The Quotient of an Abstract Interpretation

AGOSTINO CORTESI[1]
University of Venezia, Italy

GILBERTO FILE'
University of Padova, Italy

WILLIAM WINSBOROUGH
Transarc Corporation, USA

Abstract

Within the abstract interpretation framework, abstract domains are used to represent interesting properties of the concrete domain. For instance, properties that enhance the optimization of the analyzed programs. An abstract domain D expresses, in general, several properties of the concrete domain.

We describe a method for identifying, for any abstract domain D and for each property P expressed by D, the subset of D that is useful for computing P-information. We call it the *quotient of D with respect to P*. We also give a necessary and sufficient condition for having that the quotient is an abstraction of D. This property seems essential for applications such as that described below.

As an illustration of the usefulness of the notion of quotient, we show that rather sophisticated comparisons between domains, can be carried out using it. Assume to have two abstract domains that both compute some property P, but that also express distinct properties and thus are incomparable as a whole. Such domains can be compared with respect to the precision with which they compute P-information, by comparing their quotients with respect to P.

Using this method, two well-known abstract domains for Prolog programs, `Prop` and `Sharing`, are compared with respect to the precision with which they compute groundness information.

Categories and Subject Descriptors: D.3.4[**Programming Languages**]: Processors — *compilers, optimization*; F.3.1[**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs; F.3.2[**Logics and Meanings of Programs**]: Semantics of Programming Languages; F.4.1[**Mathematical Logic and Formal Languages**]: Mathematical Logic — *logic programming.*

[1]**Correspondence Address:** Agostino Cortesi, University of Venezia, Corso di Laurea in Scienze dell'Informazione, via Torino 155, 30170 MESTRE-VENEZIA (ITALY). Email: cortesi@dsi.unive.it. Phone: (39)(41) 290 8428. Fax: (39)(41) 290 8419

1 Introduction

Abstract Interpretation is a general method for defining data-flow analyses. "Ad hoc" analyses, that can be viewed as instances of the abstract interpretation approach, were already used in the 60's, [22], for the optimization of imperative languages. However, the formal foundations of the approach have been given only in 1977 by the Cousots in [5]. In this framework, a data-flow analysis is specified by describing a domain of data-descriptions and operations on these data-descriptions that mimic the concrete operations of the language. A tuple $\mathbf{D}=\langle D, \mu_1, .., \mu_k \rangle$, where D is the set of data-descriptions (called *abstract domain*) and the μ_i are the operations on D (called *abstract operations*), is called an *abstract interpretation*. The correctness of the data-flow analysis induced by $\mathbf{D}$ is guaranteed when some safety condition holds between $\mathbf{D}$ and the *concrete interpretation* $\mathbf{C}=\langle C, o_1, .., o_k \rangle$. In this paper we will take as safety condition that:

(*i*) D and C are complete lattices,

(*ii*) there is a Galois insertion between D and C (with γ and α concretization and abstraction functions, respectively), and

(*iii*) for any $c \in C$ and for any $d \in D$, $c \sqsubseteq_C \gamma(d) \;\Rightarrow\; o_i(c) \sqsubseteq_C \gamma(\mu_i(d))$.

When conditions (*i*) and (*ii*) are satisfied, we will say that D *abstracts* C, when also condition (*iii*) holds then we say that $\mathbf{D}$ *abstracts* $\mathbf{C}$. Weaker safety conditions are also sufficient for guaranteeing correctness [18, 12]. However we need $(i) - (iii)$ for the purposes of the present paper.

In an abstract interpretation that expresses and thus computes several properties, we want to identify which part is useful for computing each property. Let us be more precise. Consider an abstract interpretation $\mathbf{D}$ that expresses a property P. We want to identify the subset $\mathcal{Q}_P(D)$ of D that expresses exactly the part of D that is useful for computing P. We call it the *quotient* of D with respect to P.

The notion of quotient is a tool for obtaining more insight in the functionalities of complex abstract interpretations. Such insight can be useful in several ways. In the present paper we show that quotients are useful for comparing abstract interpretations. Assume we want to compare the relative precision of two abstract interpretations $\mathbf{D}$ and $\mathbf{L}$ in the computation of a given property P. This comparison may be impossible relying on the classical notion of Galois insertion (or connection). In fact, $\mathbf{D}$ and $\mathbf{L}$ may express information that is irrelevant for computing P and makes them incomparable using the classical formal tools. We show that the comparison can be done by considering the quotients $\mathcal{Q}_P(D)$ and $\mathcal{Q}_P(L)$. Being able to perform such sophisticated comparisons is obviously useful for choosing the best available abstract interpretation for a particular purpose. Such comparisons are also useful when combining different abstract interpretations in order to obtain a more powerful one, as is suggested in [12] and [4]. By appropriate comparisons one may check whether such combinations of $\mathbf{D}$ and $\mathbf{L}$ are worthwhile at all and also one can "tune" the combination by allowing it only for those properties P for which it may turn out to be profitable: those for which $\mathcal{Q}_P(D)$ and $\mathcal{Q}_P(L)$ are incomparable.

The main achievements of the paper are described below.

(a) For any abstract interpretation $\mathbf{D}$ and any property P expressed by D, we define the quotient $\mathcal{Q}_P(D)$ of D wrt P and give a condition that guarantees that $\mathcal{Q}_P(D)$ is a complete lattice that abstracts D.

(b) We show that if $\mathbf{D}$ and $\mathbf{L}$ are optimal for the computation of a property P and if $\mathcal{Q}_P(D)$ abstracts $\mathcal{Q}_P(L)$, then $\mathbf{L}$ is at least as precise as $\mathbf{D}$ for computing P. Moreover, if $\mathcal{Q}_P(D)$ abstracts $\mathcal{Q}_P(L)$ strictly, (i.e., $\mathcal{Q}_P(L)$ does not abstract $\mathcal{Q}_P(D)$), then $\mathbf{L}$ is strictly more precise than $\mathbf{D}$ for computing P.

(c) As an application of this theory, we compare two abstract interpretations that are well-known for the analysis of Prolog programs: **Prop**, [19, 8], and **Sharing**, [16]. We stress the fact that we consider here complete descriptions of **Prop** and **Sharing**, i.e., with all the abstract operations needed for their use in static analysis. Although both these interpretations compute the property of variable groundness, they are not directly comparable (i.e. neither one abstracts the other) because **Prop** computes also possible equivalence through disjunctions and **Sharing** computes also variable sharing. We compare the (relative) precision of these two analyses in the computation of variable groundness (GR) and we show that $\mathcal{Q}_{\mathtt{GR}}(\mathtt{Prop})$ is Prop itself, whereas $\mathcal{Q}_{\mathtt{GR}}(\mathtt{Sharing})$ is the set of formulas which are conjunctions of formulas of the form: $\wedge W \rightarrow x$, where W is a set of variables and x is a variable. Such formulas are called *definite* and form a proper subset, called Def [14, 2], of the formulas in Prop. Hence, $\mathcal{Q}_{\mathtt{GR}}(\mathtt{Sharing})$ strictly abstracts Prop and thus, from (b) above, it follows that **Prop** is strictly more precise than **Sharing** for the computation of variable groundness.

The notions and results of points (a) and (b) above are new. The construction of the quotients of Prop and Sharing with respect to GR of point(c) is also new. That $\mathcal{Q}_{\mathtt{GR}}(\mathtt{Sharing})$ coincides with Def is an interesting result on its own right. In fact, both Sharing and Def are well-known domains that have been defined independently and that are both useful for the analysis of logic programs.

The present paper is an extended and improved version of [9]. The main improvement is the introduction of the notion of quotient. In fact in [9], we observed that, in order to compare **D** and **L**, it was necessary to define a reference domain R and then compare the workings of **D** and **L** projected on R. Clearly, R plays the same role as $\mathcal{Q}_P(D)$ and $\mathcal{Q}_P(L)$ in the present approach, but in [9] we did not have any systematic way to obtain it. Its definition was only based on the insight one had over **D** and **L**.

Another improvement of the present paper concerns the criteria adopted for comparing the precision of interpretations. The criteria used now is strictly stronger (and more natural) than that of [9]. Thus the relation between Prop and Sharing shown in the present paper is actually stronger than that contained in [9].

The idea of identifying relevant parts of an abstract domain, that has given rise to the notion of quotient, has also inspired the recent work [7] in which the notion of *complement* between domains is defined. The reader may be interested to know that in that paper, among several examples of application of the complement, the authors consider Sharing and Def ($= \mathcal{Q}_{\mathtt{GR}}(\mathtt{Sharing})$, cf. point(c) above) and characterize the complement of Def wrt Sharing, i.e., the domain that represents what is left of Sharing when Def is taken away from it.

Finally, it is important to remark that, even though we have applied it to the comparison of two abstract interpretations for the data-flow analysis of Prolog programs, the notion of quotient, presented in this paper, is in no way bound to logic programming applications only. On the contrary, it can be useful for studying abstract interpretations for all programming paradigms.

The paper is organized as follows. Section 2 contains the preliminary definitions. The definition of quotient is given in Section 3 together with the general result mentioned in (b) above. The application of the theory to the comparison of **Prop** and **Sharing**, mentioned in (c) above, is described in Section 4. The Appendix contains some definitions and technical lemmas that are needed for the application part.

2 Preliminaries

This section consists of three parts. The first introduces some classical notions of abstract interpretation theory and some known results (see [5, 6, 1]). The second part contains the definitions of two types of optimality. The final part introduces the criterion for comparing abstract interpretations that will be used in the rest of the paper.

2.1 Galois Insertions and their Composition

In what follows a function's domain and range are indicated by subscripts: ε_{XY} is a function from X to Y. The ordering and the least upper bound operator defined in X are denoted by $\sqsubseteq_X$ and $\sqcup_X$, respectively.

Definition 2.1 (Galois connection and insertion) *Let C and D be posets and consider two functions of the following types: $\gamma_{DC} : D \longrightarrow C$ and $\alpha_{CD} : C \longrightarrow D$ The 4-tuple $G_{CD} = (\gamma_{DC}, C, D, \alpha_{DC})$ is a* Galois connection *if*

$$\forall c \in C \text{ and } \forall d \in D : \alpha_{CD}(c) \sqsubseteq_D d \Leftrightarrow c \sqsubseteq_C \gamma_{DC}(d)$$

G_{CD} is a Galois insertion when γ_{DC} is injective or, equivalently, when α_{CD} is onto. When G_{CD} is a Galois insertion then we say that D abstracts C.

In a Galois connection or insertion G_{CD}, γ_{DC} and α_{CD} are called the concretization and the abstraction function, respectively. The following are well-known properties of these functions, see [6].

Proposition 2.2 *Let G_{CD} be a Galois connection/insertion,*

1. *$\gamma_{DC} \circ \alpha_{CD}$ is extensive, i.e., $\forall c \in C,\ ; \gamma_{DC} \circ \alpha_{CD}(c) \sqsupseteq_C c$;*
2. *$\alpha_{CD} \circ \gamma_{DC}$ is reductive, i.e., $\forall d \in D, \alpha_{CD} \circ \gamma_{DC}(d) \sqsubseteq_D d$;*
3. *G_{CD} is an insertion if and only if $\alpha_{CD} \circ \gamma_{DC}$ is the identity.*
4. *if α_{CD} and γ_{DC} form a Galois connection, then one of the two functions determines the other one. More precisely, for $d \in D$, $\gamma_{DC}(d) = \sqcup_C \{c \in C \mid \alpha_{CD}(c) \sqsubseteq_D d\}$, and similarly, for $c \in C, \alpha_{CD}(c) = \sqcap_D \{d \in D \mid c \sqsubseteq_C \gamma_{DC}(d)\}$. Each function is called the* adjoint *of the other one.*
5. *$\alpha_{CD} \circ \gamma_{DC} \circ \alpha_{CD} = \alpha_{CD}$ and similarly, $\gamma_{DC} \circ \alpha_{CD} \gamma_{DC} = \gamma_{DC}$.*

A function $\alpha : C \longrightarrow D$, where C and D are po-sets, is *additive* when $\forall X \subseteq C$ such that $\sqcup_C X$ exists, $\alpha(\sqcup_C X) = \sqcup_D \{\alpha(x) \mid x \in X\}$.

