is the case for the domains Def [1] and Pos [16, 5, 1], which are commonly used instead of the basic domain G by Jones and Søndergaard ([14]), encoding the property of groundness.

Let $\pi \in uco(\wp(Sub))$ be a fixed property of interest and let us consider some domain $\alpha \in uco(\wp(Sub))$ such that $\alpha \sqsubseteq \pi$. Our idea is to consider, instead of the domain α, the abstract domain α^π obtained by abstracting each element $a \in \alpha$ by σ^π. The domain α^π is formally defined as follows:

$$\alpha^\pi \stackrel{\text{def}}{=} \curlywedge(\sigma^\pi(\alpha)) = \curlywedge(\{\sigma^\pi(a) \mid a \in \alpha\}).$$

By Property (1), $\sigma^\pi(a)$ is obtained by enriching a with all and only the substitutions which behave like a when the result of the unification is observed in π. Let us first show some properties of α^π.

Proposition 1. *Let $\alpha, \pi \in uco(\wp(Sub))$ with $\alpha \sqsubseteq \pi$. Then α^π enjoys the following properties.*

1. $\sigma^\pi \sqsubseteq \alpha^\pi \sqsubseteq \pi$
2. $\alpha^\pi \circ \alpha = \sigma^\pi \circ \alpha$
3. α^π *is complete for* $\otimes$ *w.r.t.* α*, i.e., for all* $a, b \in \alpha$*, it holds:*
$$\alpha^\pi(a \otimes b) = \alpha^\pi(\alpha^\pi(a) \otimes \alpha^\pi(b)).$$
4. $lfp(\alpha^\pi \circ T^s_P) \leq \alpha^\pi(lfp(\alpha \circ T^s_P))$

Note that, consistently with Point 3, it may well happen that α^π is complete for $\otimes$ w.r.t. α, even if α^π and α are possibly unrelated domains. In particular, it is not required for α^π to be an abstraction of α. When α is more concrete than α^π, this notion boils down to the standard notion of completeness. Point 4 is a direct consequence of Point 3 and ensures us that abstract computations on the domain α^π are more precise than abstract computations on α when the result is observed in α^π. Since $\alpha^\pi \sqsubseteq \pi$, as a consequence α^π is, at least, as precise as α when we are interested in observing the property π. Moreover, the domain α^π still enjoys Property (1) for any abstract object $a \in \alpha$, as stated in the next proposition.

Proposition 2. *Let $\alpha, \pi \in uco(\wp(Sub))$ with $\alpha \sqsubseteq \pi$. Then, for all $x \in \wp(Sub)$, $\alpha^\pi(\alpha(x))$ enjoys Property* (1)*, i.e., $\forall \theta \in Sub$, it holds:*

$$\theta \in \alpha^\pi(\alpha(x)) \iff \forall c, d \in \wp(Sub)\ \alpha(x) \otimes c \subseteq \pi(d) \Rightarrow \theta \otimes c \subseteq \pi(d).$$

Proposition 1 states that the new domain α^π is a concretization of the property of interest π and, more importantly, it is complete for the unification with objects of α. This means that α^π is at least as precise as α in computing the operation of unification. These two properties are obviously the fundamental requirements for any domain candidate for improving the precision of α. It is worth noting that many other domains enjoy these properties, for instance σ^π and α itself. Let us denote by $C(\alpha, \pi)$ the collection of all the abstractions of π which are complete for $\otimes$ w.r.t. α, i.e., :

$$C(\alpha, \pi) \stackrel{\text{def}}{=} \{\beta \in uco(\wp(Sub)) \mid \beta \sqsubseteq \pi, \beta \text{ complete for } \otimes \text{ w.r.t. } \alpha\}.$$

According to the above definition, for all $\beta \in C(\alpha, \pi)$ it holds:

$$\forall c, d \in \wp(Sub)\ \ \beta(\beta(\alpha(c)) \otimes \beta(\alpha(d))) = \beta(\alpha(c) \otimes \alpha(d))$$

which, in turn, implies the following property:

$$\forall c, d \in \wp(Sub)\ \ \beta(\beta(c) \otimes \beta(d)) \leq \beta(\alpha(\alpha(c) \otimes \alpha(d))).$$

The latter property precisely says that, given any two objects $c, d \in \wp(Sub)$, the result of any abstract computation on the domain β (i.e., $\beta(\beta(c) \otimes \beta(d))$) is more precise than the corresponding computation on the domain α (i.e.,

$\alpha(\alpha(c) \otimes \alpha(d)))$ when the result is observed in β. This property, combined with the assumption that $\beta \sqsubseteq \pi$, ensures us that the domain β is, at least, as precise as α in computing the operation of unification, when we are interested in observing the property π only. It is worth remarking that we do not require that $\beta(\beta(c) \otimes \beta(d)) \leq \alpha(\alpha(c) \otimes \alpha(d))$, since we are not interested in directly comparing the absolute precision of the two domains α and β.

