ELSEVIER Theoretical Computer Science 216 (1999) 159–211

Theoretical Computer Science

Fundamental Study

The reduced relative power operation on abstract domains

Roberto Giacobazzi[a],*, Francesco Ranzato[b]

[a] *Dipartimento di Informatica, Università di Pisa, Corso Italia 40, 56125 Pisa, Italy*

[b] *Dipartimento di Matematica Pura ed Applicata, Università di Padova, Via Belzoni 7, 35131 Padova, Italy*

Received April 1996; revised July 1997
Communicated by G. Levi

Abstract

In the context of standard abstract interpretation theory, a reduced relative power operation for functionally composing abstract domains is introduced and studied. 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 reduced cardinal power operation. The relationship between reduced relative power and Nielson's tensor product of abstract domains is also investigated. The case of autodependencies, i.e. base and exponent are the same domain, turns out to be particularly interesting: Under certain hypotheses, the domain of autodependencies corresponds to a powerset-like completion of the base abstract domain, providing a compact set-theoretic representation for autodependencies. Two relevant applications of the reduced relative power operation in the fields of logic program analysis and semantics design are presented. Notably, it is proved that the well-known abstract domain *Def* for logic program ground-dependency analysis can be characterized as the domain of autodependencies of the standard abstract domain representing plain groundness information only; on the semantics side, it is shown how reduced relative power can be exploited in order to systematically derive compositional semantics for logic programs. © 1999—Elsevier Science B.V. All rights reserved

Keywords: Abstract interpretation; Abstract domain; Reduced relative power; Separated abstract domain; Logic program analysis and semantics

1. Introduction

One of the appealing features of Cousot and Cousot's abstract interpretation theory is its ability to provide systematic methodologies to compositionally design complex

* Corresponding author. E-mail: giaco@di.unipi.it.

abstract domains from more simple ones (cf. [21]). 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 [29, 37], we call domain *refinements*, are devoted to enhance the precision of domains and provide high-level facilities to tune an abstract interpretation in accuracy and complexity.

Abstract domain refinements turn out to be useful also in transforming an attribute independent analysis into a more precise relational analysis (cf. [44]). Let us illustrate an example involving logic program analysis. Consider the following simple logic program defining an ancestor-like relation *anc*, where the relation R is specified by some database. It is then reasonable to assume that any computed answer in a successful derivation for a query $R(s,t)$ will ground both the arguments s and t.

$$c_1 : anc(X,Z) \; : - \; anc(X,Y), R(Y,Z).$$

$$c_2 : anc(X,Y) \; : - \; X = Y.$$

A typical groundness analysis is intended to statically detect whether any computed answer of any successful derivation for a generic predicate p will make ground some arguments of p. The most obvious domain for a groundness analysis is therefore a simple two-point lattice $Gr = \{g, ?\}$, where $g \leqslant ?$ (cf. [45]). Here, a type judgment $X:g$ says that a variable X is surely bound to a ground term, while $X:?$ represents uncertainty, i.e. it says that X can be bound to any term. Thus, by our assumptions on R, we have that $\langle R(X,Y), X:g \wedge Y:g \rangle$ represents the successful computations for R. Hence, an obvious analysis using Gr yields $\langle anc(X,Y), X:? \wedge Y:g \rangle$ for clause c_1, and $\langle anc(X,Y), X:? \wedge Y:? \rangle$ for clause 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 variables of the 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, and this propagates the groundness from Y (which is derived from R) to X. In order to overcome this problem, a richer abstract domain, called *Def*, has been introduced in the beginning of the 1990s (cf. [49]). *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. 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.

Some 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 [22]). Notable examples are Cousot and Cousot's *reduced cardinal power* [21] and Nielson's *tensor product* [55] operations. This paper originated by observing that the relational domain *Def* can be systematically derived from *Gr* by a generic procedure of domain refinement, which is a generalization of Cousot and Cousot's reduced cardinal power operation on abstract domains. This observation led us to devise a systematic operation for combining domains in abstract interpretation, called *reduced relative power*, which generalizes Cousot and Cousot reduced cardinal power [21].

1.1. Reduced relative power operation

Among the various abstract domain refinements existing in the literature, Cousot and Cousot's reduced cardinal 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 theoretical issues (e.g. see [16,21,22,24,30,39]) to applications (e.g. see [13,22,24,30,42,43,53,63]), as far as reduced cardinal power is concerned very little has been done after [21], 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. [21, 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 richer domain can be systematically derived by considering the lattice of all monotone functions 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 [21, 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 fact, in [21] the collecting semantics is defined by a lattice of assertions which is a Boolean algebra. In the present work, we carry on such an idea, and, by generalizing Cousot and Cousot's definition of reduced cardinal power, we introduce an operation of *reduced relative power* on 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 parametric with respect to a generic operation $\odot$ used to combine concrete denotations (hence, this justifies 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 $\lambda x.\alpha_2(d \odot \gamma_1(x))$ from D_1 to D_2, where d ranges over concrete values, 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 hinting that they establish a dependency relation 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

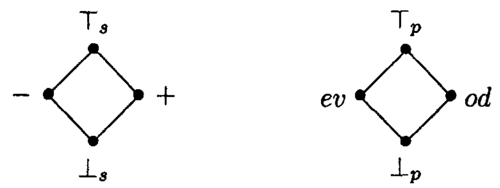


Fig. 1. The abstract domains *Sign*, on the left, and *Parity*, on the right.

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 this generalization is necessary to handle a number of important situations, ranging from abstract domains used in static program analysis to semantics design, where standard reduced cardinal power would not be otherwise applicable.

Let us see a simple example of reduced relative power, which in this case boils down to Cousot and Cousot's reduced cardinal power. Consider the simple and well-known abstract domains *Parity* and *Sign* for integer variable analysis, both depicted in Fig. 1. The objects in *Sign* and *Parity* 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 turns out that the monotone function $\{\top_s \mapsto ev,\ + \mapsto \bot_p,\ - \mapsto \bot_p,\ \bot_s \mapsto \bot_p\}$ from *Sign* to *Parity* is given by the dependency $\lambda x.\alpha_p(\{0\} \cap \gamma_s(x))$ in $Sign \overset{\cap}{\longmapsto} Parity$. Such dependency $\lambda x.\alpha_p(\{0\} \cap \gamma_s(x))$ represents the largest set of integers T which, once intersected with $\mathbb{Z}$ gives even numbers (i.e., T consists of even numbers only), 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 $\{0\}$, a new object which is represented neither in *Parity* nor in *Sign*.

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 $D_1 \overset{\wedge}{\longmapsto} D_2$ (relatively to the concrete operation $\wedge$ of *glb*) is a sublattice of, or even is equal to, the lattice of all the 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 operations of reduced relative power and Nielson's *tensor product* [54, 55] in abstract interpretation, a problem already raised by Nielson [54, p. 124] for Cousot and Cousot's reduced cardinal power. In fact, 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 refer to *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 a representation as a powerset-like completion for the domain of meet-autodependencies of a separated abstract domain.

1.2. Applications

In the field of logic program groundness analysis, we show that the well-known abstract domain of definite propositional formulae Def, introduced by Dart [25] for studying groundness in deductive databases and successively used by Marriott and Søndergaard [49] for ground-dependency analysis of logic programs, is isomorphic to the reduced relative power having the simpler Jones and Søndergaard's [45] Gr domain, representing (nonrelational) plain groundness information, as base and exponent. The reduced relative power is here necessary to deal with a non-Boolean concrete domain of order-ideals of substitutions, where Cousot and Cousot's reduced cardinal power would not be otherwise applicable. It is shown that Gr turns out to be a separated abstract domain, and therefore its dual-Moore-set completion is isomorphic to Def.

As hinted in [19, 23], 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 operation for designing new and more descriptive program semantics by incrementally composing some already existing and simpler ones. We apply the reduced relative power operation 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. 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 enjoys several advantages with respect to previous techniques for designing compositional semantics of logic programs, such as those described in [10] which constitute the basis for many compositional semantics in the s-semantics style (cf. [9, 15, 32]). 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 computed answer substitutions [28] and call patterns [32], Clark's [12, 28], Herbrand's [27] and Heyting's [46] semantics, and other semantics which can be derived by abstract interpretation from a more concrete (e.g. operational) logic program semantics (see [14, 34]). Moreover, this approach is general and independent from specific semantic representations (e.g., by means of atoms, clauses, etc.).

1.3. Structure of the paper

The remaining part of the paper is structured as follows. In Section 2, we introduce the notation used throughout the paper and recall some basic notions of abstract interpretation. In Section 3, the definition of reduced relative power is introduced, and its basic properties are studied. 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, the case of autodependencies is studied; in particular, we show that, for separated 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. A preliminary account of this paper has been presented in [36].

2. Preliminaries

Throughout the paper we will assume familiarity with the standard notions of lattice theory (see e.g. [7, 41]), abstract interpretation (see [20–22]), and logic programming (see e.g. [3, 47]). In this section, we briefly introduce some notation and recall some well-known notions.

2.1. Mathematical notation

Let A and B be sets. The powerset of A is denoted by $\wp(A)$, the cardinality of A is denoted by $|A|$, $A\backslash B$ denotes the set-theoretic difference between A and B, and $A \subset B$ denotes proper inclusion. If $X \subseteq A$ then $\overline{X}$ is the set-theoretic complement of X in A. A^* denotes the set of all finite (possibly empty) sequences of objects of A, where sequences are typically denoted by $\langle a_1, \dots, a_n \rangle$, or simply by $a_1, \dots, a_n$, 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 $s_1 :: s_2$, or simply by juxtaposition $s_1 s_2$. The sequence of symbols with first element a followed by the sequence s is denoted by $a|s$. If $\approx$ is an equivalence relation on A then, when clear from the context, an equivalence class $[a]_\approx$ is simply denoted by $[a]$. Also, we will often abuse notation by letting an element to denote its corresponding equivalence class. ω denotes the first infinite ordinal.