Proposition 2.3 *Let $\alpha_{CD} : C \longrightarrow D$, where C and D are complete lattices. The function α_{CD} is additive iff α_{CD}, together with its adjoint concretization, forms a Galois connection between C and D.*

proof: The ($\Rightarrow$) direction is shown in [6](Proposition 7). For the other direction, assume that α_{CD} and its adjoint γ_{DC} form a Galois connection. We must show that

$$\forall X \subseteq C,\ \alpha_{CD}(\sqcup_C X) = \sqcup_D \{\alpha_{CD}(x) \mid x \in X\}$$

Clearly, $\alpha_{CD}(\sqcup_C X)$ is an upper bound of $\{\alpha_{CD}(x) \mid x \in X\}$ because $\forall x \in X,\ \sqcup_C X \sqsupseteq_C x$. We show now that it is the least upper bound. Let d be an upper bound of $\{\alpha_{CD}(x) \mid x \in X\}$, i.e., $\forall x \in X,\ \alpha_{CD}(x) \sqsubseteq_D d$. Thus, by definition of Galois connection, $\forall x \in X,\ x \sqsubseteq_C \gamma_{DC}(d)$. This implies that $\gamma_{DC}(d) \sqsupseteq_C \sqcup_C X$. It suffices now to apply α_{CD} to both members of the inequality, to obtain: $\alpha_{CD} \circ \gamma_{DC}(d) \sqsupseteq_D \alpha_{CD}(\sqcup_C X)$, and thus, by Proposition 2.2 (2), we have that, $d \sqsupseteq_D \alpha_{CD}(\sqcup_C X)$. $\square$

It is well-known, see [6], that in place of considering two domains C and D, where D abstracts C, one can view D as a particular subset of C: the subset of C containing the fixpoints of an upper closure operator on C.

Definition 2.4 (Upper Closure Operator) *Given a poset C, an* upper closure operator *(uco) on C, is a function $\rho : C \to C$, that is monotonic, idempotent and extensive (i.e. $\forall c \in C, \rho(c) \sqsupseteq_C c$). The set of fixpoints of ρ is $\{c \in C \mid \rho(c) = c\}$. This set is indicated by $\rho(C)$.*

The following result shows that Galois Insertions and uco's are two equivalent ways of representing abstractions. The proof can be found in [6]. In what follows, if G_{CD} is a Galois connection/insertion, then $\gamma_{DC}(D) = \{\gamma_{DC}(d) \mid d \in D\}$.

Proposition 2.5 *Let $G_{CD} = (\gamma_{DC}, C, D, \alpha_{DC})$ be a Galois Insertion. Then $\gamma_{DC} \circ \alpha_{CD}$ is an uco on C whose set of fixpoints is $\gamma_{DC}(D)$. Vice Versa, if ρ is an uco on C, then $\rho(C)$ can be viewed as a new domain that abstracts C via a Galois Insertion that has ρ as abstraction and the identity as concretization.*

The following result, shown in [6], will be useful in the sequel.

Proposition 2.6 *If C is a complete lattice and D is a poset and there exists a Galois Insertion between them, then also D is a complete lattice.*

The idea of the previous result is that *joins* and *meets* on D "can be computed on C and then abstracted on D". More precisely, if $G_{CD} = (\gamma_{DC}, C, D, \alpha_{DC})$ is a Galois Insertion, then $\forall X \subseteq D, \sqcup_D X = \alpha_{DC}(\sqcup_C \{\gamma_{DC}(d) \mid d \in X\})$. A similar relation holds for the *meet*.

In this paper we will always consider domains that are complete lattices and will assume the existence of Galois insertions between them.

Definition 2.7 (Interpretation) *An interpretation is a tuple $\mathbf{D} = \langle D, \mu_D \rangle$ where D is a complete lattice and μ_D is a continuous function of type $D \longrightarrow D$.*

In general, an interpretation contains more than one operation, and the operations may have arity greater than one and may take some other arguments besides elements of D. For the sake of clarity we consider the simplest setting, as the generalization of definitions and results is immediate.

Definition 2.8 (Abstraction) *An interpretation $\mathbf{D} = \langle D, \mu_D \rangle$ abstracts an interpretation $\mathbf{C} = \langle C, \mu_C \rangle$ if*

1. *there is a Galois insertion $(\gamma_{DC},\ C, D, \alpha_{CD})$ and*
2. $\forall c \in C : \forall d \in D : c \sqsubseteq_C \gamma_{DC}(d) \ \Rightarrow\ \mu_C(c) \sqsubseteq_C \gamma_{DC}(\mu_D(d))$.

We say that $\mathbf{D}$ properly abstracts $\mathbf{C}$ if $\mathbf{D}$ abstracts $\mathbf{C}$ but $\mathbf{C}$ does not abstract $\mathbf{D}$

An abstract interpretation is intended to report information about a program's execution behavior. When $\mathbf{L}$ abstracts $\mathbf{D}$ we know that the analysis induced by $\mathbf{D}$ is at least as informative as the analysis induced by $\mathbf{L}$. In the rest of the paper we denote domains by capital letters C, D, L, R possibly subscripted, and we denote interpretations by boldface capital letters **C**,**D**,**L**,**R**.

It is well known that if R abstracts D and D abstracts C, then R abstracts C. The proofs of the propositions listed below are straightforward.

Proposition 2.9 (Same-Order Composition) *Let G_{DR} and G_{CD} be Galois insertions. Their composition, denoted $G_{CD} \circ G_{DR}$ is the Galois insertion $G_{CR} = (\gamma_{RC}, C, R, \alpha_{CR})$, where $\gamma_{RC} = \gamma_{DC} \circ \gamma_{RD}$ and $\alpha_{CR} = \alpha_{DR} \circ \alpha_{CD}$.*

The following two Propositions are shown in [9].

Proposition 2.10 (Opposite-Order Composition) *Assume that G_{CL} and G_{CD} are Galois insertions. Let $\varepsilon_{DL} = \alpha_{CL} \circ \gamma_{DC}$ and $\varepsilon_{LD} = \alpha_{CD} \circ \gamma_{LC}$. The following holds:*

i) *ε_{DL} and ε_{LD} are monotone;*

ii) $\forall d \in D, \forall \ell \in L \quad \varepsilon_{LD}(\varepsilon_{DL}(d)) \sqsupseteq_D d$ *and* $\varepsilon_{DL}(\varepsilon_{LD}(\ell)) \sqsupseteq_L \ell$.

Proposition 2.11 *Assume that G_{CL} and G_{CD} are Galois insertions. The following conditions are equivalent:*

i) *$G_{DL} = (\alpha_{CD} \circ \gamma_{LC}, D, L, \alpha_{CL} \circ \gamma_{DC})$ is a Galois insertion;*

ii) $\gamma_{LC}(L) \subseteq \gamma_{DC}(D)$;

iii) $\forall c_1, c_2 \in C.\ \alpha_{CD}(c_1) = \alpha_{CD}(c_2) \Rightarrow \alpha_{CL}(c_1) = \alpha_{CL}(c_2)$.

Definition 2.12 *If G_{CL} G_{CD} and $G_{DL} = (\alpha_{CD} \circ \gamma_{LC}, D, L, \alpha_{CL} \circ \gamma_{DC})$ are Galois insertions, we say that G_{DL} is coherent with respect to C.*

Coherency and same-order composition are strongly related.

Proposition 2.13 *Let G_{CD} G_{CL} and G_{DL} be Galois insertions. Then G_{DL} is coherent with respect to C if and only if $G_{CL} = G_{CD} \circ G_{DL}$.*

proof:

($\Rightarrow$) We want to show that $\gamma_{LC} = \gamma_{DC} \circ \gamma_{LD}$, which, since G_{DL} is coherent wrt C and thus $\gamma_{LD} = \alpha_{CD} \circ \gamma_{LC}$, can be rewritten in:

$$\gamma_{LC} = \gamma_{DC} \circ \alpha_{CD} \circ \gamma_{LC}$$

This equality is verified because, by Proposition 2.11 (ii), $\forall l \in L,\ \gamma_{LC}(l) \in \gamma_{DC}(D)$ and thus, by Proposition 2.2 (5), $\forall\ l \in L,\ \gamma_{DC} \circ \alpha_{CD}(\gamma_{LC}(l)) = \gamma_{LC}(l)$.

That $\alpha_{CL} = \alpha_{DL} \circ \alpha_{CD}$ is shown as follows. By coherency, $\alpha_{DL} \circ \alpha_{CD} = \alpha_{CL} \circ \gamma_{DC} \circ \alpha_{CD}$. Since $\forall c \in C,\ \alpha_{CD}(c) = \alpha_{CD}(\gamma_{DC} \circ \alpha_{CD}(c))$, cf. Proposition 2.2 (5), using Proposition 2.11 (iii), one obtains that $\forall c \in C,\ \alpha_{CL}(c) = \alpha_{CL}(\gamma_{DC} \circ \alpha_{CD}(c))$.

($\Leftarrow$) By hypothesis, $\gamma_{DC} \circ \gamma_{LD} = \gamma_{LC}$. It suffices to apply α_{CD} to both members of this equation and use Proposition 2.2 (3), to obtain the desired relation:

$$\gamma_{LD} = \alpha_{CD} \circ \gamma_{LC}$$

That also $\alpha_{DL} = \alpha_{CD} \circ \gamma_{DC}$ can be shown similarly. ∎

2.2 Optimalities

As usual, an abstract interpretation is optimal if it mimics the concrete one in the best possible way. We also introduce a weaker notion in which we project the result of the operation on a more abstract domain.

Definition 2.14 (optimalities) *Consider the interpretations $\mathbf{D}=\langle D, \mu_D \rangle$ and $\mathbf{C}=\langle C, \mu_C \rangle$. Assume that $\mathbf{D}$ abstracts $\mathbf{C}$. $\mathbf{D}$ is optimal if $\forall d \in D : \mu_D(d) = \alpha_{CD}(\mu_C(\gamma_{DC}(d)))$.*
Let now R be a domain which abstracts D, and let $\alpha_{CR} = \alpha_{DR} \circ \alpha_{CD}$. We say that $\mathbf{D}$ is R-optimal if $\forall d \in D : \alpha_{DR}(\mu_D(d)) = \alpha_{CR}(\mu_C(\gamma_{DC}(d)))$.

Lemma 2.15 *Let $\mathbf{C}$, $\mathbf{D}$, and R as in the previous definition. If $\mathbf{D}$ is optimal then $\mathbf{D}$ is also R-optimal.*

proof: Let $d \in D$.

$$\begin{array}{rrcll} & \mu_D(d) & = & \alpha_{CD}(\mu_C(\gamma_{DC}(d))) & \text{as } \mathbf{D} \text{ is optimal} \\ \Rightarrow & \alpha_{DR}(\mu_D(d)) & = & \alpha_{DR}(\alpha_{CD}(\mu_C(\gamma_{DC}(d)))) & \text{applying } \alpha_{DR} \text{ to both sides} \\ \Rightarrow & \alpha_{DR}(\mu_D(d)) & = & \alpha_{CR}(\mu_C(\gamma_{DC}(d))) & \text{by definition of } \alpha_{CR} \end{array}$$

∎

Observe that the notion of R-optimality is a generalization of the notion of optimality. In fact, $\mathbf{D}$ is optimal iff $\mathbf{D}$ is D-optimal.

2.3 General Comparison Criterion

Let P be an abstract domain that represents a property we are interested in. Assume that the two interpretations $\mathbf{L}$ and $\mathbf{D}$ also represent this property. This fact is modeled by the assumption that P abstracts both L and D. We want to compare the precision of $\mathbf{L}$ and $\mathbf{D}$ with respect to the way they compute P, according to the following intuitive idea. $\mathbf{L}$ is at least as precise as $\mathbf{D}$ with respect to P if every sequence of concrete operations is better or equally approximated by $\mathbf{L}$ than by $\mathbf{D}$ when considering only the information representable in P.

Definition 2.16 (comparison criterion) *Let $\mathbf{D} = \langle D, \mu_D \rangle$ and $\mathbf{L} = \langle L, \mu_L \rangle$ be interpretations abstracting $\mathbf{C} = \langle C, \mu_C \rangle$. Let P be a domain abstracting both D and L. Let also μ_C^i denote the i-th fold composition of μ_C and μ_D^i and μ_L^i the corresponding sequences of operators of D and L.*

- $\mathbf{L}$ *is at least as precise as* $\mathbf{D}$ *with respect to* P *if*

$$\forall c \in C, \forall i \geq 0 : \ \alpha_{LP}(\mu_L^i(\alpha_{CL}(c))) \sqsubseteq_P \alpha_{DP}(\mu_D^i(\alpha_{CD}(c))).$$

- $\mathbf{L}$ *is strictly more precise than* $\mathbf{D}$ *with respect to* P *if* $\mathbf{L}$ *is at least as precise as* $\mathbf{D}$ *but the converse does not hold.*

3 The Quotient of an Interpretation

This section consists of two parts. In the first one, two domains D and P are considered, where P represents a particular property expressed by D. An equivalence relation r_p on D is defined that identifies the classes of elements of D that are equivalent wrt the computation of P-information. It is shown that, when r_p is additive, it is possible to define an abstraction of D, called the *quotient of D wrt P*, that represents exactly the information of D that is used to compute P-information. In the second part of the section we show the relevance of quotients for comparing the precision with which different interpretations compute a given property.

Throughout this section, we always assume that G_{CP}, G_{CD}, G_{DP} are Galois insertions with G_{DP} coherent with respect to C (i.e., $G_{CP} = G_{CD} \circ G_{DP}$), and that the interpretation $\mathbf{D} = \langle D, \mu_D \rangle$ abstracts the interpretation $\mathbf{C} = \langle C, \mu_C \rangle$ with μ_D optimal.