In general, given two closures $\beta_1, \beta_2 \in C(\alpha, \pi)$, the closure $\beta_1 \sqcup \beta_2$ may not be in $C(\alpha, \pi)$ and, in particular, $\sqcup C(\alpha, \pi)$ may not be in $C(\alpha, \pi)$. For instance, it easy to see that both α and α^π belong to $C(\alpha, \pi)$, but, in general, $\alpha \sqcup \alpha^\pi$ is not in $C(\alpha, \pi)$ (see Section 6.1 for a counterexample). The idea is to restrict our attention to those domains $\beta \in C(\alpha, \pi)$ which satisfy Property (1) on the abstract objects. In other words, we require that for any $a \in \alpha$, it holds: $\theta \in \beta(a)$ if and only if Property (1) holds. This ensures us that β has been designed by adding to each abstract object $a \in \alpha$ the greatest amount of substitutions, still preserving the observable property π. The next proposition proves that α^π is indeed the most abstract element in $C(\alpha, \pi)$ which satisfies Property (1) for all abstract objects.

Proposition 3. *Let $\alpha, \pi \in uco(\wp(Sub))$ with $\alpha \sqsubseteq \pi$. Then α^π is the most abstract domain in $C(\alpha, \pi)$ which enjoys Property (1) for all $a \in \alpha$.*

The above result implies that, for any domain $\beta \in C(\alpha, \pi)$, β is complete for $\otimes$ w.r.t. α^π if and only if it is a concretization of α^π, as the next theorem shows.

Theorem 2. *Let $\alpha, \pi \in uco(\wp(Sub))$ with $\alpha \sqsubseteq \pi$. For any $\beta \in C(\alpha, \pi)$, it holds:*

$$\beta \textit{ is complete for } \otimes \textit{ w.r.t. } \alpha^\pi \iff \beta \sqsubseteq \alpha^\pi.$$

This theorem says that any domain which is at least as precise as α^π must be a concretization of α^π. Thus, the domain α^π cannot be further reduced without loosing precision. It is worth remarking that in the previous theorem, α^π is not compared to its proper abstractions only, but to any domain in $C(\alpha, \pi)$, i.e., to all the domains which contain π and which are complete for $\otimes$ w.r.t. α. Hence, this result is much stronger then a mere result of completeness, where the domain is compared to its abstractions/concretizations only.

5 An application to pair-sharing analysis

In this section, we show an application of our construction to the property of variable independence. In particular, we are interested in observing *pair-independence*, i.e., independence of pairs of variables. We say that two variables are independent if the terms they are bound to, have no variable in common. In other words, two variables are independent if they are bound to terms which do not share any variable. Information about pair-independence is typically used in order to exploit the so-called AND-parallelism [13, 17], i.e., to allow a parallel execution of different atoms in the same goal, and for the safe elimination of the occur-check [21], in order to speed-up the unification algorithm.

Variable independence analysis is strictly related to sharing analysis. In fact, many domains for sharing analyses in the literature (e.g., Sh [12, 13] and ASub [21]) actually are concerned with *possible* sharing of variables, i.e., the property of interest is to determine when variables are allowed to share. The information they encode is not that a set of variables definitely share, but, instead, that a set of variables possibly share. This duality has lead to a non-uniform (and quite confusing) terminology in the literature for the different properties of (definite) variable independence and (possible) variable sharing. With a slight abuse of terminology (and adopting the current name in the literature) we will refer to the property of (definite) independence of pairs of variables as pair-sharing, which has to be intended as *possible* pair-sharing.

Even if many different domains for sharing analysis have been proposed, the abstract domain Sh by Jacobs and Langen [12, 13] is undoubtly the most popular and more frequently used for sharing analysis. The increasing amount of results concerning this domain has contributed to confirm the adequacy and efficiency of Sh. Hence, in the following, we apply our results to the domain Sh, where the property of interest is pair-sharing. We proceed as follows. First, we formally define the abstract domain PSh encoding the property of interest (which turns out to be isomorphic to the domain for pair-sharing analysis described in [2]) and the optimal semantics for PSh, denoted by σ^{PSh}.

In order to compute $\mathsf{Sh}^{\mathsf{PSh}} = \curlywedge(\sigma^{\mathsf{PSh}}(\mathsf{Sh}))$ we proceed modularly[1]. It is well-known that the domain Sh can be decomposed into two parts, namely Sh^{+} and Def ([4]). We compute separately $\sigma^{\mathsf{PSh}}(\mathsf{Sh}^{+})$ and $\sigma^{\mathsf{PSh}}(\mathsf{Def})$, which happen to be both abstract domains. Then we show that $\mathsf{Sh}^{\mathsf{PSh}}$ is the reduced product of $\sigma^{\mathsf{PSh}}(\mathsf{Sh}^{+})$ and $\sigma^{\mathsf{PSh}}(\mathsf{Def})$. Finally, we show that $\mathsf{Sh}^{\mathsf{PSh}}$ is smaller in size than Sh and, at the same time, it is strictly more powerful in detecting pair-sharing.

