Università di Pisa

Dipartimento di Informatica

Technical Report : TR-97-13

Functional Compositions of Abstract Interpretations

Roberto Giacobazzi, Francesco Ranzato

July 02, 1997

ADDR: Corso Italia 40,56125 Pisa,Italy. TEL: +39-50-887111. FAX: +39-50-887226

Functional Compositions of Abstract Interpretations

Roberto Giacobazzi
Dipartimento di Informatica
Università di Pisa
Corso Italia 40, 56125 Pisa, Italy
giaco@di.unipi.it

Francesco Ranzato
Dipartimento di Matematica Pura ed Applicata
Università di Padova
Via Belzoni 7, 35131 Padova, Italy
franz@math.unipd.it

Abstract

In the context of standard abstract interpretation theory, we define and study a systematic operator of reduced relative power for composing functionally abstract domains. The reduced relative power of two abstract domains D_1 (the exponent) and D_2 (the base) consists in a suitably defined lattice of monotone functions from D_1 to D_2, called dependencies, and it is a generalization of the Cousot and Cousot operator of reduced (cardinal) power. The relationship between the reduced relative power and Nielson's tensor product is also investigated. The case of autodependencies (when base and exponent are the same domain) turns out to be particularly interesting: Under certain hypotheses, the domain of autodependencies corresponds to a suitable powerset-like completion of the base abstract domain, providing a compact lattice-theoretic representation for autodependencies. The usefulness of the reduced relative power is shown by a number of applications to abstract domains used both for program analysis and for semantics design. Notably, we prove that the domain *Def* for logic program ground-dependency analysis can be characterized as autodependencies of a standard more abstract domain representing plain groundness only, and we show how to exploit reduced relative power to systematically derive compositional logic program semantics.

Keywords: Abstract interpretation, abstract domain, reduced relative power, separated abstract domain, logic program analysis and semantics.

1 Introduction

One of the most appealing features of Cousot and Cousot's abstract interpretation theory is its ability to provide systematic methodologies to compositionally design complex abstract domains from more simple ones (cf. [20]). New abstract domains can be systematically obtained by combining simpler ones or by lifting them by adding new information. Such operators on abstract domains, that, following a general approach pursued in [28, 37], we call domain *refinements*, are devoted to enhance the precision of domains and provide high-level facilities to tune a static program analysis in accuracy and cost.

Abstract domain refinements turn out to be particularly useful in transforming an *attribute independent* analysis into a more precise *relational* analysis (cf. [42]). Let us see an example involving logic programs. Consider the following classical logic program defining an ancestor relation *anc*, where the relation R is specified in some database. Assume that a query $R(X,Y)$ succeeds only if both its arguments X and Y are ground.

$$\begin{array}{llll} c_1: & anc(X,Z) & :- & anc(X,Y), R(Y,Z). \\ c_2: & anc(X,Y) & :- & X = Y. \end{array}$$

A logic program groundness analysis is traditionally intended to statically discover whether the argument of the head of a clause will be ground in any of its successful derivations. The most obvious domain for a plain groundness analysis is a two point lattice $Ground = \{g, ?\}$, where $g \leq ?$ (cf. [43]). Here, a type judgment $X : g$ says that X is bound to a ground term, while $X : ?$

represents uncertainty, i.e. it says that X is bound to any term. If we assume that $\langle R(X, Y), X : g \wedge Y : g\rangle$ represents the successful computations for R, then an obvious analysis using *Ground* yields $\langle anc(X, Z), X : ? \wedge Y : g\rangle$ for c_1 and $\langle anc(X, Y), X : ? \wedge Y : ?\rangle$ for c_2. Evidently, such an analysis is very rough, as it is unable to discover that, in c_1, X will be ground in any successful derivation. This is due to the fact that no relational information is kept about the arguments of the program clause c_2. In fact, a relation between X and Y is established by unifying c_2 with $anc(X, Y)$ in the body of c_1. Thus, this relation propagates the groundness from Y (which is derived from R) to X. In order to solve this problem, a new abstract domain, called *Def*, has been introduced in the beginning of '90 (cf. [47]). *Def* consists of definite propositional formulae. For instance, an implicational formula $X : g \leftrightarrow Y : g$ represents the relational information that X is ground iff Y is ground. By using *Def*, we get a more precise groundness analysis, namely $\langle anc(X, Y), X : g \wedge Y : g\rangle$ for c_1 and $\langle anc(X, Y), X : g \leftrightarrow Y : g\rangle$ for c_2. In particular, this analysis is now able to detect that any successful derivation for the first clause will have both arguments of *anc* ground. However, although *Def* is a well understood domain for groundness analysis, it is not yet clear whether *Def* could be automatically obtained by a general procedure for building relational domains from nonrelational ones.

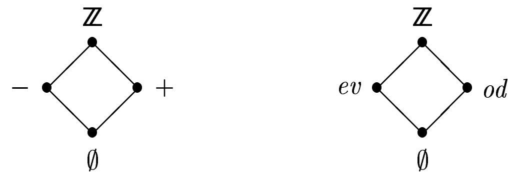
Several proposals have been put forward in the literature to systematically build relational domains either by composition of simpler (possibly nonrelational or attribute independent) ones, or by some domain completion (see [21]). Notable examples are Cousot and Cousot's *reduced* (*cardinal*) *power* in [20] and Nielson's *tensor product* in [53].

Reduced Relative Power. This paper originated by observing that the relational domain *Def* could be systematically derived from *Gr* as an instance of a generic procedure of domain refinement, which is a generalization of Cousot and Cousot's reduced cardinal power of abstract domains.

Among the various abstract domain refinements existing in the literature, Cousot and Cousot's reduced power is probably the less known one. In fact, while for the well known *reduced product* and *disjunctive completion* a comprehensive literature is available, ranging from the most theoretical aspects (e.g. see [16, 20, 23, 36]) to applications (e.g. see [13, 21, 23, 29, 41, 51, 62]), as far as reduced power is concerned very little has been done after [20], both in theory and in applications.

Reduced cardinal power has been introduced by Cousot and Cousot in the very last section of their POPL'79 paper (cf. [20, Section 10.2]), as a systematic methodology for combining abstract domains. The basic idea was that, given a pair of abstract domains D_1 and D_2, a new domain can be systematically derived by considering the lattice of all monotone mappings from D_1 to D_2, i.e. the cardinal power with base D_2 and exponent D_1. Such functions aimed to represent a functional relation between program properties expressed by D_2 with those expressed by D_1. However, Cousot and Cousot's definition and related correctness result (cf. [20, Theorem 10.2.0.1]) were given in a particular context of a collecting semantics of program assertions, and successively Cousot and Cousot did not broaden their definition to a more general setting. In [20] collecting semantics is specified by a lattice of assertions which is a Boolean algebra. In this work, we carries on such an idea, and, by generalizing the Cousot and Cousot definition of reduced power, we introduce the notion of *reduced relative power* of abstract domains. The reduced relative power $D_1 \overset{\odot}{\longmapsto} D_2$ of two domains D_1 (the exponent) and D_2 (the base) is defined in a general and standard abstract interpretation setting, and it is parameteric with respect to a generic operation $\odot$ used to combine concrete denotations (hence the adjective "relative", while "reduced" stems from the usual process of reduction of abstract domains). $D_1 \overset{\odot}{\longmapsto} D_2$ consists of all the monotone functions from D_1 to D_2 having the shape $\lambda x.\alpha_2(d \odot \gamma_1(x))$, where d is any concrete denotation, γ_1 is the concretization map for D_1, and α_2 is the abstraction map for D_2. Such functions are called *dependencies*, with the aim of indicating that a dependency relation is established between the values of D_1 and D_2. Roughly speaking, the operation $\odot$ should be thought of as a kind of combinator of concrete denotations: The *glb* is a typical example of such an operation, although some nontrivial applications (as that presented in Section 7) may require a different and less restrictive concrete combinator. Intuitively, a dependency $\lambda x.\alpha_2(d \odot \gamma_1(x))$ encodes how the abstract domain D_2 is able to represent the reaction of the concrete value d whenever it is combined via $\odot$ with an object described by D_1. The main assumption guaranteeing the correctness of our definitions is that the concrete domain endowed with the operation $\odot$ gives rise to a weak form of quantale, called *semi-quantale*, which includes Boolean algebras as special cases. Provided that such hypothesis is satisfied, reduced relative power can be applied to any two abstract domains in the standard framework of abstract interpretation. As our applications show, it turns out that our generalization is necessary to handle a number of important situations, ranging from domains used in program analysis to semantics design, where Cousot and Cousot's reduced power would not be otherwise applicable.

Let us see a naïve example of reduced relative power, which in this case boils down into the more classical reduced power. Consider the following simple and well known abstract domains *Parity* and *Sign* for integer variable analysis. The objects in *Parity* and *Sign* have an obvious meaning as sets of integer numbers.



When the operator for combining concrete values is given by intersection of sets of integers, it is easy to see that the monotone function $\{\mathbb{Z} \mapsto ev, + \mapsto \emptyset, - \mapsto \emptyset, \emptyset \mapsto \emptyset\}$ from *Sign* to *Parity* is given by the dependency $\lambda x.\alpha_p(\{0\} \cap \gamma_s(x))$ in $Sign \xmapsto{\cap} Parity$. $\lambda x.\alpha_p(\{0\} \cap \gamma_s(x))$ represents the largest set of integers T which, once intersected with $\mathbb{Z}$ gives an even number, and, once intersected with the set of either positive or negative integers gives the empty set. Clearly, these constraints mean that the concretization of $\lambda x.\alpha_p(\{0\} \cap \gamma_s(x))$ is given by $\{0\}$.

Many important results are obtained in relating reduced relative power to some well known function spaces, like the lattices of additive functions and lower closure operators.

We investigate the relationship between reduced relative power and additive functions, and we give certain sufficient conditions guaranteeing that a reduced relative power $D_1 \xmapsto{\wedge} D_2$ (relatively to the concrete operation $\wedge$ of *glb*) is a sublattice of, or even is equal to, the lattice of all additive functions from D_1 to D_2. The interest in these results is twofold. First, the reduced relative power inherits some relevant lattice-theoretic properties of the lattice of additive functions. Second, our results permit to give a precise relationship between the operators of reduced relative power and Nielson's ([52, 53]) *tensor product* in abstract interpretation, a problem already recognized by Nielson in [52, pag. 124] for Cousot and Cousot's reduced power. In fact, under these hypotheses, additive functions and tensor product are isomorphic up to a certain form of duality.

When the base and exponent domains of a reduced relative power coincide, we speak of *autodependencies*. As far as meet-autodependencies (i.e. relatively to the concrete operation of *glb*) are concerned, we identify a class of abstract domains, termed *separated*, such that the lattice of meet-autodependencies of some abstract domain A is isomorphic to the lattice of all lower closure operators on A iff A is separated. Since the lattice of lower closure operators on A is always isomorphic to a suitable family of subsets of A (the family of dual-Moore-sets of A, called the *dual-Moore-set completion* of A), this result provides an efficient representation as a powerset-like completion for the domain of meet-autodependencies of a separated abstract domain.

Applications. In the field of groundness logic program analysis, we show that the well known abstract domain of definite propositional formulae *Def*, introduced by Dart in [24] for studying groundness in deductive databases, and then used by Marriott and Søndergaard in [47] for ground-dependency analysis of logic programs, is isomorphic to the reduced relative power having the simpler Jones and Søndergaard's domain for (nonrelational) plain groundness analysis (denoted *Gr*, cf. [43]) as base and exponent. Our reduced relative power is here necessary to handle a concrete domain of order-ideals of substitutions, where Cousot and Cousot's reduced power would not be otherwise applicable, being not a Boolean algebra. We show that *Gr* is a separated abstract domain, and therefore its dual-Moore-set completion is isomorphic to *Def* as well.

As hinted in [22, 18], abstract interpretation may be fruitfully used to derive and compare program semantics at different levels of abstraction. In this context, the reduced relative power can therefore be interpreted as a systematic operator for designing new and more descriptive program semantics by incrementally composing some already existing and simpler ones. We apply our reduced relative power operator in logic programming, which is a fine example of a high-level language enjoying a simple semantic definition. We show how to systematically derive a new compositional (w.r.t. union of sets of clauses, i.e. programs) semantics for logic programs, by reduced relative power of an existing well-known (noncompositional) semantics characterizing computed answer substitutions [27]. Here, the concrete domain consists of sets of program execution traces, which are combined in the reduced relative power by a noncommutative operator of trace-unfolding. Our approach has several advantages with respect to previous techniques for designing compositional semantics of logic programs (e.g., those described in [9], which is the basis for the construction of most of the compositional semantics in the s-semantics style, cf. [8, 15, 31]). In fact, any semantics which can be derived by abstract interpretation from a more concrete one, can also be composed by reduced relative power. This could be the case for a number of well known semantics such as: The semantics of call patterns [31] and computed answer substitutions [27], Clark's [12, 27], Herbrand's [26] and Heyting's [44] semantics, and other semantics which can be derived by abstract interpretation from a more concrete (e.g. operational) logic program semantics (see [14, 33]). Moreover, this approach is general and independent from specific semantic representations (e.g., by means of atoms, clauses, etc.).

Structure of the Paper. The rest of the paper is structured as follows. In Section 2, we introduce the notation used throughout paper and recall some basic notions of abstract interpretation. In Section 3, we give the definition of reduced relative power, and study its basic properties. In Section 4, we consider the case of additive dependencies, and we relate reduced relative power with the space of additive functions. In Section 5, we study the case of autodependencies; in particular, we show that for a certain class of abstract domains, meet-autodependencies coincide with the space of lower closure operators. Sections 6 and 7 contain, respectively, the applications in logic program analysis and semantics. Finally, Section 8 discusses related work, while Section 9 concludes. The Appendix contains the technical proofs of some results of Section 7. A preliminary account of parts of this paper has been presented in [35].

2 Preliminaries

Throughout the paper we will assume familiarity with the standard notions of lattice theory (see e.g. [7, 39]), abstract interpretation ([19, 20]), and logic programming (see e.g. [3, 45]). In this section, we briefly introduce some notation and recall some well known notions.

2.1 Mathematical Notation

Let A be a set. The powerset of A is denoted by $\wp(A)$, and the cardinality of A is denoted by $|A|$. If B is another set then $A \setminus B$ denotes the set-theoretic difference between A and B, and $A \subset B$ denotes proper inclusion. If $X \subseteq A$, $\overline{X}$ is the set-theoretic complement of X in A. A^* denotes the set of finite sequences of objects of A, where sequences are typically denoted by $\langle a_1, ..., a_n \rangle$, or simply $a_1, ..., a_n$, for a_i's symbols in A. The empty sequence is denoted by Λ. Concatenation of sequences $s_1, s_2 \in A^*$ is denoted by $s_1 :: s_2$. The sequence of symbols with first element a followed by the sequence s is denoted by $a \mid s$. If $\approx$ is an equivalence relation on A then, when clear from the context, we denote an equivalence class $[a]_\approx$ simply by $[a]$, and often abuse notation by letting the elements of a set denote their corresponding equivalence class.

The set A equipped with a partial order $\leq$ is denoted by $\langle A, \leq \rangle$. If A is a poset, we usually denote $\leq_A$ the corresponding partial order. By $x < y$ we mean $x \leq y$ and $x \neq y$. The dual of a poset A is denoted by A^{op}. A complete lattice A with partial ordering $\leq$, least upper bound (*lub*, also called join) $\vee$, greatest lower bound (*glb*, also called meet) $\wedge$, greatest element (or top)

$\top = \wedge\emptyset = \vee A$, and least element (or bottom) $\bot = \vee\emptyset = \wedge A$, is denoted by $\langle A, \leq, \vee, \wedge, \top, \bot\rangle$. Whenever A is a lattice, $\vee_A$, $\wedge_A$, $\top_A$ and $\bot_A$ denote the corresponding basic operators and elements. We will often abuse notation by denoting lattices with their poset notation. We use $\cong$ to denote isomorphism of ordered structures. If A is a pre-ordered set and $I \subseteq A$ then $\downarrow I = \{x \in A \mid \exists y \in I.\ x \leq_A y\}$. For $x \in A$, $\downarrow x$ is a shorthand for $\downarrow\{x\}$. $\wp^{\downarrow}(A)$ denotes the set of *order-ideals* of A, where $I \subseteq A$ is an order-ideal if $I = \downarrow I$. It turns out that $\langle \wp^{\downarrow}(A), \subseteq, \cup, \cap, A, \emptyset\rangle$ is a complete lattice. A complete lattice A is *completely distributive* if for any $\{x_j^i\}_{j\in J(i)}^{i\in I} \subseteq A$, $\wedge_{i\in I}(\vee_{j\in J(i)} x_j^i) = \vee_{\varphi\in\mathcal{F}_{I\mapsto J}}(\wedge_{i\in I} x_{\varphi(i)}^i)$, where $\mathcal{F}_{I\mapsto J}$ is the set of all the functions φ defined over I and such that $\forall i \in I.\ \varphi(i) \in J(i)$. By exchanging $\vee$ and $\wedge$ in the previous equality, we get an equivalent formulation of complete distributivity. A complete lattice A is *completely meet-distributive* or a *complete Heyting algebra* (cHa for short), if for any $\{x_i\}_{i\in I} \subseteq A$ and $y \in A$, $y \wedge (\vee_{i\in I} x_i) = \vee_{i\in I}(y \wedge x_i)$. Let A be a meet semilattice with bottom $\bot$. The *pseudocomplement* of $a \in A$, if it exists, is the unique element $a^* \in A$ such that $a \wedge a^* = \bot$ and $(a \wedge x = \bot) \Rightarrow (x \leq a^\star)$. A is *pseudocomplemented* if every element in A admits the pseudo-complement. Clearly, any cHa A is pseudocomplemented, where for any $a \in A$, $a^* = \vee\{x \in A \mid a \wedge x = \bot\}$. A lattice A with bottom $\bot$ and top $\top$ is *complemented* if for any $a \in A$ there exists a (possibly non unique) element $a^* \in A$ (called *complement* of a) such that $a \wedge a^* = \bot$ and $a \vee a^* = \top$. A lattice is *Boolean* if it is complemented and distributive. It is well known that in a Boolean lattice complements are unique. A complete Boolean lattice is also called a *complete Boolean algebra* (cBa for short). It is well known that any cBa is a cHa (where the notions of pseudocomplement and complement coincide). It is also well known that any $\langle \wp^{\downarrow}(A), \subseteq\rangle$ is a cHa, which, in general, is not a cBa. If A is a poset with bottom $\bot$ then an element $a \in A$ is an *atom* if a covers $\bot$, i.e., $a \neq \bot$ and for all $x \in A$, $\bot < x \leq a$ implies $x = a$. By $At(A)$ we denote the set of all the atoms of A. A complete lattice A is *atomic* if every element $x \in A$ different from $\bot$ is the lub of the atoms which precede x, i.e. $a = \vee_A\{a \in At(A) \mid x \leq_A a\}$. *Dual-atoms* and *dual-atomicity* are dually defined. It is well known that any atomic complete Boolean algebra is always isomorphic to $\langle \wp(X), \subseteq\rangle$ for some set X. An element $x \in A$ of a complete lattice is (*completely*) *join-irreducible* if for all $X \subseteq A$, $x = \vee_A X \Rightarrow x \in X$. Further, x is (*completely*) *join-reducible* if it is not join-irreducible, i.e. if there exists $X \subseteq A$ such that $x = \vee_A X$ and $x \notin X$. The subset of join-irreducible elements of a complete lattice A is denoted by $JI(A)$.

We write $f : A \longmapsto B$ to mean that f is a (total) function from A to B. We sometimes use Church's lambda notation for functions, so that a function f is often denoted by $\lambda x.f(x)$. The identity function $\lambda x.x$ is sometimes denoted by id. If $C \subseteq A$ then $f(C) = \{f(x) \mid x \in C\}$. By $g \circ f$ we denote the composition of the functions f and g, i.e. $\lambda x.g(f(x))$. The set of fixpoints of a function $f : A \longmapsto A$ is denoted by $fp(f)$, and, assuming that A is a poset, the least fixpoint is denoted by $lfp(f)$, if it exists. For a complete lattice A, the ordinal α (upper) power of a function $f : A \longmapsto A$ for an element $x \in A$, is denoted by $f^\alpha(x)$ and it is defined as x if $\alpha = 0$, $f(f^{\alpha-1}(x))$ if α is a successor ordinal, and $\vee_{\delta<\alpha} f^\delta(x)$ if α is a limit ordinal. (ω denotes the first limit ordinal.) Let $\langle A, \leq_A, \vee_A, \wedge_A, \top_A, \bot_A\rangle$ and $\langle B, \leq_B, \vee_B, \wedge_B, \top_B, \bot_B\rangle$ be complete lattices, and $f : A \longmapsto B$. f is *strict* if $f(\bot_A) = \bot_B$. f is *(co-)additive* if for any $C \subseteq A$, $f(\vee_A C) = \vee_B f(C)$ ($f(\wedge_A C) = \wedge_B f(C)$). f is *(co-)continuous* whenever this condition holds for any chain $C \subseteq A$. The complete lattice of total (monotone, continuous, additive) functions from the complete lattice A into the complete lattice B, equipped with the functional pointwise order (i.e., $f \sqsubseteq g$ iff $\forall x \in A.\ f(x) \leq_B g(x)$), is denoted by $A \longmapsto B$ ($A \overset{\mathrm{m}}{\longmapsto} B$, $A \overset{\mathrm{c}}{\longmapsto} B$, $A \overset{\mathrm{a}}{\longmapsto} B$). In $A \longmapsto B$ and $A \overset{\mathrm{m}}{\longmapsto} B$, the *lub* $\sqcup$ and the *glb* $\sqcap$ are defined pointwise, i.e., if F is a subset of functions then, for any $x \in A$, $(\sqcup F)(x) = \vee_{f\in F} f(x)$, and $(\sqcap F)(x) = \wedge_{f\in F} f(x)$, while in $A \overset{\mathrm{c}}{\longmapsto} B$ and $A \overset{\mathrm{a}}{\longmapsto} B$ just the *lub* $\sqcup$ is defined pointwise.

2.2 Abstract Interpretation, Galois Connections and Closure Operators

The standard Cousot and Cousot theory of abstract interpretation ([19]) is based on the notion of Galois connection. If C and A are posets, and $\alpha : C \longmapsto A$, $\gamma : A \longmapsto C$ are monotone maps such that $\forall x \in C.\ x \leq_C \gamma(\alpha(x))$ and $\forall y \in A.\ \alpha(\gamma(y)) \leq_A y$, then we call (C, α, A, γ) a *Galois connection* (G.c. for short) between C and A. If in addition $\forall y \in A.\ \alpha(\gamma(y)) = y$, then we call

(C, α, A, γ) a *Galois insertion* (G.i. for short) of A in C. We also recall that the notion of G.c. is equivalent to that of *adjunction*: If $\alpha : C \longmapsto A$ and $\gamma : A \longmapsto C$ then (C, α, A, γ) is a G.c. iff $\forall x \in C. \forall y \in A.\ \alpha(x) \leq_A y \Leftrightarrow x \leq_C \gamma(y)$. The map α (γ) is called the *left-* (*right-*)*adjoint* to γ (α). Galois connections and insertions enjoy the following well known properties (that we will freely use, often without mention, throughout the paper). Let us assume that (C, α, A, γ) is a Galois connection.

(i) $\alpha \circ \gamma \circ \alpha = \alpha$ and $\gamma \circ \alpha \circ \gamma = \gamma$.

(ii) If A and C are both bounded (i.e., they have top and bottom elements), then $\alpha(\bot_C) = \bot_A$ and $\gamma(\top_A) = \top_C$.

(iii) α preserves *lub's* (i.e., if for some $S \subseteq C$ the lub $\vee_C S$ exists, then the lub $\vee_D \alpha(S)$ exists and $\alpha(\vee_C S) = \vee_D \alpha(S)$) and γ preserves *glb's*. In particular, if A and C are complete lattices then α is additive and γ is co-additive.

(iv) α uniquely determines γ, and γ uniquely determines α. In particular, if A and C are complete lattices, then any additive map $\alpha : C \longmapsto A$ uniquely determines a G.c. (C, α, A, γ), where its right-adjoint is $\gamma = \lambda y. \vee_C \{x \in C \mid \alpha(x) \leq_A y\}$ (and this dually holds for any co-additive map $\gamma : A \longmapsto C$).

(v) Let (C, α, A, γ) be a G.i. Then, α is onto and γ is 1-1, and if C is a complete lattice then A is a complete lattice as well.

Two G.c.'s $(C_i, \alpha_i, A_i, \gamma_i)$, $i = 1, 2$, are *isomorphic* if (1) $C_1 \cong C_2$, with isomorphism $\jmath : C_1 \longmapsto C_2$, (2) $A_1 \cong A_2$, with isomorphism $\iota : A_1 \longmapsto A_2$, and (3) $\iota \circ \alpha_1 = \alpha_2 \circ \jmath$ (or, equivalently, $\jmath \circ \gamma_1 = \gamma_2 \circ \iota$). It is not difficult to show that for two Galois insertions, $(C, \alpha_1, A_1, \gamma_1) \cong (C, \alpha_2, A_2, \gamma_2)$ iff $\gamma_1(D_1) = \gamma_2(D_2)$. We also say that two G.c.'s $(C, \alpha_{C,D}, D, \gamma_{D,C})$ and $(D, \alpha_{D,A}, A, \gamma_{A,D})$ are *coherent* with a third G.c. (C, α, A, γ) if $\alpha = \alpha_{D,A} \circ \alpha_{C,D}$ (or, equivalently, $\gamma = \gamma_{D,C} \circ \gamma_{A,D}$).

In the setting of abstract interpretation, if (C, α, A, γ) is a Galois connection, C and A are called, respectively, the *concrete* and the *abstract domain*, and they are assumed to be complete lattices, whereas α and γ are called the *abstraction* and the *concretization* maps, respectively. Also, A is called an *abstraction* of C, and C a *concretization* of A. If (C, α, A, γ) is a Galois insertion, each value of the abstract domain A is useful in the representation of the concrete domain C as all the elements of A represent distinct members of C. Note that, any G.c. may be lifted to a G.i. identifying in an equivalence class those values of the abstract domain with the same concrete meaning. This process is known as *reduction* of the abstract domain.