3.1 Definition and Properties of the Quotient

First, we characterize elements of D that are equivalent, with respect to P, in any computation sequence.

Definition 3.1 (associated relation) *The equivalence relation r_p on D associated to P is defined by*

$$(d_1, d_2) \in r_p \quad \Leftrightarrow \quad \forall i \geq 0 : \alpha_{DP}(\mu_D^i(d_1)) = \alpha_{DP}(\mu_D^i(d_2))$$

Observe in particular that $(d_1, d_2) \in r_p$ implies $\alpha_{DP}(d_1) = \alpha_{DP}(d_2)$. In the sequel $[d]_p$ denotes the set $\{d' \in D \mid (d, d') \in r_p\}$.

Clearly, the intuition suggests that the quotient of D wrt P should be a set Q that has one element corresponding to each equivalence class of D wrt r_p. Unfortunately, this is not always the case. In fact, it is easy to find equivalence relations on D for which such a Q is not an abstraction of D (whereas we want the quotient of D to abstract it). Below we will show that if r_p is *additive*, then such a Q is an abstraction of D and it will, in fact, be the quotient we are looking for. After having proven this fact, we will show in Theorem 3.7 that the additivity of r_p is equivalent to the additivity of the abstraction function that connects D to Q. This relationship should not be surprising in view of Proposition 2.3 that, in the present context, shows that the additivity of the abstraction function implies the existence of a Galois insertion between D and Q.

Definition 3.2 (Additivity) *The relation* $r_{_P}$ *is additive when* $\forall S \subseteq r_{_P}$*, if* $S_1 = \{a \,|\, (a,b) \in S\}$ *and* $S_2 = \{b \,|\, (a,b) \in S\}$*, it is true that* $(\sqcup_D S_1, \sqcup_D S_2) \in r_{_P}$.

In what follows some important consequences of the additivity of $r_{_P}$ are shown.

Lemma 3.3 *If the relation* $r_{_P}$ *on* D *associated to* P *is additive, then*

$$\forall d \in D : (\sqcup_D [d]_{_P}, d) \in r_{_P}, \;\; i.e. \;\; \sqcup_D [d]_{_P} \in [d]_{_P}.$$

proof: It is sufficient to observe that, by additivity of $r_{_P}$, if $[d]_{_P} = \{d_i : i \in I\}$ then $(\sqcup_D\{d_i : i \in I\}, \sqcup_D\{d\}) = (\sqcup_D\{d_i : i \in I\}, d) \in r_{_P}$. ☐

Lemma 3.4 *If the relation* $r_{_P}$ *on* D *associated to* P *is additive, then*

$$\forall d_1, d_2 \in D : d_1 \sqsubseteq_D d_2 \;\Rightarrow\; \sqcup_D [d_1]_{_P} \sqsubseteq_D \sqcup_D [d_2]_{_P}.$$

proof: Let $\hat{d_1} = \sqcup_D [d_1]_{_P}$ and $\hat{d_2} = \sqcup_D [d_2]_{_P}$. By Lemma 3.3, $(d_1, \hat{d_1}) \in r_{_P}$ and $(d_2, \hat{d_2}) \in r_{_P}$. Thus, by additivity of $r_{_P}$, $(d_1 \sqcup_D d_2, \hat{d_1} \sqcup_D \hat{d_2}) \in r_{_P}$. By hypothesis, $d_1 \sqsubseteq_D d_2$, so we get $(d_2, \hat{d_1} \sqcup_D \hat{d_2}) \in r_{_P}$, i.e. $\hat{d_1} \sqcup_D \hat{d_2} \in [d_2]_{_P}$. Therefore, by the definition of $\hat{d_2}$, $\hat{d_1} \sqcup_D \hat{d_2} \sqsubseteq_D \hat{d_2}$, and thus $\hat{d_1} \sqsubseteq_D \hat{d_2}$. ☐

Let us give now the definition of quotient. After that we will show that the additivity of $r_{_P}$ implies that the quotient enjoys all the properties we wanted and in particular that it abstracts D.

Definition 3.5 (Quotient) *The quotient of* D *with respect to* P *is the set* $\mathcal{Q}_P(D)$ *defined by:*

$$\mathcal{Q}_P(D) = \{\sqcup_D [d]_{_P} \mid d \in D\}.$$

$\mathcal{Q}_P(D)$ *is a subset of* D *and thus it is partially ordered.*

Theorem 3.6 *If the associated relation* $r_{_P}$ *is additive, the following facts hold.*

i) $\mathcal{Q}_P(D)$ *is a complete lattice that abstracts* D*;*

ii) P *abstracts* $\mathcal{Q}_P(D)$ *coherently wrt* C*;*

proof

i) We will show that $\mathcal{Q}_P(D)$ is the set of fixpoints of the following *uco* $\rho_{_\mathcal{Q}}$ on D:

$$\forall d \in D. \;\; \rho_{_\mathcal{Q}}(d) = \sqcup_D [d]_{_P}$$

Thus we must show that $\rho_{_\mathcal{Q}}$ is extensive, idempotent and monotone:

- *it is extensive:* by definition;
- *it is idempotent:* using Lemma 3.3 it is simple to see that: $\forall d \in D, \sqcup_D[\sqcup_D [d]_{_P}] = \sqcup_D [d]_{_P}$
- *it is monotone:* immediate from Lemma 3.4.

Obviously $\mathcal{Q}_P(D) = \rho_{_\mathcal{Q}}(D)$ and thus, from Proposition 2.5, it follows that there is a Galois insertion between D and $\mathcal{Q}_P(D)$ with abstraction $\rho_{_\mathcal{Q}}$ and the identity as concretization. Moreover, from Proposition 2.6, we have that $\mathcal{Q}_P(D)$ is a complete lattice.

ii) In order to show that P abstracts $\mathcal{Q}_P(D)$, by Proposition 2.11(ii), it suffices to show that $\gamma_{PD}(P) \subseteq \rho_{\mathcal{Q}}(D)$. Precisely we want to show that:

$$\forall b \in P.\ \ \gamma_{PD}(b) = \sqcup_D [\gamma_{PD}(b)]_p$$

Let $a = \sqcup_D [\gamma_{PD}(b)]_p$. Obviously, the following point (1) holds: $a \ \sqsupseteq_D \ \gamma_{PD}\ (b)$.
Observe now that, by Lemma 3.3, $\alpha_{DP}(a) = \alpha_{DP} \circ \gamma_{PD}(b) = b$ ($\alpha \circ \gamma$ is the identity) from which, applying γ_{PD} on both sides, one obtains:

$$a \sqsubseteq_D \gamma_{PD} \circ \alpha_{DP}(a) \ = \ \gamma_{PD}(b) \qquad (\gamma \circ \alpha \text{ is extensive}).$$

This together with (1) shows what we wanted. Thus, by Proposition 2.11(i), there is a Galois insertion $G_{QP} = (\alpha_{DQ} \circ \gamma_{PD}, \mathcal{Q}_p(D), P, \alpha_{DP} \circ \gamma_{QD})$ that is coherent wrt D. Thus, by Proposition 2.13, $G_{DP} = G_{DQ} \circ G_{QP}$. Since, by hypothesis, G_{DP} is coherent wrt C, it is the case that, $G_{CP} = G_{CD} \circ G_{DP}$, and thus, $G_{CP} = G_{CD} \circ G_{DQ} \circ G_{QP} = G_{CQ} \circ G_{QP}$ which, by Proposition 2.13, proves that G_{QP} is coherent wrt C. ∎

It is easy to see that the join on $\mathcal{Q}_P(D)$, denoted $\sqcup_Q$, is as follows: $u_1 \sqcup_Q u_2 = \sqcup_D [u_1 \sqcup_D u_2]_p$. The meet is defined similarly. As already announced, the additivity of r_Q is equivalent to that of the abstraction function $\rho_{\mathcal{Q}}$ (defined in the proof of Theorem 3.6). Recall that $\rho_{\mathcal{Q}} : D \to \mathcal{Q}_P(D)$ is additive if $\forall X \subseteq D,\ \ \rho_{\mathcal{Q}}(\sqcup_D X) = \sqcup_Q \{\rho_{\mathcal{Q}}(x) \mid x \in X\}$.

Theorem 3.7 *r_p is additive iff $\rho_{\mathcal{Q}}$ is additive.*

proof
By the additivity of r_p, it is true that:

$$\forall X \subseteq D, \bigsqcup_D X \ \ r_p \ \bigsqcup_D \{\sqcup_D [a]_p \mid a \in X\}$$

Using the definition of r_p, this is equivalent to:

$$\sqcup_D [\sqcup_D X]_p \ \ = \ \ \sqcup_D [\sqcup_D \{\sqcup_D [a]_p \mid a \in X\}]_p$$

It suffices now to observe that,

$\sqcup_D [\sqcup_D X]_p$	$=$	$\rho_{\mathcal{Q}}(\sqcup_D X)$	by definition of $\rho_{\mathcal{Q}}$
and that			
$\sqcup_D [\sqcup_D \{\sqcup_D [a]_p \mid a \in X\}]_p$	$=$	$\sqcup_Q \{\rho_{\mathcal{Q}}(a) \mid a \in X\}$	by definition of $\rho_{\mathcal{Q}}$ and of $\sqcup_Q$ given before this theorem.

∎

3.2 Comparison of Quotients

The results of this subsection show the important role that the notion of quotient plays in the comparison of two abstract interpretations. For the sake of clarity, in **ASS** below we summarize the notation and the hypotheses that we will use in the following theorems.

ASS (*a*) **D** and **L** are abstract interpretations, **C** is the concrete interpretation and P is the abstract domain that represents the property that is being studied.

(*b*) G_{CD}, G_{CL}, G_{CP}, G_{DP} and G_{LP} are Galois insertions. The last two are coherent with respect to C.

(*c*) $R_1 = \mathcal{Q}_P(D)$ and $R_2 = \mathcal{Q}_P(L)$. By Theorem 3.6, there are Galois insertions G_{R_1P} and G_{R_2P} that are coherent with respect to C.

$$
\begin{tikzcd}
 & P \arrow[dl] \arrow[dr] & \\
R_1 = \mathcal{Q}_P(D) \arrow[rr] \arrow[d] & & R_2 = \mathcal{Q}_P(L) \arrow[d] \\
D \arrow[dr] & & L \arrow[dl] \\
 & C &
\end{tikzcd}
$$

Figure 1: Domain abstractions in Theorems 3.10 and 3.11

(d) **D** and **L** are, respectively, R_1- and R_2-optimal.

(e) R_1 abstracts R_2 coherently with respect to C.

The following fact is a simple consequence of the assumptions **ASS** above.

Lemma 3.8 $G_{R_2P} = G_{R_2R_1} \circ G_{R_1P}$ *and thus, in particular,* $\alpha_{R_2P} = \alpha_{R_1P} \circ \alpha_{R_2R_1}$.

Figure 1 summarizes the relations existing among all domains considered. The arrows correspond to γ-functions.

Theorem 3.9 *Assume* **ASS** *above. Then* **L** *is at least as precise as* **D** *for computing* P.

proof: Consider any sequence π_D of operations of **D**. From the fact that **D** is R_1-optimal and from the construction of the quotient R_1, it is true that for any $d \in D$, the computation $\pi_D(d)$ can be "read" over R_1, as far as the P-information is concerned. More precisely, assume that d_i is the result after the first $i \geq 0$ steps of the computation $\pi_D(d)$, and let $t_i = \alpha_{DR_1}(d_i)$, then, for each i, $\alpha_{DP}(d_i) = \alpha_{R_1P}(t_i)$. A similar fact holds for any computation π_L on **L**. For such a π_L, let $l_0, l_1, .., l_i$ be the intermediate results and $k_0, .., k_i$ be the corresponding values in R_2.