5.1 Sharing and groundness

In this paper, we consider a finite set of variables of interest $VI \subset \mathcal{V}$, which are the relevant variables. According to this, abstract domains are restricted to have variables in VI and do not explicitly show the set of relevant variables they refer to. Let us recall the definitions of the abstract domains $\mathsf{Sh}, \mathsf{Sh}^{+}$ and Def, in order to fix the notation. All domains refer to the (finite) set of variables VI. For the definition of Sh we follow the notation in [2] and define an element of Sh as a collection of nonempty sets of variables. The domain Sh is defined as follows:

$$\mathsf{Sh} \stackrel{\text{def}}{=} \wp(\wp(VI) \setminus \{\emptyset\}).$$

For the abstraction function $\alpha_{\mathsf{Sh}} : \wp(Sub) \mapsto \mathsf{Sh}$ from the concrete domain of substitutions to Sh and the corresponding closure $\rho_{\mathsf{Sh}} \in uco(\wp(Sub))$ we refer to [12, 13]. We briefly recall the definitions of the operations on Sh. The *closure*

[1] Note that, in general, given $A, B, \pi \in \wp(Sub)$, it does not hold that $(A \sqcap B)^{\pi} = A^{\pi} \sqcap B^{\pi}$.

under union function $\cdot^* : \mathsf{Sh} \mapsto \mathsf{Sh}$ is defined as follows, for any $A \in \mathsf{Sh}$:

$$A^* \stackrel{\text{def}}{=} \{T \subseteq VI \mid \exists n \geq 1, \exists T_1, \ldots, T_n \in A, T = \bigcup_{1 \leq i \leq n} T_i\}.$$

The *extraction of the relevant components* $rel : \wp(VI) \times \mathsf{Sh} \mapsto \mathsf{Sh}$, for $V \subseteq VI$ and $A \in \mathsf{Sh}$ is given by:

$$rel(V, A) \stackrel{\text{def}}{=} \{T \in A \mid T \cap V \neq \emptyset\}.$$

For $A, B \in \mathsf{Sh}$, the *binary union* $bin : \mathsf{Sh} \times \mathsf{Sh} \mapsto \mathsf{Sh}$ is defined as:

$$bin(A, B) \stackrel{\text{def}}{=} \{T_1 \cup T_2 \mid T_1 \in A, T_2 \in B\}.$$

Finally, the *unification* operation $\mathbf{amgu} : \mathsf{Sh} \times Sub \mapsto \mathsf{Sh}$ is inductively given, for $A \in \mathsf{Sh}$ and $\{\, x \mapsto t \,\}, \theta \in Sub$, by:

$$\begin{aligned}
\mathbf{amgu}(A, \{\, x \mapsto t \,\}) \quad &\stackrel{\text{def}}{=} (A \setminus rel(vars(t) \cup \{x\})) \\
&\quad\ \cup\ bin(rel(\{x\}, B)^*, rel(vars(t), B)^*) \\[1ex]
\mathbf{amgu}(A, \{\, x \mapsto t \,\} \circ \theta) \ &\stackrel{\text{def}}{=} \mathbf{amgu}(\mathbf{amgu}(A, \{\, x \mapsto t \,\}), \theta).
\end{aligned}$$

It is well-known that the domain Sh can be decomposed into two parts, namely Sh^+ and Def ([4]). The abstract domain Sh^+ is defined as the subset of Sh whose elements contain all the singletons.

$$\mathsf{Sh}^+ \stackrel{\text{def}}{=} \{A \in \mathsf{Sh} \mid \forall x \in VI\ \{\, x \,\} \in A\}.$$

The corresponding closure operator $\alpha_{\mathsf{Sh}^+} \in uco(\mathsf{Sh})$ is given by $\alpha_{\mathsf{Sh}^+}(A) \stackrel{\text{def}}{=} A \cup \{\{\, x \,\} \mid x \in VI\}$. Moreover, we denote by $\rho_{\mathsf{Sh}^+} \in uco(\wp(Sub))$ the closure on the concrete domain $\wp(Sub)$ corresponding to Sh^+.

The domain PSD has been recently proposed in [2] for capturing pair-sharing dependency. As we will see in the following, PSD is strictly related to our construction. We briefly recall the definition of PSD (see [2, 22] for details). PSD is defined by a closure operator on the domain Sh in the following way. For any $T \subseteq VI$, let $pairs(T) = \{P \in \wp(T) \mid |P| = 2\}$. We say that T is *redundant* for an abstract object $A \in \mathsf{Sh}$ if and only if $|T| > 2$ and $pairs(T) = \cup\{pairs(S) \mid S \in A, S \subset T\}$. In other words, T is redundant for A if all its pairs can be recovered from the sets $S \in A$ which are contained in T. The closure operator $\alpha_{\mathsf{PSD}} \in uco(\mathsf{Sh})$ is defined as follows:

$$\alpha_{\mathsf{PSD}}(A) \stackrel{\text{def}}{=} A \cup \{T \subseteq VI \mid T \text{ is redundant for } A\}.$$

We denote by $\mathsf{G} \stackrel{\text{def}}{=} \wp(VI)$ the basic domain for groundness analysis by Jones and Søndergaard ([14]), where each $V \subseteq VI$ denotes the set of substitutions which ground every variable in V. We denote by $\rho_{\mathsf{G}} : \wp(Sub) \mapsto \mathsf{G}$ the abstraction function from the concrete domain of substitutions to G. The domains Def [1]

and Pos [16,6,1] can be characterized, starting from G, in the following way ([20]):

$$\mathsf{Def} \stackrel{\text{def}}{=} \bigwedge(\{a \multimap b \mid a, b \in \gamma(\mathsf{G})\})$$

$$\mathsf{Pos} \stackrel{\text{def}}{=} \bigwedge(\{a \multimap b \mid a \in \mathsf{Def}, b \in \gamma(\mathsf{G})\}).$$