An *upper* (*lower*) *closure operator* on a poset $\langle C, \leq \rangle$ is an operator $\rho : C \longmapsto C$ which is monotone, idempotent and extensive (reductive), viz. for all $x \in A$, $x \leq \rho(x)$ $(\rho(x) \leq x)$. The set of all upper (lower) closure operators on C is denoted by $uco(C)$ $(lco(C))$. It is well known that for any G.c. (C, α, A, γ), $\alpha \circ \gamma \in lco(A)$ and $\gamma \circ \alpha \in lco(C)$. In the following, we only deal with properties of upper closures, since for lower closures the corresponding properties can be retrieved by duality. If $\langle C, \leq, \vee, \wedge, \top, \bot \rangle$ is a complete lattice then each closure operator $\rho \in uco(C)$ is uniquely determined by the set of its fixpoints, which is its image, i.e. $\rho(C) = fp(\rho)$ (where $\rho = \lambda y. \wedge \{x \in fp(\rho) \mid y \leq x\}$). A set $X \subseteq C$ is the set of fixpoints of an upper closure operator iff X is a *Moore-set* of C, i.e. $X = Mc(X) = \{\wedge Y \mid Y \subseteq X\}$ (note that $\top \in Mc(X)$). For any $X \subseteq C$, $Mc(X)$ is the *Moore-closure* of X in C (dually, $dMc(X) = \{\vee Y \mid Y \subseteq X\}$ is the *dual-Moore-closure* of X in C). Moreover, if ρ is an upper closure then its set of fixpoints $\rho(C)$ is a complete lattice $\langle \rho(C), \leq, \lambda X. \rho(\vee_C X), \wedge_C, \top_C, \rho(\bot_C) \rangle$. $\rho(C)$ is a complete meet subsemilattice of C, while $\rho(C)$ is a complete sublattice of C iff ρ is additive. The complete lattice of all upper closures on a complete lattice C is denoted by $\langle uco(C), \sqsubseteq, \sqcup, \sqcap, \lambda x.\top, \lambda x.x \rangle$, where for every $\rho, \eta \in uco(C)$, $\{\rho_i\}_{i \in I} \subseteq uco(C)$ and $x \in C$:

- $\rho \sqsubseteq \eta$ iff $\forall x \in C.\ \rho(x) \leq \eta(x)$, or, equivalently, $\rho \sqsubseteq \eta$ iff $\eta(C) \subseteq \rho(C)$;
- $(\sqcup_{i \in I} \rho_i)(x) = x \ \Leftrightarrow\ \forall i \in I.\ \rho_i(x) = x$;

- $(\sqcap_{i\in I}\rho_i)(x) = \wedge_{i\in I}\rho_i(x)$;
- $\lambda x.\top$ is the top element, whereas $\lambda x.x$ is the bottom element.

It is also known that $uco(C)$ is dual-atomic, where the dual-atoms of $uco(C)$ are all and only those closures ϕ_x for $x \in C \setminus \{\top\}$, such that $\phi_x(C) = \{x, \top\}$, i.e., for any $y \in C$,

$$\phi_x(y) = \begin{cases} x & \text{if } y \leq x; \\ \top & \text{otherwise.} \end{cases}$$

Further details on closure operators can be found in [48, 64].

2.3 Logic Programming Notation

In the following, Σ, Π and *Var* will respectively denote a set of function symbols, a set of predicate symbols and a denumerable set of variables, defining a fixed first-order language $\mathcal{L}$. The set of terms and atoms over $\mathcal{L}$ are denoted by *Term* and *Atom*, respectively. A *goal* is a (possibly empty) sequence of atoms. A (*definite*) *clause* is an object $h \leftarrow b_1, ..., b_n$, with $n \geq 0$, where h is an atom, called the *head*, and $b_1, ..., b_n$ is a goal, called the *body*. The set of clauses built from elements of *Atom* is denoted by *Clause*. A *unit-clause* is a clause with an empty body. Atoms and unit-clauses will be considered equivalent notions. A (*definite*) *logic program* is a (possibly infinite[1]) set of clauses, and the set of logic programs is denoted by *Program*. $s_1 \equiv s_2$ denotes syntactic equality between syntactic objects s_1 and s_2. Tuples of syntactic objects of the same type (like variables, terms, atoms, etc.) are sometimes denoted by $\bar{s}$. Sometimes, we will abuse notation by denoting with $\bar{s}$ both the tuple and the set of corresponding syntactic objects. In this case, by $x \in \bar{s}$ we mean any element x in the sequence $\bar{s}$. The set of variables (predicates) that occur in a syntactic object s is denoted by $var(s)$ ($pred(s)$). A syntactic object s is *ground* if $var(s) = \emptyset$. A *substitution* is a mapping from *Var* to *Term* which acts as the identity almost everywhere. A substitution σ is typically specified by its set of nontrivial bindings, viz. $\sigma = \{x/\sigma(x) \mid \sigma(x) \neq x\}$. The empty substitution is denoted by ε. If σ is a substitution then $gr(\sigma)$ denotes the set of variables which are grounded by σ, i.e. $gr(\sigma) = \{x \in Var \mid var(\sigma(x)) = \emptyset\}$. The application of a substitution σ to a syntactic object s is denoted by $s\sigma$. The standard composition of substitutions θ and σ is denoted by $\theta\sigma$. If Θ is a set of substitutions, then $s\Theta = \{s\theta \mid \theta \in \Theta\}$. Unless explicitly stated otherwise, *Sub* is the set of idempotent substitutions. A *variable renaming* is a (not necessarily idempotent) substitution which is a bijection on *Var*. A syntactic object t is *more general* than t' (denoted $t' \leq t$) iff there exists a substitution σ such that $t' = t\sigma$. Syntactic objects s_1 and s_2 are *equivalent up to renaming*, denoted $s_1 \sim s_2$, iff $s_1 \leq s_2$ and $s_2 \leq s_1$. $Term_{/\sim}$, $Atom_{/\sim}$, and $Clause_{/\sim}$ are partially ordered with respect to $\leq$. With abuse of notation, they will be still denoted by *Term*, *Atom*, and *Clause*. Since all the definitions in the paper are independent on syntactic variable names, we will let a syntactic object denote its equivalence class by renaming. For a syntactic object s and a set of (equivalence classes by renaming of) objects I, we denote by $\langle c_1, \ldots, c_n \rangle \ll_s I$ ($n \geq 0$), that $c_1, \ldots, c_n$ are representatives of elements of I renamed apart from s and from each other (i.e. $\forall i, j \in [1, n]$. $var(c_i) \cap var(s) = \emptyset$ & $(i \neq j \Rightarrow var(c_i) \cap var(c_j) = \emptyset)$). If S is a set of syntactic objects then $inst(S) = \{s' \mid s' \leq s, s \in A\}$ and $ground(S) = \{s \mid var(s) = \emptyset, s \in inst(A)\}$ denote, respectively, the set of instances and ground instances of elements in S.

The notion of unification can be given in terms of equations and equation solving. An equation has the form $t = u$ where t, u are syntactic objects and $=$ is interpreted as syntactic equality. Given a set e of equations, e is unifiable iff there exists a substitution θ such that for all $(t = u) \in e$, $t\theta \equiv u\theta$. We denote by $Unif(e)$ the (possibly empty) set of unifiers of e. We fix a partial function *mgu* which maps a pair of syntactic objects (or a set of equations) into an idempotent most general unifier of the objects, if such exists. This is not restrictive, since it is well known that all the idempotent *mgu* of a set of equations are equivalent up to renaming. For sequences of syntactic objects $\langle t_1, ..., t_n \rangle$ and $\langle u_1, ..., u_n \rangle$, $n \geq 1$, $mgu(\langle t_1, ..., t_n \rangle, \langle u_1, ..., u_n \rangle)$ is an equivalent notation for $mgu(\{t_1 = u_1, ..., t_n = u_n\})$. For a set of equations e, $mgu(e)$ is an idempotent element in the

[1]As usual when reasoning on logic program semantics, we allow a logic program to be infinite.

set $\{\vartheta \in \mathit{Unif}(e) \mid \forall \vartheta' \in \mathit{Unif}(e).\ \vartheta \leq \vartheta'\}$. $mgu(e) = \mathit{fail}$ means that e is not unifiable. If $\theta = \{x_1/t_1, ..., x_n/t_n\}$ is a substitution, then we denote $\mathcal{E}(\theta) = \{x_1 = t_1, ..., x_n = t_n\}$. It is well known that for nonempty Herbrand universes, $\theta = mgu(e)$ iff e has the same solutions as $\mathcal{E}(\theta)$.

3 Reduced Relative Power of Abstract Domains

Reduced cardinal power of abstract domains has been introduced by Cousot and Cousot in the very last section of their POPL'79 paper (cf. [20, Section 10.2]). The basic idea was that, given a pair of abstract domains D_1 and D_2, a new domain can be systematically derived by considering the lattice of all monotone mappings from D_1 to D_2, i.e. the cardinal power with base D_2 and exponent D_1. However, Cousot and Cousot's definition and related correctness result (cf. [20, Theorem 10.2.0.1]) were given in a particular context of a collecting semantics of program assertions, which was a complete Boolean lattice. In this section, we carries on the original Cousot and Cousot idea to a more general and standard abstract interpretation environment. This is necessary in order to both provide a better understanding of Cousot's reduced power of domains, and to let the reduced power construction be applicable in the context of abstract interpretation of possibly non Boolean concrete domains, where the basic binary operator for combining concrete objects may not be set intersection. As we will see later in Section 6 and 7, this extension allows us to interpret well known domains and semantics in terms of this reduced relative power.

Definition 3.1 Let D, D_1, D_2 be posets, $\odot : D \times D \longmapsto D$ be a monotone[2] operation, $\gamma_1 : D_1 \longmapsto D$ and $\alpha_2 : D \longmapsto D_2$ be monotone functions. The poset of *dependencies* with base D_2 and exponent D_1 (w.r.t. D and $\odot$) is the set of functions $D_1 \overset{\odot}{\longmapsto} D_2 = \{\lambda x.\alpha_2(d \odot \gamma_1(x)) \in D_1 \longmapsto D_2 \mid d \in D\}$, partially ordered by the pointwise ordering relation $\sqsubseteq$. $\square$

These functions $\lambda x.\alpha_2(d \odot \gamma_1(x))$ from D_1 to D_2, called dependencies, constitute the basic blocks of our approach. Roughly speaking, in an abstract interpretation setting, the operation $\odot$ should be thought of as a kind of combinator of concrete denotations: We will see later that the *glb* is a typical example of such an operation, although some nontrivial applications (as that presented in Section 7) may require a different and less restrictive kind of concrete combinator. Intuitively, a dependency $\lambda x.\alpha_2(d \odot \gamma_1(x))$ encodes how the abstract domain D_2 is able to represent the reaction of the concrete value d whenever it is combined via $\odot$ with an object described by D_1. The above definition only requires the weakest hypotheses so that every dependency is monotone.

Lemma 3.2 $D_1 \overset{\odot}{\longmapsto} D_2 \subseteq D_1 \overset{\mathrm{m}}{\longmapsto} D_2$.

Proof. If $d \in D$ then $\lambda x.\alpha_2(d \odot \gamma_1(x))$ is monotone, since $\gamma_1, \odot$ and α_2 are monotone. $\blacksquare$

Obviously, our basic requirement is that the poset of dependencies actually is a correctly defined abstract domain, where every dependency $\lambda x.\alpha_2(d \odot \gamma_1(x))$ is the abstraction of the concrete value $d \in D$. *Left-additivity* of the operation $\odot$ (i.e., for any $Y \subseteq D$ and $x \in D$, $(\vee Y) \odot x = \vee_{y \in Y}\, y \odot x$) suffices to this aim. The following notion of *semi-quantale* incorporates into a unique algebraic structure this hypothesis.

Definition 3.3 A *semi-quantale* is a structure $\langle D, \leq, \odot \rangle$, where $\langle D, \leq \rangle$ is a complete lattice and $\odot : D \times D \longmapsto D$ is associative, monotone and left-additive. $\square$

Whenever a semi-quantale plays the rôle of a concrete domain, we also term it *concrete semi-quantale*. As we will see later, this hypothesis will be sufficient to prove many basic results of our theory.

Algebrically, semi-quantales are weaker structures than the well-known *quantales* used to give semantic models to various kinds of logics. Mulvey firstly introduced in [50] the term quantale in connection with his work on noncommutative algebras. A structure $\langle D, \leq, \odot \rangle$ is a *quantale* if

[2] That is, $\forall x, y, z \in D.\ (x \leq z) \Rightarrow ((x \odot y \leq z \odot y) \ \ \&\ \ (y \odot x \leq y \odot z))$.

$\langle D, \leq \rangle$ is a complete lattice and $\odot$ is associative and both left- and right-additive. More recently (cf. [1]), the term quantale is used in place of what Mulvey called *unital quantale*, viz. a quantale $\langle D, \leq, \odot \rangle$ such that there exists a unit element $\mathbf{1} \in D$ such that $\forall x \in D.\ x \odot \mathbf{1} = \mathbf{1} \odot x = x$. We follow the terminology of Mulvey [50] and Rosenthal [56] without assuming the existence of a unit. With respect to quantales, our semi-quantales may lack right-additivity. Moreover, it should be noted that, in analogy with quantales, we do not require commutativity of $\odot$: Later in Example 3.10 and Section 7, we will consider noncommutative operators for concrete composition. Left-additivity of $\odot$ in semi-quantales allows us to prove the following basic lemma.

Lemma 3.4 *If $\langle D, \leq, \odot \rangle$ is a semi-quantale, D_1 and D_2 are complete lattices, and α_2 is additive, then $\alpha : D \longmapsto (D_1 \overset{\mathrm{m}}{\longmapsto} D_2)$ defined as $\alpha(d) = \lambda x.\alpha_2(d \odot \gamma_1(x))$, is additive (and therefore the left-adjoint function of a Galois connection).*

Proof. If $\{d_i\}_{i \in I} \subseteq D$ then

$$\begin{array}{rcll} \alpha(\vee_{i\in I} d_i) & = & \lambda x.\alpha_2((\vee_{i\in I} d_i) \odot \gamma_1(x)) & \\ & = & \lambda x.\alpha_2(\vee_{i\in I}(d_i \odot \gamma_1(x))) & (\odot \text{ is left-additive}) \\ & = & \lambda x. \vee_{i\in I}\ \alpha_2(d_i \odot \gamma_1(x)) & (\alpha_2 \text{ is additive}). \end{array}$$

Since $\lambda x. \vee_{i\in I}\ \alpha_2(d_i \odot \gamma_1(x))$ is the pointwise *lub* of $\{\lambda x.\alpha_2(d_i \odot \gamma_1(x))\}_{i\in I}$, this concludes the proof. ∎

We are now ready for the following basic definition.

Definition 3.5 Let $\langle D, \leq, \odot \rangle$ be a semi-quantale, D_1 and D_2 be complete lattices, $\gamma_1 : D_1 \overset{\mathrm{m}}{\longmapsto} D$ be a monotone function, and $(D, \alpha_2, D_2, \gamma_2)$ be a Galois connection. The *reduced relative power* with base D_2 and exponent D_1 is $D_1 \overset{\odot}{\longmapsto} D_2$, ordered pointwise. □

The following basic result proves the correctness of the above definition.

Theorem 3.6 *Under the hypotheses of Definition 3.5, the function $\alpha : D \longmapsto (D_1 \overset{\odot}{\longmapsto} D_2)$ defined as $\alpha(d) = \lambda x.\alpha_2(d \odot \gamma_1(x))$, is the left-adjoint map of a Galois insertion $(D, \alpha, D_1 \overset{\odot}{\longmapsto} D_2, \gamma)$.*

Proof. Since α is evidently onto, this is an immediate consequence of Lemma 3.4. ∎

Thus, under the basic hypotheses of Definition 3.5, by mapping a concrete value $d \in D$ to the dependency $\lambda x.\alpha_2(d \odot \gamma_1(x))$, we actually get a correct abstraction map for the reduced relative power. In the following, we will often assume that γ_1 is the right-adjoint in a Galois connection $(D, \alpha_1, D_1, \gamma_1)$. In this case, we call D_2 and D_1, respectively, the *base* and *exponent* abstract domains. In the term "reduced relative power", the adjective "relative" refers to the concrete combinator, while the adjective "reduced" is justified by the fact that $D_1 \overset{\odot}{\longmapsto} D_2$ is just the reduction of $D_1 \overset{\mathrm{m}}{\longmapsto} D_2$ w.r.t. the Galois connection of Lemma 3.4. Since D is assumed to be a complete lattice, by point (v) in Section 2.2, $D_1 \overset{\odot}{\longmapsto} D_2$ is a complete lattice too. In particular, the following result holds.

Proposition 3.7 *$\langle D_1 \overset{\odot}{\longmapsto} D_2, \sqsubseteq \rangle$ is a complete lattice, where the lub is defined pointwise, the top element is $\lambda x.\alpha_2(\top_D \odot \gamma_1(x))$ and the bottom element is $\lambda x.\alpha_2(\bot_D \odot \gamma_1(x))$.*

Proof. Let $F = \{\lambda x.\alpha_2(d_i \odot \gamma_1(x))\}_{i\in I} \subseteq D_1 \overset{\odot}{\longmapsto} D_2$, for some $\{d_i\}_{i\in I} \subseteq D$. Then, the following equalities show that $\sqcup F$, the pointwise *lub* of F, is in $D_1 \overset{\odot}{\longmapsto} D_2$:

$$\begin{array}{rcll} \sqcup F & = & \lambda x. \vee_{i\in I}\ \alpha_2(d_i \odot \gamma_1(x)) & \\ & = & \lambda x.\alpha_2(\vee_{i\in I}(d_i \odot \gamma_1(x))) & (\alpha_2 \text{ is additive}) \\ & = & \lambda x.\alpha_2((\vee_{i\in I} d_i) \odot \gamma_1(x)) & (\odot \text{ is left-additive}). \end{array}$$

Moreover, it is immediate to see that $\lambda x.\alpha_2(\top_D \odot \gamma_1(x))$ and $\lambda x.\alpha_2(\bot_D \odot \gamma_1(x))$ are, respectively, the top and the bottom in $D_1 \overset{\odot}{\longmapsto} D_2$. ∎

Of course, when $\odot$ is the *glb* $\wedge$ of D, to ask for $\langle D, \leq, \wedge\rangle$ to be a semi-quantale in Definition 3.5 corresponds to require that $\langle D, \leq\rangle$ is a cHa. In this case, a generic function $\lambda x.\alpha_2(d \wedge_D \gamma_1(x)) \in D_1 \stackrel{\wedge}{\longmapsto} D_2$ is called a *meet-dependency*. Further, note that when $\odot = \vee_D$, $\langle D, \leq, \vee\rangle$ is obviously a semi-quantale.

The Cousot and Cousot reduced cardinal power can be retrieved as an instance of Definition 3.5, where the concrete semi-quantale is $\langle \wp(X), \subseteq, \cap\rangle$, for some set X. In fact, their concrete domain of program assertions $\langle Var \longmapsto \{false, true\}, \Rightarrow\rangle$ (cf. [20]) is isomorphic to $\langle \wp(Var), \subseteq\rangle$, where the logical conjunction of program assertions simply amounts to set-intersection. Definition 3.5 and Theorem 3.6 provide the right generalization in our framework of Cousot and Cousot's original definition of reduced cardinal power and related correctness result [20, Theorem 10.2.0.1]. When the concrete semi-quantale is $\langle \wp(X), \subseteq, \cap\rangle$, it is possible to provide an explicit definition for the right-adjoint map in the Galois connection of Lemma 3.4, and therefore for the concretization map γ of Theorem 3.6 (a dual formulation can be obtained whenever the order is super-inclusion and $\odot$ is set-union).

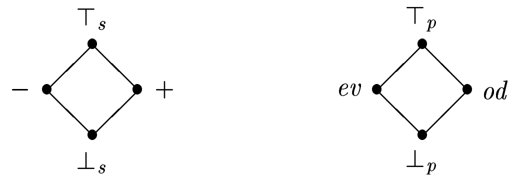
Proposition 3.8 *Let $\langle \wp(X), \subseteq, \cap\rangle$ be the concrete semi-quantale, and let $(\wp(X), \alpha_1, D_1, \gamma_1)$ and $(\wp(X), \alpha_2, D_2, \gamma_2)$ be two Galois connections. Then, the right-adjoint of α in Lemma 3.4 is $\gamma : (D_1 \stackrel{\mathrm{m}}{\longmapsto} D_2) \longmapsto D$ defined as $\gamma(f) = \{z \in X \mid \forall x \in D_1.\, (\alpha_1(\{z\}) \leq_1 x) \Rightarrow (\alpha_2(\{z\}) \leq_2 f(x))\}$.*

Proof. We show that $\alpha(S) \sqsubseteq f \;\Leftrightarrow\; S \subseteq \gamma(f)$, for any $S \in \wp(X)$ and $f \in D_1 \stackrel{\mathrm{m}}{\longmapsto} D_2$.
($\Rightarrow$) We show that if $z \in S$ then $z \in \gamma(f)$. Assume that $x \in D_1$ and $\alpha_1(\{z\}) \leq_1 x$. Then, $z \in \gamma_1(x)$. By hypothesis, $\alpha_2(S \cap \gamma_1(x)) \leq_2 f(x)$; therefore, $\alpha_2(\{z\}) \leq_2 \alpha_2(S \cap \gamma_1(x)) \leq_2 f(x)$.
($\Leftarrow$) Firstly, note that if $T \subseteq \gamma(f)$ then $\forall x \in D_1.\, (\alpha_1(T) \leq_1 x) \Rightarrow (\alpha_2(T) \leq_2 f(x))$. In fact, if $x \in D_1$ and $\alpha_1(T) \leq_1 x$ then $\forall t \in T.\, \alpha_1(\{t\}) \leq_1 x$; hence, $\forall t \in T.\, \alpha_2(\{t\}) \leq_2 f(x)$, which in turn implies $\alpha_2(T) \leq_2 f(x)$. Now, we want to prove that $\forall y \in D_1.\, \alpha_2(S \cap \gamma_1(y)) \leq_2 f(y)$. If $y \in D_1$ then $S \cap \gamma_1(y) \subseteq S \subseteq \gamma(f)$. Thus, we get $\forall x \in D_1.\, (\alpha_1(S \cap \gamma_1(y)) \leq_1 x) \Rightarrow (\alpha_2(S \cap \gamma_1(y)) \leq_2 f(x))$. From this, since $\alpha_1(S \cap \gamma_1(y)) \leq_1 \alpha_1(\gamma_1(y)) \leq_1 y$, we get $\alpha_2(S \cap \gamma_1(y)) \leq_2 f(y)$, as desired. ■

It is important to observe that the concretization map provided in Proposition 3.8 is very natural. In fact, it would not be difficult to see that for any $f \in D_1 \stackrel{\mathrm{m}}{\longmapsto} D_2$, $\gamma(f) = \cup\{S \subseteq X \mid \forall x \in D_1.\, (\alpha_1(S) \leq_1 x) \Rightarrow (\alpha_2(S) \leq_2 f(x))\}$, while the above proof also shows that $\forall x \in D_1.\, (\alpha_1(\gamma(f)) \leq_1 x) \Rightarrow (\alpha_2(\gamma(f)) \leq_2 f(x))$. Thus, a given function $f \in D_1 \stackrel{\mathrm{m}}{\longmapsto} D_2$ represents precisely the greatest concrete value $S \in \wp(X)$, such that for any input element x of D_1, if the abstraction of S in D_1 is more precise than x then the abstraction of S in D_2 is more precise than the output of x through f.

The following simple example, which formalizes the one sketched in the introduction, illustrates how the reduced relative power actually work. Note that, in this case, our construction boils down into the classical reduced cardinal power.

Example 3.9 Consider the following two classical abstract domains *Sign* and *Parity* for sign and parity analysis of an integer variable ([19, 20]).



The corresponding G.i.'s are $(\wp(\mathbb{Z}), \alpha_s, Sign, \gamma_s)$ and $(\wp(\mathbb{Z}), \alpha_p, Parity, \gamma_p)$, where the abstraction and concretization mappings are as follows.

$$\alpha_s(S) = \begin{cases} \bot_s & \text{if } S = \emptyset \\ + & \text{if } \emptyset \neq S \subseteq \{y \in \mathbb{Z} \mid y > 0\} \\ - & \text{if } \emptyset \neq S \subseteq \{y \in \mathbb{Z} \mid y < 0\} \\ \top_s & \text{otherwise} \end{cases} \qquad \gamma_s(x) = \begin{cases} \emptyset & \text{if } x = \bot_s \\ \{y \in \mathbb{Z} \mid y > 0\} & \text{if } x = + \\ \{y \in \mathbb{Z} \mid y < 0\} & \text{if } x = - \\ \mathbb{Z} & \text{if } x = \top_s \end{cases}$$

$$\alpha_p(S) = \begin{cases} \bot_p & \text{if } S = \emptyset \\ od & \text{if } \emptyset \neq S \subseteq \{y \in \mathbb{Z} \mid y \text{ is odd}\} \\ ev & \text{if } \emptyset \neq S \subseteq \{y \in \mathbb{Z} \mid y \text{ is even}\} \\ \top_p & \text{otherwise} \end{cases} \gamma_p(x) = \begin{cases} \emptyset & \text{if } x = \bot_p \\ \{y \in \mathbb{Z} \mid y \text{ is odd}\} & \text{if } x = o \\ \{y \in \mathbb{Z} \mid y \text{ is even}\} & \text{if } x = e \\ \mathbb{Z} & \text{if } x = \top_p \end{cases}$$

Here, the concrete semi-quantale is $\langle \wp(\mathbb{Z}), \subseteq, \cap \rangle$. The reduced relative power $Sign \xrightarrow{\cap} Parity$ with exponent $Sign$ and base $Parity$ is therefore given by the set of functions $\{\lambda x.\alpha_p(S \cap \gamma_s(x)) \in Sign \xrightarrow{\text{m}} Parity \mid S \in \wp(\mathbb{Z})\}$, and, by Theorem 3.6, it is related to the concrete domain $\wp(\mathbb{Z})$ by a G.i. $(\wp(\mathbb{Z}), \alpha, Sign \xrightarrow{\cap} Parity, \gamma)$. The domain $Sign \xrightarrow{\cap} Parity$ is able to represent several sets of integers which are representable neither in $Sign$ nor in $Parity$. For example, $Sign \xrightarrow{\cap} Parity$ represents the following sets of integers:

- *The value* 0

$$\lambda x.\alpha_p(\{0\} \cap \gamma_s(x)) = \begin{cases} \top_s & \mapsto ev \\ - & \mapsto \bot_p \\ + & \mapsto \bot_p \\ \bot_s & \mapsto \bot_p \end{cases} \xmapsto{\gamma} \{0\}$$

- *The positive and even integers*

$$\lambda x.\alpha_p(\{x \in \mathbb{Z} \mid x > 0,\, x \text{ even}\} \cap \gamma_s(x)) = \begin{cases} \top_s & \mapsto ev \\ - & \mapsto \bot_p \\ + & \mapsto ev \\ \bot_s & \mapsto \bot_p \end{cases} \xmapsto{\gamma} \{x \in \mathbb{Z} \mid x > 0,\, x \text{ even}\}$$

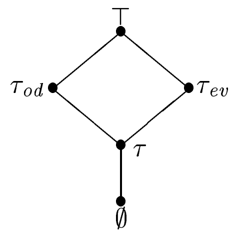
- *The positive or even integers*

$$\lambda x.\alpha_p(\{x \in \mathbb{Z} \mid x > 0 \text{ or } x \text{ even}\} \cap \gamma_s(x)) = \begin{cases} \top_s & \mapsto \top_p \\ - & \mapsto ev \\ + & \mapsto \top_p \\ \bot_s & \mapsto \bot_p \end{cases} \xmapsto{\gamma} \{x \in \mathbb{Z} \mid x > 0 \text{ or } x \text{ even}\}$$

For instance, the function $f = \lambda x.\alpha_p(\{0\} \cap \gamma_s(x))$ gives ev only when the argument is $\top_s$ (that represents any integer), while gives $\bot_p$ otherwise. Thus, this function defines the greatest set of integers which, once intersected with $\mathbb{Z}$ gives a set of even numbers (i.e. it may only contain even numbers), and, once intersected with the set of either positive or negative integers gives the empty set: Clearly, this means that $\gamma(f) = \{0\}$. On the other hand, that $\gamma(f) = \{0\}$ can also be easily verified by exploiting Proposition 3.8. □

The following example illustrates how reduced relative power can be defined relative to a concrete semi-quantale $\langle D, \leq, \odot \rangle$, where $\odot$ is not commutative. A further relevant application showing the importance of our general approach will be discussed later in Section 7.