We will use the above fact and the notation introduced in the sequel of the proof. Let for any concrete value $c \in C$, $d_0 = \alpha_{CD}(c)$ and $l_0 = \alpha_{CL}(c)$. Let also π_D and π_L be sequences of corresponding operations of **D** and **L**. In what follows the d_i, t_i, l_i, and k_i are in the relation explained above. In order to prove the result, it suffices to show the following fact:

$$(\star) \qquad \forall i \geq 0,\ \gamma_{R_1C}(t_i) \sqsupseteq_C \gamma_{R_2C}(k_i).$$

In fact, from ($\star$) it follows that $\alpha_{R_2R_1}(k_i) \sqsubseteq_{R_1} t_i$. By Lemma 3.8 above and the monotonicity of α_{R_1P}, $\alpha_{R_2P}(k_i) = \alpha_{R_1P}(\alpha_{R_2R_1}(k_i)) \sqsubseteq_P \alpha_{R_1P}(t_i)$. We proceed by induction on i.

basis Let us first consider $i = 0$. We want to show that

$$(1) \qquad \gamma_{R_1C}(t_0) \sqsupseteq_C \gamma_{R_2C}(k_0).$$

By Proposition 2.9, $t_0 = \alpha_{CR_1}(c)$ and $k_0 = \alpha_{CR_2}(c)$. Since R_1 abstracts R_2 coherently wrt C, the following two points hold:

a) $\alpha_{CR_1} = \alpha_{R_2R_1} \circ \alpha_{CR_2}$. Hence, $t_0 = \alpha_{R_2R_1}(k_0)$.

b) $\gamma_{R_1C} = \gamma_{R_2C} \circ \gamma_{R_1R_2}$

¿From point (a) and (b), using the extensivity of $\gamma_{R_1R_2} \circ \alpha_{R_2R_1}$ (cf. Section 2.1), we get immediately that:

$$\gamma_{R_1C}(t_0) = \gamma_{R_2C} \circ \gamma_{R_1R_2} \circ \alpha_{R_2R_1}(k_0) \geq_C \gamma_{R_2C}(k_0)$$

Thus (1) holds.

step Let us now prove ($\star$) for $i > 0$. We want to show that

$$(2) \quad \gamma_{R_1C}(t_{i-1}) \sqsupseteq_C \gamma_{R_2C}(k_{i-1}) \qquad \Rightarrow \qquad (3) \quad \gamma_{R_1C}(t_i) \sqsupseteq_C \gamma_{R_2C}(k_i).$$

By the assumption of R_1- and R_2-optimality of **D** and **L**, we know that

$$t_i = \alpha_{CR_1}(\mu_C(\gamma_{R_1C}(t_{i-1})))$$

where μ_C is the concrete operation corresponding to the i-th operation of π_D and π_L. A similar relation holds for k_i.

By assumption (2) and the monotonicity of μ_C it follows that

$$\mu_C(\gamma_{R_1C}(t_{i-1})) \sqsupseteq_C \mu_C(\gamma_{R_2C}(k_{i-1}))$$

and thus, by the same reasoning used for the case $i = 0$, we have that (3) holds. □

Theorem 3.10 *Assume, in addition to* **ASS**, *that R_1 properly abstracts R_2. Then* **L** *is strictly more precise than* **D** *for computing P.*

proof: By assumption it is true that:

$$(1) \quad \gamma_{R_1C}(R_1) \subseteq \gamma_{R_2C}(R_2) \text{ and } \gamma_{R_2C}(R_2) \not\subseteq \gamma_{R_1C}(R_1)$$

Hence there is $r_0 \in R_2$ such that $\gamma_{R_2(}r_0) \notin \gamma_{R_1C}(R_1)$. Let $c_0 = \gamma_{R_2C}(r_0)$, and $l_0 = \gamma_{R_2L}(r_0)$. Note that, since $\gamma_{R_2C} = \gamma_{LC} \circ \gamma_{R_2L}$, $\gamma_{LC}(l_0) = c_0$.

Let now $\hat{c} = \gamma_{R_1C} \circ \alpha_{CR_1}(c_0)$. Clearly, by their definitions, $\hat{c} \sqsupset_C c_0$ and, $\alpha_{CR_1}(c_0) = \alpha_{CR_1}(\hat{c})$, from which the following fact (2) holds:

Fact (2): $\alpha_{CD}(c_0)$ and $\alpha_{CD}(\hat{c})$ are elements of D that are equivalent wrt the computation of P, i.e., they are in the same equivalence class of the relation on D associated to P.

Note also that, by (1), $\hat{c} \in \gamma_{R_2C}(R_2)$. Let us now abstract $\hat{c}$ into L and R_2. We call $l = \alpha_{CL}(\hat{c})$ and $r_2 = \alpha_{LR_2}(l)$.

By assumption, we know that for all corresponding sequences s_D and s_L of operations of **D** and **L**, respectively, it is true that:

$$\alpha_{DP}(s_D(\alpha_{CD}(\hat{c}))) \sqsupseteq_P \alpha_{LP}(s_L(\alpha_{CL}(\hat{c})))$$

¿From this, using Fact (2), we obtain:

$$(3) \quad \alpha_{DP}(s_D(\alpha_{CD}(c_0))) = \alpha_{DP}(s_D(\alpha_{CD}(\hat{c}))) \sqsupseteq_P \alpha_{LP}(s_L(\alpha_{CL}(\hat{c})))$$

Observe now that, since $\hat{c} \sqsupset_C c_0$, and both are in $\gamma_{R_2C}(R_2)$, it is true that, $r_2 \sqsupset_{R_2} r_0$. Thus, there exists a computation sequence π_L of operations of **L** such that

$$\alpha_{DP}(\pi_L(l)) \neq \alpha_{DP}(\pi_L(l_0))$$

Since π_L is composed of continuous (and thus monotone) operations, we have:

$$(4) \quad \alpha_{DP}(\pi_L(l)) \sqsupset_P \alpha_{DP}(\pi_L(l_0))$$

It suffices now to put together (3) and (4) to show the thesis.

$$\alpha_{DP}(\pi_D(\alpha_{CD}(c_0))) \sqsupseteq_P \alpha_{DP}(\pi_L(\alpha_{CL}(\hat{c}))) = \alpha_{DP}(\pi_L(l)) \sqsupset_P \alpha_{DP}(\pi_L(l_0)) = \alpha_{DP}(\pi_L(\alpha_{CL}(c_0)))$$

□

4 Applications

We apply the theory developed in the previous section for comparing two well-known abstract interpretations for logic programming: **Prop** [2, 8, 10, 19] and **Sharing** [16]. This section is organized as follows. After some preliminary definitions concerning substitutions, in section 4.2, we recall from [10] the concrete interpretation **Rsub** and the two abstract interpretations we wish to compare. The domain GR representing groundness and the characterizations of the quotients $\mathcal{Q}_{\text{GR}}(\texttt{Sharing})$ and $\mathcal{Q}_{\text{GR}}(\texttt{Prop})$ are described in Section 4.3. The main result of the application part is in Section 4, where we apply Theorems 3.9 and 3.10 for proving that **Prop** is strictly more precise than **Sharing** with respect to the precision in computing groundness.

We point out that the interpretations **Prop** and **Sharing** that we compare are complete interpretations, in the sense that they include all the operations needed for the static analysis, viz. forward/backward unification, least upper bound, and projection. In order to describe in a simple way all the operations (and, in particular, projection), we adopt the approach introduced in [10]. In this approach, the (non trivial) values of a domain are pairs in which the first component is "the usual value", and the second component explicitly specifies the variables about which the first component provides information. These variables are often called *the variables of interest*.

4.1 Preliminaries

Let $\mathbf{V}$ be a countable set of variables. $FP(\mathbf{V})$ denotes the set of finite subsets of variables of $\mathbf{V}$. A *substitution* σ is a function in that maps variables in $\mathbf{V}$ to terms over $\mathbf{V}$ and an alphabet of function symbols, and such that $\sigma x \neq x$ only for a finite number of variables x. The *set of support* of σ is given by $supp(\sigma) = \{x \mid \sigma x \neq x\}$. The *variable range* of σ is given by $var\text{-}range(\sigma) = \bigcup\{ Var(\sigma x) \mid x \in supp(\sigma)\}$, where $Var(t)$ denotes the set of variables occurring in t. The set of variables occurring in σ is given by $Var(\sigma) = supp(\sigma) \cup var\text{-}range(\sigma)$. A substitution is typically specified by listing its non-trivial bindings. So $\sigma = \{x/\sigma x \mid x \in supp(\sigma)\}$.

Consider two substitutions σ_1 and σ_2. If there exists ϑ such that $\sigma_2 = \vartheta \circ \sigma_1$, then σ_1 is *more general* than σ_2, which we write $\sigma_2 \trianglelefteq \sigma_1$. In this case, we say that σ_2 an instance of σ_1.

We write *Subst* for the set of idempotent substitutions. Although *Subst* is not closed under composition, in a step of the execution of a logic program in which $\vartheta \circ \sigma$ is constructed, it is always the case that, $var\text{-}range(\vartheta) \cap supp(\sigma) = \emptyset$, which, provided that ϑ and σ are idempotent, ensures that $\vartheta \circ \sigma$ is also idempotent.

As we will consider sequences of concrete/abstract operations of "real" domains, i.e. containing not only unary operations, as it was assumed in Definition 2.7 for the sake of simplicity, it is necessary to make precise this notion for any set of operations. Assume to have an interpretation $\mathbf{D} = \langle D, \mu_1, \ldots, \mu_k\rangle$. A *derived operator* over $\mathbf{D}$ is a term t constructed using the symbols in $\mu_1, \ldots, \mu_k$, the values in D, values of any other domain that may be required by the operations (for instance, substitutions are needed in the unification operations), and exactly one variable. The following example illustrates this notion for $\mathbf{Z} = \langle Z, +, *\rangle$, where Z represents the set of all integers completed with top and bottom elements.

Example 4.1 *A derived operator for* $\mathbf{Z}$ *is* $t = +(*(x, 3), +(2, 1))$.

Clearly, a derived operator t is a function $t : D \longrightarrow D$. Intuitively, the result of the function t for a given value $d \in D$ is obtained by evaluating $t(d)$ interpreting the function symbols in t according to $\mathbf{D}$. In the above example, $t(0) = 3$ and $t(2) = 9$.

4.2 The interpretations Rsub, Prop, and Sharing

The interpretations **Rsub**, **Prop**, and **Sharing** consist each of a domain and three operations: unification, projection, and least upper bound. Since some of the operations are quite technical, we have chosen to recall them in the Appendix, and to describe here only the domains, their partial orders, and the concretization/abstraction functions relating them.

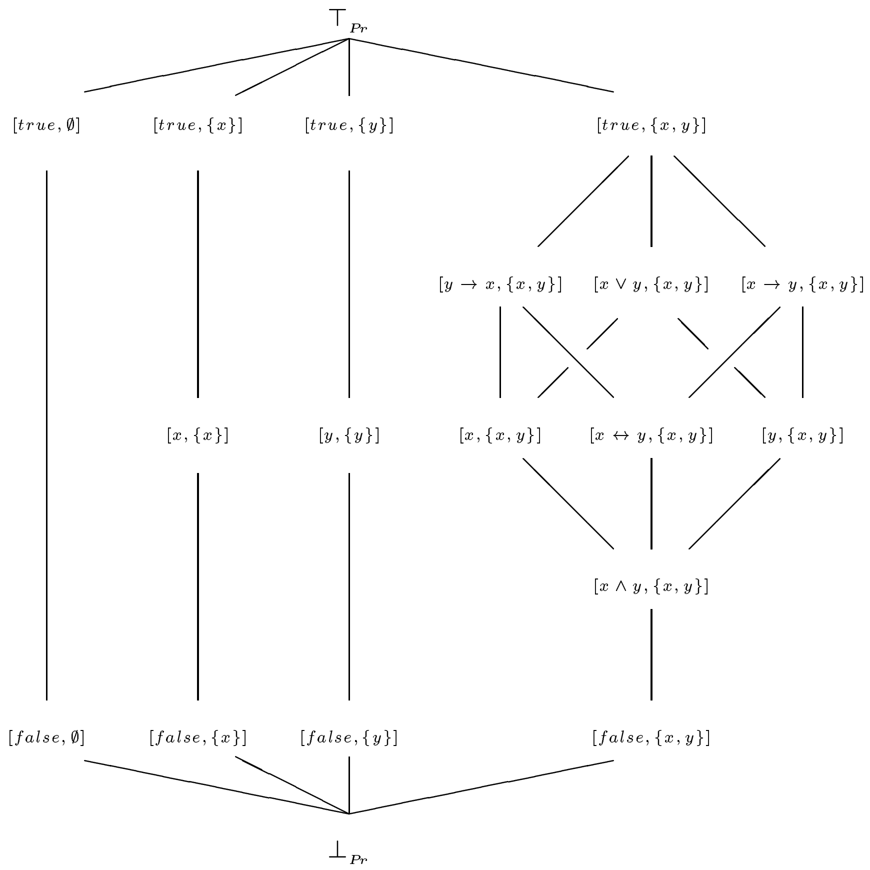

Figure 2: The domain Prop for $\mathbf{V}$=$\{x, y\}$

4.2.1 The "concrete" domain Rsub

The "concrete" domain Rsub [10] is the complete lattice

$$\texttt{Rsub} = [\wp(\mathit{Subst}) \times \mathit{FP}(\mathbf{V})] \cup \{\top_{Rs}, \perp_{Rs}\}.$$

Rsub stands for restricted substitutions. The partial order of Rsub is defined, on nontrivial elements, by $[\Sigma_1, U_1] \sqsubseteq_{Rs} [\Sigma_2, U_2]$ iff $U_1 = U_2$ and $\Sigma_1 \subseteq \Sigma_2$. The operations od **Rsub** are described in the Appendix.