The set A endowed with a partial order $\leqslant$ is denoted by $\langle A, \leqslant \rangle$. If A is a poset, we usually denote by $\leqslant_A$ the corresponding partial order relation. By $x < y$ we mean $x \leqslant y$ and $x \neq y$. The dual of a poset A is denoted by A^{op}. A complete lattice A with partial ordering $\leqslant$, 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 $\langle A, \leqslant, \vee, \wedge, \top, \bot \rangle$. Whenever A is a lattice, $\vee_A$, $\wedge_A$, $\top_A$ and $\bot_A$ denote the corresponding basic operations and elements.

We will often abuse notation by denoting lattices with their poset notation. We use $\cong$ to denote isomorphism between ordered structures. If A is a pre-ordered set and $I \subseteq A$ then $\downarrow I \overset{\text{def}}{=} \{x \in A \mid \exists y \in I.\, x \leqslant_A y\}$. 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 $I = \downarrow I$ (note that we assume the empty set to be an order-ideal). 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 subset $\{x^i_j\}^{i \in I}_{j \in J(i)} \subseteq A$, where I and, for any $i \in I$, $J(i)$ are sets of indices, $\bigwedge_{i \in I}(\bigvee_{j \in J(i)} x^i_j) = \bigvee_{\varphi \in J^I}(\bigwedge_{i \in I} x^i_{\varphi(i)})$, where J^I is the set of all the functions $\varphi : I \to \bigcup_{i \in I} J(i)$ such that for any $i \in I$, $\varphi(i) \in J(i)$. By exchanging $\vee$ and $\wedge$ in the previous equality, one gets 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 $y \in A$, $y \wedge (\bigvee_{i \in I} x_i) = \bigvee_{i \in I}(y \wedge x_i)$. Let A be a meet semilattice with bottom $\bot$. The *pseudocomplement* of $a \in A$, if it exists, is the unique element $a^\star \in A$ such that $a \wedge a^\star = \bot$ and for any $x \in A$, $(a \wedge x = \bot) \Rightarrow (x \leqslant a^\star)$. A is *pseudocomplemented* if every element in A admits the pseudocomplement. Clearly, any cHa A is pseudocomplemented, where for any $a \in A$, $a^\star = \vee\{x \in A \mid a \wedge x = \bot\}$. 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 complete lattice of order-ideals $\langle \wp^{\downarrow}(A), \subseteq \rangle$ 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., $\bot < a$ and for all $x \in A$, $\bot < x \leqslant a$ 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 $x \in A \setminus \{\bot\}$ is the *lub* of the atoms which precede x, i.e. $x = \vee_A \{a \in At(A) \mid a \leqslant_A x\}$. *Dual-atoms* and *dual-atomicity* are dually defined. It is well known that any atomic cBa 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 $f : A \mapsto B$ to mean that f is a (total) function from A to B. We sometimes use Church's lambda notation for functions. The identity function $\lambda x.x$ is sometimes denoted by *id*. If $S \subseteq A$ then $f(S) = \{f(x) \mid x \in S\}$. By $g \circ f$ we denote the composition of the functions f and g, i.e. the function $\lambda x.g(f(x))$. The set of fixpoints of a function $f : A \mapsto A$ is denoted by $fp(f)$, and, assuming that A is a poset, if the least fixpoint of f exists then it is denoted by $lfp(f)$. For a complete lattice A, given $f : A \mapsto A$, for any $i \in \mathbb{N}$, the ith power $f^i : A \mapsto A$ of f is inductively defined as follows: For any $x \in A$, $f^0(x) \overset{\text{def}}{=} x$ and $f^{i+1}(x) \overset{\text{def}}{=} f(f^i(x))$. Let $\langle A, \leqslant_A, \vee_A, \wedge_A, \top_A, \bot_A \rangle$ and $\langle B, \leqslant_B, \vee_B, \wedge_B, \top_B, \bot_B \rangle$ be complete lattices and $f : A \mapsto B$. f is *strict* if $f(\bot_A) = \bot_B$. f is *additive* if for any $S \subseteq A$, $f(\vee_A S) = \vee_B f(S)$ (note that an additive f is strict as well), while f is *continuous* whenever this condition holds for any chain $S \subseteq A$. *Co-additivity* is dually defined. The complete lattice of total (respectively, monotone, continuous, additive) functions from the complete lattice A into the complete lattice B, endowed with the standard pointwise order $\sqsubseteq$ (i.e., $f \sqsubseteq g$ iff $\forall x \in A.\ f(x) \leqslant_B g(x)$), is denoted by $A \mapsto B$ (respectively, $A \xmapsto{\mathrm{m}} B$, $A \xmapsto{\mathrm{c}} B$, $A \xmapsto{\mathrm{a}} B$). Recall that in $A \mapsto B$ and $A \xmapsto{\mathrm{m}} B$, the *lub* $\sqcup$ and the *glb* $\sqcap$ are defined pointwise, i.e., if F is a family of functions then, for any $x \in A$, $(\sqcup F)(x) = \vee_B\{f(x) \mid f \in F\}$, and $(\sqcap F)(x) = \wedge_B\{f(x) \mid f \in F\}$, while in $A \xmapsto{\mathrm{c}} B$ and $A \xmapsto{\mathrm{a}} B$ just the *lub* $\sqcup$ is defined pointwise.

2.2. Abstract domains, Galois connections and closure operators

The standard Cousot and Cousot theory of abstract domains is based on the notion of Galois connection (cf. [20, 21]). If C and A are posets, and $\alpha : C \mapsto A$, $\gamma : A \mapsto C$ are monotone maps such that $\forall x \in C.\ x \leqslant_C \gamma(\alpha(x))$ and $\forall y \in A.\ \alpha(\gamma(y)) \leqslant_A y$, then (α, C, A, γ) is a called a *Galois connection* (G.c. for short) between C and A. If in addition $\forall y \in A.\ \alpha(\gamma(y)) = y$, then (α, C, A, γ) is called a *Galois insertion* (G.i. for short) of A in C. Let us recall that the notion of G.c. is equivalent to that of *adjunction*: If $\alpha : C \mapsto A$ and $\gamma : A \mapsto C$ then (α, C, A, γ) is a G.c. iff $\forall x \in C. \forall y \in A.\ \alpha(x) \leqslant_A y \Leftrightarrow x \leqslant_C \gamma(y)$. The map α (γ) is called the *left-* (*right-*)*adjoint* to γ (α). Galois connections and insertions enjoy the following well-known properties (see e.g. [22]) that we will freely use, often without mention, throughout the paper. Let us assume that (α, C, A, γ) is a G.c.

(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 as well, 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 vice versa. In particular, if A and C are complete lattices, then any additive map $\alpha : C \mapsto A$ uniquely determines a G.c. (α, C, A, γ), where its right-adjoint is $\gamma \overset{\mathrm{def}}{=} \lambda y.\ \vee_C \{x \in C \mid \alpha(x) \leqslant_A y\}$. This dually holds for any co-additive map $\gamma : A \mapsto C$.

(v) Let (α, C, 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 $(\alpha_i, C_i, A_i, \gamma_i)$, $i = 1, 2$, are *isomorphic* when (1) $C_1 \cong C_2$, with isomorphism $\jmath : C_1 \mapsto C_2$, (2) $A_1 \cong A_2$, with isomorphism $\imath : A_1 \mapsto A_2$, and (3) $\imath \circ \alpha_1 = \alpha_2 \circ \jmath$ (or, equivalently, $\jmath \circ \gamma_1 = \gamma_2 \circ \imath$). It is not difficult to show that for two G.i.'s, $(\alpha_1, C, A_1, \gamma_1) \cong (\alpha_2, C, A_2, \gamma_2)$ iff $\gamma_1(A_1) = \gamma_2(A_2)$. A G.c. (α, C, A, γ) is the composition of two G.c.'s $(\alpha_{C,D}, C, D, \gamma_{D,C})$ and $(\alpha_{D,A}, D, A, \gamma_{A,D})$ if $\alpha = \alpha_{D,A} \circ \alpha_{C,D}$ (equivalently, $\gamma = \gamma_{D,C} \circ \gamma_{A,D}$).

In the setting of abstract interpretation, when (α, C, A, γ) is a G.c., 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 *concretization* maps, respectively. Also, A is called an *abstraction* of C, and C a *concretization* of A. If

(α, C, A, γ) is a G.i., 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 values of C. It is known 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 (cf. [21]).