Example 3.10 Consider the set ℓ of finite-length lists, where the elements of a list range over a family of types Υ ($|l|$ denotes the length of a list l). The domain $Par(\tau)$, parametric on the type $\tau \in \Upsilon$, expresses the following uniform property of lists: For a list l, either its even or its odd position elements have type τ.



$Par(\tau)$

It is easy to prove that the following concretization map gives rise to a G.i. $(\wp(\ell), \alpha, Par(\tau), \gamma)$:

$$\gamma(x) = \begin{cases} \emptyset & \text{if } x = \emptyset \\ \{l \in \ell \,|\, \forall i \in [1, |l|].\ l(i) \in \tau\} & \text{if } x = \tau \\ \{l \in \ell \,|\, \forall i \in [1, |l|].\ i \ mod \ 2 \neq 0 \Rightarrow l(i) \in \tau\} & \text{if } x = \tau_{od} \\ \{l \in \ell \,|\, \forall i \in [1, |l|].\ i \ mod \ 2 = 0 \Rightarrow l(i) \in \tau\} & \text{if } x = \tau_{ev} \\ \ell & \text{if } x = \top \end{cases}$$

Consider the operation of concatenation $@ : \wp(\ell) \times \wp(\ell) \to \wp(\ell)$ such that for any two sets of lists $A, B \in \wp(\ell)$, $A@B = \{l :: l' \ \mid\ l \in A, l' \in B\}$, where :: is the ordinary list concatenation. Obviously, $\langle \wp(\ell), \subseteq, @\rangle$ is a quantale, where the singleton set containing the empty list [ ] is a unit. By Theorem 3.6, we can consider the reduced relative power $Par(\tau) \stackrel{@}{\longmapsto} Par(\tau)$. A number of properties of lists can be characterized as elements in $Par(\tau) \stackrel{@}{\longmapsto} Par(\tau)$. For instance, we have:

- The set of all lists of odd-length having odd-position elements of type τ is represented by the function $\{\emptyset \mapsto \emptyset, \tau \mapsto \tau_{od}, \tau_{od} \mapsto \top, \tau_{ev} \mapsto \tau_{od}, \top \mapsto \top\}$;
- The set of all lists of even-length having odd-position elements of type τ is represented by the function $\{\emptyset \mapsto \emptyset, \tau \mapsto \tau_{od}, \tau_{od} \mapsto \tau_{od}, \tau_{ev} \mapsto \top, \top \mapsto \top\}$. $\square$

The following result shows that, whenever the concrete semi-quantale $\langle D, \leq, \odot\rangle$ has a right-unit, i.e. $\mathbf{1} \in D$ such that $\forall x \in D.\ x \odot \mathbf{1} = x$, and D_1 represents this unit, i.e. there exists $\mathbf{1}^\sharp$ such that $\gamma_1(\mathbf{1}^\sharp) = \mathbf{1}$, then by mapping a dependency $\lambda x.\alpha_2(d \odot \gamma_1(x))$ to $\alpha_2(d)$ we get a coherent Galois connection between D_2 and $D_1 \overset{D}{\underset{\odot}{\longmapsto}} {}_2$. Thus, according to a general approach proposed in [28, 37], $D_1 \stackrel{\odot}{\longmapsto} D_2$ is a *refinement* of D_2, i.e. $D_1 \overset{D}{\underset{\odot}{\longmapsto}} {}_2$ represents all the information representable by D_2. .

Proposition 3.11 *Let $\langle D, \leq, \odot\rangle$ be a semi-quantale with a right-unit $\mathbf{1} \in D$, and $(D, \alpha_1, D_1, \gamma_1)$ and $(D, \alpha_2, D_2, \gamma_2)$ be G.c.'s such that there exists $\mathbf{1}^\sharp \in D_1$. $\gamma_1(\mathbf{1}^\sharp) = \mathbf{1}$. Then, $(D_1 \stackrel{\odot}{\longmapsto} D_2, \bar{\alpha}, D_2, \bar{\gamma})$ is a G.c., where, for any $d \in D$ and $y \in D_2$, $\bar{\alpha}(\lambda x.\alpha_2(d \odot \gamma_1(x))) = \alpha_2(d)$ and $\bar{\gamma}(y) = \lambda x.\alpha_2(\gamma_2(y) \odot \gamma_1(x))$. Moreover, $(D, \alpha, D_1 \stackrel{\odot}{\longmapsto} D_2, \gamma)$ and $(D_1 \stackrel{\odot}{\longmapsto} D_2, \bar{\alpha}, D_2, \bar{\gamma})$ are coherent with $(D, \alpha_2, D_2, \gamma_2)$.*

Proof. First, observe that $\bar{\alpha}$ is well-defined, namely, if $\lambda x.\alpha_2(d \odot \gamma_1(x)) = \lambda x.\alpha_2(e \odot \gamma_1(x))$, for $d, e \in D$, then $\alpha_2(d) = \alpha_2(d \odot \gamma_1(\mathbf{1}^\sharp)) = \alpha_2(e \odot \gamma_1(\mathbf{1}^\sharp)) = \alpha_2(e)$, viz. $\bar{\alpha}(\lambda x.\alpha_2(d \odot \gamma_1(x))) = \bar{\alpha}(\lambda x.\alpha_2(e \odot \gamma_1(x)))$. Thus, let us show that for any $d \in D$ and $y \in D_2$, $\bar{\alpha}(\lambda x.\alpha_2(d \odot \gamma_1(x))) \leq_2 y \Leftrightarrow \lambda x.\alpha_2(d \odot \gamma_1(x)) \sqsubseteq \bar{\gamma}(y)$, viz. $\alpha_2(d) \leq_2 y \ \Leftrightarrow\ \lambda x.\alpha_2(d \odot \gamma_1(x)) \sqsubseteq \lambda x.\alpha_2(\gamma_2(y) \odot \gamma_1(x))$.
$(\Rightarrow)$ From $\alpha_2(d) \leq_2 y$ we get $d \leq \gamma_2(y)$. Thus, if $x \in D_1$ then $\alpha_2(d \odot \gamma_1(x)) \leq_2 \alpha_2(\gamma_2(y) \odot \gamma_1(x))$.
$(\Leftarrow)$ For $x = \mathbf{1}^\sharp$, we get $\alpha_2(d) = \alpha_2(d \odot \gamma_1(\mathbf{1}^\sharp)) \leq_2 \alpha_2(\gamma_2(y) \odot \gamma_1(\mathbf{1}^\sharp)) = \alpha_2(\gamma_2(y))$. Hence, applying γ_2, we have $d \leq \gamma_2(\alpha_2(d)) \leq \gamma_2(\alpha_2(\gamma_2(y)))$. But, by i in Section 2.2, $\gamma_2(\alpha_2(\gamma_2(y))) = \gamma_2(y)$, from which $\alpha_2(d) \leq_2 y$ follows.
Since $\alpha_2 = \bar{\alpha} \circ \alpha$ trivially holds, we also get coherence. ■

The following example shows the importance of a right-unit in the hypotheses of Proposition 3.11.

Example 3.12 Consider the Example 3.10 above. Clearly, the singleton set $\{[\ ]\}$ is a left- and right-unit for the concrete semi-quantale $\langle \wp(\ell), \subseteq, @\rangle$. However, for a given type τ, $Par(\tau)$ does not represent $\{[\ ]\}$. Thus, Proposition 3.11 is not applicable. In fact, while $\emptyset$, τ, and $\top$ can be represented in $Par(\tau) \xmapsto{@} Par(\tau)$, respectively, by $\lambda X.\emptyset$, $\lambda X.\alpha(\{[\ ]\}@\gamma(X))$ and $\lambda X.\ell$, neither τ_{ev} nor τ_{od} can be represented in $Par(\tau) \xmapsto{@} Par(\tau)$. Therefore, $Par(\tau) \xmapsto{@} Par(\tau)$ is not a refinement of $Par(\tau)$. □

As far as meet-dependencies are concerned, the following result is a consequence of Proposition 3.11.

Corollary 3.13 *If D is a cHa and $(D, \alpha_1, D_1, \gamma_1)$, $(D, \alpha_2, D_2, \gamma_2)$ are Galois connections, then $(D_1 \xmapsto{\wedge} D_2, \bar{\alpha}, D_2, \bar{\gamma})$ is a Galois connection, where, for any $d \in D$ and $y \in D_2$, $\bar{\alpha}(\lambda x.\alpha_2(d \wedge \gamma_1(x))) = \alpha_2(d)$ and $\bar{\gamma}(y) = \lambda x.\alpha_2(\gamma_2(y) \wedge \gamma_1(x))$.*

Proof. If D is a cHa then $\langle D, \leq, \wedge\rangle$ is a unital quantale, having $\top_D$ as unit. Moreover, since $(D, \alpha_1, D_1, \gamma_1)$ is a Galois connection then $\gamma_1(\top_1) = \top_D$ holds. Therefore, the thesis follows by Proposition 3.11. ■

4 Additive Dependencies

In general, for a concrete semi-quantale $\langle D, \leq, \odot\rangle$, from a lattice-theoretic point of view, dependencies in $D_1 \xmapsto{\odot} D_2$ are merely monotone functions. In this section, we give sufficient conditions guaranteeing that $D_1 \xmapsto{\odot} D_2$ is included in, or even isomorphic to, $D_1 \xmapsto{\text{a}} D_2$, i.e. the complete lattice of all additive functions from D_1 to D_2. Our results also permit to give a precise relationship between the operators of reduced relative power and *tensor product* in abstract interpretation,[3] a problem already recognized by Nielson in [52, pag. 124] for Cousot and Cousot's reduced cardinal power. In fact, the tensor product[4] $D_1 \otimes D_2$ of two complete lattices D_1 and D_2 is (isomorphic to) the complete lattice $(D_1 \xmapsto{\text{a}} D_2^{\text{op}})^{\text{op}}$ (cf. [6, 52, 60])[5]. Thus, we also have that $D_1 \xmapsto{\text{a}} D_2 = (D_1 \otimes D_2^{\text{op}})^{\text{op}}$. Therefore, by duality, every known result for the tensor product also holds for the space of additive functions. For instance, it is known (cf. [60, Theorems 2.6 and 4.6]) that $D_1 \otimes D_2$ is a completely distributive (Boolean) lattice iff D_1 and D_2 are completely distributive (Boolean) lattices. Thus, as a sample consequence, this would permit to apply the efficient methods discussed in [61] for immediate fixpoint computation (which are typically useful in program analysis), that are based on certain algebraic properties of distributive (Boolean) lattices.

As shown by the following example, in general, $D_1 \xmapsto{\odot} D_2$ neither includes nor is included in $D_1 \xmapsto{\text{a}} D_2$.

Example 4.1 Consider $Sign \xmapsto{\cap} Parity$ of Example 3.9. In this case, it turns out that $f = \lambda x.\ \alpha_p(\{0\} \cap \gamma_s(x)) \notin Sign \xmapsto{\text{a}} Parity$: In fact, $f(+ \vee_s -) = f(\top_s) = ev$ while $f(+) \vee_p f(-) = \bot_p$. On the other hand, the additive constant function $\lambda x.ev \in Sign \xmapsto{\text{a}} Parity$ does not belong to $Sign \xmapsto{\cap} Parity$: In fact, for any $S \in \wp(\mathbb{Z})$, we have that $\alpha_p(S \cap \gamma_s(\bot_s)) = \alpha_p(\emptyset) = \bot_p \neq ev$. □

We say that a Galois connection (D, α, A, γ) (between complete lattices) is *additive* if γ is additive, or, equivalently, if $\gamma \circ \alpha$ is an additive (upper closure) operator on D (by contrast, recall that α is always additive). If (D, α, A, γ) is additive, then A is called a *disjunctive* abstract domain ([20]). In this case, as recalled in Section 2.2, $A \cong \gamma(\alpha(D))$ is a complete sublattice of D. It is well known since [20] that every abstract domain can be lifted to its *disjunctive completion*: Examples of disjunctive completions can be found e.g. in [21, 23, 29, 36, 41]. The interest in abstract interpretation based on disjunctive domains is evident: Disjunctive abstract domains allow a precise interpretation for disjunction, i.e. concrete and abstract *lub's* coincide up

[3] See e.g. [52, 53, 54, 55] for some applications of the tensor product in abstract interpretation.

[4] See [52, 60] for other equivalent definitions of the tensor product.

[5] The presence of duality in tensor product is due to the fact that we use Galois connections in a dualized form with respect to [60]. In fact, our notion of Galois connection corresponds in [60] to the notion of *residuated mapping*. Therefore, all the results cited from [60] are here derived by duality.

to isomorphic representation of domain objects, and this is essential for a precise abstraction of multiple computation paths (e.g. in functional and logic programming). In our case, when the exponent abstract domain is disjunctive, every dependency actually is an additive function, i.e. $D_1 \overset{\odot}{\longmapsto} D_2 \subseteq D_1 \overset{\mathrm{a}}{\longmapsto} D_2$.

Proposition 4.2 *If $\langle D, \leq, \odot\rangle$ is a quantale and $(D, \alpha_1, D_1, \gamma_1)$ is an additive Galois connection, then $D_1 \overset{\odot}{\longmapsto} D_2$ is a complete join subsemilattice of $D_1 \overset{\mathrm{a}}{\longmapsto} D_2$.*

Proof. The hypotheses say that $\odot$ is both left- and right-additive. The following equalities show that each dependency is additive. If $d \in D$ and $\{x_i\}_{i\in I} \subseteq D_1$ then:

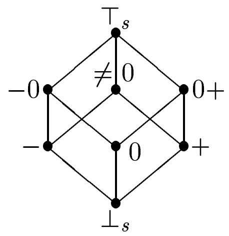
$$\begin{array}{rcll} \alpha_2(d \odot \gamma_1(\vee_{i\in I} x_i)) & = & \alpha_2(d \odot (\vee_{i\in I} \gamma_1(x_i))) & \text{(by additivity of } \gamma_1) \\ & = & \alpha_2(\vee_{i\in I}(d \odot \gamma_1(x_i))) & \text{(by right-additivity of } \odot) \\ & = & \vee_{i\in I} \alpha_2(d \odot \gamma_1(x_i)) & \text{(by additivity of } \alpha_2). \end{array}$$

As recalled in Section 2.1, *lub's* in $D_1 \overset{\mathrm{a}}{\longmapsto} D_2$ are pointwise. Thus, by Proposition 3.7, this observation concludes the proof. ■

Remark 4.3 Under the hypotheses of Proposition 4.2, any dependency $f \in D_1 \overset{\odot}{\longmapsto} D_2$ can be *weakly-inverted*[6], i.e., f actually yields a Galois connection (D_1, f, D_2, f^-), where $f^- = \lambda x. \vee \{y \in D_1 \mid f(y) \leq x\}$ (see point (iv) in Section 2.2). Note that any weak-inverse f^- is co-additive. We denote by $D_2 \rightsquigarrow D_1$ the set of all weak-inverse functions of $D_1 \overset{\odot}{\longmapsto} D_2$, i.e. $D_2 \rightsquigarrow D_1 = \{f^- \in D_2 \longmapsto D_1 \mid f \in D_1 \overset{\odot}{\longmapsto} D_2\}$, equipped with the usual pointwise ordering. It turns out that $D_2 \rightsquigarrow D_1$ specifies, in a dual way, the same information as $D_1 \overset{\odot}{\longmapsto} D_2$, i.e. $(D_2 \rightsquigarrow D_1)^{\mathrm{op}}$ is isomorphic to $D_1 \overset{\odot}{\longmapsto} D_2$. In fact, if $f, g \in D_1 \overset{\odot}{\longmapsto} D_2$ then (D_1, f, D_2, f^-) and (D_1, g, D_2, g^-) are G.c.'s, and it is easy to see that $f \sqsubseteq g$ iff $g^- \sqsubseteq f^-$. Thus, this implies that $D_1 \overset{\odot}{\longmapsto} D_2 \cong (D_2 \rightsquigarrow D_1)^{\mathrm{op}}$. □

It is worth remarking that the inclusion of $D_1 \overset{\odot}{\longmapsto} D_2$ in $D_1 \overset{\mathrm{a}}{\longmapsto} D_2$, as specified by Proposition 4.2 is in general strict, as shown by the following example.

Example 4.4 Consider the following standard abstract domain $Sign^{\pm 0}$ for sign analysis of integer variables ([19, 20]).



$Sign^{\pm 0}$

$(\wp(\mathbb{Z}), \alpha, Sign^{\pm 0}, \gamma)$ is an additive Galois insertion, where the abstraction and concretization maps are as follows ($\mathbb{Z}_{>0}$, $\mathbb{Z}_{\geq 0}$, $\mathbb{Z}_{<0}$, $\mathbb{Z}_{\leq 0}$, and $\mathbb{Z}_{\neq 0}$ denote, respectively, the sets of positive, nonnegative, negative, nonpositive, and nonzero integer numbers).

$$\alpha(S) = \begin{cases} \perp_s & \text{if } S = \emptyset \\ 0 & \text{if } S = \{0\} \\ + & \text{if } \emptyset \neq S \subseteq \mathbb{Z}_{>0} \\ 0+ & \text{if } \{0\} \subset S \subseteq \mathbb{Z}_{\geq 0} \\ - & \text{if } \emptyset \neq S \subseteq \mathbb{Z}_{<0} \\ -0 & \text{if } \{0\} \subset S \subseteq \mathbb{Z}_{\leq 0} \\ \neq 0 & \text{if } S \subseteq \mathbb{Z}_{\neq 0} \text{ and } S \cap \mathbb{Z}_{<0} \neq \emptyset,\ S \cap \mathbb{Z}_{>0} \neq \emptyset \\ \top_s & \text{otherwise} \end{cases} \qquad \gamma(x) = \begin{cases} \emptyset & \text{if } x = \perp_s \\ \{0\} & \text{if } x = 0 \\ \mathbb{Z}_{>0} & \text{if } x = + \\ \mathbb{Z}_{\geq 0} & \text{if } x = 0+ \\ \mathbb{Z}_{<0} & \text{if } x = - \\ \mathbb{Z}_{\leq 0} & \text{if } x = -0 \\ \mathbb{Z}_{\neq 0} & \text{if } x = \neq 0 \\ \mathbb{Z} & \text{if } x = \top_s \end{cases}$$

[6] We adopt here the term weak-inverse for the right-adjoint of a function. A similar abuse of terminology was also in [40].

By Proposition 4.2, $Sign^{\pm 0}$ can be combined with the domain *Parity*, defined in Example 3.9, so that $Sign^{\pm 0} \overset{\sqcap}{\longmapsto} Parity \subseteq Sign^{\pm 0} \overset{\mathrm{a}}{\longmapsto} Parity$. However, this inclusion is strict. In fact, consider $f : Sign^{\pm 0} \overset{\mathrm{a}}{\longmapsto} Parity$ defined by:

$$f(x) = \begin{cases} \bot_p & \text{if } x = \bot_s, \\ od & \text{if } x = 0, \\ \top_p & \text{otherwise.} \end{cases}$$

It turns out that $f \notin Sign^{\pm 0} \overset{\sqcap}{\longmapsto} Parity$: In fact, for any $S \subseteq \mathbb{Z}$, $\alpha_p(S \cap \gamma(0)) \neq od$. □

As a first hypothesis, in the following, we will only consider meet-dependencies, i.e., we will assume that the concrete semi-quantale is a cHa $\langle D, \leq, \wedge \rangle$, that $\gamma_1 : D_1 \longmapsto D$ is a monotone map, and that $(D, \alpha_2, D_2, \gamma_2)$ is a G.c.

It is known (cf. [60, Theorem 2.5]) that for two complete lattices D_1 and D_2, any function in the tensor product $D_1 \otimes D_2$ can be generated as a *lub* (in $D_1 \otimes D_2$) of a suitable set of functions belonging to the *basis* of the tensor product, which is a certain subset $\mathcal{B}(D_1 \otimes D_2) \subseteq D_1 \otimes D_2$. Since $D_1 \otimes D_2^{\mathrm{op}} = (D_1 \overset{\mathrm{a}}{\longmapsto} D_2)^{\mathrm{op}}$, by duality, we have that any additive function $f \in D_1 \overset{\mathrm{a}}{\longmapsto} D_2$ can be generated as a *glb* in $D_1 \overset{\mathrm{a}}{\longmapsto} D_2$ (i.e. *lub* in $D_1 \otimes D_2^{\mathrm{op}}$) of a suitable subset of $\mathcal{B}(D_1 \otimes D_2^{\mathrm{op}})$, where $\mathcal{B}(D_1 \otimes D_2^{\mathrm{op}}) \subseteq D_1 \overset{\mathrm{a}}{\longmapsto} D_2$. In the following, we will denote $\mathcal{B}(D_1 \otimes D_2^{\mathrm{op}})$ simply by $\mathcal{B}(D_1, D_2)$. From the definition of basis of the tensor product given in [60], by duality, we have that $\mathcal{B}(D_1, D_2) = \{\ell_a^b \in D_1 \overset{\mathrm{a}}{\longmapsto} D_2 \mid a \in D_1, b \in D_2\}$, where for any $a, x \in D_1$ and $b \in D_2$:

$$\ell_a^b(x) = \begin{cases} \bot & \text{if } x = \bot, \\ b & \text{if } \bot < x \leq a, \\ \top & \text{if } x \not\leq a. \end{cases}$$

It is an easy task to show that any ℓ_a^b actually is an additive function in $D_1 \overset{\mathrm{a}}{\longmapsto} D_2$, and that for any $\{\ell_{a_i}^{b_i}\}_{i \in I} \subseteq \mathcal{B}(D_1, D_2)$, the pointwise *lub* is given by $\sqcup_{i \in I} \ell_{a_i}^{b_i} = \ell_{\wedge_{i \in I} a_i}^{\vee_{i \in I} b_i}$. Notice also that if $\ell_a^b \in \mathcal{B}(D_1, D_2)$ then $\ell_b^a \in \mathcal{B}(D_2, D_1)$. Moreover, analogously to Remark 4.3, any ℓ_a^b induces a Galois connection $(D_1, \ell_a^b, D_2, (\ell_a^b)^-)$, and it would not be difficult to prove that $(\ell_a^b)^- = \ell_b^a$.

In our context, these additive functions $\ell_a^b \in \mathcal{B}(D_1, D_2)$ play the rôle of (meet) *generators* of $D_1 \overset{\mathrm{a}}{\longmapsto} D_2$ (recall that, in general, the meet in $D_1 \overset{\mathrm{a}}{\longmapsto} D_2$ is not defined pointwise). The following result provides a sufficient condition ensuring that any generator in $\mathcal{B}(D_1, D_2)$ is a meet dependency in $D_1 \overset{\wedge}{\longmapsto} D_2$.

Lemma 4.5 *Suppose that γ_1 is strict and, for any $a \in D_1$, $b \in D_2$, there exists $d_a^b \in D$ such that:* (a) $\forall x \in D_1.\ \bot < x \leq a \Rightarrow \alpha_2(d_a^b \wedge \gamma_1(x)) = b$ *and* (b) $\forall x \in D_1.\ x \not\leq a \Rightarrow \alpha_2(d_a^b \wedge \gamma_1(x)) = \top$. *Then, $\mathcal{B}(D_1, D_2) \subseteq D_1 \overset{\wedge}{\longmapsto} D_2$.*

Proof. For any $\ell_a^b \in \mathcal{B}(D_1, D_2)$, the hypotheses directly imply that $\ell_a^b = \lambda x.\alpha_2(d_a^b \wedge \gamma_1(x))$. ■

The strictness condition for γ_1 (viz. $\gamma_1(\bot) = \bot$) provides here a sufficient condition such that the dependency $\lambda x.\ \alpha_2(d_a^b \wedge \gamma_1(x))$ in the theorem above is a strict function. This is a quite reasonable condition, saying that $\bot$ is the only concrete object whose abstraction in D_1 is the bottom element.

The key point in order to include $\mathcal{B}(D_1, D_2)$ into $D_1 \overset{\wedge}{\longmapsto} D_2$ is therefore the existence of concrete values $d_a^b \in D$ for which conditions (a) and (b) of Lemma 4.5 are verified. A sufficient condition for the existence of such elements is provided by the following result.

Theorem 4.6 *Let $(D, \alpha_1, D_1, \gamma_1)$ be a Galois connection, and assume that the following conditions hold:*

(i) *D_1 is a cBa,*
(ii) *γ_1 is strict,*
(iii) $\forall x \in D_1. \forall y \in D_2.\ x \neq \bot \Rightarrow \alpha_2(\gamma_1(x) \wedge \gamma_2(y)) = y$.

Then, $\mathcal{B}(D_1, D_2) \subseteq D_1 \stackrel{\wedge}{\longmapsto} D_2$.