4.2.2 The domain Prop

For any set of variables $U \in \mathit{FP}(\mathbf{V})$, by $\wedge U$ we denote the formula consisting of the conjunction of the variables in U. For any $U \in \mathit{FP}(\mathbf{V})$, a *positive* formula ([2, 21]) on U is any propositional formula containing only variables in U and that is satisfied by the truth-assignment that assigns *true* to all variables in U. The set of positive formulas on U is denoted Pos_U. ¿From now on, in order to avoid burdensome notation, we simply write f for the class of formulas equivalent to f and assume that Pos_U consists of classes of equivalent formulas. We also adopt the usual convention of representing a truth-assignment a on U as the set $\{x \in U \mid a(x) = \mathit{true}\}$.

Notice that for any $U \in \mathit{FP}(\mathbf{V})$, $\mathit{Pos}_U \cup \{\mathsf{F}\}$ is a complete lattice with least upper bound and greatest lower bound, respectively, $\vee$ (logical disjunction) and $\wedge$ (logical conjunction), appropriately extended to classes of equivalent formulas.

The domain Prop is as follows:

$$\texttt{Prop} = \{\, [f, U] : U \in \mathit{FP}(\mathbf{V}),\ f \in \mathit{Pos}_U \cup \{\mathsf{F}\} \,\} \cup \{\top_{Pr}, \perp_{Pr}\}.$$

Prop is partially ordered: $\top_{Pr}$ is the largest element and $\bot_{Pr}$ is the smallest; for the other elements, $[f_1, U_1] \leq_{Pr} [f_2, U_2]$ if and only if $U_1{=}U_2$ and $f_1 \models f_2$.

Figure 2 depicts the domain Prop for $\mathbf{V} = \{x, y\}$. The lines represent the ordering relation among the (equivalence classes of) formulas.

That positive formulas are useful for computing variable groundness in logic programs is well-known, see [2, 8, 10, 19, 12]. The intuition behind the relation between positive formulas and substitutions, is as follows. Each substitution defines a truth-assignment, and, since groundness is a property closed under instantiation, we say that a formula approximates a substitution when it is true wrt the truth-assignments defined by all instances of that substitution.

The truth-assignment of substitution σ is $assign\sigma$: $assign\,\sigma\,x = true$ *iff* σ grounds x. The concretization expressing the relation between Prop and Rsub, is

$$\gamma_{PrRs} : \texttt{Prop} \longrightarrow \texttt{Rsub}$$
$$\gamma_{PrRs}(d) = \begin{cases} \top_{Rs} & \text{if } d = \top_{Pr}; \\ \bot_{Rs} & \text{if } d = \bot_{Pr}; \\ [\{\sigma \in Subst \mid \forall \sigma' \trianglelefteq \sigma \,.\, assign\,\sigma' \models f\}, U] & \text{if } d = [f, U]. \end{cases}$$

The function $\alpha_{RsPr} : \texttt{Rsub} \longrightarrow \texttt{Prop}$ is the usual adjoint [6] of γ_{PrRs}, i.e., $\alpha_{RsPr}(c) = \sqcap_{Pr}\{d \in \texttt{Prop} \mid \gamma_{PrRs}(d) \sqsupseteq_{Rs} c\}$. The tuple ($\gamma_{PrRs}$,Rsub, Prop,$\alpha_{RsPr}$) is a Galois Insertion [10]. In [10] it is also shown that the operations of **Prop** (see Appendix) are optimal.

Lemma 4.2 ([10]) Prop *is optimal.*

The definite formulas in U, denoted Def_U [2, 14], consist of the formulas $f \in Pos_U$ that satisfy the following model intersection property: consider any two models M_1 and M_2 of f, if $M = M_1 \cap M_2$, then $M \models f$. The name "definite" for such formulas comes from the following well-known syntactical characterization: for each $f \in Def_U$ there is a formula on U equivalent to f and consisting of a conjunction of definite implications of the form $\wedge W \to x$.

Observe that the set Def_U is properly included in Pos_U. For instance, the formula $x \vee y$ of $Pos_{\{x,y\}}$ does not belong to $Def_{\{x,y\}}$. In fact, let $M_1 = \{x\}$, and $M_2 = \{y\}$, it is immediate to see that both M_1 and M_2 are models of $x \vee y$, whereas $M = M_1 \cap M_2 = \emptyset$ is not a model of this formula.

In the same way as positive formulas were used for defining Prop , it is possible to define a domain using the definite formulas:

$$\texttt{Def} = \{\, [f, U] \mid U \in FP(\mathbf{V}), f \in Def_U \cup \{\mathsf{F}\}\} \cup \{\top_{Df}, \bot_{Df}\}$$

Obviously, Def abstracts Prop with the identity as concretization and its adjoint as abstraction. From this it also follows that Def also abstracts Rsub with the same concretization as Prop.

4.2.3 The domain Sharing

The abstract domain Sharing proposed by Jacobs and Langen in [16] in order to represent variable aliasing, covering, and groundness is defined by

$$\texttt{Sharing} = \{[A, U] \mid A \subseteq \wp(U), A \neq \emptyset \Rightarrow \emptyset \in A, U \in FP(\mathbf{V})\} \cup \{\top_{Sh}, \bot_{Sh}\}.$$

Sharing is partially ordered: $\top_{Sh}$ is the largest element and $\bot_{Sh}$ is the smallest one; for the other elements, $[A_1, U_1]$, $\sqsubseteq_{Sh}$ $[A_2, U_2]$ iff $U_1{=}U_2$ and $A_1 \subseteq A_2$. The domain Sharing for $\mathbf{V} = \{x, y\}$ is depicted in Figure 3. Even though the lattice structure is similar to that of Prop, the two domains represent different informations of the concrete domain Rsub.

Jacobs and Langen [16] proved that Sharing enjoys a Galois insertion into $\wp(Subst)$. This can be immediately extended to our concrete domain Rsub. We recall briefly the construction of the abstraction of this insertion. For $x \in \mathbf{V}$,$U \subseteq \mathbf{V}$, and $\sigma \in Subst$, let $share(\sigma, x, U)$ be the set of variables in U whose images under σ contain the variable x, i.e. $share(\sigma, x, U) = \{y \in U \mid x \in Var(\sigma y)\}$. For $[\Sigma, U] \in \texttt{Rsub}$,

$$\alpha_{RsSh}([\Sigma, U]) \quad = \quad [\{share(\sigma, x, U) \mid \sigma \in \Sigma,\ x \in \mathbf{V}\}, U].$$

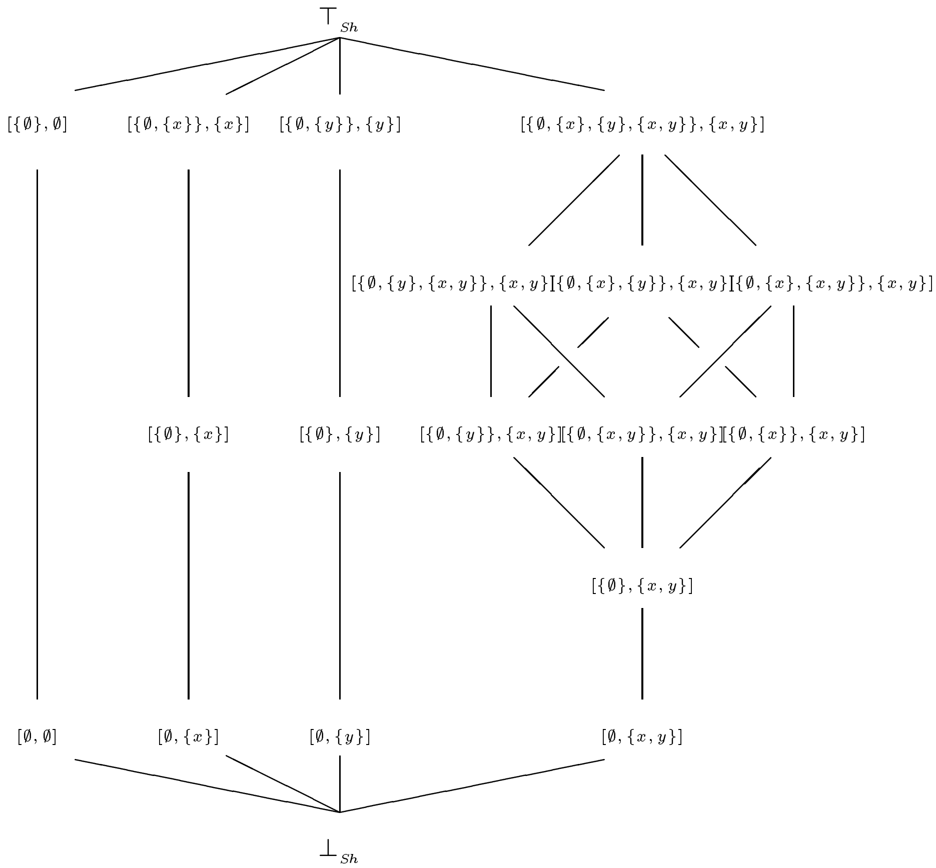

Figure 3: The domain `Sharing` for $\mathbf{V}=\{x,y\}$

The concretization γ_{ShRs} is the usual adjoint of the abstraction. Let $[A, U] = \alpha_{RsSh}([\{\sigma\}, U])$. Each $S \in A$ is a set of variables that under σ share a variable. Every variable $x \in U$ such that $Var(\sigma x) = \emptyset$ will not appear in A. In [11] it is shown that the operations of **Sharing** are optimal.

Lemma 4.3 ([11]) **Sharing** *is optimal.*

4.3 Quotients with respect to Groundness

The interpretations **Sharing** and **Prop** are incomparable with respect to the notion of abstraction [9]. The intuition behind this result is the following. On the one hand, by means of disjunctions, `Prop` represents also *possible equivalence* (and thus also groundness), whereas `Sharing` does not. On the other hand, `Sharing` represents *variable independence* that is not expressible in `Prop`. However, as both interpretations compute groundness information, we are interested in comparing their precision in the computation of groundness.

4.3.1 The domain GR

The simplest domain that represents variable groundness is `GR` as follows. Given an element $[\Sigma, U] \in$ `Rsub`, its groundness information can be represented by the set of variables grounded by every substitution in Σ.

$$\texttt{GR} = \{[A, U] : A \in \wp(U) \cup \{\bot\}, U \in FP(\mathbf{V})\} \cup \; \{\top_{Gr}, \bot_{Gr}\}.$$

The set `GR` is partially ordered as follows. $\top_{Gr}$ is the top element, and $\bot_{Gr}$ the bottom one. $[B_1, U_1] \sqsubseteq_{Gr} [B_2, U_2]$ if $U_1 = U_2$ and $B_1 \supseteq B_2$. Obviously, `GR` is a complete lattice. The least upper bound of two elements $[B_1, U_1]$ and $[B_2, U_2]$ is defined as

$$[B_1, U_1] \sqcup_{Gr} [B_2, U_2] = \begin{cases} [B_1 \cap B_2, U_1] & \text{if } U_1 = U_2 \\ \top_{Gr} & \text{otherwise} \end{cases}$$

It is easy to see that there are Galois insertion between `GR` on the one side and `Rsub`, `Prop Def` and `Sharing` on the other. We only specify the concretization and abstraction functions, as proving that each pair of functions forms a Galois insertion, is an easy exercise.

$$\begin{cases} \gamma_{GrRs}([B, U]) = [\Sigma, U], \text{ where } \Sigma = \{\sigma \in Subst \mid \forall x \in B, Var(\sigma x) = \emptyset\} \\ \alpha_{RsGr}([\Sigma, U]) = [\bigcap_{\sigma \in \Sigma}\{x \in U \mid Var(\sigma x) = \emptyset\}, U] \end{cases}$$

$$\begin{cases} \gamma_{GrPr}([B, U]) = \gamma_{GrDf}([B, U] = [\wedge B, U] \\ \alpha_{PrGr}([f, U]) = \alpha_{DfGr}([f, U]) = [\{x \mid f \models x\}, U] \end{cases}$$

$$\begin{cases} \gamma_{GrSh}([B, U]) = [\wp(U \setminus B), U] \\ \alpha_{ShGr}([A, U]) = [U \setminus (\cup A), U] \end{cases}$$

These Galois insertions together with those that connect `Prop`,`Def`, and `Sharing` to `Rsub`, defined above, are coherent with respect to `Rsub`.

4.3.2 The Quotient of Prop with respect to GR

In Section 3 we have proved that the quotient of an interpretation with respect to a given domain is a domain abstracting the starting domain, provided the associated relation is additive.

The quotient of `Prop` with respect to `GR` is `Prop` itself. This is due to the fact that none of the formulas of `Prop` is irrelevant for the computation of groundness [10].

Lemma 4.4 *Let* $[f_1, U], [f_2, U] \in$ `Prop`*. If* $f_1 \neq f_2$*, there exists a derived operator* t *using the operations of* **Prop***, such that* $\alpha_{PrGr}(t([f_1, U])) \neq \alpha_{PrGr}(t([f_2, U]))$.