An *upper* (*lower*) *closure operator* on a poset $\langle C, \leqslant \rangle$ is an operator $\rho : C \mapsto C$ which is monotone, idempotent and extensive (reductive), i.e. for all $x \in A$, $x \leqslant \rho(x)$ ($\rho(x) \leqslant x$). 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. (α, C, A, γ), $\alpha \circ \gamma \in lco(A)$ and $\gamma \circ \alpha \in uco(C)$. In the following, we only deal with properties of upper closures, since for lower closures the corresponding properties can be retrieved by duality. Each closure operator $\rho \in uco(C)$ is uniquely determined by the set of its fixpoints, which is its image, i.e. $\rho(C) = fp(\rho)$. Let C be a complete lattice. In this case, $\rho \in uco(C)$ is characterized as follows: $\rho = \lambda y. \wedge \{x \in fp(\rho) \mid y \leqslant x\}$. A subset $X \subseteq C$ is the set of fixpoints of some upper closure operator iff X is a *Moore-set* of C (or, equivalently, a complete meet subsemilattice of C), i.e. $X = Mc(X) \overset{\text{def}}{=} \{\wedge Y \mid Y \subseteq X\}$ (note that $\top = \wedge\emptyset \in Mc(X)$). For any $X \subseteq C$, $Mc(X)$ is called the *Moore-closure* of X in C (dually, $dMc(X) \overset{\text{def}}{=} \{\vee Y \mid Y \subseteq X\}$ is called the *dual-Moore-closure* of X in C). Moreover, if $\rho \in uco(C)$ then its set of fixpoints $\rho(C)$ gives rise to the complete lattice $\langle \rho(C), \leqslant, \lambda X. \rho(\vee_C X), \wedge_C, \top_C, \rho(\bot_C) \rangle$. Thus, $\rho(C)$ is a complete meet subsemilattice (i.e. a Moore-set) 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 $\forall x \in C.\ \rho(x) \leqslant \eta(x)$, or, equivalently, $\rho \sqsubseteq \eta$ iff $\eta(C) \subseteq \rho(C)$;
- $(\bigsqcup_{i \in I} \rho_i)(x) = x \Leftrightarrow \forall i \in I.\ \rho_i(x) = x$;
- $(\bigsqcap_{i \in I} \rho_i)(x) = \bigwedge_{i \in I} \rho_i(x)$;
- $\lambda x. \top$ 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 $x \in C \setminus \{\top\}$, defined by $\phi_x(C) \overset{\text{def}}{=} \{x, \top\}$, i.e., for any $y \in C$,

$$\phi_x(y) \overset{\text{def}}{=} \begin{cases} x & \text{if } y \leqslant x, \\ \top & \text{otherwise.} \end{cases}$$

More on closure operators can be found in [18, 21, 50, 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 $\mathscr{L}$. The set of terms and atoms over $\mathscr{L}$ are denoted by *Term* and *Atom*, respectively. A *goal* is a (possibly empty) sequence of atoms (the empty goal is denoted by Λ). A (*definite*) *clause* is an object $h \leftarrow b_1, \ldots, b_n$, with $n \geqslant 0$, where h is an atom, called the *head*, and $b_1, \ldots, b_n$ 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 finite set of nontrivial bindings, i.e. $\sigma = \{x/\sigma(x) \mid \sigma(x) \neq x\}$. If σ is a substitution then $gr(\sigma)$ denotes the set of variables which are grounded by σ, i.e. $gr(\sigma) \overset{\text{def}}{=} \{x \in Var \mid var(\sigma(x)) = \emptyset\}$. 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 $\theta\sigma$, and is defined by $\theta\sigma \overset{\text{def}}{=} \lambda x.(x\theta)\sigma$. If Θ is a set of substitutions, then $s\Theta \overset{\text{def}}{=} \{s\theta \mid \theta \in \Theta\}$. *Sub* denotes the set of idempotent substitutions. A *variable renaming* is a (not necessarily idempotent) substitution which is a bijection on *Var*. Given two syntactic objects s and t, t is *more general* than s, denoted $s \preccurlyeq t$, iff there exists a substitution σ such that $s \equiv t\sigma$. Syntactic objects s and t are *equivalent up to renaming*, denoted $s \sim t$, iff $s \preccurlyeq t$ and $t \preccurlyeq s$. The quotients $Atom_{/\sim}$ and $Clause_{/\sim}$ turn out to be partially ordered with respect to $\preccurlyeq$. With abuse of notation, they will be still denoted by *Atom* and *Clause*. Since all the definitions in the paper are independent of 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) syntactic objects I, $\langle c_1, \ldots, c_n \rangle \ll_s I$ ($n \geqslant 0$) denotes 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 $ground(S) \overset{\text{def}}{=} \{s' \mid s \in S,\ s' \preccurlyeq s,\ var(s') = \emptyset\}$ denotes the set of 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 $s = t$, where s and t are syntactic objects and $=$ is interpreted as syntactic equality. Given a set e of equations, e is unifiable iff there exists a substitution θ, called unifier, such that for all $(s = t) \in e$, $s\theta \equiv t\theta$. 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 pair of objects, if such exists. This is not restrictive, since it is well known that all the idempotent *mgu*'s of a set of equations are equivalent up to renaming (cf. [47]). Hence, for a set of equations e, $mgu(e) = \theta$ means that e is unifiable (thus admits a

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

most general unifier) and θ is an idempotent most general unifier of e, i.e., θ is an idempotent substitution in the set $\{\sigma \in \mathit{Unif}(e) \mid \forall \sigma' \in \mathit{Unif}(e).\ \sigma' \preccurlyeq \sigma\}$. For sequences of syntactic objects $\langle s_1, \ldots, s_n \rangle$ and $\langle t_1, \ldots, t_n \rangle$, with $n \geqslant 1$, $\mathit{mgu}(\langle s_1, \ldots, s_n \rangle, \langle t_1, \ldots, t_n \rangle)$ is an equivalent notation for $\mathit{mgu}(\{s_1 = t_1, \ldots, s_n = t_n\})$.

3. The reduced relative power operation

The reduced cardinal power operation on abstract domains has been introduced by Cousot and Cousot in the very last section of their POPL'79 paper (cf. [21, 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 [21, Theorem 10.2.0.1] were given for a particular collecting concrete domain of program assertions, which was a complete Boolean lattice. We carry on the original Cousot and Cousot idea to a more general and standard abstract interpretation environment. This turns out to be useful in order to provide a better understanding of the potential behind Cousot and Cousot's reduced cardinal power construction as well as to let this construction be applicable in the context of abstract interpretation of possibly non Boolean concrete domains.

Definition 3.1. Let D, D_1, D_2 be posets, $\odot : D \times D \mapsto D$ be a monotone[2] operation, $\gamma_1 : D_1 \mapsto D$ and $\alpha_2 : D \mapsto D_2$ 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 $D_1 \overset{\odot}{\longmapsto} D_2 \overset{\text{def}}{=} \{\lambda x.\alpha_2(d \odot \gamma_1(x)) \in D_1 \mapsto D_2 \mid d \in D\}$, partially ordered by the pointwise ordering relation $\sqsubseteq$.

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 and generalize analogous objects considered in Cousot and Cousot's reduced cardinal power. Roughly speaking, in an abstract interpretation setting, the operation $\odot$ should be thought of as a sort 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. As stated by the following straightforward lemma, the very weak hypotheses of Definition 3.1 guarantee that every dependency is monotone.

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

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

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 corresponding concrete value $d \in D$. *Left-additivity* of the operation $\odot$ (i.e., for any $Y \subseteq D$ and $x \in D$, $(\vee Y) \odot x = \bigvee_{y \in Y} y \odot x$) 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 $\langle D, \leqslant, \odot \rangle$, where $\langle D, \leqslant \rangle$ is a complete lattice and $\odot : D \times D \mapsto D$ is associative, monotone and left-additive.

Whenever a semi-quantale plays the role of a concrete domain, we also term it *concrete semi-quantale*. As we will see later, this hypothesis will be sufficient to prove the basic results concerning the reduced relative power operation.

Algebraically, semi-quantales are weaker structures than the well-known *quantales*, which are semantic models of various kinds of logics. Mulvey [52] firstly introduced the term quantale in connection with his work on noncommutative C*-algebras. A structure $\langle D, \leqslant, \odot \rangle$ is a *quantale* if $\langle D, \leqslant \rangle$ 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*, i.e., a quantale $\langle D, \leqslant, \odot \rangle$ 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 [52] and Rosenthal [58] without assuming the existence of a unit. With respect to quantales, our semi-quantales require monotonicity and 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 operations for concrete composition. Left-additivity of $\odot$ in semi-quantales allows us to prove the following basic lemma.

Lemma 3.4. *If $\langle D, \leqslant, \odot \rangle$ is a semi-quantale, D_1 and D_2 are complete lattices, and α_2 is additive, then $\alpha : D \mapsto (D_1 \xmapsto{\mathrm{m}} D_2)$ defined as $\alpha(d) \stackrel{\text{def}}{=} \lambda x.\alpha_2(d \odot \gamma_1(x))$, is additive (and therefore the left-adjoint function of a G.c.).*

Proof. If $\{d_i\}_{i \in I} \subseteq D$ then, by left-additivity of $\odot$, we have

$$\begin{aligned}
\alpha\left(\bigvee_{i \in I} d_i\right) &= \lambda x.\alpha_2\left(\left(\bigvee_{i \in I} d_i\right) \odot \gamma_1(x)\right) \\
&= \lambda x.\alpha_2\left(\bigvee_{i \in I} (d_i \odot \gamma_1(x))\right) \\
&= \lambda x.\bigvee_{i \in I} \alpha_2(d_i \odot \gamma_1(x)).
\end{aligned}$$

Since $\lambda x.\bigvee_{i\in I}\alpha_2(d_i \odot \gamma_1(x))$ is the pointwise *lub* in $D_1 \xmapsto{\mathrm{m}} D_2$ 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 key definition.

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

The following basic result, which is an immediate consequence of Lemma 3.4, proves the correctness of the above definition.

Theorem 3.6. *Under the hypotheses of Definition* 3.5, *the map* $\alpha : D \mapsto (D_1 \xmapsto{\odot} D_2)$ *defined as* $\alpha(d) \stackrel{\text{def}}{=} \lambda x.\alpha_2(d \odot \gamma_1(x))$, *is the left-adjoint of a G.i.* $(\alpha, D, D_1 \xmapsto{\odot} D_2, \gamma)$.

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 G.c. $(\alpha_1, D, D_1, \gamma_1)$. In this case, D_2 and D_1 are called, 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 $D_1 \xmapsto{\odot} D_2$ is just the reduction of $D_1 \xmapsto{\mathrm{m}} D_2$ w.r.t. the G.c. of Lemma 3.4. Since D is assumed to be a complete lattice, by point (v) in Section 2.2, $D_1 \xmapsto{\odot} D_2$ is a complete lattice too. In particular, the following result holds.