Let us denote, respectively, by $\rho_{\mathsf{Def}}, \rho_{\mathsf{Pos}} \in uco(\wp(Sub))$ the closure operators corresponding to the domains Def and Pos. It is well-known that Def induces a Galois insertion on Sh [3]. We denote by $\rho_{\mathsf{ShDef}} \in uco(\mathsf{Sh})$ the closure operator on Sh corresponding to Def, thus $\rho_{\mathsf{ShDef}}(\mathsf{Sh})$ is isomorphic to Def (see [3]). Intuitively, an element $\cap X \multimap x \in \mathsf{Def}$ corresponds to the element in Sh $\{T \subseteq VI \mid x \in T \Rightarrow X \cap T \neq \emptyset\}$. In the following, we shall abuse the notation and call Def its isomorphic image $\rho_{\mathsf{ShDef}}(\mathsf{Sh})$. This does not give rise to ambiguity, since one can distinguish the domains according to the context.

The next result summarizes the behavior of PSD on the two components of Sh, i.e., Sh^+ and Def.

Proposition 4.

1. $\alpha_{\mathsf{PSD}}(\mathsf{Def}) = \mathsf{Def}$
2. $\alpha_{\mathsf{PSD}} \circ \alpha_{\mathsf{Sh}^+} = \alpha_{\mathsf{Sh}^+} \circ \alpha_{\mathsf{PSD}}$
3. $\mathsf{PSD} = \alpha_{\mathsf{PSD}}(\mathsf{Sh}^+) \sqcap \mathsf{Def}$.

Note that, by Point 2 above, $\alpha_{\mathsf{PSD}}(\mathsf{Sh}^+)$ turns out to be an abstract domain. Therefore, the reduced product in Point 3 is well-defined.

5.2 Pair-sharing

We say that two variables x and y are *independent* for the substitution θ when $vars(\theta(x)) \cap vars(\theta(y)) = \emptyset$. Let us denote by I_{xy} the set of substitutions for which x and y are independent:

$$I_{xy} \stackrel{\text{def}}{=} \{\theta \in Sub \mid vars(\theta(x)) \cap vars(\theta(y)) = \emptyset\}.$$

Our basic domain PSh for pair-sharing is given by the most abstract domain which contains all the objects I_{xy}, for any $x, y \in VI$, with $x \neq y$. In the following, we shall assume that $|VI| \geq 2$.

$$\mathsf{PSh} \stackrel{\text{def}}{=} \bigwedge(\{I_{xy} \mid x, y \in VI, x \neq y\}).$$

It is immediate to see that it holds $I_{xy} = \gamma_{\mathsf{Sh}}(\{T \subseteq VI \mid \{x, y\} \nsubseteq T\})$. Therefore, PSh induces a Galois insertion $(\alpha_{\mathsf{PSh}}, \mathsf{Sh}, \mathsf{PSh}, \gamma_{\mathsf{PSh}})$ defined as follows, for any $A \in \mathsf{Sh}$ and $B \in \mathsf{PSh}$:

$$\alpha_{\mathsf{PSh}}(A) \stackrel{\text{def}}{=} \cap \{I_{xy} \mid x, y \in VI, x \neq y, \forall T \in A\ \{x, y\} \nsubseteq T\}$$

$$\gamma_{\mathsf{PSh}}(B) \stackrel{\text{def}}{=} \{T \subseteq VI \mid x, y \in VI, x \neq y, T \neq \emptyset, B \subseteq I_{xy} \Rightarrow \{x, y\} \nsubseteq T\}.$$

6 Optimal semantics for PSh

Now that we have formally defined the basic domain encoding the pair-sharing property, we are ready to compute $\mathsf{Sh}^{\mathsf{PSh}} = \curlywedge(\sigma^{\mathsf{PSh}}(\mathsf{Sh}))$. We proceed modularly on the two parts of Sh, namely Sh^{+} and Def. We start by computing $\sigma^{\mathsf{PSh}}(\mathsf{Sh}^{+})$. The next theorem states that, for any $c \in \wp(Sub)$, the object $\sigma^{\mathsf{PSh}}(\rho_{\mathsf{Sh}^{+}}(c))$ coincides with the abstraction α_{PSD} on $\rho_{\mathsf{Sh}^{+}}(c)$.

Theorem 3.

$$\sigma^{\mathsf{PSh}} \circ \rho_{\mathsf{Sh}^{+}} = \rho_{\mathsf{PSD}} \circ \rho_{\mathsf{Sh}^{+}}.$$

In order to better understand the behavior of the optimal semantics on the elements of Def, let us first characterize the bottom element $\perp_{\sigma^{\mathsf{PSh}}}$ of σ^{PSh}, given by $\sigma^{\mathsf{PSh}}(\emptyset)$. Since the domain $\mathsf{Sh}^{\mathsf{PSh}}$ is an abstraction of σ^{PSh}, it is clear that any object $c \in \mathsf{Sh}^{\mathsf{PSh}}$ must approximate the bottom of σ^{PSh}.