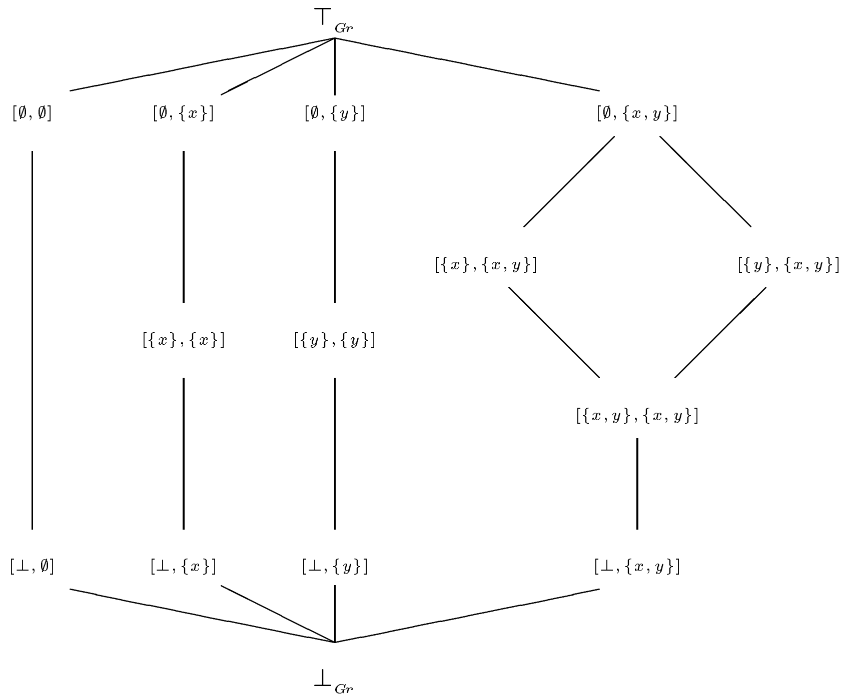

Figure 4: The domain GR for $\mathbf{V}=\{x, y\}$

Corollary 4.5 *r_{Pr} is the identity on* Prop*, and thus it is obviously additive.*

Theorem 4.6 $\mathcal{Q}_{\text{GR}}(\texttt{Prop}) = \texttt{Prop}$.

proof: Follows immediately from Corollary 4.5 and Theorem 3.6. □

4.3.3 The Quotient of Sharing with respect to GR

Let r_{Sh} be the relation on Sharing associated to GR. It will be shown that, differently from what we just saw for Prop, the equivalence classes of r_{Sh} are not singletons. However, r_{Sh} is additive and thus $\mathcal{Q}_{\text{GR}}(\texttt{Sharing})$ exists. Moreover, we will show that $\mathcal{Q}_{\text{GR}}(\texttt{Sharing})$ is isomorphic to Def. Because of this fact, the comparison between $\mathcal{Q}_{\text{GR}}(\texttt{Sharing})$ and $\mathcal{Q}_{\text{GR}}(\texttt{Prop})$ will be extremely simple. Some results of this section need rather technical proofs. For the readability sake these proofs are given in the Appendix.

That Sharing expresses information about groundness is well-known, cf. [16]. A formalization of this intuition was first attempted in [9] where it is shown that between Def and Sharing there is a Galois connection. The following even stronger result has been shown recently in [7].

Theorem 4.7 *The domain* Def *abstracts* Sharing *with the following abstraction and concretization functions:*
For $[A, U] \in$ Sharing*, let*

$$\mathcal{C}([A, U]) = \wedge\{\wedge W_1 \to x \mid W_1 \subseteq U, x \in U, \text{ and } \forall N \in A : x \in N \Rightarrow (W_1 \cap N) \neq \emptyset\}.$$

$$\alpha_{ShDf}([A, U]) = \begin{cases} [\mathsf{F}, U] & \text{if } A = \emptyset \\ [\mathcal{C}([A, U]), U] & \text{otherwise} \end{cases}$$

$$\gamma_{DfSh}([f, U]) = [\{U \setminus M \mid M \models f\}, U]$$

This Galois insertion is coherent wrt Rsub*.*

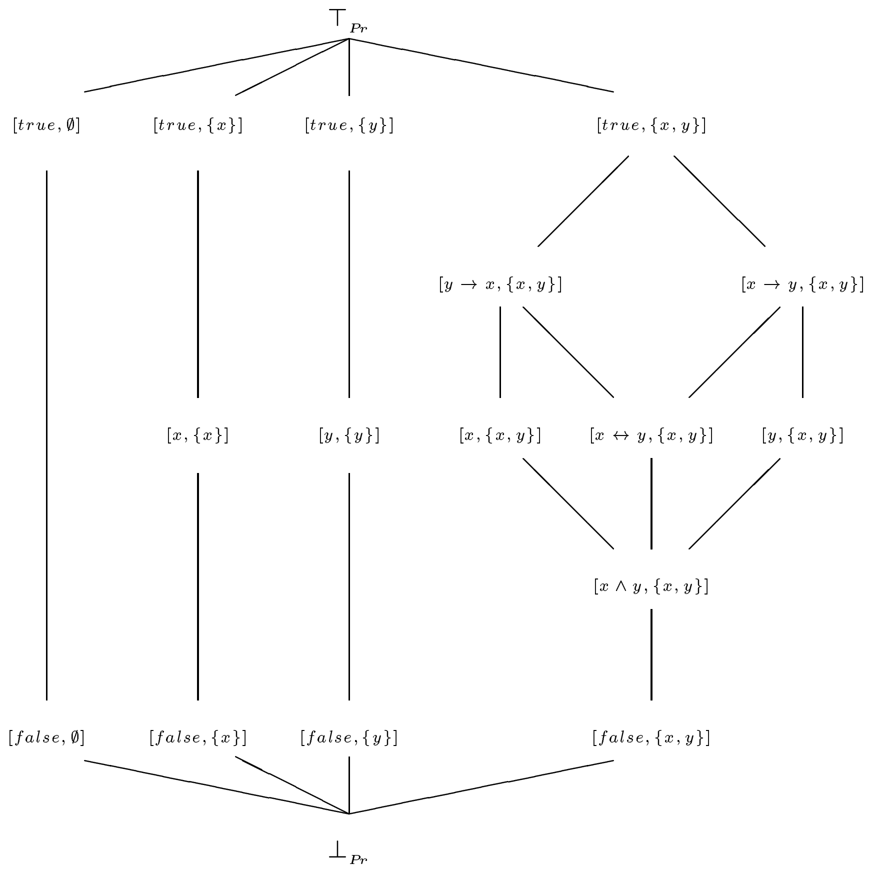

Figure 5: The quotient $\mathcal{Q}_{\texttt{GR}}(\texttt{Sharing})$ for $\mathbf{V}=\{x, y\}$

According to Definition 3.1, the relation on $\texttt{Sharing}$ associated to $\texttt{GR}$ is defined as follows. Let $S_1, S_2 \in \texttt{Sharing}$, and let t be any derived operator on the operations of **Sharing**.

$$(S_1, S_2) \in r_{Sh} \;\Leftrightarrow\; \alpha_{ShGr}(t(S_1)) = \alpha_{ShGr}(t(S_2)).$$

The following Theorem characterizes r_{Sh} using the abstractions of the elements of $\texttt{Sharing}$ into $\texttt{Def}$. Its proof is in the Appendix.

Theorem 4.8 *Let $S_1, S_2 \in \texttt{Sharing}$, where $S_1, S_2 \in \texttt{Sharing}$.*

$$(S_1, S_2) \in r_{Sh} \;\Leftrightarrow\; \alpha_{ShDf}(S_1) = \alpha_{ShDf}(S_2).$$

The existence of the quotient $\mathcal{Q}_{\texttt{GR}}(\texttt{Sharing})$ is guaranteed by the following result.

Theorem 4.9 *r_{Sh} is additive.*

proof: Consider $X \subseteq r_{Sh}$. Let for $i \in [1, 2], X_i = \{S_i \mid (S_1, S_2) \in X\}$. We want to show that

$$(\sqcup_{Sh} X_1,\; \sqcup_{Sh} X_2) \in r_{Sh}$$

By Theorem 4.8,

$$(\sqcup_{Sh} X_1,\; \sqcup_{Sh} X_2) \in r_{Sh} \;\Leftrightarrow\; \alpha_{ShDf}(\sqcup_{Sh} X_1) = \alpha_{ShDf}(\sqcup_{Sh} X_2)$$

$$\begin{tikzcd}
& \texttt{GR} \arrow[dl] \arrow[dr] & \\
\mathcal{Q}_{\texttt{GR}}(\texttt{Sharing}) \arrow[rr] \arrow[d] & & \mathcal{Q}_{\texttt{GR}}(\texttt{Prop}) = \texttt{Prop} \arrow[d] \\
\texttt{Sharing} \arrow[dr] & & \texttt{Prop} \arrow[dl] \\
& \texttt{Rsub} &
\end{tikzcd}$$

Figure 6: Domain abstractions

Since, by Theorem 4.7, α_{ShDf} together with its adjoint, forms a Galois insertion, by Proposition 2.3, it is additive and therefore, the following holds:

$$\begin{array}{rcll}
\alpha_{ShDf}(\sqcup_{Sh} X_1) & = & \sqcup_{Df}\{\alpha_{ShDf}(x) \mid x \in X_1\} & \text{by the additivity} \\
& = & \sqcup_{Df}\{\alpha_{ShDf}(x) \mid x \in X_2\} & \text{by definition of } X_1 \text{ and } X_2 \\
& = & \alpha_{ShDf}(\sqcup_{Sh} X_2) & \text{again by the additivity.}
\end{array}$$

□

The following Theorem characterizes the quotient of `Sharing` wrt `GR`.

Theorem 4.10 $\mathcal{Q}_{\texttt{GR}}(\texttt{Sharing}) = \texttt{Def}$

proof: It is easy to show that the sets `Def` and $\{\sqcup_{Sh}[S]_{r_{Sh}} \mid S \in \texttt{Sharing}\}$ are isomorphic: the abstraction α_{ShDf} is a bijection that preserves the orders of the two sets. In fact, on the one hand, if $[A_1, U] \sqsubseteq_{Sh} [A_2, U]$ then $A_1 \subseteq A_2$ and thus $\mathcal{C}([A_1, U]) \models \mathcal{C}([A_2, U])$ (see Theorem 4.7 for the definition of $\mathcal{C}$). On the other hand, if $f_1 \models f_2$, with $[f_1, U]$ and $[f_2, U]$ in `Def`, then f_1 has less models than f_2, and thus,

$$\{U \setminus M \mid M \models f_1\} \subseteq \{U \setminus M \mid M \models f_2\}.$$

Hence, $\gamma_{DfSh}([f_1, U]) \sqsubseteq_{Sh} \gamma_{DfSh}([f_2, U])$.

□

Figure 5 depicts, for the case that $\mathbf{V}$=$\{x, y\}$, the quotient of `Sharing` with respect to `GR`, which is the domain `Def`. Observe that the elements $[\{\{x\},\{y\},\{x,y\}\},\{x,y\}]$ and $[\{\{x\},\{y\}\},\{x,y\}]$ belong to the same equivalence class $[true, \{x, y\}]$ via r_{Sh}. The only difference between these two elements is that the first one represents also substitutions σ such that σx and σy share a common variable. However, this distinction is irrelevant when considering only groundness computation. In fact, both elements simply say that x and y are completely unrelated with respect to groundness.

4.4 Comparison of Prop and Sharing wrt `GR`

We can finally compare the two interpretations **Prop** and **Sharing** using the new theory developed in Section 3.

Theorem 4.11 **Prop** *is strictly more precise than* **Sharing** *with respect to the domain* `GR` *representing groundness.*

proof: It suffices to show that Theorems 3.9 and 3.10 are applicable, that is, we have to show that all the assumptions **ASS** (a)–(e) of Section 3.2 are satisfied:

- *Point(a) and (b):* have been stated in Sections 4.2.2, 4.2.3 and 4.3.1.
- *Point(c):* is shown in Theorems 4.6 and 4.10.
- *Point(d):* is stated in Lemmas 4.2 and 4.3.
- *Point(e):* is stated in Section 4.2.2.

$\square$

5 Conclusions

In this paper we addressed the problem of exactly characterizing the part of an abstract domain which is useful for the computation of a given property. To this end, we introduced the notion of quotient of an abstract interpretation. We showed that the comparison of abstract interpretations wrt a common property can be performed by comparing their quotients wrt that property. As an example, we applied this technique to the comparison of two well-known abstract interpretations for logic programs: **Prop** and **Sharing**.

Other algebraic operators on abstract domains and abstract interpretations have been proposed in the literature, namely the reduced product [6], the open product [12], the powerset [15], and the complement [7]. An interesting subject for future work is the study of the interaction between the quotient and these operators. For instance one may wonder whether the quotient of a reduced product is the reduced product of the quotients and also whether the quotient of the powerset of a domain D is the powerset of the quotient of D.