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

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

$$\begin{aligned}\sqcup F &= \lambda x.\bigvee_{i\in I}\alpha_2(d_i \odot \gamma_1(x))\\ &= \lambda x.\alpha_2\left(\bigvee_{i\in I}(d_i \odot \gamma_1(x))\right)\\ &= \lambda x.\alpha_2\left(\left(\bigvee_{i\in I} d_i\right) \odot \gamma_1(x)\right).\end{aligned}$$

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 $D_1 \xmapsto{\odot} D_2$. □

Of course, whenever $\odot$ is the *glb* $\wedge$ of D, to ask for $\langle D, \leqslant, \wedge\rangle$ to be a semi-quantale in Definition 3.5 corresponds to require that $\langle D, \leqslant\rangle$ is a cHa. In this case, a dependency $\lambda x.\alpha_2(d \wedge \gamma_1(x)) \in D_1 \overset{\wedge}{\longmapsto} D_2$ is called a *meet-dependency*. Further, note that when $\odot = \vee$, $\langle D, \leqslant, \vee\rangle$ is trivially 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 $\langle \mathit{Var} \mapsto \{\mathit{false}, \mathit{true}\}, \Rightarrow\rangle$ (cf. [21]) is isomorphic to $\langle \wp(\mathit{Var}), \subseteq\rangle$, where the logical conjunction of program assertions simply amounts to set-intersection. Definition 3.5 and Theorem 3.6 provide the right generalization of Cousot and Cousot's original definition of reduced cardinal power and related correctness result [21, Theorem 10.2.0.1]. When the concrete quantale is $\langle \wp(X), \subseteq, \cap\rangle$, it is possible to provide an explicit characterization for the right-adjoint map in the G.c. 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 quantale, for some set X, and $(\alpha_1, \wp(X), D_1, \gamma_1)$ and $(\alpha_2, \wp(X), D_2, \gamma_2)$ be two G.c.'s. Then, the right-adjoint of α in Lemma 3.4 is the function $\gamma : (D_1 \overset{m}{\longmapsto} D_2) \mapsto D$ defined as $\gamma(f) \overset{\text{def}}{=} \{z \in X \mid \forall x \in D_1. (\alpha_1(\{z\}) \leqslant_1 x) \Rightarrow (\alpha_2(\{z\}) \leqslant_2 f(x))\}$.*

Proof. Let us show that $\alpha(S) \sqsubseteq f \Leftrightarrow S \subseteq \gamma(f)$, for any $S \in \wp(X)$ and $f \in D_1 \overset{m}{\longmapsto} D_2$.

($\Rightarrow$) Let $z \in S$, and let us show that $z \in \gamma(f)$. Assume that $x \in D_1$ and $\alpha_1(\{z\}) \leqslant_1 x$. Then, $z \in \gamma_1(x)$. By hypothesis, $\alpha_2(S \cap \gamma_1(x)) \leqslant_2 f(x)$, and therefore, $\alpha_2(\{z\}) \leqslant_2 \alpha_2(S \cap \gamma_1(x)) \leqslant_2 f(x)$.

($\Leftarrow$) Firstly, note that if $T \subseteq \gamma(f)$ then $\forall x \in D_1. (\alpha_1(T) \leqslant_1 x) \Rightarrow (\alpha_2(T) \leqslant_2 f(x))$. In fact, if $x \in D_1$ and $\alpha_1(T) \leqslant_1 x$ then $\forall t \in T.\ \alpha_1(\{t\}) \leqslant_1 x$; hence, $\forall t \in T.\ \alpha_2(\{t\}) \leqslant_2 f(x)$, which in turn implies $\alpha_2(T) \leqslant_2 f(x)$. Now, we want to prove that $\forall y \in D_1.\ \alpha_2(S \cap \gamma_1(y)) \leqslant_2 f(y)$. If $y \in D_1$ then $S \cap \gamma_1(y) \subseteq S \subseteq \gamma(f)$. Thus, we get $\forall x \in D_1.\ (\alpha_1(S \cap \gamma_1(y)) \leqslant_1 x) \Rightarrow (\alpha_2(S \cap \gamma_1(y)) \leqslant_2 f(x))$. From this, since $\alpha_1(S \cap \gamma_1(y)) \leqslant_1 \alpha_1(\gamma_1(y)) \leqslant_1 y$, we get $\alpha_2(S \cap \gamma_1(y)) \leqslant_2 f(y)$, as desired. □

Let us remark that the characterization of the concretization map provided by Proposition 3.8 follows the intuition. In fact, it would not be too difficult to see that for any $f \in D_1 \overset{m}{\longmapsto} D_2$, $\gamma(f) = \cup\{S \subseteq X \mid \forall x \in D_1. (\alpha_1(S) \leqslant_1 x) \Rightarrow (\alpha_2(S) \leqslant_2 f(x))\}$ holds; further, the above proof also shows that $\forall x \in D_1. (\alpha_1(\gamma(f)) \leqslant_1 x) \Rightarrow (\alpha_2(\gamma(f)) \leqslant_2 f(x))$. Thus, a given function $f \in D_1 \overset{m}{\longmapsto} D_2$ 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 obtained by applying f to x.

The following simple example, which formalizes that sketched in the introduction, illustrates how the reduced relative power actually works. In this case, our construction boils down to the Cousot and Cousot reduced cardinal power.

Example 3.9. Consider the abstract domains *Sign* and *Parity* for sign and parity analysis of integer variables already depicted in Fig. 1. The corresponding G.i.'s are $(\alpha_s, \wp(\mathbb{Z}), \mathit{Sign}, \gamma_s)$ and $(\alpha_p, \wp(\mathbb{Z}), \mathit{Parity}, \gamma_p)$, where the obvious abstraction and concretization mappings are as follows.

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

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

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

$$\gamma_p(x) = \begin{cases} \emptyset & \text{if } x = \bot_p, \\ \{y \in \mathbb{Z} \mid y \text{ is even}\} & \text{if } x = ev, \\ \{y \in \mathbb{Z} \mid y \text{ is odd}\} & \text{if } x = od, \\ \mathbb{Z} & \text{if } x = \top_p. \end{cases}$$

Here, the concrete (semi-)quantale is $\langle \wp(\mathbb{Z}), \subseteq, \cap \rangle$. $\mathit{Sign} \xrightarrow{\cap} \mathit{Parity}$ is therefore given by the set of functions $\{\lambda x . \alpha_p(S \cap \gamma_s(x)) \in \mathit{Sign} \xrightarrow{\mathrm{m}} \mathit{Parity} \mid S \in \wp(\mathbb{Z})\}$, and, by Theorem 3.6, it is related by the G.i. $(\alpha, \wp(\mathbb{Z}), \mathit{Sign} \xrightarrow{\cap} \mathit{Parity}, \gamma)$ to the concrete domain $\wp(\mathbb{Z})$. It turns out that $\mathit{Sign} \xrightarrow{\cap} \mathit{Parity}$ is able to represent several sets of integers which are representable neither in *Sign* nor in *Parity*. For example, $\mathit{Sign} \xrightarrow{\cap} \mathit{Parity}$ represents the following sets of integers:
– *The value* 0:

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

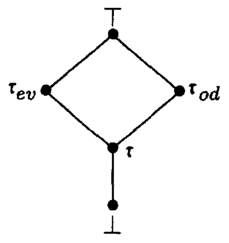


Fig. 2. The abstract domain $Par(\tau)$.

– *The positive and even integers*:

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

– *The positive or even integers*:

$$\lambda x.\alpha_p(\{x\in\mathbb{Z}\mid x>0\ \text{or}\ x\ \text{even}\}\cap\gamma_s(x)) =$$

$$=\begin{cases}\top_s & \mapsto \top_p\\ - & \mapsto ev\\ + & \mapsto \top_p\\ \bot_s & \mapsto \bot_p\end{cases}\ \overset{\gamma}{\longmapsto}\ \{x\in\mathbb{Z}\mid x>0\ \text{or}\ x\ \text{even}\}.$$

For instance, the dependency $f\overset{\text{def}}{=}\lambda x.\alpha_p(\{0\}\cap\gamma_s(x))$ 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 next example shows how reduced relative power can be defined relatively to a concrete semi-quantale $\langle D,\leqslant,\odot\rangle$, where $\odot$ is not commutative. A further relevant application showing the importance of our generalized 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 abstract domain $Par(\tau)$ in Fig. 2, 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. It is routine to prove that the following map γ gives rise to a G.i. $(\alpha, \wp(\ell), Par(\tau), \gamma)$:

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

Consider the operation of concatenation $@ : \wp(\ell) \times \wp(\ell) \to \wp(\ell)$ such that for any two sets of lists $A, B \in \wp(\ell)$, $A @ B \overset{\text{def}}{=} \{l :: l' \mid l \in A, l' \in B\}$, 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 and where, for any $A \in \wp(\ell)$, $A @ \emptyset = \emptyset = \emptyset @ A$. By Theorem 3.6, we can consider the reduced relative power $Par(\tau) \overset{@}{\longmapsto} Par(\tau)$. Then, a number of properties of lists can be characterized as elements in $Par(\tau) \overset{@}{\longmapsto} Par(\tau)$. For instance, it is easy to check that:

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

The following result shows that if the concrete semi-quantale $\langle D, \leqslant, \odot\rangle$ has a right-unit, i.e., there exists $\mathbf{1} \in D$ such that $\forall x \in D. x \odot \mathbf{1} = x$, and D_1 represents this unit, i.e., there exists $\mathbf{1}^\sharp \in D_1$ such that $\gamma_1(\mathbf{1}^\sharp) = \mathbf{1}$, then, by mapping a dependency $\lambda x.\alpha_2(d \odot \gamma_1(x))$ to $\alpha_2(d)$, D_2 can be viewed as a coherent abstraction of $D_1 \overset{\odot}{\longmapsto} D_2$. Thus, according to a general approach proposed in [29, 37], $D_1 \overset{\odot}{\longmapsto} D_2$ turns out to be a *refinement* of D_2, i.e., $D_1 \overset{\odot}{\longmapsto} D_2$ represents at least all the information represented by D_2.