Proposition 5. $\perp_{\sigma^{\mathsf{PSh}}} = \bigcap \{\gamma_{\mathsf{G}}(x) \cup \gamma_{\mathsf{G}}(y) \mid x, y \in VI, x \neq y\}$.

The previous proposition shows that the least element of σ^{PSh} is given by the intersection of all the possible disjunctions $\gamma_{\mathsf{G}}(x) \cup \gamma_{\mathsf{G}}(y)$, where $\gamma_{\mathsf{G}}(x)$ is the set of substitutions which ground x. This result is quite surprising, since it implies that the optimal semantics for detecting pair-sharing does not need to take into account the information given by the groundness of a single variable. By exploiting the distributive property of the concrete domain, the object $\perp_{\sigma^{\mathsf{PSh}}}$ can be equivalently characterized as follows:

$$\perp_{\sigma^{\mathsf{PSh}}} = \bigcup_{x \in VI} \bigcap_{v \in VI \setminus \{x\}} \gamma_{\mathsf{G}}(v).$$

Note that, for any $x \in VI$, the object $\bigcap_{v \in VI \setminus \{x\}} \gamma_{\mathsf{G}}(v)$ is the set of substitutions which ground every variable but x. It is clear that this information suffices for proving that x is independent form any other variable. Therefore, the disjunction of all these objects is the least amount of information which we need in order to observe pair-independence. In other words, from the pair-sharing point of view, it is equivalent to know either that a variable x is ground or that every variable but x is ground. In both cases, x does not share with any other variable. Finally, note that $\perp_{\sigma^{\mathsf{PSh}}}$ does not belong to Sh, and therefore nor to Def (but it is easily seen that $\perp_{\sigma^{\mathsf{PSh}}} \in \mathsf{Pos}$). Hence, we expect that the domain $\mathsf{Sh}^{\mathsf{PSh}}$ will be incomparable to Sh, as we will see in the following.

Since any object in $\mathsf{Sh}^{\mathsf{PSh}}$ approximates $\perp_{\sigma^{\mathsf{PSh}}}$, it follows that for any object $a \in \mathsf{Def}$, $\sigma^{\mathsf{PSh}}(a)$ must approximate $a \cup \perp_{\sigma^{\mathsf{PSh}}}$. The next result proves that $\sigma^{\mathsf{PSh}}(\rho_{\mathsf{Def}}(a))$ coincides with the abstraction in Pos of the object $a \cup \perp_{\sigma^{\mathsf{PSh}}}$.

Theorem 4. *Let $d \in \wp(Sub)$.*

$$\sigma^{\mathsf{PSh}}(\rho_{\mathsf{Def}}(d)) = \rho_{\mathsf{Pos}}(\rho_{\mathsf{Def}}(d) \cup \perp_{\sigma^{\mathsf{PSh}}})).$$

The next theorem proves that the result of the application of σ^{PSh} to an element of Sh can be computed modularly on the two components of Sh, i.e., Sh^{+} and Def.

Theorem 5. *Let $A \in \gamma(\mathsf{Sh})$. $\sigma^{\mathsf{PSh}}(A) = \rho_{\mathsf{PSD}}(\rho_{\mathsf{Sh}^{+}}(A)) \cap \rho_{\mathsf{Pos}}(\rho_{\mathsf{Def}}(A) \cup \perp_{\sigma^{\mathsf{PSh}}})$.*

As a consequence, the domain $\mathsf{Sh}^{\mathsf{PSh}}$ can be specified as the reduced product of the two parts.

Corollary 1. $\mathsf{Sh}^{\mathsf{PSh}} = \rho_{\mathsf{PSD}}(\mathsf{Sh}^{+}) \sqcap \{\rho_{\mathsf{Pos}}(A \cup \perp_{\sigma^{\mathsf{PSh}}}) \mid A \in \mathsf{Def}\}$.

6.1 Comparing Sh to $\mathsf{Sh}^{\mathsf{PSh}}$

Since pair-sharing is clearly a decidable domain, from Section 3 we immediately inherits the following results for $\mathsf{Sh}^{\mathsf{PSh}}$.

Corollary 2.

- $\mathsf{Sh}^{\mathsf{PSh}} \sqsubseteq \mathsf{PSh}$
- $\mathsf{Sh}^{\mathsf{PSh}}$ *is complete for $\otimes$ w.r.t.* Sh*, i.e., for all $A, B \in \gamma_{\mathsf{Sh}}(\mathsf{Sh})$, it holds:*

$$\rho_{\mathsf{Sh}^{\mathsf{PSh}}}(A \otimes B) = \rho_{\mathsf{Sh}^{\mathsf{PSh}}}(\rho_{\mathsf{Sh}^{\mathsf{PSh}}}(A) \otimes \rho_{\mathsf{Sh}^{\mathsf{PSh}}}(B))$$

- *For any $\beta \in C(\mathsf{Sh}, \mathsf{PSh})$, it holds:*

$$\beta \textit{ is complete for } \otimes \textit{ w.r.t. } \mathsf{Sh}^{\mathsf{PSh}} \iff \beta \sqsubseteq \mathsf{Sh}^{\mathsf{PSh}}.$$