Another question that deserves further study is what one can do for comparing two domains when the present framework cannot be applied, for instance, when one of the associated relations is not additive. In this case it may still be possible to perform a comparison by lifting the domains to their powersets and comparing their quotients. In fact, quotients always exist for domains obtained through the powerset operation.

References

[1] Abramski, S., and Hankin, C., Abstract Interpretation of Declarative Languages. Ellis Horwood, Chichester, U.K., 1987.

[2] Armstrong, T., and Marriott,K., and Schachte, P., and Søndergaard, H., "Boolean Functions for Dependency Analysis: Algebraic Properties and Efficient Representations". *Proc. Static Analysis Symposium*, LNCS 864:266–280. Springer-Verlag, 1994.

[3] Barbuti, R., Giacobazzi, R., and Levi, G., "A General Framework for Semantics-based Bottom-up Abstract Interpretation of Logic Programs". *ACM TOPLAS* 15(1):133–181, 1993.

[4] Codish M., Mulkers A., Bruynooghe M., Garcia de la Banda M., Hermenegildo M., "Improving abstract interpretations by combining domains". *ACM TOPLAS* vol. 17(1) pp. 28-44 (1995).

[5] Cousot, P. and Cousot, R., "Abstract interpretation: a unified framework for static analysis of programs by construction of approximation of fixpoints." *Proc. Fourth ACM POPL*, 1977.

[6] Cousot, P. and Cousot, R., "Abstract Interpretation and applications to logic programs". *Journal of Logic Programming*, 13, 1992.

[7] A. Cortesi, G. File', R. Giacobazzi, C. Palamidessi, and F . Ranzato, "Complementation in Abstract Interpretation", *ACM TOPLAS*, vol. 19(1), pp.7-47, 1997.

[8] Cortesi, A., Filè, G. and Winsborough W., "Prop revisited: Propositional formula as abstract domain for groundness analysis." *Proc. Sixth IEEE LICS*, Amsterdam 1991, pp. 322-327.

[9] Cortesi, A., Filè, G. and Winsborough W., "Comparison of Abstract Interpretations." *Proc. 19th Int. Colloquium on Automata, Languages and Programming ICALP'92*, Springer Verlag LNCS 623, pp. 523-534 (1992)

[10] Cortesi, A., Filè, G. and Winsborough W., "Optimal Groundness Analysis using Propositional Logic." *Journal of Logic Programming*, vol. 27(2), pp.137-167. (1996)

[11] Cortesi, A., Filè, G., "Optimalities in the Abstract Domain Sharing." Manuscript, University of Padova, 1996.

[12] Cortesi, A., Van Hentenryck P. and Le Charlier B., "Combinations of Abstract Domains for Logic Programming". *Proc. 21th ACM POPL*, Portland 1994, ACM Sigplan Notices, 29(6), pp.227-239, (1994).

[13] Davey, B.A., and Priestley, H.A., Introduction to Lattices and Order. Cambridge University Press. 1990.

[14] Dart, P.W., "On derived dependencies and connected databases." *Journal of Logic Programming*, vol. 11, pp.163-188 (1991).

[15] File,G., and Ranzato F., "Improving Abstract Interpretations by Systematic Lifting to the Powerset." *Proc. ILPS '94*, MIT Press, 1994, pp.655-669.

[16] Jacobs, D. and Langen, A., "Accurate and efficient approximation of variable aliasing in logic programs." *Journal of Logic Programming*, vol.13, pp.291–314, 1992.

[17] Lassez,J.L., Maher, M.J., and Marriott, K.G., "Unification Revisited". In J.Minker (ed.) *Foundation of Deductive Databases and Logic Programming*, 1986.

[18] Marriott, K., "Frameworks for Abstract Interpretation", *Acta Informatica*, vol. 30(2), pp.103–129",1993.

[19] Marriott, K. and Søndergaard, H., "Notes for a tutorial on abstract interpretation of logic programs." Proc. NACLP. Cleveland, 1989.

[20] Marriott, K., Søndergaard, H. and Jones, N., "Denotational Abstract Interpretation of Logic Programs." *ACM TOPLAS*, Vol.16, pp.607–648, 1994.

[21] Marriott, K. and Søndergaard, H., "Precise and Efficient Groundness Analysis for logic programs." *ACM LOPLAS*, Vol.2 (1-4), pp. 181–196, 1993.

[22] Naur, P., "Checking of operands types in Algol compilers." BIT vol. 5, pp.151-163, 1965.

[23] Van Hentenryck P., Cortesi, A. and Le Charlier B., "Evaluation of the domain Prop", *Journal of Logic Programming*, vol.23(3), pp.237-278 (1995).

6 Appendix

The Appendix consists in four parts. In the first three, we formally define the operations in **Rsub**, **Prop**, and **Sharing**. Then, we show some technical lemmas that lead to the proof of Theorem 4.8.

6.1 Operations in Rsub

Let E be a set of term equations. If a substitution σ makes $\sigma(t_1)$ syntactically identical to $\sigma(t_2)$ for each $(t_1 = t_2) \in E$, σ is called a *unifier* of E. A *most general unifier* of E is a unifier σ of E that is more general than any other unifier of E. We denote by $mgu(E)$ any idempotent most general unifier of E. It is not necessary to specify which most general unifier is considered, because, from the relationship existing among the idempotent most general unifiers of a given set of equations, [17], it is immediate to see that each of them carries the same information about the properties we are interested in, namely, variable groundness and sharing.

A set of equations E is in *solved form* if it has the form $\{x_1 = t_1, \ldots, x_n = t_n\}$, where each x_i is a distinct variable occurring in none of the terms t_j. Given a set of equations $E = \{x_1 = t_1, \ldots, x_n = t_n\}$ in solved form, the substitution $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$ is an idempotent most general unifier of E; we denote E by $Eq(\sigma)$.

Least upper bound The operation $\sqcup_{Rs}$, which produces the least upper bound of two elements of $\mathtt{Rsub}$, is as follows: for any $k \in \mathtt{Rsub}$, $\top_{Rs} \sqcup_{Rs} k = \top_{Rs}$, $\bot_{Rs} \sqcup_{Rs} k = k$, for the other elements,

$$[\Sigma_1, U_1] \sqcup_{Rs} [\Sigma_2, U_2] = \begin{cases} [\Sigma_1 \cup \Sigma_2, U_1] & \text{if } U_1 = U_2 \\ \top_{Rs} & \text{otherwise} \end{cases}$$

Projection The concrete projection $\pi_{Rs} : \mathtt{Rsub} \times FP(\mathbf{V}) \to \mathtt{Rsub}$ maps $([\Sigma, U_1], U_2)$ to $[\Sigma, U_1 \cap U_2]$.

Thus projection only changes the second component leaving the first one unchanged. One may think that the fact that the projection does not eliminate from the sustitutions the variables that are projected out, may cause problems of variable capture. As usual, variable captures can be avoided using appropriate renamings.

Unification In order to define the concrete unification $\mathbf{U}_{Rs}$, it is convenient to introduce first the following function $\mathbf{u}_{Rs}$:

$$\begin{array}{rrcl} \mathbf{u}_{Rs} : & Subst \times Subst \times Subst & \longrightarrow & Subst \\ & (\sigma_1, \sigma_2, \delta) & \longmapsto & mgu(Eq(\sigma_1) \cup Eq(\sigma_2) \cup Eq(\delta)) \end{array}$$

$\mathbf{U}_{Rs}$ is strict: if either of the first two arguments is $\bot_{Rs}$, the result is $\bot_{Rs}$. Otherwise, if one of these is $\top_{Rs}$, the result is $\top_{Rs}$. The other cases are as follows:

$$\begin{array}{rrcl} \mathbf{U}_{Rs} : & \mathtt{Rsub} \times \mathtt{Rsub} \times Subst & \longrightarrow & \mathtt{Rsub} \\ & ([\Sigma_1, U_1],\ [\Sigma_2, U_2], \delta) & \longmapsto & [\{\mathbf{u}_{Rs}(\sigma_1, \sigma_2, \delta) | \sigma_1 \in \Sigma_1 \ \&\ \sigma_2 \in \Sigma_2\}, U_1 \cup U_2 \cup Var(\delta)] \end{array}$$

6.2 Operations in Prop

Least upper bound For all $d \in \mathtt{Prop}, \top_{Pr} \sqcup_{Pr} d = \top_{Pr}$ and $\bot_{Pr} \sqcup_{Pr} d = d$; for the other elements,

$$[f_1, U_1] \sqcup_{Pr} [f_2, U_2] = \begin{cases} [f_1 \vee f_2, U_1] & \text{if } U_1 = U_2 \\ \top_{Pr} & \text{otherwise} \end{cases}$$

Projection The abstract projection π_{Pr} amounts to existentially quantifying a formula [8, 20]. The existential quantification of a propositional formula obeys $\exists x.f \equiv f\langle x/\mathsf{T}\rangle \vee f\langle x/\mathsf{F}\rangle$.

$$\pi_{Pr}([f, U], V) = [\exists (U \setminus V).\ f, U \cap V].$$

Unification The abstract unification is obtained by means of logical conjunctions. For $\delta = \{x_i/t_i \mid 1 \leq i \leq n\} \in \mathit{Subst}$, let $\varphi_\delta = \wedge\{x_i \leftrightarrow (\wedge Var(t_i)) \mid 1 \leq i \leq n\}$.

$$\begin{array}{rrcl}\mathbf{U}_{Pr}: & \texttt{Prop} \times \texttt{Prop} \times \mathit{Subst} & \longrightarrow & \texttt{Prop} \\ & ([f_1, U_1], [f_2, U_2], \delta) & \longmapsto & [f_1 \wedge f_2 \wedge \varphi_\delta,\ U_1 \cup U_2 \cup \mathit{Var}(\delta)].\end{array}$$

6.3 Operations in Sharing

Least upper bound The least upper bound of any two nontrivial elements $[A_1, U_1]$ and $[A_2, U_2]$ is defined by

$$[A_1, U_1] \sqcup_{Sh} [A_2, U_2] = \begin{cases} [A_1 \cup A_2, U_1] & \text{if } U_1 = U_2 \\ \top_{Sh} & \text{otherwise.} \end{cases}$$

Projection The projection on **Share** is the identity on the bottom and top elements. In the other cases it is defined by means of set intersection.

$$\begin{array}{l}\pi_{Sh}: \texttt{Sharing} \times \mathit{FP}(\mathbf{V}) \longrightarrow \texttt{Sharing} \\ ([S_1, U_1], U_2) \longmapsto [\{A \cap U_2 \mid A \in S_1\}, U_1 \ \cap\ U_2]\end{array}$$

Unification In order to define the abstract unification function $\mathbf{U}_{Sh}$ for the `Sharing` domain, we need the following auxiliary functions [16]:

- The *closure under union* of $A \in \wp(\wp(\mathbf{V}))$, denoted A^*, is the smallest superset of A satisfying $X \in A^* \wedge Y \in A^* \rightarrow (X \cup Y) \in A^*$.
- The part of $A \in \wp(\wp(\mathbf{V}))$ that is *relevant* to a term t, denoted $rel(A, t)$, is the set $\{S \in A \mid Var(t) \cap S \neq \emptyset\}$.
- If $A, A' \in \wp(\wp(\mathbf{V}))$ then $A \otimes A' = \{(S \cup S') \mid S \in A \textit{ and } S' \in A'\}$.
- The basic unification step is performed by

$$\mathbf{u}_{Sh}: \wp(\wp(\mathbf{V})) \times \mathit{Subst} \longrightarrow \wp(\wp(\mathbf{V}))$$

$$\begin{array}{c}\forall A_0 \in \wp(\wp(\mathbf{V})),\ \forall \delta \in \mathit{Subst}, \delta = \{x_1/t_1, \ldots, x_m/t_m\} \\ \mathbf{u}_{Sh}(A_0, \delta) = \mathbf{amgu}.\mathbf{args}([x_1, \ldots, x_m], [t_1, \ldots, t_m], A_0)\end{array}$$

$$\begin{array}{c}\mathbf{amgu}.\mathbf{args}([\,], [\,], B) = B \\ \mathbf{amgu}.\mathbf{args}([x_1|\overline{x}], [t_1|\overline{t}], B) = \mathbf{amgu}.\mathbf{args}(\overline{x}, \overline{t}, \mathbf{amgu}(x_1, t_1, B)) \\ \mathbf{amgu}(x, t, B) = (B \setminus (rel(x, B) \cup rel(t, B))) \cup (rel(x, B) \otimes rel(t, B))^*\end{array}$$

- The backward/forward unification $\mathbf{U}_{Sh}: \texttt{Sharing} \times \texttt{Sharing} \times \mathit{Subst} \longrightarrow \texttt{Sharing}$ is defined as follows. Let
  $[A, U], [A', U'] \in$ **Sharing**, with $U \cap U' = \emptyset$, and let $\delta \in \mathit{Subst}$ such that $Var(\delta) \subseteq U \cup U'$.

$$\mathbf{U}_{Sh}([A, U], [A', U'], \delta) = [\mathbf{u}_{Sh}(A \cup A', \delta), U \cup U'].$$

6.4 Technical Results

The goal of this section is to show Theorem 4.8. Some preliminary results are necessary.