Proposition 3.11. *Let $\langle D, \leqslant, \odot\rangle$ be a semi-quantale with a right-unit $\mathbf{1} \in D$, and $(\alpha_1, D, D_1, \gamma_1)$, $(\alpha_2, D, D_2, \gamma_2)$ be G.c.'s such that there exists $\mathbf{1}^\sharp \in D_1$ with $\gamma_1(\mathbf{1}^\sharp) = \mathbf{1}$. Then, $(\bar{\alpha}, D_1 \overset{\odot}{\longmapsto} D_2, D_2, \bar{\gamma})$ is a G.c., where, for all $d \in D$ and $y \in D_2$, $\bar{\alpha}(\lambda x.\alpha_2(d \odot \gamma_1(x))) \overset{\text{def}}{=} \alpha_2(d)$ and $\bar{\gamma}(y) \overset{\text{def}}{=} \lambda x.\alpha_2(\gamma_2(y) \odot \gamma_1(x))$. Further, $(\alpha_2, D, D_2, \gamma_2)$ is the composition of $(\alpha, D, D_1 \overset{\odot}{\longmapsto} D_2, \gamma)$ and $(\bar{\alpha}, D_1 \overset{\odot}{\longmapsto} D_2, D_2, \bar{\gamma})$.*

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 $\alpha_2(d) = \alpha_2(d \odot \gamma_1(\mathbf{1}^\sharp)) = \alpha_2(e \odot \gamma_1(\mathbf{1}^\sharp)) = \alpha_2(e)$, i.e. $\bar{\alpha}(\lambda x. \alpha_2(d \odot \gamma_1(x))) = \bar{\alpha}(\lambda x.\alpha_2(e \odot \gamma_1(x)))$. Thus, let us show that for any $d \in D$ and $y \in D_2$,

$\bar{\alpha}(\lambda x.\alpha_2(d \odot \gamma_1(x))) \leqslant_2 y \Leftrightarrow \lambda x.\alpha_2(d \odot \gamma_1(x)) \sqsubseteq \bar{\gamma}(y)$, i.e. $\alpha_2(d) \leqslant_2 y \Leftrightarrow \lambda x.\alpha_2(d \odot \gamma_1(x)) \sqsubseteq \lambda x.\alpha_2(\gamma_2(y) \odot \gamma_1(x))$.
($\Rightarrow$) From $\alpha_2(d) \leqslant_2 y$ we get $d \leqslant \gamma_2(y)$. Thus, if $x \in D_1$ then $\alpha_2(d \odot \gamma_1(x)) \leqslant_2 \alpha_2(\gamma_2(y) \odot \gamma_1(x))$.
($\Leftarrow$) For $x = \mathbf{1}^\sharp$, we get $\alpha_2(d) = \alpha_2(d \odot \gamma_1(\mathbf{1}^\sharp)) \leqslant_2 \alpha_2(\gamma_2(y) \odot \gamma_1(\mathbf{1}^\sharp)) = \alpha_2(\gamma_2(y))$. Hence, by applying γ_2, we have $d \leqslant \gamma_2(\alpha_2(d)) \leqslant \gamma_2(\alpha_2(\gamma_2(y)))$. But, by (i) in Section 2.2, $\gamma_2(\alpha_2(\gamma_2(y))) = \gamma_2(y)$, from which $\alpha_2(d) \leqslant_2 y$ follows.

Since $\alpha_2 = \bar{\alpha} \circ \alpha$ trivially holds, we also get the last statement of composition. □

The following is a consequence of Proposition 3.11 specialised to meet-dependencies.

Corollary 3.12. *Let D be a cHa and let $(\alpha_1, D, D_1, \gamma_1)$ and $(\alpha_2, D, D_2, \gamma_2)$ be G.c.'s. Then, $(\bar{\alpha}, D_1 \xrightarrow{\wedge} D_2, D_2, \bar{\gamma})$ is a G.c., where, for any $d \in D$ and $y \in D_2$, $\bar{\alpha}(\lambda x.\alpha_2(d \wedge \gamma_1(x))) \overset{\text{def}}{=} \alpha_2(d)$ and $\bar{\gamma}(y) \overset{\text{def}}{=} \lambda x.\alpha_2(\gamma_2(y) \wedge \gamma_1(x))$.*

Proof. If D is a cHa then $\langle D, \leqslant, \wedge \rangle$ is a unital quantale, having $\top_D$ as unit. Moreover, since $(\alpha_1, D, D_1, \gamma_1)$ is a G.c. then, by (ii) in Section 2.2, $\gamma_1(\top_1) = \top_D$ holds. Therefore, the thesis follows by Proposition 3.11. □

4. Additive dependencies

In general, for a concrete semi-quantale $\langle D, \leqslant, \odot \rangle$, from a lattice-theoretic point of view, dependencies in $D_1 \xrightarrow{\odot} D_2$ are merely monotone functions. In this section, we give sufficient conditions guaranteeing that $D_1 \xrightarrow{\odot} D_2$ is included in, or even isomorphic to, $D_1 \xrightarrow{\text{a}} D_2$, 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 operations of reduced relative power and *tensor product* in abstract interpretation,[3] a problem already raised by Nielson [54, p. 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 $(D_1 \xrightarrow{\text{a}} D_2^{\text{op}})^{\text{op}}$ (cf. [6, 54, 61]). Thus, we also have that $D_1 \xrightarrow{\text{a}} D_2 = (D_1 \otimes D_2^{\text{op}})^{\text{op}}$. Therefore, by duality, every known result for the tensor product also holds for the space of additive functions. For instance, it is known (cf. [61, 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

[3] See e.g. [54–57] for some applications of the tensor product in abstract interpretation.
[4] See [54, 61] for other equivalent definitions of tensor product.

in [62] for immediate fixpoint computation (which are typically useful in static program analysis), that are based on certain algebraic properties of distributive (Boolean) lattices.

As shown by the next example, in general $D_1 \overset{\odot}{\longmapsto} D_2$ is not a subset of $D_1 \overset{\mathrm{a}}{\longmapsto} D_2$.

Example 4.1. Consider the reduced relative power $Sign \overset{\cap}{\longmapsto} Parity$ of Example 3.9. In this case, it turns out that $f \overset{\mathrm{def}}{=} \lambda x.\alpha_p(\{0\} \cap \gamma_s(x)) \notin Sign \overset{\mathrm{a}}{\longmapsto} Parity$: In fact, $f(+ \vee_s -) = f(\top_s) = ev$, whilst $f(+) \vee_p f(-) = \bot_p$.

We say that a G.c. (α, D, 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 (α, D, A, γ) is additive, then A is called a *disjunctive* abstract domain (cf. [21]). 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 [21] that every abstract domain can be lifted to its *disjunctive completion*: Examples of disjunctive completions can be found, e.g. in [22, 24, 30, 39, 42, 43]. The interest in abstract interpretation based on disjunctive domains is clear: Disjunctive abstract domains allow a precise interpretation for disjunction, i.e. concrete and abstract *lub's* coincide up 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, it turns out that 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 $\langle D, \leqslant, \odot \rangle$ is a quantale and $(\alpha_1, D, D_1, \gamma_1)$ is an additive G.c., then $D_1 \overset{\odot}{\longmapsto} D_2$ is a complete join subsemilattice of $D_1 \overset{\mathrm{a}}{\longmapsto} D_2$.*

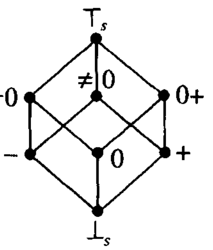
Proof. By hypothesis, $\odot$ is both left- and right-additive. The following equalities show that each dependency is additive. If $d \in D$ and $\{x_i\}_{i \in I} \subseteq D_1$, then

$$\begin{aligned}\alpha_2\left(d \odot \gamma_1\left(\bigvee_{i\in I} x_i\right)\right) &= \alpha_2\left(d \odot \left(\bigvee_{i\in I} \gamma_1(x_i)\right)\right)\\ &= \alpha_2\left(\bigvee_{i\in I}(d \odot \gamma_1(x_i))\right)\\ &= \bigvee_{i\in I} \alpha_2(d \odot \gamma_1(x_i)).\end{aligned}$$

As recalled in Section 2.1, *lub's* in $D_1 \overset{\mathrm{a}}{\longmapsto} D_2$ are pointwise. Thus, by Proposition 3.7, $D_1 \overset{\odot}{\longmapsto} D_2$ is a complete join subsemilattice of $D_1 \overset{\mathrm{a}}{\longmapsto} D_2$. □

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

Example 4.3. Consider the following standard abstract domain $Sign^{\pm 0}$ for sign analysis of integer variables [20, 21]:



Sign$^{\pm 0}$

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

$$\alpha(S) = \begin{cases} \bot_s & \text{if } S = \emptyset, \\ 0 & \text{if } S = \{0\}, \\ + & \text{if } \emptyset \neq S \subseteq \mathbb{Z}_{>0}, \\ 0+ & \text{if } \{0\} \subset S \subseteq \mathbb{Z}_{\geqslant 0}, \\ - & \text{if } \emptyset \neq S \subseteq \mathbb{Z}_{<0}, \\ -0 & \text{if } \{0\} \subset S \subseteq \mathbb{Z}_{\leqslant 0}, \\ \neq 0 & \text{if } S \subseteq \mathbb{Z}_{\neq 0},\ S \cap \mathbb{Z}_{<0} \neq \emptyset,\ S \cap \mathbb{Z}_{>0} \neq \emptyset, \\ \top_s & \text{otherwise}, \end{cases}$$

$$\gamma(x) = \begin{cases} \emptyset & \text{if } x = \bot_s, \\ \{0\} & \text{if } x = 0, \\ \mathbb{Z}_{>0} & \text{if } x = +, \\ \mathbb{Z}_{\geqslant 0} & \text{if } x = 0+, \\ \mathbb{Z}_{<0} & \text{if } x = -, \\ \mathbb{Z}_{\leqslant 0} & \text{if } x = -0, \\ \mathbb{Z}_{\neq 0} & \text{if } x = \neq 0, \\ \mathbb{Z} & \text{if } x = \top_s. \end{cases}$$

By Proposition 4.2, $Sign^{\pm 0}$ can be combined with the domain *Parity*, defined in Example 3.9, so that $Sign^{\pm 0} \xrightarrow{\sqcap} Parity \subseteq Sign^{\pm 0} \xrightarrow{a} Parity$. This inclusion turns out to be strict. In fact, consider $f : Sign^{\pm 0} \xrightarrow{a} Parity$ defined by

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

It turns out that $f \notin Sign^{\pm 0} \xrightarrow{\sqcap} Parity$: 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 $\langle D, \leqslant, \wedge \rangle$, that $\gamma_1 : D_1 \mapsto D$ is a monotone map, and that $(\alpha_2, D, D_2, \gamma_2)$ is a G.c.

It is known from Shmuely's paper [61, Theorem 2.5] that for any 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 so-called *basis* of the tensor product, which is a certain subset $\mathscr{B}(D_1 \otimes D_2) \subseteq D_1 \otimes D_2$. Since $D_1 \otimes D_2^{\mathrm{op}} = (D_1 \xmapsto{\mathrm{a}} D_2)^{\mathrm{op}}$, by duality, we have that any additive function $f \in D_1 \xmapsto{\mathrm{a}} D_2$ can be generated as a *glb* in $D_1 \xmapsto{\mathrm{a}} D_2$ (i.e. *lub* in $D_1 \otimes D_2^{\mathrm{op}}$) of a suitable subset of $\mathscr{B}(D_1 \otimes D_2^{\mathrm{op}})$, where $\mathscr{B}(D_1 \otimes D_2^{\mathrm{op}}) \subseteq D_1 \xmapsto{\mathrm{a}} D_2$. In the following, we will denote $\mathscr{B}(D_1 \otimes D_2^{\mathrm{op}})$ simply by $\mathscr{B}(D_1, D_2)$. From the definition of basis of the tensor product given by Shmuely [61], by duality, we have that $\mathscr{B}(D_1, D_2) \overset{\mathrm{def}}{=} \{\ell_a^b \in D_1 \xmapsto{\mathrm{a}} D_2 \mid a \in D_1, b \in D_2\}$, where for any $a, x \in D_1$ and $b \in D_2$:

$$\ell_a^b(x) \overset{\mathrm{def}}{=} \begin{cases} \bot & \text{if } x = \bot, \\ b & \text{if } \bot < x \leqslant a, \\ \top & \text{if } x \nleqslant a. \end{cases}$$

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

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

Lemma 4.4. *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) $\forall x \in D_1 . \bot_{D_1} < x \leqslant a \Rightarrow \alpha_2(d_a^b \wedge \gamma_1(x)) = b$ *and* (b) $\forall x \in D_1 .\ x \nleqslant a \Rightarrow \alpha_2(d_a^b \wedge \gamma_1(x)) = \top_{D_2}$. *Then, $\mathscr{B}(D_1, D_2) \subseteq D_1 \xmapsto{\wedge} D_2$.*

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

The strictness condition for γ_1 (i.e. $\gamma_1(\bot_{D_1}) = \bot_D$) provides here a sufficient condition in order that $\lambda x . \alpha_2(d_a^b \wedge \gamma_1(x))$ is a strict function. This is a quite reasonable condition, saying that $\bot_D$ is the only concrete object whose abstraction in D_1 is the bottom.

The key point in order to include $\mathscr{B}(D_1, D_2)$ into $D_1 \xmapsto{\wedge} D_2$ is therefore the existence of concrete values $d_a^b \in D$ for which conditions (a) and (b) of Lemma 4.4 are verified. A sufficient condition for the existence of such elements is given by the following result.

Theorem 4.5. *Let $(\alpha_1, D, D_1, \gamma_1)$ be a G.c., and assume that the following conditions hold*:
(i) D_1 *is a cBa*,
(ii) γ_1 *is strict*,
(iii) $\forall x \in D_1 . \forall y \in D_2 . x \neq \bot_{D_1} \Rightarrow \alpha_2(\gamma_1(x) \wedge \gamma_2(y)) = y$.
Then, $\mathscr{B}(D_1, D_2) \subseteq D_1 \overset{\wedge}{\longmapsto} D_2$.

Proof. We have to prove that for any $a \in D_1$ and $b \in D_2$, $\ell_a^b \in D_1 \overset{\wedge}{\longmapsto} D_2$. Let $a \in D_1$ and $b \in D_2$, and define $d_a^b \overset{\text{def}}{=} \gamma_1(a^*) \vee \gamma_2(b) \in D$. 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 4.4. Assume that $\bot_{D_1} < x \leqslant a$. Note that in a Boolean lattice, if $x \leqslant a$ then $x \wedge a^* = \bot$: This holds because $x \wedge a^* \leqslant a \wedge a^* = \bot$. Then, we have that

$$\begin{aligned}
\alpha_2(d_a^b \wedge \gamma_1(x)) &= \\
\alpha_2((\gamma_1(a^*) \vee \gamma_2(b)) \wedge \gamma_1(x)) &= \text{(by distributivity and co-additivity of } \gamma_1\text{)} \\
\alpha_2(\gamma_1(a^* \wedge x) \vee (\gamma_1(x) \wedge \gamma_2(b))) &= \text{(since } a^* \wedge x = \bot_{D_1}\text{)} \\
\alpha_2(\gamma_1(\bot_{D_1}) \vee (\gamma_1(x) \wedge \gamma_2(b))) &= \text{(by strictness of } \gamma_1\text{)} \\
\alpha_2(\gamma_1(x) \wedge \gamma_2(b)) &= \text{(by hypothesis (iii))} \\
b.&
\end{aligned}$$

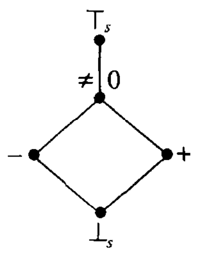
Assume that $x \nleqslant a$. By distributivity of D and co-additivity of γ_1, we have that $\alpha_2(d_a^b \wedge \gamma_1(x)) = \alpha_2((\gamma_1(x) \wedge \gamma_2(b)) \vee \gamma_1(a^* \wedge x))$. Thus, $\alpha_2(d_a^b \wedge \gamma_1(x)) \geqslant \alpha_2(\gamma_1(a^* \wedge x))$. Note that, because D_1 is Boolean, $a^* \wedge x = \bot_{D_1}$ implies $x \leqslant a^{**} = a$. Hence, $a^* \wedge x = \bot_{D_1}$ iff $x \leqslant a$. Then, because $x \nleqslant a$, we have $a^* \wedge x \neq \bot_{D_1}$. Moreover, since, by G.c., $\gamma_2(\top_{D_2}) = \top_D$, we have that $\alpha_2(\gamma_1(a^* \wedge x)) = \alpha_2(\gamma_1(a^* \wedge x) \wedge \gamma_2(\top_{D_2}))$. Thus, by (iii) and since $a^* \wedge x \neq \bot_{D_1}$, we get $\alpha_2(\gamma_1(a^* \wedge x)) = \top_{D_2}$, and therefore $\alpha_2(d_a^b \wedge \gamma_1(x)) = \top_{D_2}$. $\square$

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

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 4.5, i.e. $d_a^b \overset{\text{def}}{=} \gamma_1(a^*) \vee \gamma_2(b)$. Notice that requiring D_1 to be Boolean in Theorem 4.5 is crucial. In fact, the weaker notion of relative pseudocomplementation (or even pseudocomplementation) is not sufficient, as shown in Example 4.6 below. Indeed, in the proof of Theorem 4.5 it is needed that for all

$a \in D_1$, $a = a^{**}$, and it is well known (cf. [41]) 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 4.5 is tight.

Example 4.6. Consider the following abstract domain $Sign^{\neq 0}$.



$Sign^{\neq 0}$

$(\alpha, \wp(\mathbb{Z}), Sign^{\neq 0}, \gamma)$ is an additive G.i. (actually, it turns out that $Sign^{\neq 0}$ is the disjunctive completion of the domain *Sign* in Example 3.9), where γ is the restriction on $Sign^{\neq 0}$ of the concretization defined in Example 4.3. $Sign^{\neq 0}$ 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 4.5. However, $Sign^{\neq 0}$ is not Boolean: In fact, $\neq 0$ does not admit a complement. It turns out that $Sign^{\neq 0} \overset{\sqcap}{\longmapsto} Parity$ does not contain $\mathscr{B}(Sign^{\neq 0}, Parity)$. In particular, the generator

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

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

Fact 4.7. $\ell^{ev}_{\neq 0} \notin Sign^{\neq 0} \overset{\sqcap}{\longmapsto} Parity$.