Proof. We have to prove that for any $a \in D_1$ and $b \in D_2$, $\ell_a^b \in D_1 \stackrel{\wedge}{\longmapsto} D_2$. Let $a \in D_1$ and $b \in D_2$, and define $d_a^b = \gamma_2(b) \vee \gamma_1(a^*) \in D$. Recall that a^* denotes the unique complement of a in the cBa D_1. We show that such d_a^b satisfies the hypotheses of Lemma 4.5. Assume $\bot < x \leq a$. Note that in a Boolean lattice, if $x \leq a$ then $x \wedge a^* = \bot$: This holds because $x \wedge a^* \leq a \wedge a^* = \bot$. Then:

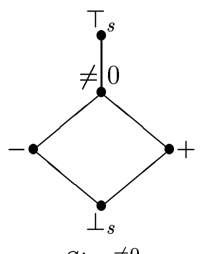
$$\begin{array}{rcll} \alpha_2(d_a^b \wedge \gamma_1(x)) & = & \alpha_2((\gamma_2(b) \vee \gamma_1(a^*)) \wedge \gamma_1(x)) & \\ & = & \alpha_2((\gamma_1(x) \wedge \gamma_2(b)) \vee \gamma_1(a^* \wedge x)) & \text{(by distributivity and } \gamma_1 \text{ co-additivity)} \\ & = & \alpha_2(\gamma_1(x) \wedge \gamma_2(b)) & \text{(since } x \wedge a^* = \bot \text{ and } \gamma_1(\bot) = \bot) \\ & = & b & \text{(since } x \neq \bot \text{ and by (iii)).} \end{array}$$

Assume $x \not\leq a$. By distributivity of D and co-additivity of γ_1, we have that $\alpha_2(d_a^b \wedge \gamma_1(x)) = \alpha_2((\gamma_1(x) \wedge \gamma_2(b)) \vee \gamma_1(a^* \wedge x))$. Thus, $\alpha_2(d_a^b \wedge \gamma_1(x)) \geq \alpha_2(\gamma_1(a^* \wedge x))$. Note that, because D_1 is Boolean, $a^* \wedge x = \bot$ implies $x \leq a^{**} = a$. Hence, $a^* \wedge x = \bot$ iff $x \leq a$. Then, because $x \not\leq a$, we have $a^* \wedge x \neq \bot$. Moreover, since, by G.c., $\gamma_2(\top) = \top$, we have that $\alpha_2(\gamma_1(a^* \wedge x)) = \alpha_2(\gamma_1(a^* \wedge x) \wedge \gamma_2(\top))$. Thus, by (iii) and since $a^* \wedge x \neq \bot$, we get $\alpha_2(\gamma_1(a^* \wedge x)) = \top$, and therefore $\alpha_2(d_a^b \wedge \gamma_1(x)) = \top$. ∎

Hence, this result provides sufficient conditions in order that any additive function in $D_1 \stackrel{\mathrm{a}}{\longmapsto} D_2$ can be constructed as a *glb* (in $D_1 \stackrel{\mathrm{a}}{\longmapsto} D_2$) of dependencies belonging to $D_1 \stackrel{\wedge}{\longmapsto} D_2$. However, it should be noted that, in general, $D_1 \stackrel{\wedge}{\longmapsto} D_2$ is not a meet subsemilattice of $D_1 \stackrel{\mathrm{a}}{\longmapsto} D_2$, and therefore $\mathcal{B}(D_1, D_2) \subseteq D_1 \stackrel{\wedge}{\longmapsto} D_2$ does not imply that $D_1 \stackrel{\wedge}{\longmapsto} D_2$ contains $D_1 \stackrel{\mathrm{a}}{\longmapsto} D_2$. On the other hand, if $D_1 \stackrel{\wedge}{\longmapsto} D_2 \subseteq D_1 \stackrel{\mathrm{a}}{\longmapsto} D_2$ (by Proposition 4.2, this happens, e.g., whenever D_1 is disjunctive), then under the hypotheses of the above result any dependency in $D_1 \stackrel{\wedge}{\longmapsto} D_2$ can be represented as a *glb* (in $D_1 \stackrel{\mathrm{a}}{\longmapsto} D_2$ and therefore in $D_1 \stackrel{\wedge}{\longmapsto} D_2$) of dependencies in $\mathcal{B}(D_1, D_2)$.

In the following, for any $a \in D_1$ and $b \in D_2$, d_a^b will denote a generic element of D as built in the proof of Theorem 4.6, i.e. $d_a^b = \gamma_2(b) \vee \gamma_1(a^*)$. Notice that requiring D_1 to be Boolean in Theorem 4.6 is crucial. In fact, the weaker notion of relative pseudocomplementation (or even pseudocomplementation) is not sufficient, as shown in Example 4.7 below. Indeed, in the proof of Theorem 4.6 we need that for all $a \in D_1$, $a = a^{**}$, and it is well known (cf. [39, 63]) that for a pseudocomplemented lattice this holds if and only if D_1 is Boolean. Actually, the following example shows that condition (i) in Theorem 4.6 is tight.

Example 4.7 Consider the following abstract domain $\mathit{Sign}^{\neq 0}$.



$\mathit{Sign}^{\neq 0}$

$(\wp(\mathbb{Z}), \alpha, \mathit{Sign}^{\neq 0}, \gamma)$ is an additive Galois insertion (actually, it turns out that $\mathit{Sign}^{\neq 0}$ is the disjunctive completion of the domain *Sign* in Example 3.9), where γ is the restriction on $\mathit{Sign}^{\neq 0}$ of the concretization defined in Example 4.4. $\mathit{Sign}^{\neq 0}$ is evidently distributive (and therefore, since it is finite, a cHa), and, together with *Parity* of Example 3.9, it satisfies conditions (ii) and (iii) of Theorem 4.6. However, $\mathit{Sign}^{\neq 0}$ is not Boolean: In fact, $(\neq 0)^{**} = \top_s > (\neq 0)$. It turns out that $\mathit{Sign}^{\neq 0} \stackrel{\sqcap}{\longmapsto} \mathit{Parity}$ does not contain $\mathcal{B}(\mathit{Sign}^{\neq 0}, \mathit{Parity})$. In particular, the generator

$$\ell_{\neq 0}^{ev}(x) = \begin{cases} \bot_p & \text{if } x = \bot_s \\ ev & \text{if } \bot < x \leq \neq 0 \\ \top_p & \text{if } x = \top_s \end{cases}$$

does not belong to $Sign^{\neq 0} \overset{\cap}{\longmapsto} Parity$.

Fact 4.8 $\ell^{ev}_{\neq 0} \notin Sign^{\neq 0} \overset{\cap}{\longmapsto} Parity$.

Proof. Assume by contradiction that $\ell^{ev}_{\neq 0} \in Sign^{\neq 0} \overset{\cap}{\longmapsto} Parity$. Thus, there exists $S \subseteq \mathbb{Z}$ such that $\alpha_p(S \cap \gamma(\neq 0)) = ev$ and $\alpha_p(S \cap \gamma(\top_s)) = \top_p$. Then, we have:

$$\begin{array}{rcll} \alpha_p(S \cap \gamma(\top_s)) & = & \alpha_p(S \cap (\gamma(\neq 0) \cup \{0\})) & (\gamma(\top_s) = \mathbb{Z} = \gamma(\neq 0) \cup \{0\}) \\ & = & \alpha_p((S \cap \gamma(\neq 0)) \cup (S \cap \{0\})) & \\ & = & \alpha_p(S \cap \gamma(\neq 0)) \vee \alpha_p(S \cap \{0\}) & (\text{by additivity of } \alpha_p) \\ & = & ev \vee \alpha_p(S \cap \{0\}) & (\alpha_p(S \cap \gamma(\neq 0)) = ev) \\ & = & ev & (\forall S \subseteq \mathbb{Z}.\ \alpha_p(S \cap \{0\}) \leq ev), \end{array}$$

which is a contradiction. ∎

In particular, due to a too weak form of complementation of $Sign^{\neq 0}$, the element $d^{ev}_{\neq 0} = \gamma(\bot_s) \cup \gamma_p(ev) = \{x \in \mathbb{Z} \mid x \text{ even}\}$ in the proof of Theorem 4.6, is such that

$$\alpha_p(d^{ev}_{\neq 0} \cap \gamma(x)) = \begin{cases} \bot_p & \text{if } x = \bot_s \\ ev & \text{otherwise.} \end{cases}$$

Thus, we can conclude that $\mathcal{B}(Sign^{\neq 0}, Parity) \not\subseteq Sign^{\neq 0} \overset{\cap}{\longmapsto} Parity$, and, by Proposition 4.2, $Sign^{\neq 0} \overset{\cap}{\longmapsto} Parity \subset Sign^{\neq 0} \overset{a}{\longmapsto} Parity$. □

The following easy lemma shows some properties concerning distributivity and complementation. This lemma will be useful later.

Lemma 4.9 *Let D and A be complete lattices and (D, α, A, γ) be an additive Galois insertion.*

(1) *If D is (completely meet-)distributive then A is (completely meet-)distributive.*

(2) *If $a^* \in A$ is a complement of $a \in A$ then $\gamma(a^*) \in D$ is a complement of $\gamma(a) \in D$.*

Proof. (1) We prove only the complete meet-distributive case. The noncomplete one is obviously analogous. Let $x \in A$ and $Y \subseteq A$:

$$\begin{array}{rcll} x \wedge (\vee Y) & = & \alpha(\gamma(x \wedge (\vee Y))) & (\text{by Galois insertion}) \\ & = & \alpha(\gamma(x) \wedge (\vee \gamma(Y))) & (\text{by } \gamma \text{ (co-)additivity}) \\ & = & \alpha(\bigvee_{y \in Y} \gamma(x) \wedge \gamma(y)) & (\text{by complete meet-distributivity}) \\ & = & \alpha(\bigvee_{y \in Y} \gamma(x \wedge y)) & (\text{by } \gamma \text{ co-additivity}) \\ & = & \bigvee_{y \in Y} \alpha(\gamma(x \wedge y)) & (\text{by } \alpha \text{ additivity}) \\ & = & \bigvee_{y \in Y} x \wedge y & (\text{by Galois insertion}). \end{array}$$

(2) We have to prove that $\gamma(a) \wedge \gamma(a^*) = \bot$ and $\gamma(a) \vee \gamma(a^*) = \top$. Notice that additivity of γ implies strictness of γ. By γ co-additivity and strictness, $\gamma(a) \wedge \gamma(a^*) = \gamma(a \wedge a^*) = \gamma(\bot) = \bot$. By γ additivity, $\gamma(a) \vee \gamma(a^*) = \gamma(a \vee a^*) = \gamma(\top) = \top$. ∎

It is worth noting that in Theorem 4.6, as well as in Lemma 4.5, it is not required that D_1 is disjunctive. Note however that if $(D, \alpha_1, D_1, \gamma_1)$ is additive, then the strictness condition (ii) always holds. Moreover, in this case, by Lemma 4.9 (1), D_1 is a cHa, and, in particular, it is pseudocomplemented. Thus, in order to verify that D_1 is Boolean (viz. condition (i) of Theorem 4.6), it is sufficient to check whether pseudocomplementation in D_1 is idempotent, i.e. $\forall x \in D_1.\ x = x^{**}$.

Condition (iii) of Theorem 4.6 gives a constraint on the interaction between the abstract domains D_1 and D_2. One consequence of condition (iii) is that D_2 represents no information on the (nonbottom) abstract values of D_1, i.e., $\forall x \neq \bot \in D_1.\ \alpha_2(\gamma_1(x)) = \top$. We can observe that while condition (iii) is satisfied for the abstract domains $D_1 = Sign$ and $D_2 = Parity$ of Example 3.9, it does not hold if $Sign$ is replaced by $Sign^{\pm 0}$ of Example 4.4: In fact, $\alpha_2(\gamma_1(0) \cap \gamma_2(od)) = \bot_p \neq od$.

This is consistent with Example 4.4, where it is shown that $\mathit{Sign}^{\pm 0} \stackrel{\cap}{\longmapsto} \mathit{Parity}$ does not contain the generator $\ell_0^{od} \in \mathcal{B}(\mathit{Sign}^{\pm 0}, \mathit{Parity})$ (denoted by f in that example). Since $\mathit{Sign}^{\pm 0}$ and Parity satisfy both conditions (i) and (ii) of Theorem 4.6, this observation also implies that condition (iii) in Theorem 4.6 is tight. On the other hand, the following example shows that $\mathit{Parity} \stackrel{\cap}{\longmapsto} \mathit{Sign}$ contains the whole set of generators of $\mathit{Parity} \stackrel{\mathrm{a}}{\longmapsto} \mathit{Sign}$.

Example 4.10 Consider the domains *Sign* and *Parity* of Example 3.9. It is easy to prove that *Sign* is the only domain among *Sign*, $\mathit{Sign}^{\neq 0}$, and $\mathit{Sign}^{\pm 0}$ satisfying conditions (i)–(iii) of Theorem 4.6, whenever joined, either as base or as exponent, with *Parity*. However, note that, while $\mathit{Sign} \stackrel{\cap}{\longmapsto} \mathit{Parity}$ contains, by Theorem 4.6, the generators of $\mathit{Sign} \stackrel{\mathrm{a}}{\longmapsto} \mathit{Parity}$, it may also contain nonadditive functions. This because *Sign* is not a disjunctive abstraction of $\wp(\mathbb{Z})$ (cf. Proposition 4.2). On the other hand, since $(\wp(\mathbb{Z}), \alpha_p, \mathit{Parity}, \gamma_p)$ is an additive Galois insertion, by Proposition 4.2, we have that $\mathit{Parity} \stackrel{\cap}{\longmapsto} \mathit{Sign} \subseteq \mathit{Parity} \stackrel{\mathrm{a}}{\longmapsto} \mathit{Sign}$. Moreover, by Theorem 4.6, $\mathit{Parity} \stackrel{\cap}{\longmapsto} \mathit{Sign}$ also contains all the generators of $\mathit{Parity} \stackrel{\mathrm{a}}{\longmapsto} \mathit{Sign}$. The generators are here all constructed as dependencies by using the following sets of integer numbers:

$$\begin{gathered}
d_{\perp_p}^{\perp_s} = d_{\perp_p}^{+} = d_{\perp_p}^{-} = d_{\perp_p}^{\top_s} = d_{ev}^{\top_s} = d_{od}^{\top_s} = d_{\top_p}^{\top_s} = \mathbb{Z}, \quad d_{\top_p}^{\perp_s} = \emptyset, \\
d_{\top_p}^{+} = \{x \in \mathbb{Z} \mid x > 0\}, \quad d_{\top_p}^{-} = \{x \in \mathbb{Z} \mid x < 0\}, \\
d_{ev}^{\perp_s} = \{x \in \mathbb{Z} \mid x \text{ odd}\}, \quad d_{od}^{\perp_s} = \{x \in \mathbb{Z} \mid x \text{ even}\}, \\
d_{ev}^{+} = \{x \in \mathbb{Z} \mid x > 0\} \cup \{x \in \mathbb{Z} \mid x \text{ odd}\}, \quad d_{ev}^{-} = \{x \in \mathbb{Z} \mid x < 0\} \cup \{x \in \mathbb{Z} \mid x \text{ odd}\}, \\
d_{od}^{+} = \{x \in \mathbb{Z} \mid x > 0\} \cup \{x \in \mathbb{Z} \mid x \text{ even}\}, \quad d_{od}^{-} = \{x \in \mathbb{Z} \mid x < 0\} \cup \{x \in \mathbb{Z} \mid x \text{ even}\}.
\end{gathered}$$

Thus, any dependency in $\mathit{Parity} \stackrel{\cap}{\longmapsto} \mathit{Sign}$ can be represented as a *glb* (in $\mathit{Parity} \stackrel{\mathrm{a}}{\longmapsto} \mathit{Sign}$ and therefore in $\mathit{Parity} \stackrel{\cap}{\longmapsto} \mathit{Sign}$) of some generators. For instance, the dependency $\lambda x.\alpha_s(\{-1, 2\} \cap \gamma_p(x)) = \{\perp_p \mapsto \perp_s, ev \mapsto +, od \mapsto -, \top_p \mapsto \top_s\}$ is the *glb* (in this case pointwise) of the generators $\lambda x.\alpha_s(d_{ev}^{+} \cap \gamma_p(x)) = \{\perp_p \mapsto \perp_s, ev \mapsto +, od \mapsto \top_p, \top_p \mapsto \top_s\}$ and $\lambda x.\alpha_s(d_{od}^{-} \cap \gamma_p(x)) = \{\perp_p \mapsto \perp_s, ev \mapsto \top_s, od \mapsto -, \top_p \mapsto \top_s\}$. □

Let us assume that $\langle D, \leq_D \rangle = \langle \wp(X), \subseteq \rangle$, for some set X. Then, under the hypotheses of Theorem 4.6, the following result shows that for any $a \in D_1$ and $b \in D_2$, the concretization of the generator ℓ_a^b is given by $\gamma(\ell_a^b) = \gamma_1(a^*) \cup \gamma_2(b)$. This gives a logical counterpart to generators: In fact, because by Lemma 4.9 $\gamma_1(a^*) \cup \gamma_2(b) = \overline{\gamma_1(a)} \cup \gamma_2(b)$, any ℓ_a^b represents precisely the classical implicational relation $x \in \gamma_1(a) \Rightarrow x \in \gamma_2(b)$.

Lemma 4.11 *Let $\langle D, \leq_D \rangle = \langle \wp(X), \subseteq \rangle$, for some set X, $(D, \alpha_1, D_1, \gamma_1)$ be an additive Galois insertion, and $(D, \alpha_2, D_2, \gamma_2)$ be a Galois insertion. Let us assume that conditions* (i) *and* (iii) *of Theorem 4.6 are satisfied. Then, for any $a \in D_1$ and $b \in D_2$, $\gamma(\ell_a^b) = \gamma_1(a^*) \cup \gamma_2(b)$.*

Proof. Firstly, observe that by Lemma 4.9 (1), D_1 is a cHa. By definition, $\gamma(\ell_a^b)$ is the largest set in $\wp(X)$ such that $\lambda x.\alpha_2(\gamma(\ell_a^b) \cap \gamma_1(x)) = \ell_a^b$. Moreover, by the proof of Theorem 4.6, $\lambda x.\alpha_2((\gamma_1(a^*) \cup \gamma_2(b)) \cap \gamma_1(x)) = \ell_a^b$. Thus, assume, by contradiction, that $\gamma_1(a^*) \cup \gamma_2(b) \subset \gamma(\ell_a^b)$. Hence, if $d = \gamma(\ell_a^b) \setminus (\gamma_1(a^*) \cup \gamma_2(b))$, then $d \neq \emptyset$. Since, by hypothesis, D_1 is Boolean and $(\wp(X), \alpha_1, D_1, \gamma_1)$ is additive, by Lemma 4.9 (2), for any $x \in D_1$, $\gamma_1(x^*) = \overline{\gamma_1(x)}$. Moreover, $\gamma_1(\perp) = \emptyset$ holds. Hence, from $d \cap \gamma_1(a^*) = \emptyset$, we get $d \subseteq \gamma_1(a)$. Further, $a \neq \perp$, otherwise $\gamma_1(a^*) = \overline{\gamma_1(a)} = X$, which implies $d \cap \gamma_1(a^*) = d \neq \emptyset$, which would be a contradiction. We have that $\alpha_2(\gamma(\ell_a^b) \cap \gamma_1(a)) = \ell_a^b(a) = b$. Thus, the following equalities hold:

$$\begin{array}{rcll}
b & = & \alpha_2(\gamma(\ell_a^b) \cap \gamma_1(a)) & \\
 & = & \alpha_2((d \cup \gamma_1(a^*) \cup \gamma_2(b)) \cap \gamma_1(a)) & \\
 & = & \alpha_2((d \cap \gamma_1(a)) \cup (\gamma_1(a^*) \cap \gamma_1(a)) \cup (\gamma_2(b) \cap \gamma_1(a))) & \\
 & = & \alpha_2((d \cap \gamma_1(a)) \cup (\gamma_2(b) \cap \gamma_1(a))) & \text{(by complementation)} \\
 & = & \alpha_2(d \cap \gamma_1(a)) \vee \alpha_2(\gamma_2(b) \cap \gamma_1(a)) & \text{(by additivity of } \alpha_2\text{)} \\
 & = & \alpha_2(d \cap \gamma_1(a)) \vee b & \text{(by (iii) of Theorem 4.6).}
\end{array}$$

Hence, $\gamma_2(\alpha_2(d \cap \gamma_1(a))) \subseteq \gamma_2(b)$. Since $d \subseteq \gamma_1(a)$, we get $\gamma_2(\alpha_2(d)) \subseteq \gamma_2(b)$, which implies that $d \subseteq \gamma_2(b)$. But this is a contradiction for $d \neq \emptyset$, because $d \cap \gamma_2(b) = \emptyset$. ■

The last result of this section shows that under the hypotheses of Lemma 4.11, if D_1 is an atomic cBa (and, as recalled in Section 2.1, this means that D_1 is isomorphic to a powerset $\langle \wp(Y), \subseteq \rangle$ for some set Y) then $D_1 \stackrel{\wedge}{\longmapsto} D_2 = D_1 \stackrel{\mathrm{a}}{\longmapsto} D_2$ (and therefore, since both are ordered pointwise, they actually are the same complete lattice). Note that, under these hypotheses and in view of Remark 4.3, $D_1 \otimes D_2 = (D_2^{\mathrm{op}} \rightsquigarrow D_1)$.

Theorem 4.12 *Assume that the hypotheses in Lemma 4.11 hold and that D_1 is an atomic cBa. Then, $D_1 \stackrel{\wedge}{\longmapsto} D_2 = D_1 \stackrel{\mathrm{a}}{\longmapsto} D_2$.*

Proof. First notice that condition (i) in Theorem 4.6 is satisfied (since γ_1 is additive). Then, by Proposition 4.2 and Theorem 4.6, we have that $\mathcal{B}(D_1, D_2) \subseteq D_1 \stackrel{\wedge}{\longmapsto} D_2 \subseteq D_1 \stackrel{\mathrm{a}}{\longmapsto} D_2$. We prove that $D_1 \stackrel{\mathrm{a}}{\longmapsto} D_2 \subseteq D_1 \stackrel{\wedge}{\longmapsto} D_2$. Let $f \in D_1 \stackrel{\mathrm{a}}{\longmapsto} D_2$. Recall that $At(D_1)$ is the set of all the atoms of D_1. We prove that $f = \sqcap\{\ell_a^{f(a)} \mid a \in At(D_1)\}$ (where $\sqcap$ is the *glb* in $D_1 \stackrel{\wedge}{\longmapsto} D_2$). By Lemma 4.11, we know that for any $a \in D_1$ and $b \in D_2$, $\gamma(\ell_a^b) = \gamma_1(a^*) \cup \gamma_2(b)$. Recall that $(D, \alpha, D_1 \stackrel{\wedge}{\longmapsto} D_2, \gamma)$ is G.i. Also, observe that for any $a, x \in At(D_1)$, if $x \neq a$ then $x \leq a^*$: In fact, $x \wedge a = \bot$ implies $x \leq a^*$. Then, for any $x \in At(D_1)$, we have:

$$
\begin{array}{rcll}
(\sqcap\{\ell_a^{f(a)} \mid a \in At(D_1)\})(x) & = & \alpha(\gamma(\sqcap\{\ell_a^{f(a)} \mid a \in At(D_1)\}))(x) & (\text{since } \alpha \circ \gamma = \lambda x.x) \\
& = & \alpha(\cap\{\gamma(\ell_a^{f(a)}) \mid a \in At(D_1)\})(x) & \\
& & & (\text{by co-additivity of } \gamma) \\
& = & \alpha_2(\bigcap_{a \in At(D_1)} (\gamma_1(a^*) \cup \gamma_2(f(a))) \cap \gamma_1(x)) & \\
& & & (\text{by co-additivity of } \gamma_1) \\
& = & \alpha_2(\bigcap_{a \in At(D_1)} (\gamma_1(a^* \wedge x) \cup (\gamma_2(f(a)) \cap \gamma_1(x)))) & \\
& & & (\text{since } x \neq a \Rightarrow a^* \wedge x = x) \\
& = & \alpha_2(\gamma_2(f(x)) \cap \gamma_1(x)) & \\
& & & (\text{since } x \neq \bot, \text{ and by (iii) in Theorem 4.6}) \\
& = & f(x). &
\end{array}
$$

Since any additive function on an atomic domains is fully determined by its values on the atoms, actually we get $f = \sqcap\{\ell_a^{f(a)} \mid a \in At(D_1)\} \in D_1 \stackrel{\wedge}{\longmapsto} D_2$, and this concludes the proof. ■

Example 4.13 The base *Parity* and exponent *Sign* (both in Example 3.9) satisfy all the hypotheses of Theorem 4.12. Thus, $\mathit{Parity} \stackrel{\cap}{\longmapsto} \mathit{Sign} = \mathit{Parity} \stackrel{\mathrm{a}}{\longmapsto} \mathit{Sign}$. In particular, $\mathit{Parity} \stackrel{\cap}{\longmapsto} \mathit{Sign} = (\mathit{Parity} \otimes \mathit{Sign}^{\mathrm{op}})^{\mathrm{op}}$ also holds. As a sample consequence of this fact, since both *Parity* and *Sign* are finite cBa's, we get that $\mathit{Parity} \stackrel{\cap}{\longmapsto} \mathit{Sign}$ is a finite cBa as well. Actually, (the image of) $\mathit{Parity} \stackrel{\cap}{\longmapsto} \mathit{Sign}$ (in $\wp(\mathbb{Z})$) can be easily computed by considering the closure by intersection of all the elements d_a^b, for $a \in \mathit{Parity}$ and $b \in \mathit{Sign}$, of Example 4.10. □

5 Autodependencies

An important case of reduced relative power of abstract domains is given whenever the base and exponent domains coincide. In this case the elements of the reduced relative power are called *autodependencies*.

Definition 5.1 Let $\langle D, \leq, \odot \rangle$ be a semi-quantale and (D, α, A, γ) be a Galois connection. The lattice of *autodependencies* of A is $A \stackrel{\odot}{\longmapsto} A$. □

Thus, $A \stackrel{\odot}{\longmapsto} A = \{\lambda x.\alpha(d \odot \gamma(x)) \in A \stackrel{\mathrm{m}}{\longmapsto} A \mid d \in D\}$. In the following, we adopt a more concise notation and denote $A \stackrel{\odot}{\longmapsto} A$ by $\mathit{Dep}_{\odot}(A)$. Note that $\mathit{Dep}_{\odot}(A)$ is precisely the set of all best correct approximations[7] of the family of concrete operations $\{\lambda x.d \odot x\}_{d \in D} \subseteq D \stackrel{\mathrm{m}}{\longmapsto} D$. By

[7] If (D, α, A, γ) is a Galois connection and $f : D \longmapsto D$ then the *best correct approximation* of f in A is defined as $\alpha \circ f \circ \gamma : A \longmapsto A$ (cf. [20]).

Proposition 3.7, $Dep_{\odot}(A)$ is a complete lattice w.r.t. the pointwise ordering $\sqsubseteq$, where the *lub* is defined pointwise, the top is $\lambda x.\alpha(\top_D \odot \gamma(x))$, and the bottom is $\lambda x.\alpha(\bot_D \odot \gamma(x))$. We denote by $(D, \alpha^{\natural}, Dep_{\odot}(A), \gamma^{\natural})$ the Galois insertion of $Dep_{\odot}(A)$ in A given by Theorem 3.6. Also, note that under the hypotheses of Proposition 3.11, $Dep_{\odot}(A)$ is a coherent refinement of A.

Autodependencies for an abstract domain constitute an important case, because, under certain hypotheses, $Dep_{\odot}(A)$ coincides with well known function spaces. As a first result, when $\odot$ is an idempotent (i.e., $\forall x.\ x \odot x = x$) lower (upper) bound (i.e., $\forall x, y.\ x \odot y \le x, y$ $(x, y \le x \odot y)$) on D, we have that each autodependency is a lower (upper) closure operator on A.