This corollary summarizes the properties of the domain $\mathsf{Sh}^{\mathsf{PSh}}$ which are straightforwardly inherited by construction. In particular, $\mathsf{Sh}^{\mathsf{PSh}}$ turns out to be a concretization of PSh and, at least, as precise as Sh in computing the abstract unification. Moreover, the last property ensures us that, among all domains which are as precise as Sh, it does not exist any domain β which is precise as $\mathsf{Sh}^{\mathsf{PSh}}$ and that does not contain $\mathsf{Sh}^{\mathsf{PSh}}$. The importance of this result relies on the fact that $\mathsf{Sh}^{\mathsf{PSh}}$ is compared to all possible domains candidate to improving Sh, including those which are incomparable to Sh. A similar result as been shown in [2]. The authors proved that $\mathsf{PSD} \in C(\mathsf{Sh}, \mathsf{PSh})$ and that it does not exist any proper abstraction of PSD which is as precise as PSD. Note that in [2] the domain PSD is compared to its proper abstractions only, which is a rather restrictive hypothesis and, in general, it does not suffice in order to ensure the optimality of a given domain.

The next example shows that $\mathsf{Sh}^{\mathsf{PSh}}$ is strictly more precise than Sh under the hypothesis that $|\mathit{VI}| \geq 2$.

Example 1. Let $\mathit{VI} = \{x_1, ..., x_n\}$, with $n \geq 2$. Let $\theta, \delta \in \mathit{Sub}$ defined as follows:

$$\theta = \{x_1 \leftarrow a\} \qquad\qquad \delta = \{x_2 \leftarrow a, \ldots, x_n \leftarrow a\}$$

where a is a constant of the language. Let us unify the concrete object $c = \{\theta, \delta\}$ with the substitution $\{x_1 \leftarrow x_2\}$. On the concrete domain of substitutions we obtain:

$$\begin{aligned} c \otimes \{x_1 \leftarrow x_2\} &= \{\theta \otimes \{x_1 \leftarrow x_2\}, \delta \otimes \{x_1 \leftarrow x_2\}\} \\ &= \{\{x_1 \leftarrow a, x_2 \leftarrow a\}, \{x_1 \leftarrow a, \ldots, x_n \leftarrow a\}\} \subseteq I_{x_1,x_2}. \end{aligned}$$

Instead, when computing on Sh, we obtain:

$$\begin{aligned} \mathbf{amgu}(\alpha_{\mathsf{Sh}}(c), \{x_1 \leftarrow x_2\}) &= \mathbf{amgu}(\{\{x_i\} \mid x_i \in \mathit{VI}\}, \{x_1 \leftarrow x_2\}) \\ &= \{\{x_1, x_2\}, \{x_3\}, \ldots, \{x_n\}\}. \end{aligned}$$

Therefore, $\gamma_{\mathsf{Sh}}(\mathbf{amgu}(\alpha_{\mathsf{Sh}}(c), \{x_1 \leftarrow x_2\})) \not\subseteq I_{x_1,x_2}$.

Note that, by definition of $\mathsf{Sh}^{\mathsf{PSh}}$, it holds $\rho_{\mathsf{Sh}^{\mathsf{PSh}}}(\theta) = \rho_{\mathsf{Pos}}(\rho_{\mathsf{Def}}(\theta) \cup \bot_{\sigma^{\mathsf{PSh}}}) = \rho_{\mathsf{Pos}}(x_1 \cup \bot_{\sigma^{\mathsf{PSh}}}) = x_1 \cup (x_2 \cap \ldots \cap x_n)$ which stands for all the substitutions for which either x_1 is ground or all the variables but x_1 are ground. Hence $\delta \in \rho_{\mathsf{Sh}^{\mathsf{PSh}}}(\{\theta\})$ and therefore $\rho_{\mathsf{Sh}^{\mathsf{PSh}}}(\{\theta, \delta\}) = \rho_{\mathsf{Sh}^{\mathsf{PSh}}}(\{\theta\}) = x_1 \cup (x_2 \cap \ldots \cap x_n)$. As a consequence, we have the following computation:

$$\begin{aligned} &\rho_{\mathsf{Sh}^{\mathsf{PSh}}}(\rho_{\mathsf{Sh}^{\mathsf{PSh}}}(c) \otimes \{x_1 \leftarrow x_2\}) \\ &= \rho_{\mathsf{Sh}^{\mathsf{PSh}}}(x_1 \cup (x_2 \cap \ldots \cap x_n) \otimes \{x_1 \leftarrow x_2\}) \\ &= \rho_{\mathsf{Sh}^{\mathsf{PSh}}}((x_1 \otimes \{x_1 \leftarrow x_2\}) \cup (x_2 \cap \ldots \cap x_n) \otimes \{x_1 \leftarrow x_2\})) \\ &= \rho_{\mathsf{Sh}^{\mathsf{PSh}}}((x_1 \cap x_2) \cup (x_1 \cap \ldots \cap x_n)) \\ &= \rho_{\mathsf{Sh}^{\mathsf{PSh}}}(x_1 \cap x_2) \subseteq I_{x_1,x_2}. \end{aligned}$$

Therefore, the domain $\mathsf{Sh}^{\mathsf{PSh}}$ is precise enough to detect that x_1 and x_2 are independent, while Sh is not able to detect it. □

It is worth noting that, being PSD an abstraction of Sh, it is as precise as Sh , and therefore, according to the above example, PSD is strictly less precise than $\mathsf{Sh}^{\mathsf{PSh}}$. Surprisingly, the increasing precision of $\mathsf{Sh}^{\mathsf{PSh}}$ w.r.t. Sh is not due to an enlargement of the domain. On the contrary, if we compare the size of $\mathsf{Sh}^{\mathsf{PSh}}$ to the sizes of Sh and PSD, we find out that the cardinality of $\mathsf{Sh}^{\mathsf{PSh}}$ is strictly smaller than the cardinality of PSD, and therefore of Sh.