Proof. Assume by contradiction that $\ell^{ev}_{\neq 0} \in Sign^{\neq 0} \overset{\sqcap}{\longmapsto} Parity$. 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

$$\begin{aligned} \alpha_p(S \cap \gamma(\top_s)) &= (\text{since } \gamma(\top_s) = \mathbb{Z} = \gamma(\neq 0) \cup \{0\}) \\ \alpha_p(S \cap (\gamma(\neq 0) \cup \{0\})) &= \\ \alpha_p((S \cap \gamma(\neq 0)) \cup (S \cap \{0\})) &= \\ \alpha_p(S \cap \gamma(\neq 0)) \vee \alpha_p(S \cap \{0\}) &= (\text{since } \alpha_p(S \cap \gamma(\neq 0)) = ev) \\ ev \vee \alpha_p(S \cap \{0\}) &= (\text{since } \forall S \subseteq \mathbb{Z}. \alpha_p(S \cap \{0\}) \leqslant ev) \\ & ev \end{aligned}$$

which is a contradiction. □

Thus, we conclude that $\mathscr{B}(\mathit{Sign}^{\neq 0}, \mathit{Parity}) \not\subseteq \mathit{Sign}^{\neq 0} \xmapsto{\sqcap} \mathit{Parity}$, and therefore, by Proposition 4.2, $\mathit{Sign}^{\neq 0} \xmapsto{\sqcap} \mathit{Parity} \subset \mathit{Sign}^{\neq 0} \xmapsto{\mathrm{a}} \mathit{Parity}$.

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

Lemma 4.8. *Let D and A be complete lattices and (α, D, A, γ) be an additive G.i.*
(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 completely meet-distributive case. The noncomplete one is analogous. Let $x \in A$ and $Y \subseteq A$:

$$\begin{aligned} x \wedge (\vee Y) &= \alpha(\gamma(x \wedge (\vee Y))) \\ &= \alpha(\gamma(x) \wedge (\vee \gamma(Y))) \\ &= \alpha\left(\bigvee_{y \in Y} \gamma(x) \wedge \gamma(y)\right) \\ &= \alpha\left(\bigvee_{y \in Y} \gamma(x \wedge y)\right) \\ &= \bigvee_{y \in Y} \alpha(\gamma(x \wedge y)) \\ &= \bigvee_{y \in Y} x \wedge y. \end{aligned}$$

(2) We have to prove that $\gamma(a) \wedge \gamma(a^*) = \bot_D$ and $\gamma(a) \vee \gamma(a^*) = \top_D$. Notice that additivity of γ implies strictness of γ. By co-additivity and strictness of γ, $\gamma(a) \wedge \gamma(a^*) = \gamma(a \wedge a^*) = \gamma(\bot_A) = \bot_D$. By additivity of γ, $\gamma(a) \vee \gamma(a^*) = \gamma(a \vee a^*) = \gamma(\top_A) = \top_D$. □

Let us remark that in Theorem 4.5, as well as in Lemma 4.4, it is not required that D_1 is disjunctive. Note however that if $(\alpha_1, D, D_1, \gamma_1)$ is additive, then the strictness condition (ii) always holds. Moreover, in this case, by Lemma 4.8(1), D_1 is a cHa, and, in particular, it is pseudocomplemented. Thus, in order to verify that D_1 is Boolean (i.e., condition (i) of Theorem 4.5), it is sufficient to check whether pseudocomplementation in D_1 is idempotent.

Condition (iii) of Theorem 4.5 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 (non-bottom) abstract values of D_1, i.e., $\forall x \in D_1 \setminus \{\bot_{D_1}\}. \alpha_2(\gamma_1(x)) = \top_{D_2}$. We can observe that while condition (iii) is satisfied for the abstract domains $D_1 = \mathit{Sign}$ and $D_2 = \mathit{Parity}$ of Example 3.9, it does not hold if Sign is replaced by $\mathit{Sign}^{\pm 0}$ of Example 4.3: In fact, $\alpha_2(\gamma_1(0) \cap \gamma_2(od)) = \bot_P \neq od$. This is consistent with Example 4.3, where it is shown that $\mathit{Sign}^{\pm 0} \overset{\sqcap}{\longmapsto} \mathit{Parity}$ does not contain the generator $\ell_0^{od} \in \mathscr{B}(\mathit{Sign}^{\pm 0}, \mathit{Parity})$ (denoted by f in that example). Since $\mathit{Sign}^{\pm 0}$ and *Parity* both satisfy conditions (i) and (ii) of Theorem 4.5, this observation also implies the tightness of condition (iii) in Theorem 4.5. On the other hand, $\mathit{Parity} \overset{\sqcap}{\longmapsto} \mathit{Sign}$ contains the whole set of generators of $\mathit{Parity} \overset{\mathrm{a}}{\longmapsto} \mathit{Sign}$, as shown by the next example.

Example 4.9. Consider the abstract 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 4.5, whenever joined, either as base or as exponent, with *Parity*. However, note that, while $\mathit{Sign} \overset{\sqcap}{\longmapsto} \mathit{Parity}$ contains, by Theorem 4.5, the generators of $\mathit{Sign} \overset{\mathrm{a}}{\longmapsto} \mathit{Parity}$, it may also contain nonadditive functions (see Example 4.1). On the other hand, since $(\alpha_p, \wp(\mathbb{Z}), \mathit{Parity}, \gamma_p)$ is an additive G.i., by Proposition 4.2, we have that $\mathit{Parity} \overset{\sqcap}{\longmapsto} \mathit{Sign} \subseteq \mathit{Parity} \overset{\mathrm{a}}{\longmapsto} \mathit{Sign}$. Moreover, by Theorem 4.5, $\mathit{Parity} \overset{\sqcap}{\longmapsto} \mathit{Sign}$ also contains all the generators of $\mathit{Parity} \overset{\mathrm{a}}{\longmapsto} \mathit{Sign}$. The generators are here all constructed as dependencies by using the following sets of integer numbers $d_a^b \overset{\mathrm{def}}{=} \gamma_p(a^*) \cup \gamma_s(b)$, for $a \in \mathit{Parity}$ and $b \in \mathit{Sign}$:

$$d_{\bot_p}^{\bot_s} = d_{\bot_p}^{+} = d_{\bot_p}^{-} = d_{\bot_p}^{\top_s} = d_{ev}^{\top_s} = d_{od}^{\top_s} = d_{\top_p}^{\top_s} = \mathbb{Z}, \quad d_{\top_p}^{\bot_s} = \emptyset,$$

$$d_{\top_p}^{+} = \{x \in \mathbb{Z} \mid x > 0\}, \quad d_{\top_p}^{-} = \{x \in \mathbb{Z} \mid x < 0\},$$

$$d_{ev}^{\bot_s} = \{x \in \mathbb{Z} \mid x \text{ odd}\}, \quad d_{od}^{\bot_s} = \{x \in \mathbb{Z} \mid x \text{ even}\},$$

$$d_{ev}^{+} = \{x \in \mathbb{Z} \mid x \text{ odd}\} \cup \{x \in \mathbb{Z} \mid x > 0\}, \quad d_{ev}^{-} = \{x \in \mathbb{Z} \mid x \text{ odd}\} \cup \{x \in \mathbb{Z} \mid x < 0\},$$

$$d_{od}^{+} = \{x \in \mathbb{Z} \mid x \text{ even}\} \cup \{x \in \mathbb{Z} \mid x > 0\}, \quad d_{od}^{-} = \{x \in \mathbb{Z} \mid x \text{ even}\} \cup \{x \in \mathbb{Z} \mid x < 0\}.$$

Any dependency in $\mathit{Parity} \overset{\sqcap}{\longmapsto} \mathit{Sign}$ can be represented as a *glb* of $\mathit{Parity} \overset{\mathrm{a}}{\longmapsto} \mathit{Sign}$ of some generators. For instance, the dependency $\lambda x.\alpha_s(\{-1,2\} \cap \gamma_p(x)) = \{\top_p \mapsto \top_s, ev \mapsto +, od \mapsto -, \bot_p \mapsto \bot_s\}$ is the *glb* – in this case pointwise – of the generators $\lambda x.\alpha_s(d_{ev}^{+} \cap \gamma_p(x)) = \{\top_p \mapsto \top_s, ev \mapsto +, od \mapsto \top_p, \bot_p \mapsto \bot_s\}$ and $\lambda x.\alpha_s(d_{od}^{-} \cap \gamma_p(x)) = \{\top_p \mapsto \top_s, ev \mapsto \top_s, od \mapsto -, \bot_p \mapsto \bot_s\}$.

Let us assume that $\langle D, \leqslant_D \rangle = \langle \wp(X), \subseteq \rangle$, for some set X. In this case, under the hypotheses of Theorem 4.5, the next 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, since, by Lemma 4.8, $\gamma_1(a^*) \cup \gamma_2(b) = \overline{\gamma_1(a)} \cup \gamma_2(b)$, any ℓ_a^b represents precisely the classical implicational relation $x \in \gamma_1(a) \Rightarrow x \in \gamma_2(b)$.