Theorem 5.2 *Let $\langle D, \le, \odot\rangle$ be a semi-quantale where $\odot$ is idempotent, and (D, α, A, γ) be a Galois connection.*

(i) *If $\odot$ is a lower bound then $Dep_{\odot}(A)$ is a complete join subsemilattice of $lco(A)$.*

(ii) *If (D, α, A, γ) is a G.i. and $\odot$ is an upper bound such that for any $d, d' \in D$, $d \odot \gamma(\alpha(d')) \le \gamma(\alpha(d \odot d'))$, then $Dep_{\odot}(A)$ is a complete join subsemilattice of $uco(A)$.*

Proof. (i) We show that for any $d \in D$, $\lambda x.\alpha(d \odot \gamma(x)) \in lco(A)$. Monotonicity is given by Lemma 3.2. The following inequalities prove idempotency:

$$\begin{array}{rcll}
\alpha(d \odot \gamma(x)) & = & \alpha((d \odot d) \odot \gamma(x)) & (\odot \text{ is idempotent}) \\
& = & \alpha(d \odot (d \odot \gamma(x))) & (\odot \text{ is associative}) \\
& \le & \alpha(d \odot \gamma(\alpha(d \odot \gamma(x)))) & (\text{by G.c. and monotonicity}) \\
& \le & \alpha(\gamma(\alpha(d \odot \gamma(x)))) & (\odot \text{ is a lower bound}) \\
& = & \alpha(d \odot \gamma(x)) & (\text{by G.c.}).
\end{array}$$

Finally, by G.c. and since $\odot$ is a lower bound, $\alpha(d \odot \gamma(x)) \le \alpha(\gamma(x)) \le x$, which proves reductivity. Further, $Dep_{\odot}(A)$ is a complete join subsemilattice of $lco(A)$, since in both lattices the *lub* is defined pointwise (cf. Proposition 3.7).
(ii) We show that for any $d \in D$, $\lambda x.\alpha(d \odot \gamma(x)) \in uco(A)$. Monotonicity is given by Lemma 3.2. The following inequalities show idempotency:

$$\begin{array}{rcll}
\alpha(d \odot \gamma(\alpha(d \odot \gamma(x)))) & \le & \alpha(\gamma(\alpha(d \odot (d \odot \gamma(x))))) & (\text{by hypothesis on } \odot) \\
& = & \alpha(d \odot (d \odot \gamma(x))) & (\text{by G.c.}) \\
& = & \alpha(d \odot \gamma(x)) & (\odot \text{ is associative and idempotent}) \\
& \le & \alpha(d \odot \gamma(\alpha(d \odot \gamma(x)))) & (\text{by G.c.}).
\end{array}$$

By G.i. and since $\odot$ is an upper bound, $x = \alpha(\gamma(x)) \le \alpha(d \odot \gamma(x))$, which proves extensivity. Finally, since the *lub* in $Dep_{\odot}(A)$ is pointwise, $Dep_{\odot}(A)$ is a complete join subsemilattice of $uco(C)$ (although, the *lub* in $uco(A)$, in general, is not pointwise). ∎

In the following, we consider $\odot$ either as the *glb* or as the *lub* of D. Therefore, in the former case when $\odot = \wedge$, the concrete domain D is a cHa. A generic function in $Dep_{\wedge}(A)$ ($Dep_{\vee}(A)$) is called a *meet-autodependency* (*join-autodependency*). It is easy to show that in such cases the hypotheses for $\odot$ in Theorem 5.2 are satisfied.

Corollary 5.3 *Let (D, α, A, γ) be a Galois connection.*

(i) *If D is a cHa then $A \overset{\wedge}{\longmapsto} A$ is a complete join subsemilattice of $lco(A)$.*

(ii) *If (D, α, A, γ) is a G.i. then $A \overset{\vee}{\longmapsto} A$ is a complete join subsemilattice of $uco(A)$.*

Proof. (i) This follows by Theorem 5.2 (i), since $\wedge_D$ is obviously an idempotent lower bound.
(ii) $\vee_D$ is trivially an idempotent upper bound. Moreover, if $d, d' \in D$ then:

$$\begin{array}{rcll}
d \vee_D \gamma(\alpha(d')) & \le & \gamma(\alpha(d \vee_D \gamma(\alpha(d')))) & (\text{by G.c.}) \\
& = & \gamma(\alpha(d) \vee_A \alpha(\gamma(\alpha(d')))) & (\alpha \text{ is additive}) \\
& = & \gamma(\alpha(d) \vee_A \alpha(d')) & (\text{by G.c.}) \\
& = & \gamma(\alpha(d \vee_D d')) & (\alpha \text{ is additive}),
\end{array}$$

which proves the inequality of Theorem 5.2 (ii). ∎

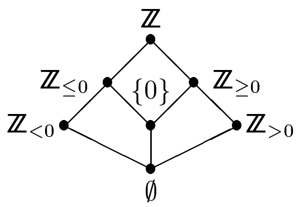
In the following, we will give a number of additional results for meet- and join-autodependencies. While meet-autodependencies provide an interesting abstract domain refinement, join-autodependencies add no further information to their input abstract domain. In fact, it turns out that $Dep_\vee(A)$ is always isomorphic to A.

Proposition 5.4 *If (D, α, A, γ) is a Galois insertion then $Dep_\vee(A) \cong A$.*

Proof. Note that any $\lambda x.\alpha(d \vee_D \gamma(x)) \in Dep_\vee(A)$ actually can be rewritten as $\lambda x.\alpha(d) \vee_A x$. Thus, $Dep_\vee(A) = \{\lambda x.a \vee_A x \mid a \in A\}$. Hence, the isomorphism is given by $(\lambda x.a \vee x) \leftrightarrow a$. In fact, it is immediate to see that that for any $a, a' \in A$, $a \le a'$ iff $\lambda x.a \vee x \sqsubseteq \lambda x.a' \vee x$. ∎

Thus, since join-autodependencies behave like a trivial identity operator on domains, in the following, we will mainly focus on meet-autodependencies only. The following is a simple example of meet-autodependencies.

Example 5.5 Consider the domain *Sign* of Example 3.9. By Corollary 5.3 (i), we have that $Dep_\cap(Sign) \subseteq lco(Sign)$. On the other hand, it is a simple task to check that each lower closure on *Sign* actually is a meet-autodependency belonging to $Dep_\cap(Sign)$, and therefore $Dep_\cap(Sign) = lco(Sign)$ (later, we will derive this fact as a consequence of a general result). Thus, $Dep_\cap(Sign)$ is precisely the following abstract domain, where any $S \in \wp(\mathbb{Z})$ identifies the dependency $\lambda x.\alpha_s(S \cap \gamma(x))$ (indeed, it is its concretization).



For instance, the lower closure having as set of fixpoints $\{\top_s, \bot_s\}$ corresponds to the dependency $\lambda x.\alpha(\{0\} \cap \gamma(x))$. □

The following result is an instance of Proposition 3.11 to the case of meet-autodependencies, and shows the explicit Galois insertion of A into $Dep_\wedge(A)$.

Corollary 5.6 *If D is a cHa and (D, α, A, γ) is a Galois connection then $(Dep_\wedge(A), \bar{\alpha}, A, \bar{\gamma})$ is a Galois insertion, where, for any $d \in D$ and $a \in A$, $\bar{\alpha}(\lambda x.\alpha(d \wedge \gamma(x))) = \alpha(d)$ and $\bar{\gamma}(a) = \lambda x.a \wedge x$. Moreover, $(D, \alpha^\natural, Dep_\odot(A), \gamma^\natural)$ and $(Dep_\wedge(A), \bar{\alpha}, A, \bar{\gamma})$ are coherent with (D, α, A, γ).*

Proof. Note that $\bar{\gamma}$ actually is an instance of the more general concretization map of Proposition 3.11, since, for any $a \in A$, $\lambda x.a \wedge x = \lambda x.\alpha(\gamma(a \wedge x)) = \lambda x.\alpha(\gamma(a) \wedge \gamma(x))$. Moreover, $\bar{\gamma}$ is injective, since by Corollary 5.3 (i), $Dep_\wedge(A) \subseteq lco(A)$, and each lower closure $\lambda x.a \wedge x$ uniquely determines a different dual-Moore-set, that is $\downarrow a$. Thus, by Proposition 3.11, $(Dep_\wedge(A), \bar{\alpha}, A, \bar{\gamma})$ is a Galois insertion. Coherence is a straight consequence of Proposition 3.11. ∎

By Corollary 5.3 (i), we know that for any abstract domain A, each meet-autodependency relative to A is a lower closure on A, i.e. $Dep_\wedge(A) \subseteq lco(A)$. In the following, we characterize precisely the class of Galois insertions (D, α, A, γ) for which $Dep_\wedge(A) = lco(A)$ holds. This class of abstract domains includes some relevant examples, such as a well known domain for logic program groundness analysis, as shown later in Section 6.

Definition 5.7 A Galois insertion (D, α, A, γ) is *separated* if for any $a \in A$, there exists $d \in D$ such that: (i) $\alpha(d) = a$, (ii) $\forall x \in A.\ (a \not\le_A x) \Rightarrow (d \wedge \gamma(x) \le_D \gamma(\bot_A))$. □

For a separated Galois insertion (D, α, A, γ), we also say that A is separated in D, or simply that A is separated, when D is clear from the context. A few observations about Definition 5.7 are needed. First, notice that the above definition implies that α is surjective, and therefore this justifies the requirement that (D, α, A, γ) is a Galois insertion. Also note that, for a given $a \in A$, the element d in Definition 5.7 can be different from $\gamma(a)$ (see Example 5.8 below). Finally, note that (i) and (ii) trivially hold for $a = \perp_A$, by choosing $d = \perp_D$. Intuitively, each element a of a separated abstract domain A approximates precisely some concrete object $d \in D$ (i.e. $\alpha(d) = a$), which is disjoint with the information represented by any abstract object which is not above x. We call such a d a separating element for a. In a sense, the object d is hidden in D just inside a, and it "separates" a from the abstract objects not above a. The following example helps to clarify the notion of separatedness.

Example 5.8 It turns out that the abstract domain *Sign* of Example 3.9 is separated in $\wp(\mathbb{Z})$. For any $a \in \mathit{Sign}$, a corresponding separating element $d_a \in \wp(\mathbb{Z})$ can be chosen as follows:

$$d_{\top_s} = \{0\}; \quad d_- = \gamma_s(-) = \mathbb{Z}_{<0}; \quad d_+ = \gamma_s(+) = \mathbb{Z}_{>0}; \quad d_{\perp_s} = \gamma_s(\perp_s) = \emptyset.$$

It is easy to check that for them the conditions (i) and (ii) in Definition 5.7 hold. Also, note that $d_{\top_s} \neq \gamma_s(\top_s) = \mathbb{Z}$, since $\mathbb{Z}$ does not satisfy the requested conditions. In this case, $d_{\top_s} = \{0\}$ intuitively separates $\top_s$ from $-$ and $+$ (since $\{0\} \cap \gamma_s(-) = \{0\} \cap \gamma_s(+) = \emptyset$). Thus, in this sense, although the concrete value $\{0\}$ is not represented in *Sign*, the property of separatedness says that it is hidden in *Sign* inside $\top_s$. □

On the other hand, the following example shows two nonseparated abstract domains.

Example 5.9 The abstract domains $\mathit{Sign}^{\neq 0}$ and $\mathit{Dep}_\cap(\mathit{Sign})$ of Examples 4.7 and 5.5, respectively, are not separated in $\wp(\mathbb{Z})$. In fact, consider $\neq 0 \in \mathit{Sign}^{\neq 0}$ and $\mathbb{Z}_{\leq 0} \in \mathit{Dep}_\cap(\mathit{Sign})$: It is straightforward to verify that for them the conditions of Definition 5.7 are not satisfied. □

Whenever the concrete domain is a cHa, the following interesting property of separated Galois insertions holds.

Proposition 5.10 *If D is a cHa and (D, α, A, γ) is a separated Galois insertion, then, for any $x \in A \setminus \{\perp_A\}$, $\vee_D\{\gamma(y) \mid y \in A,\ y <_A x\} <_D \gamma(x)$.*

Proof. Assume that $x \in A \setminus \{\perp_A\}$, and that, by contradiction, $\vee_D\{\gamma(y) \mid y \in A,\ y <_A x\} = \gamma(x)$. By separatedness, there exists a suitable $d_x \in D$ such that $\alpha(d_x) = x$. Since $d_x \leq_D \gamma(\alpha(d_x)) = \gamma(x)$, the following equalities hold:

$$\begin{array}{rcll} d_x & = & d_x \wedge_D (\vee_D\{\gamma(y) \mid y \in A,\ y <_A x\}) & \\ & = & \vee_D\{d_x \wedge_D \gamma(y) \mid y \in A,\ y <_A x\} & \text{(since } D \text{ is a cHa)} \\ & \leq_D & \gamma(\perp_A) & \text{(by separatedness).} \end{array}$$

Thus, $x = \alpha(d_x) \leq_A \alpha(\gamma(\perp_A)) = \perp_A$, i.e. $x = \perp_A$, which is a contradiction. ■

The following observation will be useful later on.

Remark 5.11 The above result implies that if $x \in A \setminus \{\perp_A\}$ is join-reducible, i.e. there exists $X \subseteq A$ such that $x \notin X$ and $x = \vee_A X$, then $\vee_D \gamma(X) <_D \gamma(\vee_A X)$. In other terms, if a G.i. (D, α, A, γ) is separated and A contains join-reducible elements different from the bottom, then (D, α, A, γ) is not additive. Thus, if A is disjunctive and contains join-reducible elements different from the bottom, then A is not separated. □

For instance, since $\mathit{Sign}^{\neq 0}$ of Example 4.7 (abstracting the cHa $\wp(\mathbb{Z})$) is disjunctive and its element $\neq 0$ is join-reducible, we get that $\mathit{Sign}^{\neq 0}$ is not separated, as we already observed in Example 5.9.

If we consider concrete domains that are powerset of some set, then the converse of Proposition 5.10 holds, thus providing an alternative characterization for separatedness.

Proposition 5.12 *Let $\langle D, \leq_D \rangle = \langle \wp(X), \subseteq \rangle$, for some set X, and (D, α, A, γ) be a Galois insertion. Then, (D, α, A, γ) is separated iff $\forall x \in A \setminus \{\bot_A\}.\ x \notin JI(A) \Rightarrow \cup\{\gamma(y) \mid y \in A,\ y <_A x\} \subset \gamma(x)$.*

Proof. By Proposition 5.10, it suffices to prove only the "if" direction. Since Definition 5.7 is satisfied for $x = \bot_A$, we assume that $x \in A \setminus \{\bot_A\}$. We distinguish two cases.
(1) x is join-irreducible. Define $a_x = \vee_A\{z \in A \mid z <_A x\}$. By join-irreducibility of x, it follows that $a_x <_A x$. Now, define $d_x = \gamma(x) \setminus \gamma(a_x)$. Clearly, $\alpha(d_x) \leq_A \alpha(\gamma(x)) = x$. If we assume that $\alpha(d_x) <_A x$, we would get $\alpha(d_x) \leq_A a_x$, from which $d_x \subseteq \gamma(a_x)$, namely $\gamma(x) \setminus \gamma(a_x) \subseteq \gamma(a_x)$, which in turn implies $\gamma(x) \subseteq \gamma(a_x)$, from which we get $x \leq a_x$, i.e. a contradiction. Thus, $\alpha(d_x) = x$. Consider now $y \in A$ such that $x \not\leq_A y$. Then, $d_x \cap \gamma(y) = (\gamma(x) \cap \gamma(y)) \setminus \gamma(a_x) = \gamma(x \wedge_A y) \setminus \gamma(a_x)$. On the other hand, $x \wedge_A y <_A x$ implies $x \wedge_A y \leq_A a_x$, from which we have $\gamma(x \wedge_A y) \subseteq \gamma(a_x)$. Hence, we get $\gamma(x \wedge_A y) \setminus \gamma(a_x) = \emptyset$, which implies $d_x \cap \gamma(y) = \emptyset$.
(2) x is join-reducible. In this case, $x = \vee_A\{z \in A \mid z <_A x\}$. Define $d_x = \gamma(x) \setminus (\cup\{\gamma(z) \mid z <_A x\})$. Note that, by hypothesis, $d_x \neq \emptyset$. Trivially, $\alpha(d_x) \leq_A x$. If we suppose that $\alpha(d_x) <_A x$, we would get $d_x = \gamma(\alpha(d_x)) \cap d_x = \emptyset$, which is a contradiction. Therefore, $\alpha(d_x) = x$. Consider now $y \in A$ such that $x \not\leq_A y$. Then, $x \wedge_A y <_A x$, from which $\gamma(x \wedge_A y) \cap d_x = \gamma(x) \cap \gamma(y) \cap d_x = \emptyset$. But $d_x \subseteq \gamma(x)$, from which $d_x \cap \gamma(y) = \emptyset$. ■

As announced above, for separated abstract domains the following important result holds.

Theorem 5.13 *Let D be a cHa and (D, α, A, γ) be a Galois insertion. Then, $Dep_\wedge(A) = lco(A)$ iff (D, α, A, γ) is separated.*

Proof. As recalled in Section 2.2, by duality, $lco(A)$ is atomic, that is each lower closure different from the bottom $\lambda x.\bot_A$ is the join of the atomic closures ϕ_a which precede it, where, for $a \in A \setminus \{\bot_A\}$, ϕ_a is defined as follows:

$$\phi_a(x) = \begin{cases} a & \text{if } x \geq_A a; \\ \bot_A & \text{otherwise.} \end{cases}$$

($\Leftarrow$) By Corollary 5.3 (i), the *lub* of $Dep_\wedge(A)$ coincides with that of $lco(A)$. Therefore, since $\lambda x.\bot_A = \lambda x.\alpha(\bot_D \wedge \gamma(x)) \in Dep_\wedge(A)$, it is sufficient to show that $Dep_\wedge(A)$ contains all the atoms of $lco(A)$ to prove that $lco(A) = Dep_\wedge(A)$. Consider any $a \in A \setminus \{\bot_A\}$. By separatedness, there exists $d_a \in D$ such that $\alpha(d_a) = a$. Hence, $\lambda x.\alpha(d_a \wedge \gamma(x)) \in Dep_\wedge(A)$. We show that $\lambda x.\alpha(d_a \wedge \gamma(x)) = \phi_a$. If $a \leq_A x$ then $\alpha(d_a \wedge \gamma(x)) = \alpha(d_a) = a$, because $d_a \leq_D \gamma(a) \leq_D \gamma(x)$. The case $a \not\leq_A x$ follows by the hypothesis of separatedness: $\alpha(d_a \wedge \gamma(x)) \leq_A \alpha(\gamma(\bot_A)) = \bot_A$.
($\Rightarrow$) Assume, by contradiction, that (D, α, A, γ) is not separated. Then, there exists $\hat{a} \in A \setminus \{\bot_A\}$, such that for any $d \in D$, if $\alpha(d) = \hat{a}$ then there exists $x \not\geq_A \hat{a}$ such that $d \wedge \gamma(x_d) \not\leq_D \gamma(\bot_A)$. Since $Dep_\wedge(A) = lco(A)$, we have that $\phi_{\hat{a}} \in Dep_\wedge(A)$. Thus, $\phi_{\hat{a}} = \lambda x.\alpha(d_{\hat{a}} \wedge \gamma(x))$, for some $d_{\hat{a}} \in D$. Note that $\phi_{\hat{a}}(\top_A) = \alpha(d_{\hat{a}}) = \hat{a}$. Hence, there exists $x_{d_{\hat{a}}} \not\geq_A \hat{a}$ such that $d_{\hat{a}} \wedge \gamma(x_{d_{\hat{a}}}) \not\leq_D \gamma(\bot_A)$. But this is a contradiction, because for any $x \not\geq_A \hat{a}$, we have $d_{\hat{a}} \wedge \gamma(x) \leq_D \gamma(\alpha(d_{\hat{a}} \wedge \gamma(x))) = \gamma(\phi_{\hat{a}}(x)) = \gamma(\bot_A)$. ■

Thus, the notion of separatedness characterizes precisely when the lattice of meet-autodependencies of an abstract domain A coincides with the lattice of lower closures on A. As a consequence, for a separated abstract domain A, the above result provides a way to represent any dependency in $Dep_\wedge(A)$ as a subset of A. In fact, recall from Section 2.2, that for any complete lattice C, the complete lattice $\langle lco(C), \sqsubseteq \rangle$ of all lower closure operators on C (ordered pointwise) is isomorphic to the complete lattice $\langle \mathcal{M}(C), \subseteq \rangle$ of all dual-Moore-sets of C, ordered by subset inclusion (recall that $X \subseteq C$ is a dual-Moore-set iff $X = dMc(X) = \{\vee Y \mid Y \subseteq X\}$ – notice that $\bot \in X$). Such an isomorphism maps any lower closure to its set of fixpoints. Thus, for a separated abstract domain A, by exploiting this isomorphism, the Galois insertion $(D, \alpha^\natural, Dep_\wedge(A), \gamma^\natural)$ becomes isomorphic to the Galois insertion $(D, \tilde{\alpha}, \mathcal{M}(A), \tilde{\gamma})$, where the left-adjoint $\tilde{\alpha}$ is defined as $\tilde{\alpha}(d) = \{x \in A \mid \alpha(d \odot \gamma(x)) = x\}$. Analogously, the Galois insertion $(Dep_\wedge(A), \bar{\alpha}, A, \bar{\gamma})$ of A in $Dep_\wedge(A)$, given by

Corollary 5.6, becomes isomorphic to the Galois insertion $(\mathcal{M}(A), \widehat{\alpha}, A, \widehat{\gamma})$, where, for any $M \in \mathcal{M}(A)$ and $a \in A$, $\widehat{\alpha}(M) = \vee_A M$ and $\widehat{\gamma}(a) = \downarrow a$. This latter observation is a consequence of the fact that the set of fixpoints of the lower closure $\bar{\gamma}(a) = \lambda x.a \wedge x$ actually is given by $\downarrow a = \widehat{\gamma}(a)$.

Thus, for a separated abstract domain A, the operator of meet-autodependencies applied to A provides a powerset-like completion of A that we call *dual-Moore-set completion* of A.

Obviously, the dual-Moore-set completion is in general uncomparable with the standard operator of disjunctive completion, as shown by the following example.

Example 5.14 We have seen in Example 5.8 that the abstract domain *Sign* is separated. Hence, its dual-Moore-set completion $\mathcal{M}(\mathit{Sign})$ has been characterized in Example 5.5. By contrast, the disjunctive completion of *Sign* is the domain $\mathit{Sign}^{\neq 0}$ of Example 4.7. It is then obvious that these two abstractions of $\wp(\mathbb{Z})$ are uncomparable. □

Whenever A is separated in D, the lattice of meet-autodependencies $\mathit{Dep}_{\wedge}(A)$ enjoys all the well known properties of closure operators (see e.g. [48, 64] for a few of them). In particular, the following lattice-theoretic properties are inherited from $\mathit{lco}(A)$.

(i) $\mathit{Dep}_{\wedge}(A)$ is atomic[8] (cf. [64]).

(ii) $\mathit{Dep}_{\wedge}(A)$ is distributive iff it is complemented iff it is Boolean iff A is a complete well-ordered chain (cf. [49]).

(iii) If A is join-continuous (namely, for any $x \in A$ and any chain $Y \subseteq A$, $x \vee (\wedge Y) = \wedge_{y \in Y}(x \vee y)$) then $\mathit{Dep}_{\wedge}(A)$ is dual-pseudocomplemented (cf. [34]), namely for any $f \in \mathit{Dep}_{\wedge}(A)$, there exists a unique $f^* \in \mathit{Dep}_{\wedge}(A)$ such that $f \sqcup f^* = \mathit{id}$, and if $f \sqcup g = \mathit{id}$ then $f^* \sqsubseteq g$, for any $g \in A \xrightarrow{\wedge} A$.

We close this section by a simple example.

Example 5.15 As shown in Example 5.8, the abstract domain *Sign* of Example 3.9 is separated. Hence, by Theorem 5.13, this yields another proof of the fact that, in Example 5.5, $\mathit{Dep}_{\cap}(\mathit{Sign})$ coincides with $\mathit{lco}(\mathit{Sign})$.
Consider instead the abstract domain *Parity* in Example 3.9. Since *Parity* is disjunctive and its top is join-reducible, by Remark 5.11, $(\wp(\mathbb{Z}), \alpha_p, \mathit{Parity}, \gamma_p)$ is not separated. Thus, by Corollary 5.3 (i) and Theorem 5.13, $\mathit{Dep}_{\cap}(\mathit{Parity}) \subset \mathit{lco}(\mathit{Parity})$. In particular, the lower closure corresponding to the dual-Moore-set $\{\top_p, \bot_p\}$ is not included in $\mathit{Dep}_{\cap}(\mathit{Parity})$:

Fact 5.16 *Let $\rho \in \mathit{lco}(\mathit{Parity})$ such that $\mathit{fp}(\rho) = \{\top_p, \bot_p\}$. Then, $\rho \notin \mathit{Dep}_{\cap}(\mathit{Sign})$.*

Proof. $\rho \in \mathit{Parity} \xrightarrow{\cap} \mathit{Parity}$, where

$$\rho(x) = \begin{cases} \top_p & \text{if } x = \top_p; \\ \bot_p & \text{otherwise.} \end{cases}$$

Assume by contradiction that there exists $S \subseteq \mathbb{Z}$ such that for any $x \in \mathit{Parity}$, $\alpha_p(S \cap \gamma_p(x)) = \rho(x)$. Since γ_p is strict, we have that, for any $T \in \wp(\mathbb{Z})$, if $\alpha_p(T) = \bot_p$ then $T = \emptyset$. Thus, we have that $S \cap \gamma_p(ev) = \emptyset = S \cap \gamma_p(od)$. Hence, since $\mathbb{Z} = \gamma_p(ev) \cup \gamma_p(od)$, we get $S = \emptyset$. But this is a contradiction because

$$\begin{aligned} \alpha_p(S \cap \gamma_p(\top_p)) &= \alpha_p(\emptyset \cap \mathbb{Z}) \\ &= \alpha_p(\emptyset) \\ &= \bot_p, \end{aligned}$$

and therefore $\alpha_p(S \cap \gamma_p(\top_p)) = \bot_p \neq \top_p = \rho(\top_p)$. ■

Thus, $\mathit{Dep}_{\cap}(\mathit{Parity}) \subset \mathit{lco}(\mathit{Parity})$. On the other hand, it would not be too difficult to show directly that $\mathit{Dep}_{\cap}(\mathit{Parity}) \cong \mathit{Parity}$. □

[8] As early observed in [52], atomicity is an interesting property in domains for abstract interpretation: Atoms represent primitive properties, and each (nonprimitive) property can be generated by collecting primitive ones.

6 Application to Logic Program Analysis

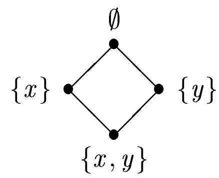
We present in this section an example of application of the reduced relative power to domains for abstract interpretation-based program analysis. We show that the well known abstract domain *Def*, introduced by Dart in [24] in the context of groundness analysis for deductive databases, and used by Marriott and Søndergaard in [47] for ground-dependency analysis of logic programs, can be systematically derived by meet-autodependencies of the more abstract (and much simpler) domain *Gr* for plain groundness analysis of logic programs, introduced by Jones and Søndergaard in [43].