Proposition 6. $|\mathsf{Sh}^{\mathsf{PSh}}| < |\mathsf{PSD}|$

As an example of two abstract objects in PSD which are identified by $\mathsf{Sh}^{\mathsf{PSh}}$, consider a variable $x \in \mathit{VI}$ and the objects $A = \emptyset$ and $B = \{\{x\}\}$ of PSD. Note that A is the bottom of Sh and B denotes the set of substitutions which ground all variables in VI but x. From the characterization of $\bot_{\sigma^{\mathsf{PSh}}}$ given in Proposition 5, it is immediate to see that $\gamma_{\mathsf{Sh}}(A) \subseteq \gamma_{\mathsf{Sh}}(B) \subseteq \bot_{\sigma^{\mathsf{PSh}}}$, and thus $\sigma^{\mathsf{PSh}}(A) = \sigma^{\mathsf{PSh}}(B) = \bot_{\sigma^{\mathsf{PSh}}}$.

For $\mathit{VI} = \{x, y\}$, the domains Sh, PSD and $\mathsf{Sh}^{\mathsf{PSh}}$ are depicted in Figure 1, where $x \cup y$ denotes the object $\gamma_{\mathsf{G}}(x) \cup \gamma_{\mathsf{G}}(y)$ (note that, in this particular case, the domains Sh and PSD do coincide). As a further example of objects which are identified by $\mathsf{Sh}^{\mathsf{PSh}}$, from Figure 1 it is easy to see that the objects $\emptyset, \{\{y\}\}, \{\{x, y\}\}, \{\{x\}\} \in \mathsf{Sh}$ all collapse in the same object $x \cup y \in \mathsf{Sh}^{\mathsf{PSh}}$ and $\{\{y\}, \{x, y\}\}, \{\{x\}, \{x, y\}\}$ collapse in $\top$.

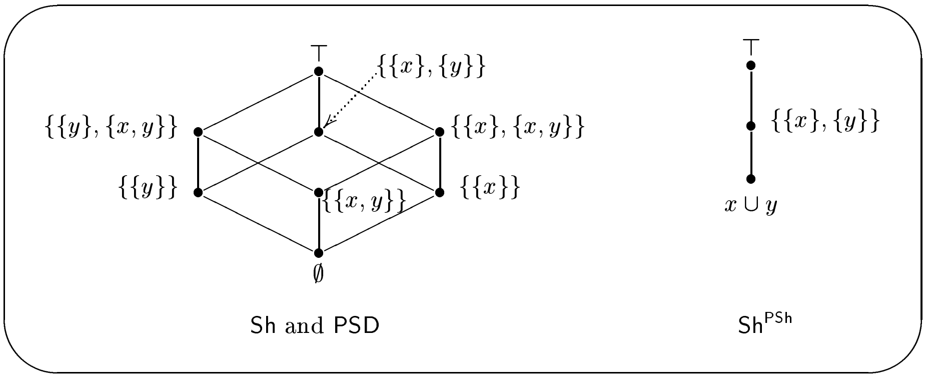

Fig. 1. Sh, PSD and $\mathsf{Sh}^{\mathsf{PSh}}$

7 Groundness analysis

In this section we apply our construction to the property of groundness. We consider as basic domain of groundness the domain G by Jones and Søndergaard ([14]). The next theorem shows that both domains Def and Pos are not further transformed by our construction.

Theorem 6.

- $\mathsf{Def}^{\mathsf{G}} = \mathsf{Def}$
- $\mathsf{Pos}^{\mathsf{G}} = \mathsf{Pos}$

This result is a consequence of the fact that Def and Pos are both abstractions of the optimal semantics for G (see [20]).

8 Conclusion

We have proposed a new technique for transforming abstract domains for logic program analysis. The basic idea is to exploit the optimal semantics of a given property in order to set up an enriched abstract domain, which improves the precision of the analysis. The main point of this construction is that the new domain is, at least, as precise as the original one. It is important to remark that the transformation we propose is not a refinement operator. In fact, given a domain α, the resulting domain α^{π} can be incomparable to the original one (e.g., this is the case for the domains Sh and $\mathsf{Sh}^{\mathsf{PSh}}$).

As an application, we have fixed the property of pair-sharing and we have instantiated our construction to the well-known domain Sh. The new domain $\mathsf{Sh}^{\mathsf{PSh}}$ obtained by transforming Sh, turns out to be more precise than Sh and, at the same time, smaller in size. A refining of the domain Sh for observing pair-sharing was first presented in [2]. The authors propose the domain PSD, proving that it is as precise as Sh and that no proper abstraction of PSD is as precise as PSD. The weak point of the construction in [2] is that the domain PSD is compared to its proper abstractions only. On the contrary, by using our optimal semantic-based transformation, we can set up abstract domains which are incomparable to the original ones. Up to our knowledge, this is the first method which allows us to construct domains which are strictly more precise and, at the same time, smaller in size than the original ones.