Lemma 6.1 *Let* $S_1, S_2 \in$ `Sharing`.

$$\alpha_{ShDf}(S_1) = \alpha_{ShDf}(S_2) \ \Rightarrow \alpha_{ShGr}(S_1) = \alpha_{ShGr}(S_2)$$

proof: It suffices to observe that $\alpha_{ShGr} = \alpha_{DfGr} \circ \alpha_{ShDf}$, cf. section 4.3.1 for the definition of the abstraction functions. □

In what follows we will use the notion of $\mathcal{C}$ introduced in section 4.3.3. For the sake of clarity, we recall its definition below: for $[A, U] \in$ `Sharing`,

$$\mathcal{C}([A,U]) = \wedge\{\wedge W_1 \to x \mid W_1 \subseteq U, x \in U, \text{ and } \forall N \in A : x \in N \Rightarrow (W_1 \cap N) \neq \emptyset\}.$$

Recall also that $\alpha_{ShDf}([A,U]) = [\mathcal{C}([A,U]), U]$.

The following lemma plays a central role in this section: it expresses the $\mathcal{C}$ meaning of the result of the $\mathbf{U}_{Sh}$ operation in terms of the conjunction of the $\mathcal{C}$ meaning of its arguments.

Lemma 6.2 *Let* $S_1 = [A_1, U_1]$, $S_2 = [A_2, U_2] \in$ `Sharing`*, with* $U_1 \cap U_2 = \emptyset$*, and let* $\delta = \{x_1/t_1, \ldots, x_n/t_n\}$ *be any substitution in Subst with* $Var(\delta) \subseteq U_1 \cup U_2$*. Let* $\varphi_\delta = \wedge\{x_i \leftrightarrow (\wedge Var(t_i)) \mid 1 \leq i \leq n\}$*.*

$$\mathcal{C}(\mathbf{U}_{Sh}(S_1, S_2, \delta)) = \mathcal{C}(S_1) \wedge \mathcal{C}(S_2) \wedge \varphi_\delta$$

proof: We show that the relation holds for the case that $\delta = \{x/t\}$. The general result easily follows by applying the basis case n times.
Let in what follows, $\mathbf{U}_{Sh}(S_1, S_2, \delta) = S' = [A', U]$, where $U = U_1 \cup U_2$. Let also $A = A_1 \cup A_2$ and $S = [A, U]$.

- $\Rightarrow$ First, we show that $\mathcal{C}(S') \models \mathcal{C}(S_1) \wedge \mathcal{C}(S_2) \wedge (x \leftrightarrow \wedge Var(t))$. The fact that $\mathcal{C}(S') \models x \leftrightarrow \wedge Var(t)$ follows easily from the definition of $\mathbf{U}_{Sh}$. Consider $rel(A, x)$ and $rel(A, t)$. If one of them is empty, then A' contains no variable in $\{x\} \cup Var(t)$. Thus, $\mathcal{C}(S') \models x \wedge (\wedge Var(t))$ and, therefore, $\mathcal{C}(S') \models x \leftrightarrow \wedge Var(t)$.

  On the other hand, if both $rel(A, x)$ and $rel(A, t)$ are non empty, then, by definition of $\mathbf{u}_{Sh}$, A' satisfies the following condition: $\forall N \in A', \quad x \in N \Leftrightarrow Var(t) \cap N \neq \emptyset$. Thus, $\mathcal{C}(S') \models x \leftrightarrow \wedge Var(t)$.

  Let us now prove that $\mathcal{C}(S') \models \mathcal{C}(S_1)$. The proof for $\mathcal{C}(S_2)$ is analogous. Consider any definite formula $f = \wedge W \to y$ such that,

  $$(1) \quad \forall N \in A_1.\ y \in N \quad \Rightarrow \quad W \cap N \neq \emptyset$$

  It is easy to see that this condition is satisfied also if we replace A_1 with A': each element $N \in A'$ is either an element of A_1 (that satisfies (1) by assumption), or it is an element of A_2 (that satisfies (1) trivially because $U_1 \cap U_2 = \emptyset$ and $\{y\} \cup W \subseteq U_1$), or it is the union of some elements of A_1 and of A_2 ((1) is satisfied by the above arguments).

- $\Leftarrow$ We will show that $\mathcal{C}(S) \wedge \varphi_\delta \models \mathcal{C}(S')$. This proves the thesis. In fact, $\mathcal{C}(S) = \mathcal{C}(S_1) \wedge \mathcal{C}(S_2)$, because in A the elements of A_1 and A_2 do not interact since they are pairwise disjoint.

  We proceed by contradiction. Assume that there is $f = \wedge W \to y$ such that $\mathcal{C}(S') \models f$, but $\mathcal{C}(S) \wedge \varphi_\delta \not\models f$. Then it must be the case that $\mathcal{C}(S) \not\models f$ and thus, there must be $B \in A$ such that $y \in B$, but $W \cap B = \emptyset$. Surely, $B \cap Var(x = t) \neq \emptyset$, otherwise $B \in A'$, that contradicts the hypothesis. Thus, $B \in rel(A, x) \cup rel(A, t)$. From this, it follows that:

  $$(2) \quad \mathcal{C}(S) \models ((\wedge W) \wedge x \wedge (\wedge Var(t))) \to y$$

  because for each $B \in A$, if $y \in B$, but $W \cap B = \emptyset$, then $(\{x\} \cup Var(t)) \cap B \neq \emptyset$.

  Assume that $B \in rel(A, t)$, the case that $B \in rel(A, x)$ is analogous. By definition of $\mathbf{u}_{Sh}$, A' contains $B \otimes rel(A, x)$. By the initial assumption, $\forall R \in rel(A, x)$ it must be that $R \cap W \neq \emptyset$. From this it follows that

  $$(3) \quad \mathcal{C}(S) \models \wedge W \to x$$

  Thus, from (2) and (3) we have

  $$\mathcal{C}(S) \wedge (x \leftrightarrow \wedge(Var(t))) \models [(x \leftrightarrow \wedge(Var(t)))] \wedge [\wedge W \to x] \wedge [(\wedge W) \wedge x \wedge (\wedge Var(t)) \to y],$$

from which one easily obtains

$$\mathcal{C}(S) \wedge (x \leftrightarrow \wedge(Var(t))) \models \wedge W \rightarrow y$$

□

In the following three lemmas, we will show, for each one of the three operations of **Sharing**, that the $\mathcal{C}$ meaning of its result is completely determined by the $\mathcal{C}$ meaning of its arguments.

Lemma 6.3 (unification) *Let* $S_1 = [A_1, U], S_2 = [A_2, U]$ *and* $S' = [A', U']$ *be elements of* `Sharing` *such that* $U \cap U' = \emptyset$. *Let also* $\delta \in Subst$ *such that* $Var(\delta) \subseteq U \cup U'$.

$$\mathcal{C}(S_1) = \mathcal{C}(S_2) \;\Rightarrow \mathcal{C}(\mathbf{U}_{Sh}(S', S_1, \delta)) = \mathcal{C}(\mathbf{U}_{Sh}(S', S_2, \delta))$$

proof: By Lemma 6.2, we know that:

$$\begin{array}{lcl} \mathcal{C}(\mathbf{U}_{Sh}(S', S_1, \delta)) & = & \mathcal{C}(S') \wedge \mathcal{C}(S_1) \wedge \varphi_\delta \\ \text{by assumption} & = & \mathcal{C}(S') \wedge \mathcal{C}(S_2) \wedge \varphi_\delta \\ & = & \mathcal{C}(\mathbf{U}_{Sh}(S', S_2, \delta)) \end{array}$$

□

Lemma 6.4 (projection) *Let* $S_1 = [A_1, U], S_2 = [A_2, U] \in$ `Sharing`*, and* U' *be a finite set of variables.*

$$\mathcal{C}(S_1) = \mathcal{C}(S_2) \;\Rightarrow\; \mathcal{C}(\pi_{Sh}(S_1, U')) = \mathcal{C}(\pi_{Sh}(S_2, U'))$$

proof: Straightforward. □

Lemma 6.5 (lub) *Let* $S_1 = [A_1, U], S_2 = [A_2, U]$ *be elements of* `Sharing` *with* $A_1, A_2 \neq \emptyset$*, and let* $S' = [A', U'] \in$ `Sharing`.

$$\mathcal{C}(S_1) = \mathcal{C}(S_2) \;\Rightarrow\; \mathcal{C}(S_1 \sqcup_{Sh} S') = \mathcal{C}(S_2 \sqcup_{Sh} S')$$

proof: Recall that, for any $S = [A, U] \in$ `Sharing`, $\alpha_{ShDf}(S) = [\mathcal{C}(S), U]$. Thus, we can use the additivity of α_{ShDf}, obtaining the following:

$$\alpha_{ShDf}(S_1 \sqcup_{Sh} S') = \alpha_{ShDf}(S_1) \sqcup_{Df} \alpha_{ShDf}(S') = \alpha_{ShDf}(S_2) \sqcup_{Df} \alpha_{ShDf}(S') = \alpha_{ShDf}(S_2 \sqcup_{Sh} S')$$

which shows the thesis.

□

Finally, we show Theorem 4.8.

Theorem 4.8 *Let* $S_1, S_2 \in$ `Sharing`*, where* $S_1 = [A_1, U_1], S_2 = [A_2, U_2]$ *and* $A_1, A_2 \neq \emptyset$.

$$(S_1, S_2) \in r_{Sh} \;\Leftrightarrow\; \alpha_{ShDf}(S_1) = \alpha_{ShDf}(S_2).$$

proof:

$\Rightarrow$ ¿From the definition of r_{Sh} it follows immediately that $U_1 = U_2$: it is sufficient to consider the empty sequence of operations to obtain

$$([A_1, U_1], [A_2, U_2]) \in r_{Sh} \;\Rightarrow\; \alpha_{ShGr}([A_1, U_1]) = \alpha_{ShGr}([A_2, U_2]) \;\Rightarrow\; U_1 = U_2.$$

It remains to show that $\mathcal{C}(S_1) = \mathcal{C}(S_2)$. Assume the converse. This means that there exists a definite formula $\psi = \wedge W \rightarrow x$ such that $\mathcal{C}(S_1) \models \psi$ and $\mathcal{C}(S_2) \not\models \psi$ (or vice versa). Notice that this means that there exists $N \in A_2$ such that $x \in N$ but $W \cap N = \emptyset$.

Consider $S_0 = [\{\emptyset\}, U_0]$ with $U_0 \cap U_1 = \emptyset$ (and thus $U_0 \cap U_2 = \emptyset$). Call $U = U_0 \cup U_1$. Consider also the substitution $\delta = \{x/a \mid x \in W\}$. Let

$$\mathbf{U}_{Sh}(S_0, S_1, \delta)) = [R_1, U] \quad \text{and} \quad \mathbf{U}_{Sh}(S_0, S_2, \delta)) = [R_2, U]$$

By the definition of $\mathbf{U}_{Sh}$, $R_1 = A_1 \setminus \{H \in A_1 \mid H \cap W \neq \emptyset\}\}$. Thus, by the hypothesis, $x \notin \cup R_1$ and thus, by definition of α_{ShGr} (cf. Section 4.3.1), x is in the first component of $\alpha_{ShGr}([R_1, U])$.
On the other hand, $R_2 = A_2 \setminus \{H \in A_2 \mid H \cap W \neq \emptyset\}$ and thus $N \in R_2$ from which, $x \in \cup R_2$. ¿From this it follows that x does not belong to the first component of $\alpha_{ShGr}([R_2, U])$. Thus we arrived to a contraddiction of our initial hypothesis.

$\Leftarrow$ Since, by definition, cf. Theorem 4.7,

$$\alpha_{ShDf}(S_1) = \alpha_{ShDf}(S_2) \ \Leftrightarrow\ U_1 = U_2 \text{ and } \mathcal{C}(S_1) = \mathcal{C}(S_2)$$

it suffices to show that for any derived operator t on $\{\sqcup_{Sh}, \pi_{Sh}, \mathbf{U}_{Sh}\}$,

$$\mathcal{C}(S_1) = \mathcal{C}(S_2) \ \wedge\ U_1 = U_2 \ \Rightarrow\ \alpha_{ShGr}(t(S_1)) = \alpha_{ShGr}(t(S_2))$$

By structural induction on the derived operator t, using Lemmas 6.3, 6.4, and 6.5, one can show that $\mathcal{C}(t(S_1)) = \mathcal{C}(t(S_2))$. ¿From this it follows that

$$(1) \qquad \alpha_{ShDf}(t(S_1)) = \alpha_{ShDf}(t(S_2))$$

because $U_1 = U_2$ by hypothesis, and the same derived operator t is applied to S_1 and S_2 and thus the two results will have equal second components. By Lemma 6.1, (1) implies the thesis.

□