We will consider sets of substitutions (we remark that we do not consider substitutions up to renaming) closed by instantiation, i.e. the concrete domain is $\langle \wp^{\downarrow}(Sub), \subseteq \rangle$, which is a cHa where *lub* and *glb* are, respectively, union and intersection. It is worth noting that this concrete domain gives rise to the so-called c-Herbrand interpretations used in the c-semantics for logic programs, as described in [27]. The concrete operator of composition on order-ideals of substitutions is given by intersection, namely $\langle \wp^{\downarrow}(Sub), \subseteq, \cap \rangle$ plays the rôle of the concrete semi-quantale.

The simplest abstract domain for plain groundness analysis, relative to a (nonempty) finite set of variables of interest $V \subseteq Var$, is given by $Gr = \langle \wp(V), \supseteq \rangle$. Any $W \in Gr$ is intended to represent every substitution that grounds at least all the variables in W. Thus, the abstraction and concretization maps are defined as follows: For any $\Theta \in \wp^{\downarrow}(Sub)$ and $W \in \wp(V)$,

- $\alpha_g(\Theta) = \{x \in V \mid \forall \theta \in \Theta.\, var(\theta(x)) = \emptyset\} = \cap_{\theta \in \Theta} gr(\theta)$,
- $\gamma_g(W) = \{\theta \in Sub \mid gr(\theta) \supseteq W\}$.

It is worth noting that γ_g actually is closed by instantiation (i.e., if $\theta \in \gamma_g(W)$ and $\theta' \leq \theta$ then $\theta' \in \gamma_g(W)$), and therefore, $\gamma_g(W)$ correctly is an order-ideal of substitutions. It is simple to show that $(\wp^{\downarrow}(Sub), \alpha_g, \wp(V), \gamma_g)$ is a Galois insertion (cf. [43]). For $V = \{x, y\}$, Gr is the simple abstract domain depicted below.



Let us now recall the definitions of the abstract domains *Def* and *Pos* (more details can be found in [4]). As before, $V \subseteq Var$ is a fixed finite set of variables of interest. It is well known that Boolean functions and propositional formulae are equivalent concepts; in the following, we will use them without distinction. A truth interpretation I for a propositional formula (on V) is any subset $I \subseteq V$, where the obvious intended meaning is that I contains all and only the true variables. For a given propositional formula f on V, $mod(f)$ denotes the set of all the interpretations that are models of f (hence, $mod(f) \subseteq \wp(V)$). We write $f \models g$, when $mod(f) \subseteq mod(g)$. Two propositional formulas f and g are equivalent when $mod(f) = mod(g)$. In the following, we will always consider propositional formulae up to this equivalence, and any formula will be implicitly intended to represent its equivalence class. Obviously, all the (equivalence classes of) propositional formulae on V form a Boolean lattice with respect to the standard ordering $\models$. A formula f is *positive* if $V \in mod(f)$ (i.e. the unital interpretation V is a model of f), while f is *definite* if f is positive and closed by intersection, i.e. if $M, N \in mod(f)$ then $M \cap N \in mod(f)$. Also, f is *monotone* if $mod(f)$ is upward closed, i.e. for all $M, N \subseteq V$, if $M \subseteq N$ and $M \in mod(f)$ then $N \in mod(f)$. *Def* (*Pos*) is the finite lattice (with respect to $\models$) of all the definite (positive) formulae on V. Also, *Con* is the finite lattice (with respect to $\models$) of all the monotone definite formulae on V; it is easy to prove that *Con* consists of all the possible (logical) conjunctions of variables (including the empty conjunction *true*). *Pos* is a Boolean lattice, where the *glb* and *lub* are given by logical conjunction and disjunction (denoted by $\wedge$ and $\vee$), respectively. *Def* is an atomic lattice (the atoms are those formulae f such that $mod(f) = \{V, W\}$, for any $W \neq V$), where the *lub* is, in general, different from logical disjunction, and can be defined as follows: $f_1 \vee_{Def} f_2 = \wedge\{f \in Def \mid f_1 \vee f_2 \models f\}$. It is simple to show that Gr is isomorphic to Con, where any $W \in Gr$ corresponds to the conjunction $\wedge W$. It is well known that Gr is an abstraction of Def, which in turn is an abstraction of Pos: This is evidently true, since both Gr and Def are closed by logical conjunction, i.e. up to isomorphic representations, both Gr and Def are Moore-sets of Pos. In particular, the abstraction for Gr is given by the Galois insertion $(Def, \lambda f.\{x \in V \mid f \models x\}, Gr, \lambda W. \wedge W)$. The abstraction and concretization maps between $\wp(Sub)$ and Def are well known. For any $f \in Def$, its concretization is defined as follows:

$$\gamma(f) = \{\sigma \in Sub \mid \forall \sigma' \text{ instance of } \sigma.\ gr(\sigma') \in mod(f)\}.$$

It is possible to show that γ is co-additive and injective, and therefore it gives rise to a Galois insertion of Def into $\wp(Sub)$. This same γ is the concretization map for Con and Pos as well. In particular, it would not be difficult to show that when restricted to $Con \cong Gr$, this map γ coincides, up to isomorphism, with the above concretization γ_g for Gr. It is also possible to give an explicit definition for the abstraction map of Def. If $\theta \in Sub$ then its Def-abstraction is defined as

$$\alpha_{\{\cdot\}}(\theta) = \exists_{\overline{V}}.\ \wedge\{x \leftrightarrow \wedge var(\theta(x)) \mid \theta(x) \neq x\},$$

where $\exists_{\overline{V}}$ is the existential quantification over the variables of noninterest, i.e. the variables in $Var \setminus V$. Then, for any $\Theta \in \wp(Sub)$, the abstraction map is given by:

$$\alpha(\Theta) = \vee_{Def}\{\alpha_{\{\cdot\}}(\theta) \mid \theta \in \Theta\}.$$

Notice that $\alpha(\emptyset) = \vee_{Def}\emptyset = \wedge V$. These two mappings α and γ form a Galois insertion of Def into $\wp(Sub)$. As an example, for $V = \{x, y, z, u\}$, the formula $x \wedge (y \leftrightarrow z)$ is an element of Def that represents all the substitutions σ such that for any instance σ' of σ the following conditions hold: (i) $\sigma'(x)$ is ground, (ii) $\sigma'(y)$ is ground iff also $\sigma'(z)$ is ground. In particular, $\sigma_1 = \{x/a, y/b, z/c\}$ and $\sigma_2 = \{x/a, y/w, z/w, v/u\}$ satisfy these properties. Thus, $\{\sigma_1, \sigma_2\} \subseteq \gamma(x \wedge (y \leftrightarrow z))$. It turns out that the concretization of any definite formula is an order-ideal of substitutions, and therefore, $(\wp^{\downarrow}(Sub), \alpha, Def, \gamma)$ is a Galois insertion as well.

It turns out that Gr is a separated abstract domain in $\wp^{\downarrow}(Sub)$.

Lemma 6.1 *$(\wp^{\downarrow}(Sub), \alpha_g, Gr, \gamma_g)$ is a separated Galois insertion.*

Proof. Consider any $W \in Gr$. Let us consider a substitution $\sigma \in Sub$ such that (i) $\forall x \in W.\ var(\sigma(x)) = \emptyset$, (ii) $\forall y \in V \setminus W.\ \sigma(x) = z$, where $z \notin V$. Evidently, it is always possible to define such an idempotent substitution σ. We show that $\downarrow\sigma \in \wp^{\downarrow}(Sub)$ is a separating element for W. Obviously, $\alpha_g(\downarrow\sigma) = W$. Moreover, consider any $S \in Gr$ such that $W \not\supseteq S$. We have to prove that $\downarrow\sigma \cap \gamma_g(S) \subseteq \gamma_g(V) = \{\theta \in Sub \mid gr(\theta) \supseteq V\}$. Consider any $\delta \in \downarrow\sigma \cap \gamma_g(S)$. Since $W \not\supseteq S$, there exists $x \in S \setminus W$. Then, $var(\delta(x)) = \emptyset$, and therefore, for any $y \in V \setminus W$, by definition of σ, $var(\sigma(y)) = \emptyset$. Thus, $gr(\delta) \supseteq V$. ∎

Thus, by Theorem 5.13, $Dep_{\cap}(Gr) = lco(Gr)$ $(= \mathcal{M}(Gr))$, and, as remarked after Theorem 5.13, $(\wp^{\downarrow}(Sub), \alpha_g^{\natural}, Dep_{\cap}(Gr), \gamma_g^{\natural}) \cong (\wp^{\downarrow}(Sub), \tilde{\alpha}, \mathcal{M}(Gr), \tilde{\gamma})$.

The next result shows that $(\wp^{\downarrow}(Sub), \tilde{\alpha}, \mathcal{M}(Gr), \tilde{\gamma})$ is isomorphic to $(\wp^{\downarrow}(Sub), \alpha, Def, \gamma)$.

Theorem 6.2 *$(\wp^{\downarrow}(Sub), \alpha, Def, \gamma) \cong (\wp^{\downarrow}(Sub), \tilde{\alpha}, \mathcal{M}(Gr), \tilde{\gamma})$.*

Proof. Since we deal with Galois insertions, as observed in Section 2.2, it is sufficient to show that $\gamma(Def) = \tilde{\gamma}(\mathcal{M}(Gr))$.
($\subseteq$) Consider any $f \in Def$. Notice that since f is a definite formula, $mod(f)$ is a dual-Moore-set of $Gr = \langle \wp(V), \supseteq \rangle$. We prove that $\gamma(f) = \tilde{\gamma}(mod(f))$. We first show that $\gamma(f) \subseteq \tilde{\gamma}(mod(f))$, i.e. $\tilde{\alpha}(\gamma(f)) = \{W \in Gr \mid \alpha_g(\gamma(f) \cap \gamma_g(W)) = W\} \subseteq mod(f)$. Thus, let us assume that for some $W \in \wp(V)$, $\alpha_g(\gamma(f) \cap \gamma_g(W)) = W$. This means that $W = \cap\{gr(\theta) \mid \theta \in \gamma(f) \cap \gamma_g(W)\}$. But since $\{gr(\theta) \mid \theta \in \gamma(f) \cap \gamma_g(W)\} \subseteq mod(f)$ and f is definite, we get that $W \in mod(f)$, as

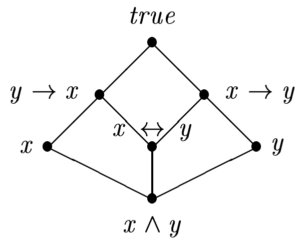
desired. We show now that $\tilde{\gamma}(mod(f)) \subseteq \gamma(f)$. To this aim, it is enough to show that for any $\Theta \in \wp^{\downarrow}(Sub)$, if $\tilde{\alpha}(\Theta) \subseteq mod(f)$ then $\Theta \subseteq \gamma(f)$. Therefore, consider any $\theta \in \Theta$ and any instance θ' of θ. Since Θ is an order-ideal, $\theta' \in \Theta$. Hence, it is sufficient to show that $gr(\theta) \in mod(f)$. To this purpose, by hypothesis, it is enough to show that $\alpha_g(\Theta \cap \gamma_g(gr(\theta))) = gr(\theta)$: But this is true, since $\theta \in \Theta \cap \gamma_g(gr(\theta))$.
($\supseteq$) Consider any $M \in \mathcal{M}(Gr)$. Since M is a dual-Moore-set of $\langle \wp(V), \supseteq \rangle$, M contains V and it is closed under intersection. This means that there exists $f_M \in Def$ such that $mod(f_M) = M$. We have seen above that for any $f \in Def$, $\gamma(f) = \tilde{\gamma}(mod(f))$. Therefore, $\gamma(f_M) = \tilde{\gamma}(mod(f_M)) = \tilde{\gamma}(M)$, and this concludes the proof. ■

Thus, equivalently, we also get that $(\wp^{\downarrow}(Sub), \gamma, Def, \alpha) \cong (\wp^{\downarrow}(Sub), \alpha^{\natural}, Dep_{\cap}(Gr), \gamma^{\natural})$. We have then fully reconstructed the abstract domain Def by the systematic operator of meet-autodependencies. The next result proves that the Galois insertion of Gr into its dual-Moore-set completion $(\mathcal{M}(Gr), \widehat{\alpha}, Gr, \widehat{\gamma})$ is isomorphic to the Galois insertion of Gr in Def.

Theorem 6.3 $(Def, \lambda f.\{x \in V \mid f \models x\}, Gr, \lambda W. \wedge W) \cong (\mathcal{M}(Gr), \widehat{\alpha}, Gr, \widehat{\gamma})$.

Proof. Recall that, for any $W \in Gr$, $\widehat{\gamma}(W) = \downarrow W = \{U \in \wp(V) \mid W \subseteq U\}$. Thus, using the isomorphism between Def and $\mathcal{M}(Gr)$ given in the proof of Theorem 6.2, it is sufficient to show that for any $W \in Gr$, $mod(\wedge W) = \{U \in \wp(V) \mid W \subseteq U\}$, and this is trivially true. ■

For the case $V = \{x, y\}$, we have that Def has the same lattice structure as in Example 5.5, since it is the dual-Moore-set completion of Gr.



By the proof of Theorem 6.2, the correspondence between formulae of Def and dual-Moore-sets of Gr is given by the model-theoretic interpretation of a formula. For $V = \{x, y\}$ we therefore have: $x \wedge y = \{\{x, y\}\}$, $x = \{\{x\}, \{x, y\}\}$, $y = \{\{y\}, \{x, y\}\}$, $x \leftrightarrow y = \{\emptyset, \{x, y\}\}$, $x \to y = \{\emptyset, \{y\}, \{x, y\}\}$, $y \to x = \{\emptyset, \{x\}, \{x, y\}\}$, and $true = \{\emptyset, \{x\}, \{y\}, \{x, y\}\}$.

It is worth noting that the operation of logical conjunction, i.e. the standard abstract unification used for ground-dependency analysis by Def (cf. [47]), can be easily implemented in Def by set-intersection on the corresponding dual-Moore-set completion of Gr. Furthermore, Def shares the lattice-theoretic properties of lower closures, being atomic and dual-pseudo-complemented. In particular, by atomicity, any formula f of Def can be represented as the *lub* of the atoms that imply f. In $\mathcal{M}(Gr)$, an atom is a dual-Moore-set $\{W, V\}$, where $W \neq V$. For instance, $x \to y \in Def$ can be represented by joining the atomic dual-Moore-sets $\{\{y\}, \{x, y\}\}$ and $\{\emptyset, \{x, y\}\}$. This technique may provide a concise representation for formulae in Def, which are usually represented by using ordered binary decision diagrams (cf. [4, 11]).

We close this section by showing that Def is not separated in $\wp^{\downarrow}(Sub)$.

Lemma 6.4 *$(\wp^{\downarrow}(Sub), \alpha, Def, \gamma)$ is not separated.*

Proof. Let us consider $x \to y \in Def$, and any $\Theta \in \wp^{\downarrow}(Sub)$ such that $\alpha(\Theta) = x \to y$. We prove that every such Θ is not a separating element for $x \to y$. We can distinguish two cases.
(i) There exists $\theta \in \Theta$ such that $\alpha_{\{\cdot\}}(\theta) = x \to y$. Then, $var(\sigma(x)) \supset var(\sigma(y)) \neq \emptyset$. Thus, there exists θ' instance of θ such that $y \in gr(\theta')$ and $x \notin gr(\theta')$. Since $\theta' \in \Theta \cap \gamma(y)$, this means that Θ is not separating.
(ii) Otherwise, there exist $\theta_1, \theta_2 \in \Theta$ such that $\alpha_{\{\cdot\}}(\theta_1) = x \leftrightarrow y$ and $\alpha_{\{\cdot\}}(\theta_2) = y$. Thus, $x \in gr(\theta_2)$, and since $\theta_2 \in \Theta \cap \gamma(y)$, we get that Θ is not separating. ■

Thus, by Corollary 5.3 (i) and Theorem 5.13, we get that $Dep_{\cap}(Def) \subset lco(Def)$. On the other hand, very recently, Scozzari in [59] characterized precisely the meet-autodependencies of Def in $\wp^{\downarrow}(Sub)$ by showing that $Dep_{\cap}(Def) = Pos$.

7 Application to Logic Program Semantics

In this section, we apply the reduced relative power in the field of logic program semantics. We show how to systematically derive new declarative semantics for logic programs by composing by reduced relative power the domains of interpretation of some well known semantics. All the proofs not presented in this section are contained in Appendix A.

In the following, a logic program semantics is a pair $\mathcal{X} = \langle C, T \rangle$, where C is a complete lattice (the semantic domain) and $T : Program \longmapsto (C \overset{m}{\longmapsto} C)$. For $P \in Program$, we denote $T(P)$ by the more customary T_P. Sometimes, with a slight abuse of notation, we denote a semantics $\langle C, T \rangle$ by $\langle C, T_P \rangle$. The (least fixpoint) semantics of a program P is given by $lfp(T_P) \in C$, and it is denoted by $[\![P]\!]^{\mathcal{X}}$. When T_P is continuous, we have that $[\![P]\!] = \vee_{n<\omega} T_P^n(\bot_C)$. If $\circ$ is a syntactic operator for composing programs, a semantics $\mathcal{X}$ is *compositional* with respect to $\circ$, if given any two programs P_1 and P_2, the semantics of $P_1 \circ P_2$ can be retrieved from the semantics of P_1 and P_2, i.e. if there exists a function @ such that, for any $P_1, P_2 \in Program$, $[\![P_1 \circ P_2]\!] = [\![P_1]\!] @ [\![P_2]\!]$. Since logic programs are sets of clauses, the standard and most obvious way of syntactically composing programs is by set union, i.e. $\circ = \cup$. Then, we will consider compositionality of programs with respect to set union.

7.1 The Concrete Semantics

As Cousot and Cousot showed in [18, 22], semantics at different levels of abstraction for a programming language can be derived by abstract interpretation of a more concrete *reference* semantics. An obvious choice for a concrete reference semantics is the operational description of the computational process, which in logic programming is *SLD resolution*. In the following, we recall some basic results from [33], where a hierarchy of logic program semantics, based on *SLD* resolution, has been introduced. We fix the Prolog left-to-right selection rule without depth-first search. The operational semantics of a given logic program P is defined through a labeled transition system defining *SLD* resolution as follows: $SLD = \langle State, \{\xrightarrow{c} \mid c \in Clause\} \rangle$, where $State = Atom^* \times Sub$, and transitions $\xrightarrow{c} \subseteq State \times State$, labeled with (program) clauses, are defined by:

$$\langle a :: \bar{b}, \sigma \rangle \xrightarrow{c} \langle body :: \bar{b}, \sigma\theta \rangle \quad \Leftrightarrow \quad c = h \leftarrow body \text{ is a renamed apart clause and } \theta = mgu(a\sigma, h).$$

We denote by $\mathbf{T}$ the set of equivalence classes, up to variable renaming, of all possible finite traces, i.e.

$$\mathbf{T} = \{ s_0 \xrightarrow{c_1} \dots \xrightarrow{c_n} s_n \mid n \geq 1,\ s_0, ..., s_n \in State,\ c_1, ..., c_n \in Clause \}.$$

We use π to denote a generic element in $\mathbf{T}$, and by π_l we denote the last state of π. For a (nonempty) sequence of clauses $\bar{c} = c_1, ..., c_n \in Clause^*$ (hence $n \geq 1$), we write $s_0 \xrightarrow{\bar{c}} s_n$ iff there exist $s_1, ..., s_{n-1}$ such that $s_0 \xrightarrow{c_1} s_1 \cdots s_{n-1} \xrightarrow{c_n} s_n \in \mathbf{T}$. We say that $\pi \in \mathbf{T}$ is a *successful* trace if $\pi = s \xrightarrow{\bar{c}} \Lambda$. The set $\mathcal{T}_P^{sld} \subseteq \mathbf{T}$ of program execution traces for a program P is inductively defined by the following rules:

$$\frac{s, s' \in State \quad c \in P \quad s \xrightarrow{c} s'}{s \xrightarrow{c} s' \in \mathcal{T}_P^{sld}} \qquad\qquad \frac{s \in State \quad \pi \in \mathcal{T}_P^{sld} \quad c \in P \quad \pi_l \xrightarrow{c} s}{\pi \xrightarrow{c} s \in \mathcal{T}_P^{sld}}.$$

A key point in this construction is that execution traces can be equivalently specified by restricting the interest to AND-compositional execution traces only. Intuitively, a set of traces $\mathcal{T}$ is AND-compositional if any execution trace starting from any (possibly nonatomic) goal can be reconstructed from the execution traces in $\mathcal{T}$ starting from its atomic subgoals only. It is worth noting that all the well-known fixpoint semantics in [26, 27, 30] are based on denotations which

1 $\dfrac{c = h \leftarrow \bar{b} \in P}{h \xrightarrow{c} \bar{b} \in \mathcal{T}_P^{and}}$	2 $\dfrac{\begin{array}{c} c = h \leftarrow b_1, ..., b_n \in P \\ \langle\langle a_i \xrightarrow{\bar{c}_i} \Lambda\rangle_{i=1}^{k-1}, a_k \xrightarrow{\bar{c}_k} \bar{b}\rangle \ll_c \mathcal{T}_P^{and} \quad (1 \leq k \leq n) \\ \theta = mgu(\langle b_1, ..., b_k\rangle, \langle a_1, ..., a_k\rangle) \end{array}}{(h \xrightarrow{c} \langle b_1, ..., b_n\rangle \xrightarrow{\bar{c}_1} ... \xrightarrow{\bar{c}_k} \bar{b} :: \langle b_{k+1}, ..., b_n\rangle)\theta \in \mathcal{T}_P^{and}}$

Table 1: Inductive definition of $\mathcal{T}_P^{and}$.

are AND-compositional in an analogous sense. The set $\mathbf{aT}$ of *plain* execution traces for atomic goals is defined as follows:

$$\mathbf{aT} = \{(h \xrightarrow{c_1} \dots \xrightarrow{c_n} \bar{b})\theta \mid h \in Atom,\ \langle h, \sigma_0\rangle \xrightarrow{c_1} \dots \xrightarrow{c_n} \langle \bar{b}, \theta\rangle \in \mathbf{T}\}.$$

Thus, a plain execution trace $\pi \in \mathbf{aT}$ is an execution trace starting from an atomic goal and where each state is of the form $\langle \bar{b}, \varepsilon\rangle$, abbreviated simply to $\bar{b}$. Hence, up to syntactic representation, $\mathbf{aT}$ is indeed a subset of $\mathbf{T}$. As before, for a (nonempty) sequence of clauses $\bar{c} = c_1, ..., c_n \in Clause^*$, for any $h \in Atom$ and $\bar{b} \in Atom^*$, we write $h \xrightarrow{\bar{c}} \bar{b}$ iff there exist $\bar{b}_1, ..., \bar{b}_{n-1} \in Atom^*$ such that $h \xrightarrow{c_1} \bar{b}_1 \cdots \bar{b}_{n-1} \xrightarrow{c_n} \bar{b} \in \mathbf{aT}$. For a given program P, the inductive definition of the set $\mathcal{T}_P^{and} \subseteq \mathbf{aT}$ of *AND-compositional plain execution traces* of P for atomic goals is given in [33] as in Table 1. Rules 1 and 2 specify a big-step SLD semantics for atomic goals as a positive inductive definition. The first rule specifies a basic plain execution trace starting from an atom h. For a given clause $c = h \leftarrow b_1, ..., b_n$, the second rule specifies how plain execution traces starting from each atom b_i in the body of c can be composed to get a trace starting from the head h: This is obtained by composing some successful traces for the first $k-1$ atoms $b_1, ..., b_{k-1}$ of the body of c, with the state (goal) produced from a trace for the k^{th} atom b_k of the body.

Following a standard technique (cf. [2]), an operator $\varphi : Program \longmapsto (\wp(\mathbf{aT}) \overset{c}{\longmapsto} \wp(\mathbf{aT}))$ can be systematically derived by the inductive definition of $\mathcal{T}_P^{and}$ in Table 1 as follows: For any $P \in Program$,

$$\begin{array}{rcl} \varphi_P(X) & = & \{h \xrightarrow{c} \bar{b} \mid c = h \leftarrow \bar{b} \in P\} \cup \\ & & \left\{ \pi\theta \;\middle|\; \begin{array}{l} c = h \leftarrow b_1, ..., b_n \in P \\ \langle\langle a_i \xrightarrow{\bar{c}_i} \Lambda\rangle_{i=1}^{k-1}, a_k \xrightarrow{\bar{c}_k} \bar{b}\rangle \ll_c X \quad (1 \leq k \leq n) \\ \theta = mgu(\langle b_1, ..., b_k\rangle, \langle a_1, ..., a_k\rangle) \\ \pi = h \xrightarrow{c} \langle b_1, ..., b_n\rangle \xrightarrow{\bar{c}_1} ... \xrightarrow{\bar{c}_k} \bar{b} :: \langle b_{k+1}, ..., b_n\rangle \end{array} \right\}. \end{array}$$

It turns out that, for any program P, φ_P is continuous on $\langle \wp(\mathbf{aT}), \subseteq\rangle$, and $\mathcal{T}_P^{and} = lfp(\varphi_P)$. The following theorem justifies the interest in plain execution traces for atomic goals.

Theorem 7.1 ([33]) *Let $G = b_1, ..., b_n$ be a goal and $\sigma \in Sub$. $\langle G, \sigma\rangle \xrightarrow{\bar{c}} \langle B, \theta\rangle \in \mathcal{T}_P^{sld}$ iff there exist $\langle\langle h_i \xrightarrow{\bar{c}_i} \Lambda\rangle_{i=1}^{k-1}, h_k \xrightarrow{\bar{c}_k} \bar{b}\rangle \ll_G \mathcal{T}_P^{and}$, $1 \leq k \leq n$, such that $\delta = mgu(\langle b_1, ..., b_k\rangle\sigma, \langle h_1, ..., h_k\rangle)$, $\bar{c} = \bar{c}_1 :: ... :: \bar{c}_k$, $B\theta \sim (\bar{b} :: \langle b_{k+1}, ..., b_n\rangle)\sigma\delta$, and $G\theta \sim G\sigma\delta$.*

In the following, we consider $\langle\langle \wp(\mathbf{aT}), \subseteq\rangle, \varphi\rangle$ as the concrete reference semantics. Elements in $\wp(\mathbf{aT})$ are simply called *trace interpretations*. This semantics models finite plain execution traces for atomic goals, and provides a least fixpoint presentation of the operational semantics of logic programs, which, by Theorem 7.1, is equivalent to standard *SLD*-resolution.

7.2 The Concrete Domain

We consider the following operator $\nabla : \wp(\mathbf{aT}) \times \wp(\mathbf{aT}) \longmapsto \wp(\mathbf{aT})$ of *trace-unfolding*.