References

1. T. Armstrong, K. Marriott, P. Schachte, and H. Søndergaard. Two classes of Boolean functions for dependency analysis. *Science of Computer Programming*, 31(1):3–45, 1998.
2. R. Bagnara, P.M. Hill, and E. Zaffanella. Set-sharing is redundant for pair-sharing. *Theoretical Computer Science*, 2000. To appear.
3. A. Cortesi, G. Filé, , and W. Winsborough. The quotient of an abstract interpretation. *Theor. Comput. Sci.*, 202(1-2):163–192, 1998.
4. A. Cortesi, G. Filé, R. Giacobazzi, C. Palamidessi, and F. Ranzato. Complementation in abstract interpretation. *ACM Trans. Program. Lang. Syst.*, 19(1):7–47, 1997.
5. A. Cortesi, G. Filè, and W. Winsborough. *Prop* revisited: Propositional formula as abstract domain for groundness analysis. In *Proc. Sixth IEEE Symp. on Logic In Computer Science*, pages 322–327, Los Alamitos, Calif., 1991. IEEE Computer Society Press.
6. A. Cortesi, G. Filé, and W. Winsborough. Optimal groundness analysis using propositional logic. *J. Logic Program.*, 27(2):137–167, 1996.
7. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the 4th ACM Symposium on Principles of Programming Languages* (*POPL '77*), pages 238–252, New York, 1977. ACM Press.
8. P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Conference Record of the 6th ACM Symposium on Principles of Programming Languages* (*POPL '79*), pages 269–282, New York, 1979. ACM Press.
9. M. Falaschi, G. Levi, M. Martelli, and C. Palamidessi. Declarative modeling of the operational behavior of logic languages. *Theoretical Computer Science*, 69(3):289–318, 1989.
10. R. Giacobazzi. "Optimal" collecting semantics for analysis in a hierarchy of logic program semantics. In C. Puech and R. Reischuk, editors, *Proc. of the 13th Int'l Symposium on Theoretical Aspects of Computer Science (STACS '96)*, volume 1046 of *Lecture Notes in Computer Science*, pages 503–514, Berlin, 1996. Springer-Verlag.
11. R. Giacobazzi, F. Ranzato, and F. Scozzari. Building complete abstract interpretations in a linear logic-based setting. In G. Levi, editor, *Static Analysis, Proceedings of the Fifth International Static Analysis Symposium SAS 98*, volume 1503 of *Lecture Notes in Computer Science*, pages 215–229, Berlin, 1998. Springer-Verlag.
12. D. Jacobs and A. Langen. Accurate and efficient approximation of variable aliasing in logic programs. In E.L. Lusk and R.A. Overbeek, editors, *Proc. of the 1989 North American Conference on Logic Programming (NACLP '89)*, Workshops in Computing, pages 154–165, Cambridge, Mass., 1989. The MIT Press.

13. D. Jacobs and A. Langen. Static analysis of logic programs for independent AND-parallelism. *J. Logic Program.*, 13(2-3):154–165, 1992.
14. N. D. Jones and H. Søndergaard. A semantics-based framework for the abstract interpretation of Prolog. In S. Abramsky and C. Hankin, editors, *Abstract Interpretation of Declarative Languages*, pages 123–142. Ellis Horwood Ltd, Chichester, UK, 1987.
15. 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, Los Altos, 1988.
16. K. Marriott and H. Søndergaard. Abstract interpretation of logic programs: the denotational approach. In A. Bossi, editor, *Proc. GULP '90*, pages 399–425, Padova, 1990.
17. K. Muthukumar and M. Hermenegildo. Compile-time derivation of variable dependency using abstract interpretation. *Journal of Logic Programming*, 13(2-3):315–347, 1992.
18. C. Palamidessi. Algebraic properties of idempotent substitutions. In M. S. Paterson, editor, *Proc. of the 17th International Colloquium on Automata, Languages and Programming*, volume 443 of *Lecture Notes in Computer Science*, pages 386–399, Berlin, 1990. Springer-Verlag.
19. F. Scozzari. *Domain theory in abstract interpretation: equations, completeness and logic.* PhD thesis, Dipartimento di Matematica, Univ. di Siena, 1999. Available at http://www.di.unipi.it/∼scozzari/.
20. F. Scozzari. Logical optimality of groundness analysis. *Theoretical Computer Science*, 2000. To appear.
21. H. Søndergaard. An application of abstract interpretation of logic programs: occur check reduction. In *Proc. ESOP '86*, volume 213 of *Lecture Notes in Computer Science*, pages 327–338, Berlin, 1986. Springer-Verlag.
22. E. Zaffanella, P.M. Hill, and R. Bagnara. Decomposing non-redundant sharing by complementation. In A. Cortesi and G. Filé, editors, *Proceedings of the 6th International Symposium on Static Analysis (SAS'99)*, volume 1694 of *Lecture Notes in Computer Science*, pages 69–84, Berlin, 1999. Springer-Verlag.