Definition 7.2 The *trace-unfolding* operator $\nabla : \wp(\mathbf{aT}) \times \wp(\mathbf{aT}) \longmapsto \wp(\mathbf{aT})$, for any $X, Y \in \wp(\mathbf{aT})$, is defined as follows:

$$X \nabla Y = \left\{ \pi\theta \;\middle|\; \begin{array}{l} c = h \xrightarrow{\bar{c}} \langle b_1, ..., b_n \rangle \in X \\ \langle\langle a_i \xrightarrow{\bar{c}_i} \Lambda \rangle_{i=1}^{k-1}, a_k \xrightarrow{\bar{c}_k} \bar{b} \rangle \ll_c Y \ (1 \leq k \leq n) \\ \theta = mgu(\langle b_1, ..., b_k \rangle, \langle a_1, ..., a_k \rangle) \\ \pi = h \xrightarrow{\bar{c}} \langle b_1, ..., b_n \rangle \xrightarrow{\bar{c}_1} ... \xrightarrow{\bar{c}_k} \bar{b} :: \langle b_{k+1}, ..., b_n \rangle \end{array} \right\}.$$ □

Thus, $X \nabla Y$ denotes the result of unfolding any trace in X by using any trace in Y. The trace-unfolding operator above satisfies the following basic algebraic properties.

Proposition 7.3 *The operator ∇ is monotone, associative, left-additive, right-continuous, and $\emptyset$ is a left-annihilator for ∇ (i.e. $\forall Y \in \wp(\mathbf{aT}).\ \emptyset \nabla Y = \emptyset$).*

Thus, $\langle \wp(\mathbf{aT}), \subseteq, \nabla \rangle$ is a semi-quantale, and therefore, by Theorem 3.6, if $(\wp(\mathbf{aT}), \alpha_1, D_1, \gamma_1)$ and $(\wp(\mathbf{aT}), \alpha_2, D_2, \gamma_2)$ are Galois connections, then $(\wp(\mathbf{aT}), \alpha, D_1 \overset{\nabla}{\longmapsto} D_2, \gamma)$ is a Galois insertion, where for any trace interpretation $I \in \wp(\mathbf{aT})$, $\alpha(I) = \lambda x.\alpha_2(I \nabla \gamma_1(x))$. A trace interpretation I is therefore abstracted in a dependency in $D_1 \overset{\nabla}{\longmapsto} D_2$ that encodes how I is abstractly unfolded in D_2 by using traces approximated in D_1. The key point is that each Galois connection $(\wp(\mathbf{aT}), \alpha, A, \gamma)$ actually induces an abstract semantics $\langle A, \alpha \circ \varphi_P \circ \gamma \rangle$, where $\alpha \circ \varphi_P \circ \gamma : A \overset{\mathrm{m}}{\longmapsto} A$ is the best correct approximation of φ_P in A, which is correct[9] with respect to the reference semantics $\langle \wp(\mathbf{aT}), \varphi_P \rangle$. Thus, any pair of such abstract semantics can be composed by reduced relative power, yet providing a new correct abstract semantics.

7.3 The s-semantics as an Abstract Semantics

As shown in [33], a number of well-known logic program semantics can be derived from $\langle \wp(\mathbf{aT}), \varphi_P \rangle$ by abstract interpretation. We consider here the well-known s-semantics characterizing the observable notion of computed answer substitution ([27]).

The *success abstraction* $\alpha_{\mathcal{S}} : \langle \wp(\mathbf{aT}), \subseteq \rangle \longmapsto \langle \wp(Atom), \subseteq \rangle$ approximates any finite successful trace with its initial state, while nonsuccessful traces are simply ignored, i.e., for any $X \in \wp(\mathbf{aT})$,

$$\alpha_{\mathcal{S}}(X) = \{ h \in Atom \mid h \xrightarrow{\bar{c}} \Lambda \in X \}.$$

It is immediate to observe that $\alpha_{\mathcal{S}}$ is additive and onto, and therefore defines a Galois insertion $(\wp(\mathbf{aT}), \alpha_{\mathcal{S}}, \wp(Atom), \gamma_{\mathcal{S}})$. It is also easy to note that the concretization map $\gamma_{\mathcal{S}}$ is additive. As shown in [33], it turns out that this Galois insertion induces an abstract semantics $\mathcal{S} = \langle \langle \wp(Atom), \subseteq \rangle, \alpha_{\mathcal{S}} \circ \varphi_P \circ \gamma_{\mathcal{S}} \rangle$ which is equivalent to the s-semantics. Let us recall that for any $P \in Program$, the immediate consequences operator of the s-semantics is $T_P^s : \wp(Atom) \overset{\mathrm{c}}{\longmapsto} \wp(Atom)$ defined as follows:

$$T_P^s(I) = \left\{ h\theta \in Atom \;\middle|\; \begin{array}{l} c = h \leftarrow b_1, ..., b_n \in P \\ \langle b'_1, ..., b'_n \rangle \ll_c I,\ n \geq 0 \\ \theta = mgu(\langle b_1, ..., b_n \rangle, \langle b'_1, ..., b'_n \rangle) \end{array} \right\}.$$

It turns out that the best correct approximation $\alpha_{\mathcal{S}} \circ \varphi_P \circ \gamma_{\mathcal{S}}$ of φ_P in $\wp(Atom)$ and T_P^s coincide. For the sake of completeness, we include the proof of this result in Appendix A.

Proposition 7.4 $T_P^s = \alpha_{\mathcal{S}} \circ \varphi_P \circ \gamma_{\mathcal{S}}$.

In a similar way, it would not be too difficult to show (see [33]) that the *Herbrand abstraction* $\alpha_{\mathcal{H}} = \lambda I.ground(\alpha_{\mathcal{S}}(I)) : \wp(\mathbf{aT}) \longmapsto \wp(ground(Atom))$ gives rise to an abstract semantics $\mathcal{H}$ equivalent to the well-known van Emden and Kowalski ground semantics in [26].

[9] In the sense of abstract interpretation, i.e. $\alpha(lfp(\varphi_P)) \leq_A lfp(\alpha \circ \varphi_P \circ \gamma)$.

7.4 Autodependencies of s-semantics

In the following, we denote simply by $Dep_\nabla(\mathcal{S})$ the domain of autodependencies induced by the Galois insertion $(\wp(\mathbf{aT}), \alpha_S, \wp(Atom), \gamma_S)$ w.r.t. the concrete semi-quantale $\langle \wp(\mathbf{aT}), \subseteq, \nabla \rangle$, where the corresponding Galois insertion is denoted by $(\wp(\mathbf{aT}), \alpha, Dep_\nabla(\mathcal{S}), \gamma)$. Therefore, for a trace-interpretation $I \in \wp(\mathbf{aT})$, $\alpha(I) = \lambda x.\alpha_\mathcal{S}(I \nabla \gamma_\mathcal{S}(x))$ is a generic dependency in $Dep_\nabla(\mathcal{S})$. As observed in Section 5, a dependency $\lambda x.\alpha_\mathcal{S}(I \nabla \gamma_\mathcal{S}(x))$ is the best correct approximation of the concrete operator $\lambda x.I \nabla x : \wp(\mathbf{aT}) \longmapsto \wp(\mathbf{aT})$. As hinted in [25], it turns out that any function $\lambda x.I \nabla x$ is a T_P-like function. In our context, this observation is formalized by the next lemma, where we use the *sequence abstraction* α_* to approximate finite traces by the pair of their initial and final states in a clause-like form: $\alpha_* : \langle \wp(\mathbf{aT}), \subseteq \rangle \longmapsto \langle \wp(Clause), \subseteq \rangle$ is defined as $\alpha_*(X) = \{h \leftarrow \bar{b} \in Clause \mid h \xrightarrow{\bar{c}} \bar{b} \in X\}$. It is easily seen that α_* is additive and onto, and therefore it gives rise to a Galois insertion $(\wp(\mathbf{aT}), \alpha_*, \wp(Clause), \gamma_*)$. As proved in [15] in a similar context, this latter G.i. induces an abstract semantics equivalent to the compositional semantics of resultants of [31].

Lemma 7.5 *For any $I \in \wp(\mathbf{aT})$ and $X \subseteq Atom$, $\alpha_\mathcal{S}(I \nabla \gamma_\mathcal{S}(X)) = T^\mathcal{S}_{\alpha_*(I)}(X)$.*

As a consequence, we get that $Dep_\nabla(\mathcal{S}) = \{T^s_I : \wp(Atom) \longmapsto \wp(Atom) \mid I \subseteq Clause\}$, and for any program P, $T^s_P = \lambda X.\alpha_\mathcal{S}(\gamma_*(P) \nabla \gamma_\mathcal{S}(X))$. If $I \in \wp(\mathbf{aT})$ then the intermediate states of any trace $\pi \in I$ are not relevant in $\lambda X.\alpha_\mathcal{S}(I \nabla \gamma_\mathcal{S}(X))$, as the following lemma shows.

Lemma 7.6 *For any $I \in \wp(\mathbf{aT})$ and $X \subseteq Atom$, $\alpha_\mathcal{S}(I \nabla \gamma_\mathcal{S}(X)) = \alpha_\mathcal{S}(\gamma_*(\alpha_*(I)) \nabla \gamma_\mathcal{S}(X))$.*

Let us now introduce a mapping $\alpha_{cl} : \langle \wp(Clause), \subseteq \rangle \longmapsto Dep_\nabla(\mathcal{S})$ defined as follows: For any set of clauses $I \subseteq Clause$, $\alpha_{cl}(I) = \lambda X.\alpha_\mathcal{S}(\gamma_*(I) \nabla \gamma_\mathcal{S}(X))$.

Lemma 7.7 *α_{cl} is additive and surjective.*

Thus, α_{cl} induces a Galois insertion $(\wp(Clause), \alpha_{cl}, Dep_\nabla(\mathcal{S}), \gamma_{cl})$. It is simple to see that composing the Galois insertions $(\wp(Clause), \alpha_{cl}, Dep_\nabla(\mathcal{S}), \gamma_{cl})$ and $(\wp(\mathbf{aT}), \alpha_*, \wp(Clause), \gamma_*)$ we exactly get the Galois insertion relating $Dep_\nabla(\mathcal{S})$ to $\wp(\mathbf{aT})$.

Proposition 7.8 *$\alpha = \alpha_{cl} \circ \alpha_*$ and $\gamma = \gamma_* \circ \gamma_{cl}$.*

Proof. By Lemma 7.6, for any $I \in \wp(\mathbf{aT})$, $\alpha_{cl}(\alpha_*(I)) = \lambda X.\alpha_\mathcal{S}(I \nabla \gamma_\mathcal{S}(X)) = \alpha(I)$, and this suffices to conclude the proof. ∎

It is worth noting that since both γ_* and γ_{cl} are additive functions, by the above result, we obtain that γ is additive as well (i.e. $Dep_\nabla(\mathcal{S})$ is a disjunctive abstraction of $\wp(\mathbf{aT})$). We denote by $\mathcal{S} \mapsto \mathcal{S} = \langle Dep_\nabla, T_P^{\mathcal{S}\mapsto\mathcal{S}} \rangle$ the abstract semantics induced by $(\wp(\mathbf{aT}), \alpha, Dep_\nabla(\mathcal{S}), \gamma)$, where $T_P^{\mathcal{S}\mapsto\mathcal{S}} = \alpha \circ \varphi_P \circ \gamma$. As observed above, γ is additive. Thus, $T_P^{\mathcal{S}\mapsto\mathcal{S}}$ is a continuous operator on $Dep_\nabla(\mathcal{S})$.

Lemma 7.9 *For any $I \in \wp(\mathbf{aT})$, $\alpha(\varphi_P(\gamma(\alpha(I)))) = \alpha(\varphi_P(I))$.*

Thus, our semantic transformer $T_P^{\mathcal{S}\mapsto\mathcal{S}}$ can be characterized in terms of φ_P as follows (the last equality exploits Lemma 7.5): For any $\lambda X.\alpha_\mathcal{S}(I \nabla \gamma_\mathcal{S}(X)) \in Dep_\nabla(\mathcal{S})$,

$$T_P^{\mathcal{S}\mapsto\mathcal{S}}(\lambda X.\alpha_\mathcal{S}(I \nabla \gamma_\mathcal{S}(X)) = \lambda X.\alpha_\mathcal{S}(\varphi_P(I) \nabla \gamma_\mathcal{S}(X)) = T^s_{\alpha_*(\varphi_P(I))}.$$

The least fixpoint abstract semantics induced by $Dep_\nabla(\mathcal{S})$ is defined as $[\![P]\!]^{\mathcal{S}\mapsto\mathcal{S}} = lfp(T_P^{\mathcal{S}\mapsto\mathcal{S}}) = \sqcup_{n<\omega}(T_P^{\mathcal{S}\mapsto\mathcal{S}})^n(\perp_{Dep_\nabla(\mathcal{S})})$ (recall that the *lub* $\sqcup$ of $Dep_\nabla(\mathcal{S})$ is defined pointwise). This semantics can be characterized as follows.

Theorem 7.10 *For any $P \in Program$, $[\![P]\!]^{\mathcal{S}\mapsto\mathcal{S}} = \lambda X.\alpha_\mathcal{S}(\mathcal{T}_P^{and} \nabla \gamma_\mathcal{S}(X))$.*

Proof. Let us consider the following equalities:

$$\begin{array}{rcll} [\![P]\!]^{s\mapsto s} & = & lfp(T_P^{s\mapsto s}) & \\ & = & \sqcup_{n<\omega}(T_P^{s\mapsto s})^n(\bot_{Dep_\nabla(\mathcal{S})}) & \\ & = & \sqcup_{n<\omega}(T_P^{s\mapsto s})^n(\alpha(\emptyset)) & \text{(by strictness of } \alpha) \\ & = & \sqcup_{n<\omega}\alpha(\varphi_P^n(\emptyset)) & \text{(by Lemma 7.9)} \\ & = & \alpha\left(\cup_{n<\omega}\varphi_P^n(\emptyset)\right) & \text{(by additivity of } \alpha) \\ & = & \alpha(lfp(\varphi_P)). & \end{array}$$

Since $lfp(\varphi_P) = \mathcal{T}_P^{and}$, we get $[\![P]\!]^{s\mapsto s} = \lambda X.\alpha_{\mathcal{S}}(\mathcal{T}_P^{and}\nabla\gamma_{\mathcal{S}}(X))$. ∎

The next result characterizes the equivalence on $\wp(Clause)$ induced by the above abstraction map α_{cl}.

Theorem 7.11 *For any $I, J \subseteq Clause$, $\alpha_{cl}(I) = \alpha_{cl}(J)$ iff $T_I^s = T_J^s$.*

Proof. ($\Rightarrow$) If $\alpha_{cl}(I) = \alpha_{cl}(J)$ then, by Lemma 7.6 and since $\gamma_* \circ \alpha_* \circ \gamma_* = \gamma_*$, we get $\lambda X.\alpha_{\mathcal{S}}(\gamma_*(I)\nabla\gamma_{\mathcal{S}}(X)) = \lambda X.\alpha_{\mathcal{S}}(\gamma_*(J)\nabla\gamma_{\mathcal{S}}(X))$. Thus, by Lemma 7.5, and since $\alpha_* \circ \gamma_* = id$, we get $T_I^s = T_J^s$.
($\Leftarrow$) Since α_* is surjective, there exist $I', J' \in \wp(\mathbf{aT})$ such that $\alpha_*(I') = I$ and $\alpha_*(J') = J$. Thus, by Lemma 7.5 and Lemma 7.6, $\alpha_{cl}(\alpha_*(I')) = \alpha_{cl}(\alpha_*(J'))$, and therefore, $\alpha_{cl}(I) = \alpha_{cl}(J)$. ∎

The following example shows that $(\wp(Clause), \alpha_{cl}, Dep_\nabla(\mathcal{S}), \gamma_{cl})$ is a strict Galois insertion. Therefore, the domain of autodependencies $Dep_\nabla(\mathcal{S})$ is strictly more abstract than the domain $\wp(Clause)$, used in [31] as the semantic domain of a compositional semantics characterizing the observable notion of resultant.

Example 7.12 It is immediate to see that $c_1 = p(x) \leftarrow q(x)$ and $c_2 = p(x) \leftarrow q(x), q(x)$ are such that $T_{\{c_1\}}^s = T_{\{c_2\}}^s$. Thus, by Theorem 7.11, $\alpha_{cl}(\{c_1\}) = \alpha_{cl}(\{c_2\})$. □

By Theorem 7.11, the equivalence $\sim_{cl}$ on $\wp(Clause)$ (viz. on programs) induced by α_{cl}, i.e. for any $P, P' \in \wp(Clause)$, $P \sim_{cl} P' \Leftrightarrow \alpha_{cl}(P) = \alpha_{cl}(P')$, characterizes precisely the semantics of autodependencies $Dep_\nabla(\mathcal{S})$ as an abstraction of $\wp(Clause)$. This equivalence is based on the correspondence between T_P-like functions and program clauses, the latter encoding the dependency between their bodies and heads. Gabbrielli *et al.* proved in [30, Theorem 5.5] that the program equivalence induced by the compositional semantics of Bossi *et al.* in [9], let us call it $\mathcal{B}$, is strictly finer than $\sim_{cl}$, i.e. for all P, P', $\mathcal{B}(P) = \mathcal{B}(P') \Rightarrow P \sim_{cl} P'$, while the converse does not hold (a counterexample can be found in [30, Example 5.14]). Moreover, it would be possible to show (see the details in [33]) that $(\wp(\mathbf{aT}), \alpha_*, \wp(Clause), \gamma_*)$ actually induces an abstract semantics equivalent to $\mathcal{B}$. Thus, by Example 7.12, we get that the abstract semantics induced by $Dep_\nabla(\mathcal{S})$ is strictly more abstract than the compositional semantics $\mathcal{B}$ of [9]. This last fact suggests an observation on the property of full abstraction of $Dep_\nabla(\mathcal{S})$. We say that a logic program semantics $\mathcal{X}$ is *fully abstract* with respect to the observable notion of computed answer substitution and the composition of programs by union, if for any pair of programs P and Q,

$$\mathcal{X}(P) = \mathcal{X}(Q)$$

iff

$$\forall R \in Program.\forall G \in Atom^*.\ \langle G, \varepsilon\rangle \xrightarrow{\bar{c}} \langle \Lambda, \theta\rangle \in \mathcal{T}_{P\cup R}^{sld} \Leftrightarrow \langle G, \varepsilon\rangle \xrightarrow{\bar{c}'} \langle \Lambda, \sigma\rangle \in \mathcal{T}_{Q\cup R}^{sld} \text{ and } G\sigma \sim G\theta.$$

By the results of s-semantics (cf. [27]), we have that $\mathcal{X}$ is fully abstract iff for all programs P and Q, $\mathcal{X}(P) = \mathcal{X}(Q) \Leftrightarrow \forall R \in Program.\ lfp(T_{P\cup R}^s) = lfp(T_{Q\cup R}^s)$. It turns out that $\mathcal{B}$ is only correct and not fully abstract (see [30, Example 5.14]). Moreover, the following result shows that $[\![\cdot]\!]^{s\mapsto s}$ actually is a compositional semantics.

Theorem 7.13 *For all $P, Q \in Program$, $[\![P \cup Q]\!]^{s\mapsto s} = [\![\gamma_{cl}([\![P]\!]^{s\mapsto s}) \cup \gamma_{cl}([\![Q]\!]^{s\mapsto s})]\!]^{s\mapsto s}$.*

Hence, the above observations suggest that $Dep_{\nabla}(\mathcal{S})$ may be closer than $\mathcal{S}$ to a fully abstract semantics for computed answer substitutions and compositionality by union. In the following, we adopt the notation used in [46] and, for a given $f : \wp(X) \longmapsto \wp(X)$, we denote by $f + id$ the function $\lambda x. f(x) \cup x : \wp(X) \longmapsto \wp(X)$. Notice that $f + id$ is the least extensive function greater than or equal to f (w.r.t. the pointwise ordering). Moreover, we follow [30], and we denote by $[\![P]\!]^{ca}$ the function $\lambda x.(T_P^s + id)^{\omega}(x) : \wp(Atom) \longmapsto \wp(Atom)$. It is known that $[\![P]\!]^{ca}$ is an upper closure operator on $\langle \wp(Atom), \subseteq \rangle$ (cf. [46]). More importantly, Gabbrielli *et al.* proved in [30, Theorem 5.17] that the semantics $[\![\cdot]\!]^{ca}$ is fully abstract. By using the full abstraction property of $[\![\cdot]\!]^{ca}$, the next example shows that $[\![\cdot]\!]^{\mathcal{S} \mapsto \mathcal{S}}$ is not fully abstract.

Example 7.14 Consider the following propositional logic programs: $P = \{q \leftarrow, \quad p \leftarrow p\}$ and $Q = \{q \leftarrow\}$. It is clear that for any program R, $lfp(T^s_{P \cup R}) = lfp(T^s_{Q \cup R})$, and therefore $[\![P]\!]^{ca} = [\![Q]\!]^{ca} = \lambda X. X \cup \{q\}$. On the other hand, note that $\mathcal{T}_P^{and} = \{q \longrightarrow \Lambda\} \cup \{\pi = p \longrightarrow p \cdots p \longrightarrow p \mid |\pi| \geq 1\}$ and $\mathcal{T}_Q^{and} = \{q \longrightarrow \Lambda\}$. Thus, by Theorem 7.10, we have that $[\![P]\!]^{\mathcal{S} \mapsto \mathcal{S}}(\{p\}) = \{p, q\} \neq \{q\} = [\![Q]\!]^{\mathcal{S} \mapsto \mathcal{S}}(\{p\})$. Hence, by full abstraction of $[\![\cdot]\!]^{ca}$, it turns out that $[\![\cdot]\!]^{\mathcal{S} \mapsto \mathcal{S}}$ is not a fully abstract semantics. □

Although $[\![\cdot]\!]^{\mathcal{S} \mapsto \mathcal{S}}$ is not fully abstract, it represents a good example on how semantics can be specified by means of suitable operators for domain combination. An appealing feature of this approach to specify semantics for programming languages is that, being based on arbitrary Galois connections, it is independent from any specific choice for representing semantic denotations. In particular, we can compose by reduced relative power arbitrary abstract semantics without being constrained to look for corresponding syntactic objects acting as denotations, like clauses, binary clauses, atoms, etc., which is instead the case in the standard s-semantics approach for logic programming (cf. [8]). In logic programming, for instance, likewise to $Dep_{\nabla}(\mathcal{S})$, one could consider $Dep_{\nabla}(\mathcal{H})$ as a variant of the compositional semantics in [32, 46], or, analogously, one can combine by reduced relative power the semantics for call patterns, partial answers, etc. Moreover, a semantic domain of functions like $Dep_{\nabla}(\mathcal{S})$ can be further approximated in different ways (e.g. by using binary relations, pointwise abstraction, etc.) as suggested in [18, 22, 23]. It should be therefore evident that the operator of reduced relative power is an important tool for designing semantics in abstract interpretation-based framework.

8 Related Work

Cousot and Cousot firstly introduced a *reduced cardinal power* of abstract domains in [20, Section 10.2]. The Cousot and Cousot reduced cardinal power was defined in a particular context of a collecting semantics of program assertions, where, in a function between abstract domains, assertions were composed by logical conjunction. As we discussed in Section 3, we have extended Cousot and Cousot's ideas to a very general setting, where an operator of composition for concrete objects is only required to give rise to a weak form of quantale, called semi-quantale, on any concrete domain. We profitted of the full generality of our theory in our applications to ground-dependency analysis and semantics of logic programming (cf. Sections 6 and 7). In fact, in the former case, the concrete domain is given by order-ideals of substitutions, and therefore is not simply a powerset of some set, while in the latter case, a concrete domain of sets of program execution traces is endowed with an operator of trace-unfolding that does not behave like a meet-operation (in particular, it is not commutative).

We have investigated the relationship between Nielson's *tensor product* ([52, 53]) and our reduced relative power of abstract domains. Nielson proposed to use the tensor product as a way to build relational domains in data flow program analysis, as opposed to the standard methodology (cf. [42]) of using the powerset of the cartesian product (see, e.g., [52, 53] for more details). The tensor product was not originally conceived as an abstract domain refinement, and therefore, its overall aim is slightly different from that of the reduced relative power. We equally investigated how the reduced relative power relates to the tensor product, especially from a lattice-theoretic point of view, hence providing an answer to a question earlier raised by Nielson in [52, pag. 124].

The *open product* of abstract domains, introduced in [17], provides a different way of expressing dependencies between domains. It has been introduced as an enhancement of the reduced product operator, by allowing the domains to exchange information each other (and with an external environment) through a particular type of functions, called queries. Due to its peculiar features, the open product differs substantially from our reduced relative power, and it is, in general, uncomparable.

Bagnara has developed in [5] an elegant meta-language for abstract domain manipulation based on a subset of a *concurrent constraint* (*cc*) language ([57]), where a domain is required to be a particular kind of an ask-and-tell constraint system. Bagnara's framework offers a way to combine such domains with asynchronous interaction provided by ask and tell operations. Because of this feature, such a combination of abstract domains relies upon (lower) closure operators, i.e. the standard semantic interpretation for *cc* agents like ask and tell (cf. [58]). Thus, the use of *cc*-like expressions to combine domains can be viewed as an enhancement of a domain with some of its lower closures, and therefore, in this sense, as an instance of our notion of meet-autodependencies of a domain. It is also worth noting that a *cc* language can often result too restrictive to express arbitrary closure operators, or dependency relations between abstract domains relative to generic (possibly non meet-like) operations of composition (as in our application in logic program semantics design of Section 7).

Finally, as far as applications are concerned, Dart introduced in [24] the domain of formulae *Def* describing groundness dependencies between the arguments of a predicate in a deductive database, and used it for a groundness analysis aiming to determine whether a database is connected. Then, successively, Marriott and Søndergaard (see [47]) used the domain *Def* for logic program ground-dependency analysis, and compared it with other domains of formulae, like *Pos*. The domain *Def* first came to our attention as a good example of a typical and well-known domain used for a relational analysis which can be understood as a reduced relative power (as indeed we showed in Section 6). In the field of logic program semantics, it is worth mentioning that Brogi and Turini in [10] also have introduced a compositional semantics based on functions as program denotations. The main difference is that in [10] the semantics of a logic program module is a T_P-like function and compositionality is achieved by function composition, while in our approach, e.g. for the case of $Dep_{\nabla}(\mathcal{S})$ treated in Section 7, the semantics of a module is given as a least fixpoint of a transformer of T_P-like functions and compositionality is achieved using similar techniques to those in [9].

9 Conclusion

This work originated by observing that some well-known domains for analysis and semantics could be systematically derived by exploiting some form of dependency among simpler (i.e. more abstract) domains. This observation led us to develop a theory of a reduced relative power operator for combining domains in abstract interpretation, generalizing the earlier Cousot and Cousot reduced cardinal power in [20]. We have presented two applications of our theory, both in the field of logic programming, which have shown how the reduced relative power may provide both a better and precise understanding of the internal structure of complex domains, and a fruitful methodology to systematically build new relational domains from existing simpler ones.

Although many results in Sections 4 and 5 on the relationship between reduced relative power and well-known function spaces, like additive functions and closure operators, are based on concrete domains which are complete Heyting or Boolean algebras, it is important to remark that the general theory is based on a much weaker algebraic structure which is strongly related with the notion of *quantale*. Recently, Giacobazzi and Scozzari [38] gave a logical interpretation of the reduced relative power of abstract domains, whenever the concrete domain is a complete Heyting algebra. These results go beyond Proposition 4.11, and prove that dependencies give rise to intuitionistic implications between domain's objects. We believe that this is a first step towards a logic-based interpretation of domain refinement operations.

Acknowledgments

This work was partly done while the authors were visiting the *LIX, Laboratoire d'Informatique, École Polytechnique*, Paris. The work of Roberto Giacobazzi has been partly supported by the EEC *Human Capital and Mobility* individual grant: "Semantic Definitions, Abstract Interpretation and Constraint Reasoning", no. ERBCHBICT930822.

A Appendix: Selected Proofs

This appendix contains the proofs of some results in Section 7.

Proposition 7.3 *The operator ∇ is monotone, associative, left-additive, right-continuous, and $\emptyset$ is a left-annihilator for ∇ (i.e. $\forall Y \in \wp(\mathbf{aT}).\ \emptyset \nabla Y = \emptyset$).*

Proof. We omit the proof of associativity of ∇, because it follows from a similar result proved in [25]. Moreover, by definition, it is immediate that ∇ is monotone, left-additive, right-continuous, and $\emptyset$ is a left-annihilator for ∇. ∎

Proposition 7.4 $T_P^s = \alpha_{\mathcal{S}} \circ \varphi_P \circ \gamma_{\mathcal{S}}$.

Proof. We prove a stronger relation between T_P^s and φ_P, namely $\alpha_{\mathcal{S}} \circ \varphi_P = T_P^s \circ \alpha_{\mathcal{S}}$. Let $I \in \wp(\mathbf{aT})$. By definition, $h \in \alpha_{\mathcal{S}}(\varphi_P(I))$ iff either (1) there exists $h \in P$, or (2) there exist $c = h' \leftarrow b_1, ..., b_n \in P$ and $\langle a_i \xrightarrow{\bar{c}_i} \Lambda \rangle_{i=1}^n \ll_c I$, such that $\theta = mgu(\langle b_1, ..., b_n\rangle, \langle a_1, ..., a_n\rangle)$ and $h \equiv h'\theta$. In both cases, it is clear that $h \in T_P^s(\alpha_{\mathcal{S}}(I))$. On the other hand, $h\theta \in T_P^s(\alpha_{\mathcal{S}}(I))$ iff either (3) $h\theta \in P$, or (4) there exist $c = h \leftarrow b_1, ..., b_n \in P$ and $\langle a_1, ..., a_n\rangle \ll_c \alpha_{\mathcal{S}}(I)$ such that $\theta = mgu(\langle b_1, ..., b_n\rangle, \langle a_1, ..., a_n\rangle)$. If (3) holds, then obviously $h\theta \in \alpha_{\mathcal{S}}(\varphi_P(I))$. If (4) holds, then there exist $a_1 \xrightarrow{\bar{c}_1} \Lambda, \ldots, a_n \xrightarrow{\bar{c}_n} \Lambda \in I$; then, $(h \xrightarrow{c} \langle b_1, ..., b_n\rangle \xrightarrow{\bar{c}_1} \ldots \xrightarrow{\bar{c}_n} \Lambda)\theta \in \varphi_P(I)$, and therefore $h\theta \in \alpha_{\mathcal{S}}(\varphi_P(I))$. This concludes the proof. ∎

Lemma 7.5 *For any $I \in \wp(\mathbf{aT})$ and $X \subseteq Atom$, $\alpha_{\mathcal{S}}(I \nabla \gamma_{\mathcal{S}}(X)) = T^s_{\alpha_*(I)}(X)$.*

Proof. As proved in [27], it turns out that for any $C \subseteq Clause$ and $X \subseteq Atom$, $T_C^s(X) = \cup_{c \in C} T^s_{\{c\}}(X)$. Thus, since ∇ is left-additive, and $\alpha_{\mathcal{S}}$ and α_* are additive, it is sufficient to prove that for any plain trace $\pi \in \mathbf{aT}$, $\alpha_{\mathcal{S}}(\{\pi\} \nabla \gamma_{\mathcal{S}}(X)) = T^s_{\alpha_*(\{\pi\})}(X)$. Let us assume that $\pi = h \xrightarrow{\bar{c}} \langle b_1, ..., b_n\rangle$.
($\subseteq$) Let $h' \in \alpha_{\mathcal{S}}(\{\pi\} \nabla \gamma_{\mathcal{S}}(X))$. Then, there exist $\langle a_i \xrightarrow{\bar{c}_i} \Lambda \rangle_{i=1}^n \ll_\pi \gamma_{\mathcal{S}}(X)$ such that $fail \neq \theta = mgu(\langle b_1, ..., b_n\rangle, \langle a_1, ..., a_n\rangle)$ and $h' = h\theta$. Hence, by G.i., $\langle a_1, ..., a_n\rangle \ll_\pi X$. Thus, by definition, $h' = h\theta \in T^s_{\alpha_*(\{\pi\})}(X)$.
($\supseteq$) Let us assume that $h' \in T^s_{\alpha_*(\{\pi\})}(X)$. Then, there exist $\langle a_1, ..., a_n\rangle \ll_\pi X$ such that $fail \neq \theta = mgu(\langle b_1, ..., b_n\rangle, \langle a_1, ..., a_n\rangle)$ and $h' = h\vartheta$. Then, there exist $\langle\langle a_i \xrightarrow{\bar{c}_i} \Lambda\rangle_{i=1}^n\rangle \ll_\pi \gamma_{\mathcal{S}}(X)$. Thus, we get $(h \xrightarrow{\bar{c}} \langle b_1, ..., b_n\rangle \xrightarrow{\bar{c}_1} \ldots \xrightarrow{\bar{c}_n} \Lambda)\theta \in (\{\pi\} \nabla \gamma_{\mathcal{S}}(X))$, and hence, $h\vartheta = h' \in \alpha_{\mathcal{S}}(\{\pi\} \nabla \gamma_{\mathcal{S}}(X))$. ∎

Lemma 7.6 *For any $I \in \wp(\mathbf{aT})$ and $X \subseteq Atom$, $\alpha_{\mathcal{S}}(I \nabla \gamma_{\mathcal{S}}(X)) = \alpha_{\mathcal{S}}(\gamma_*(\alpha_*(I)) \nabla \gamma_{\mathcal{S}}(X))$.*

Proof. One inclusion is straightforward because $\gamma_* \circ \alpha_*$ is extensive and ∇ is argumentwise monotone. Thus, let $I \in \wp(\mathbf{aT})$, and $X \subseteq Atom$, and consider $h \in \alpha_{\mathcal{S}}(\gamma_*(\alpha_*(I)) \nabla \gamma_{\mathcal{S}}(X))$. Then, there exists a trace $\pi = h' \xrightarrow{\bar{c}} \langle b_1, ..., b_n\rangle \in \gamma_*(\alpha_*(I))$, $n \geq 0$, and $\langle a_i \xrightarrow{\bar{c}_i} \Lambda \rangle_{i=1}^n \ll_\pi \gamma_{\mathcal{S}}(X)$ such that $\theta = mgu(\langle b_1, ..., b_n\rangle, \langle a_1, ..., a_n\rangle)$ and $h = h'\theta$. Moreover, note that $h \in \alpha_{\mathcal{S}}(\{\pi\} \nabla \gamma_{\mathcal{S}}(X))$ and $h' \leftarrow b_1, ..., b_n \in \alpha_*(I)$. Thus, by definition of α_*, there exists $h' \xrightarrow{\bar{c}'} \langle b_1, ..., b_n\rangle \in I$. Thus, we immediately get that $h \in \alpha_{\mathcal{S}}(I \nabla \gamma_{\mathcal{S}}(X))$. ∎

Lemma 7.7 *α_{cl} is an additive and surjective map.*

Proof. It is immediate to note that γ_* is additive. Hence, the additivity of α_{cl} follows by left-additivity of ∇ (cf. Proposition 7.3) and additivity of $\alpha_{\mathcal{S}}$.

Consider now $\lambda X.\alpha_{\mathcal{S}}(I \nabla \gamma_{\mathcal{S}}(X)) \in Dep_{\nabla}(\mathcal{S})$, for some $I \in \wp(\mathbf{aT})$. Then, by considering for any $\alpha_*(I) \subseteq Clause$, by Lemma 7.6, $\alpha_{cl}(\alpha_*(I)) = \lambda x.\alpha_{\mathcal{S}}(I \nabla \gamma_{\mathcal{S}}(X))$. ■

In the following proofs, we abuse the notation by considering a program P as a denotation for the set $\{h \xrightarrow{c} \bar{b} \mid c = h \leftarrow \bar{b} \in P\}$. In this way, note that $\alpha_*(P) = P$, and, for any $X \in \wp(\mathbf{aT})$, $\varphi_P(X) = P \cup (P \nabla X)$. Moreover, note that, because we consider atoms and unit clauses as the same objects, γ_* coincides with $\gamma_{\mathcal{S}}$ when restricted to unit clauses, and α_* coincides with $\alpha_{\mathcal{S}}$ when restricted to sets of successful traces.

Lemma 7.9 *For any $X \in \wp(\mathbf{aT})$, $\alpha(\varphi_P(\gamma(\alpha(X)))) = \alpha(\varphi_P(X))$.*

Proof. We have to prove that for any $X \subseteq \mathbf{aT}$ and $Y \subseteq Atom$, $\alpha(\varphi_P(\gamma(\alpha(X))))(Y) = \alpha(\varphi_P(X))(Y)$.
($\supseteq$) This inclusion is straightforward by monotonicity of $\alpha \circ \varphi_P$ and since $\gamma \circ \alpha$ is extensive.
($\subseteq$) By Lemma 7.5, it is sufficient to prove that $T^s_{\alpha_*(\varphi_P(\gamma(\alpha(X))))}(Y) \subseteq T^s_{\alpha_*(\varphi_P(X))}(Y)$. So, assume that $h \in T^s_{\alpha_*(\varphi_P(\gamma(\alpha(X))))}(Y)$. Then, by definition, there exists $c' = h' \leftarrow b_1, ..., b_n \in \alpha_*(\varphi_P(\gamma(\alpha(X))))$ (with $n \geq 0$) and $\langle b'_1, ..., b'_n \rangle \ll_\pi Y$ such that $\theta = mgu(\langle b_1, ..., b_n \rangle, \langle b'_1, ..., b'_n \rangle)$ and $h = h'\theta$. Therefore, there exists $\pi = h' \xrightarrow{\bar{c}} b_1, ..., b_n \in \varphi_P(\gamma(\alpha(X)))$. By definition of φ_P, we can have the following two cases.

1. If $c' = h' \leftarrow b_1, ..., b_n \in P$ then $h' \xrightarrow{c'} b_1, ..., b_n \in \varphi_P(X)$, and therefore $h \in T^s_{\alpha_*(\varphi_P(X))}(Y)$.
2. Otherwise, assume $h' \xrightarrow{\bar{c}} b_1, ..., b_n \in P \nabla \gamma(\alpha(X))$. Then, $h \in \alpha_*((P \nabla \gamma(\alpha(X))) \nabla \gamma_*(Y))$. By associativity of ∇ (cf. Proposition 7.3), $h \in \alpha_*(P \nabla (\gamma(\alpha(X)) \nabla \gamma_*(Y)))$, and since $\gamma_*(Y) = \gamma_{\mathcal{S}}(Y)$, we get $h \in \alpha_*(P \nabla (\gamma(\alpha(X)) \nabla \gamma_{\mathcal{S}}(Y)))$. Thus, we may have the following two cases.
   (a) If h is a unit-clause in P, then h is a unit-clause in $\alpha_*(\varphi_P(X))$, and therefore $h \in T^s_{\alpha_*(\varphi_P(X))}(Y)$.
   (b) Otherwise, there are $c'' = h'' \leftarrow r_1, ..., r_m \in P$ and $\langle r'_1, ..., r'_m \rangle \ll_{c''} \alpha_*(\gamma(\alpha(X)) \nabla \gamma_{\mathcal{S}}(Y))$ such that $\sigma = mgu(\langle r_1, ..., r_m \rangle, \langle r'_1, ..., r'_m \rangle)$ and $h = h''\sigma$. Thus, we also have that $\langle r'_1, ..., r'_m \rangle \ll_{c''} \alpha_{\mathcal{S}}(\gamma(\alpha(X)) \nabla \gamma_{\mathcal{S}}(Y))$. Observe now that since $\alpha(\gamma(\alpha(X))) = \alpha(X)$, we have that $\alpha_{\mathcal{S}}(\gamma(\alpha(X)) \nabla \gamma_{\mathcal{S}}(Y)) = \alpha_{\mathcal{S}}(X \nabla \gamma_{\mathcal{S}}(Y))$. So, $\langle r'_1, ..., r'_m \rangle \ll_{c''} \alpha_{\mathcal{S}}(X \nabla \gamma_{\mathcal{S}}(Y))$, and $h \in \alpha_{\mathcal{S}}(P \nabla (X \nabla \gamma_{\mathcal{S}}(Y)))$. By associativity of ∇, we get $h \in \alpha_{\mathcal{S}}((P \nabla X) \nabla \gamma_{\mathcal{S}}(Y))$. Since $P \nabla X \subseteq \varphi_P(X)$, by monotonicity of ∇ and $\alpha_{\mathcal{S}}$, we get $h \in \alpha_{\mathcal{S}}(\varphi_P(X) \nabla \gamma_{\mathcal{S}}(Y))$. Thus, by Lemma 7.5, $h \in T^s_{\alpha_*(\varphi_P(X))}(Y)$. ■

Lemma A.1 *For any $H, K \subseteq Clause$, $H \sim_{cl} K \Rightarrow T^s_{[\![H]\!]^{\mathcal{B}}} = T^s_{[\![K]\!]^{\mathcal{B}}}$.*

Proof. By a straightforward inductive argument it is easy to prove that if $T^s_H = T^s_K$ then for any $n \in \mathbb{N}$: $(T^s_{H \cup id})^n = (T^s_{K \cup id})^n$, where $id = \{p(x) \leftarrow p(x) \mid p \in \Pi\}$. For any program P, $[\![P]\!]^{\mathcal{B}} = \cup_{n<\omega} P_n$, where $P_1 = P$ and $P_{n+1} = P_n \nabla (P \cup id)$ [9]. By [9][Lemma 4.12 (2), p. 21], for any $n \in \mathbb{N}$: $T^s_{P_{n+1}} = T^s_P \circ (T^s_{P \cup id})^n$. We conclude the proof by observing that, from the above hypothesis, $T^s_H \circ (T^s_{H \cup id})^n = T^s_K \circ (T^s_{K \cup id})^n$, therefore for any $n \in \mathbb{N}$: $T^s_{H_{n+1}} = T^s_{K_{n+1}}$, namely $T^s_{[\![H]\!]^{\mathcal{B}}} = T^s_{[\![K]\!]^{\mathcal{B}}}$. ■

Theorem 7.13 *For all $P, Q \in Program$, $[\![P \cup Q]\!]^{\mathcal{S} \mapsto \mathcal{S}} = [\![\gamma_{cl}([\![P]\!]^{\mathcal{S} \mapsto \mathcal{S}}) \cup \gamma_{cl}([\![Q]\!]^{\mathcal{S} \mapsto \mathcal{S}})]\!]^{\mathcal{S} \mapsto \mathcal{S}}$.*

Proof. By Theorem 7.10, Lemma 7.5, and because $\alpha_*(\mathcal{T}^{and}_P) = [\![P]\!]^{\mathcal{B}}$ [33]: $[\![P]\!]^{\mathcal{S} \mapsto \mathcal{S}} = \alpha_{cl}([\![P]\!]^{\mathcal{B}})$. Hence $[\![P \cup Q]\!]^{\mathcal{S} \mapsto \mathcal{S}} = \alpha_{cl}([\![P \cup Q]\!]^{\mathcal{B}})$. By compositionality of $\mathcal{B}$, i.e. $[\![P \cup Q]\!]^{\mathcal{B}} = [\![[\![P]\!]^{\mathcal{B}} \cup [\![Q]\!]^{\mathcal{B}}]\!]^{\mathcal{B}}$ [9], we have: $[\![P \cup Q]\!]^{\mathcal{S} \mapsto \mathcal{S}} = \alpha_{cl}([\![[\![P]\!]^{\mathcal{B}} \cup [\![Q]\!]^{\mathcal{B}}]\!]^{\mathcal{B}})$. Hence: $[\![P \cup Q]\!]^{\mathcal{S} \mapsto \mathcal{S}} = [\![[\![P]\!]^{\mathcal{B}} \cup [\![Q]\!]^{\mathcal{B}}]\!]^{\mathcal{S} \mapsto \mathcal{S}}$. Because $[\![P]\!]^{\mathcal{S} \mapsto \mathcal{S}} = \alpha_{cl}([\![P]\!]^{\mathcal{B}})$, it is now sufficient to prove that $[\![[\![P]\!]^{\mathcal{B}} \cup [\![Q]\!]^{\mathcal{B}}]\!]^{\mathcal{S} \mapsto \mathcal{S}} = [\![\gamma_{cl}(\alpha_{cl}([\![P]\!]^{\mathcal{B}})) \cup \gamma_{cl}(\alpha_{cl}([\![Q]\!]^{\mathcal{B}}))]\!]^{\mathcal{S} \mapsto \mathcal{S}}$. By additivity of α_{cl} and γ_{cl} this is equivalent to prove that $[\![[\![P]\!]^{\mathcal{B}} \cup [\![Q]\!]^{\mathcal{B}}]\!]^{\mathcal{S} \mapsto \mathcal{S}} = [\![\gamma_{cl}(\alpha_{cl}([\![P]\!]^{\mathcal{B}} \cup [\![Q]\!]^{\mathcal{B}}))]\!]^{\mathcal{S} \mapsto \mathcal{S}}$. Because, as observed above, for any set of clauses H: $[\![H]\!]^{\mathcal{S} \mapsto \mathcal{S}} = T^s_{[\![H]\!]^{\mathcal{B}}}$ and $[\![P]\!]^{\mathcal{B}} \cup [\![Q]\!]^{\mathcal{B}} \sim_{cl} \gamma_{cl}(\alpha_{cl}([\![P]\!]^{\mathcal{B}} \cup [\![Q]\!]^{\mathcal{B}}))$, we only have to prove that if $H \sim_{cl} K$ then $[\![H]\!]^{\mathcal{S} \mapsto \mathcal{S}} = [\![H]\!]^{\mathcal{S} \mapsto \mathcal{S}}$, or equivalently $T^s_{[\![H]\!]^{\mathcal{B}}} = T^s_{[\![K]\!]^{\mathcal{B}}}$. This follows by Lemma A.1, and concludes the proof. ■

References

[1] S. Abramsky and S. Vickers. Quantales, observational logic and process semantics. *Math. Struct. Comput. Sci.*, 3(2):161–228, 1993.

[2] P. Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 739–782. North-Holland, Amsterdam, 1977.

[3] K. R. Apt. Introduction to logic programming. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 495–574. Elsevier, Amsterdam and The MIT Press, Cambridge, Mass., 1990.

[4] T. Armstrong, K. Marriott, P. Schachte, and H. Søndergaard. Two classes of Boolean functions for dependency analysis. *Sci. Comput. Program.*, 1997. To appear. A preliminary version appeared in Proc. SAS '94.

[5] R. Bagnara. A hierarchy of constraint systems for data-flow analysis of constraint logic-based languages. *Sci. Comput. Program.*, 1997. To appear.

[6] H. J. Bandelt. The tensor product of continuous lattices. *Math. Z.*, 172:89–96, 1980.

[7] G. Birkhoff. *Lattice Theory.* AMS Colloquium Publication, 3rd edition. AMS, Providence, RI, 1967.

[8] A. Bossi, M. Gabbrielli, G. Levi, and M. Martelli. The s-semantics approach: theory and applications. *J. Logic Program.*, 19-20:149–197, 1994.

[9] A. Bossi, M. Gabbrielli, G. Levi, and M. C. Meo. A compositional semantics for logic programs. *Theor. Comput. Sci.*, 122(1-2):3–47, 1994.

[10] A. Brogi and F. Turini. Fully abstract compositional semantics for an algebra of logic programs. *Theor. Comput. Sci.*, 149(2):201–229, 1995.

[11] R. Bryant. Symbolic Boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv.*, 24(3):293–318, 1992.

[12] K. L. Clark. Predicate logic as a computational formalism. Technical Report DOC 79/59, Department of Computing, Imperial College, London, 1979.

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

[14] M. Comini and G. Levi. An algebraic theory of observables. In M. Bruynooghe, editor, *Proc. of the 1994 Internat. Logic Programming Symp.* (*ILPS '94*), pages 172–186. The MIT Press, Cambridge, Mass., 1994.

[15] M. Comini, G. Levi, and M. C. Meo. Compositionality of *SLD*-derivations and their abstractions. In J. Lloyd, editor, *Proc. of the 1995 Internat. Logic Programming Symp.* (*ILPS '95*). The MIT Press, Cambridge, Mass., 1995.

[16] 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.

[17] A. Cortesi, B. Le Charlier, and P. Van Hentenryck. Combinations of abstract domains for logic programming. In *Conference Record of the 21st ACM Symp. on Principles of Programming Languages* (*POPL '94*), pages 227–239. ACM Press, New York, 1994.

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

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

[20] 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. ACM Press, New York, 1979.

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

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

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

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

[25] F. Denis and J.-P. Delahaye. Unfolding, procedural and fixpoint semantics of logic programs. In C. Choffrut and M. Jantzen, editors, *Proc. of the 8th Internat. Symp. on Theoretical Aspects of Computer Science (STACS '91)*, volume 480 of *Lecture Notes in Computer Science*, pages 511–522. Springer-Verlag, Berlin, 1991.

[26] M. H. van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *J. ACM*, 23(4):733–742, 1976.

[27] M. Falaschi, G. Levi, M. Martelli, and C. Palamidessi. Declarative modeling of the operational behavior of logic languages. *Theor. Comput. Sci.*, 69(3):289–318, 1989.

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

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

[30] M. Gabbrielli, G. Levi, and M. C. Meo. Observable behaviors and equivalences of logic programs. *Inf. Comput.*, 122(1):1–29, 1996.

[31] M. Gabbrielli, G. Levi, and M. C. Meo. Resultant semantics for Prolog. *J. Logic and Comput.*, 6(4):491–521, 1996.

[32] H. Gaifman and E. Shapiro. Fully abstract compositional semantics for logic programs. In *Conference Record of the 16th ACM Symp. on Principles of Programming Languages (POPL '89)*, pages 134–142. ACM Press, New York, 1989.

[33] 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 Internat. Symp. on Theoretical Aspects of Computer Science (STACS '96)*, volume 1046 of *Lecture Notes in Computer Science*, pages 503–514. Springer-Verlag, Berlin, 1996.

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

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

[36] R. Giacobazzi and F. Ranzato. Optimal domains for disjunctive abstract interpretation. To appear in *Sci. Comput. Program.*, 1998. A preliminary version appeared in Proc. ESOP '96.

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

[38] R. Giacobazzi and F. Scozzari. Intuitionistic implication in abstract interpretation. To appear in H. Glaser and H. Kuchen, editors, *Proc. of the 9th Internat. Symp. on Programming Languages, Implementations, Logics, and Programs (PLILP '97)*, *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1997.

[39] G. Grätzer. *General Lattice Theory.* Birkhäuser Verlag, Basel, Switzerland, 1978.

[40] C. A. R. Hoare, I. J. Hayes, H. Jifeng, C. C. Morgan, A. W. Roscoe, J. W. Sanders, I. H. Sorensen, J. M. Spivey, and B. A. Sufrin. Laws of programming. *Comm. ACM*, 30(8):672–686, 1987.

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

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

[43] 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, 1987.

[44] R. S. Kemp and G. R. Ringwood. Reynolds and Heyting models of definite clauses. Technical Report 575, Comput. Science Dept., Queen Mary and Westfield College, London, 1991.

[45] 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, Calif., 1988.

[46] M. J. Maher. Equivalences of logic programs. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 627–658. Morgan Kaufmann, Los Altos, Calif., 1988.

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

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

[49] J. Morgado. Note on complemented closure operators of complete lattices. *Portug. Math.*, 21(3):135–142, 1962.

[50] C.J. Mulvey. &. *Suppl. Rend. Circ. Mat. Palermo*, 12:99–104, 1986.

[51] K. Muthukumar and M. Hermenegildo. Combined determination of sharing and freeness of program variables through abstract interpretation. In K. Furukawa, editor, *Proc. of the 8th Internat. Conf. on Logic Programming* (*ICLP '91*), pages 49–63. The MIT Press, Cambridge, Mass., 1991.

[52] F. Nielson. *Abstract Interpretation Using Domain Theory.* PhD Thesis, CST-31-84, Department of Computer Science, University of Edinburgh, Edinburgh, Scotland, 1984.

[53] F. Nielson. Tensor products generalize the relational data flow analysis method. In M. Arató, I. Kátai, and L. Varga, editors, *Proc. of the 4th Hungarian Computer Science Conf.*, pages 211–225, 1985.

[54] F. Nielson. Two-level semantics and abstract interpretation. *Theor. Comput. Sci.*, 69:117–242, 1989.

[55] F. Nielson and H. Riis Nielson. The tensor product in Wadler's analysis of lists. In B. Krieg-Brückner, editor, *Proc. of the 4th European Symp. on Programming* (*ESOP '92*), volume 582 of *Lecture Notes in Computer Science*, pages 351–370. Springer-Verlag, Berlin, 1992.

[56] K. I. Rosenthal. Quantales and their applications. In *Pitman Research Notes in Mathematics.* Longman Scientific & Technical, London, 1990.

[57] V. Saraswat. *Concurrent Constraint Programming Languages.* The MIT Press, Cambridge, Mass., 1993.

[58] V. Saraswat, V. A. Rinard, and P. Panangaden. Semantic foundations of concurrent constraint programming. In *Conference Record of the 18th ACM Symp. on Principles of Programming Languages* (*POPL '91*), pages 333–353. ACM Press, New York, 1991.

[59] F. Scozzari. Logical optimality of groundness analysis. To appear in P. van Hentenryck, editor, *Proc. of the 4th Internat. Static Analysis Symp.* (*SAS '97*), *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1997.

[60] Z. Shmuely. The structure of Galois connections. *Pacific J. Math.*, 54(2):209–225, 1974.

[61] H. Søndergaard. Immediate fixpoints and their use in groundness analysis. In V. Chandru and V. Vinay, editors, *Proc. of the 16th Conf. on Foundations of Software Technology and Theoretical Computer Science* (*FST&TCS '96*), volume 1180 of *Lecture Notes in Computer Science*, pages 359–370. Springer-Verlag, Berlin, 1996.

[62] R. Sundararajan and J. Conery. An abstract interpretation scheme for groundness, freeness, and sharing analysis of logic programs. In R. Shyamasundar, editor, *Proc. of the 12th Conf. on Foundations of Software Technology and Theoretical Computer Science* (*FST&TCS '92*), volume 652 of *Lecture Notes in Computer Science*, pages 203–216. Springer-Verlag, Berlin, 1992.

[63] J. Varlet. Contribution à l'étude des treillis pseudo-complémentés et des treillis de Stone. *Mém. Société Royale Sci. Liège*, 8(4):71 pages, 1963.

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