Lemma 4.10. *Let $\langle \wp(X), \subseteq \rangle$, for some set X, be the concrete domain, $(\alpha_1, \wp(X), D_1, \gamma_1)$ be an additive G.i., and $(\alpha_2, \wp(X), D_2, \gamma_2)$ be a G.i. Let us assume that* *conditions* (i)–(iii) *of Theorem* 4.5 *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 4.8(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 4.5, $\lambda x.\alpha_2((\gamma_1(a^*) \cup \gamma_2(b)) \cap \gamma_1(x)) = \ell_a^b$. Thus, assume, by contradiction, that $\gamma_1(a^*) \cup \gamma_2(b) \subset \gamma(\ell_a^b)$. Hence, if $d \overset{\text{def}}{=} \gamma(\ell_a^b) \setminus (\gamma_1(a^*) \cup \gamma_2(b))$, then $d \neq \emptyset$. Since, by hypothesis, D_1 is Boolean and $(\alpha_1, \wp(X), D_1, \gamma_1)$ is additive, by Lemma 4.8(2), for any $x \in D_1$, $\gamma_1(x^*) = \overline{\gamma_1(x)}$. Moreover, $\gamma_1(\bot_{D_1}) = \emptyset$ holds. Also, from $d \cap \gamma_1(a^*) = \emptyset$, we get $d \subseteq \gamma_1(a)$. Further, $a \neq \bot$, 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:

$$\begin{aligned}
b &= \alpha_2(\gamma(\ell_a^b) \cap \gamma_1(a)) \\
&= \alpha_2((d \cup \gamma_1(a^*) \cup \gamma_2(b)) \cap \gamma_1(a)) \\
&= \alpha_2((d \cap \gamma_1(a)) \cup (\gamma_1(a^*) \cap \gamma_1(a)) \cup (\gamma_2(b) \cap \gamma_1(a))) \\
&= \alpha_2((d \cap \gamma_1(a)) \cup (\gamma_2(b) \cap \gamma_1(a))) \\
&= \alpha_2(d \cap \gamma_1(a)) \vee \alpha_2(\gamma_2(b) \cap \gamma_1(a)) \\
&= \alpha_2(d \cap \gamma_1(a)) \vee b.
\end{aligned}$$

Hence, $\alpha_2(d \cap \gamma_1(a)) \leqslant b$, from which $\gamma_2(\alpha_2(d \cap \gamma_1(a))) \subseteq \gamma_2(b)$. Since $d \subseteq \gamma_1(a)$, we get $\gamma_2(\alpha_2(d)) \subseteq \gamma_2(b)$, which implies that $d \subseteq \gamma_2(b)$. Then, since $d = d \cap \gamma_2(b) = \emptyset$, we get a contradiction by the fact that $d \neq \emptyset$. □

The last result of this section shows that under the hypotheses of Lemma 4.10, if D_1 is an atomic cBa – 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 $D_1 \overset{\wedge}{\longmapsto} D_2 = D_1 \overset{\text{a}}{\longmapsto} D_2$, and therefore, since both are ordered pointwise, they actually are the same complete lattice.

Theorem 4.11. *Assume that the hypotheses in Lemma* 4.10 *hold and that* D_1 *is an atomic cBa. Then,* $D_1 \overset{\wedge}{\longmapsto} D_2 = D_1 \overset{\text{a}}{\longmapsto} D_2$.

Proof. First, notice that condition (i) in Theorem 4.5 is satisfied (since γ_1 is additive). Then, by Proposition 4.2 and Theorem 4.5, we have that $\mathscr{B}(D_1, D_2) \subseteq D_1 \overset{\wedge}{\longmapsto} D_2 \subseteq D_1 \overset{\text{a}}{\longmapsto} D_2$. We prove that $D_1 \overset{\text{a}}{\longmapsto} D_2 \subseteq D_1 \overset{\wedge}{\longmapsto} D_2$. Let $f \in D_1 \overset{\text{a}}{\longmapsto} D_2$. Recall that $At(D_1)$ denotes 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 $D_1 \overset{\wedge}{\longmapsto} D_2$. By Lemma 4.10, 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 $(\alpha, D, D_1 \overset{\wedge}{\longmapsto} D_2, \gamma)$ is a G.i. Also, observe that for any $x, y \in At(D_1)$, if $x \neq y$ then $x \leqslant y^*$: In fact, $x \wedge y = \bot_{D_1}$ implies $x \leqslant y^*$. Then, for any $x \in At(D_1)$, we have

$$\begin{aligned}
(\sqcap\{\ell_a^{f(a)} \mid a \in At(D_1)\})(x) &= (\text{since } \alpha \circ \gamma = id)\\
\alpha(\gamma(\sqcap\{\ell_a^{f(a)} \mid a \in At(D_1)\}))(x) &= (\text{by co-additivity of } \gamma)\\
\alpha(\cap\{\gamma(\ell_a^{f(a)}) \mid a \in At(D_1)\})(x) &=\\
\alpha_2\left(\bigcap_{a\in At(D_1)} (\gamma_1(a^*) \cup \gamma_2(f(a))) \cap \gamma_1(x)\right) &= (\text{by co-additivity of } \gamma_1)\\
\alpha_2\left(\bigcap_{a\in At(D_1)} (\gamma_1(a^* \wedge x) \cup (\gamma_2(f(a)) \cap \gamma_1(x)))\right) &= (\text{since } x \neq a \Rightarrow a^* \wedge x = x)\\
\alpha_2(\gamma_2(f(x)) \cap \gamma_1(x)) &= (\text{since } x \neq \bot, \text{ and by (iii) of}\\
&\qquad \text{Theorem 4.5})\\
&\ f(x).
\end{aligned}$$

Hence, the functions f and $\sqcap\{\ell_a^{f(a)} \mid a \in At(D_1)\}$ coincide on $At(D_1)$. Since any additive function on an atomic complete lattice is clearly determined by its values on the atoms, actually we get $f = \sqcap\{\ell_a^{f(a)} \mid a \in At(D_1)\}$, and therefore $f \in D_1 \overset{\wedge}{\longmapsto} D_2$, thus concluding the proof. □

Example 4.12. The base *Parity* and the exponent *Sign* (both in Example 3.9) satisfy all the hypotheses of Theorem 4.11. Thus, $Parity \overset{\sqcap}{\longmapsto} Sign = Parity \overset{\mathrm{a}}{\longmapsto} Sign$. In particular, $Parity \overset{\sqcap}{\longmapsto} Sign = (Parity \otimes Sign^{\mathrm{op}})^{\mathrm{op}}$ also holds. As a sample consequence of this fact, since both *Parity* and *Sign* are finite cBa's, we get that $Parity \overset{\sqcap}{\longmapsto} Sign$ is a finite cBa as well. Actually, (the image of) $Parity \overset{\sqcap}{\longmapsto} Sign$ (in $\wp(\mathbb{Z})$) can be easily computed by considering the closure by intersection of all the elements d_a^b, for $a \in Parity$ and $b \in Sign$, of Example 4.9.

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 functions in the reduced relative power are called *autodependencies*.

Definition 5.1. Let $\langle D, \leqslant, \odot\rangle$ be a semi-quantale and (α, D, A, γ) be a G.c. The lattice of *autodependencies* of A is $A \overset{\odot}{\longmapsto} A$.

Thus, $A \overset{\odot}{\longmapsto} A \overset{\mathrm{def}}{=} \{\lambda x.\alpha(d \odot \gamma(x)) \in A \overset{\mathrm{m}}{\longmapsto} A \mid d \in D\}$. In the following, we adopt a more concise notation and denote $A \overset{\odot}{\longmapsto} A$ by $Dep_{\odot}(A)$. Note that $Dep_{\odot}(A)$ is precisely the set of all the best correct approximations[5] of the family of concrete operators $\{\lambda x.d \odot x\}_{d\in D} \subseteq D \xrightarrow{m} D$. By 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 $(\alpha^{\natural}, D, Dep_{\odot}(A), \gamma^{\natural})$ the G.i. 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 refinement of A.

Autodependencies on an abstract domain constitute an important case of study, 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., $\forall x,y.x\odot y \leqslant x,y$ $(x,y \leqslant x \odot y)$ – on D, we have that each autodependency is a lower (upper) closure operator on A.

Theorem 5.2. *Let $\langle D, \leqslant, \odot\rangle$ be a semi-quantale such that $\odot$ is idempotent and let (α, D, A, γ) be a G.c.*

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

(ii) *If (α, D, A, γ) is a G.i. and $\odot$ is an upper bound such that for any $d, d' \in D$, $d \odot \gamma(\alpha(d')) \leqslant \gamma(\alpha(d \odot d'))$, 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. Idempotency is proved as follows:

$$\begin{aligned}
\alpha(d \odot \gamma(x)) &= \alpha((d \odot d) \odot \gamma(x)) \\
&= \alpha(d \odot (d \odot \gamma(x))) \\
&\leqslant \alpha(d \odot \gamma(\alpha(d \odot \gamma(x)))) \\
&\leqslant \alpha(\gamma(\alpha(d \odot \gamma(x)))) \\
&= \alpha(d \odot \gamma(x)).
\end{aligned}$$

Finally, since $\odot$ is a lower bound, $\alpha(d \odot \gamma(x)) \leqslant \alpha(\gamma(x)) \leqslant x$, 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 the idempotency:

$$\begin{aligned}
\alpha(d \odot \gamma(\alpha(d \odot \gamma(x)))) &\leqslant \alpha(\gamma(\alpha(d \odot (d \odot \gamma(x))))) \\
&= \alpha(d \odot (d \odot \gamma(x))) \\
&\leqslant \alpha(d \odot \gamma(\alpha(d \odot \gamma(x)))).
\end{aligned}$$

[5] If (α, D, A, γ) is a G.c. and $f : D \mapsto D$, then the *best correct approximation* of f in A is defined as $\alpha \circ f \circ \gamma : A \mapsto A$ (cf. [21]).

Since $\alpha \circ \gamma = id$ and $\odot$ is an upper bound, $x = \alpha(\gamma(x)) \leqslant \alpha(d \odot \gamma(x))$, which proves extensivity. Finally, since the *lub* in $Dep_{\odot}(A)$ is pointwise, $Dep_{\odot}(A)$ is a complete join subsemilattice of $uco(A)$ (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 (α, D, A, γ) be a G.c.*
(i) *If D is a cHa then $Dep_{\wedge}(A)$ is a complete join subsemilattice of $lco(A)$.*
(ii) *If (α, D, A, γ) is a G.i. then $Dep_{\vee}(A)$ is a complete join subsemilattice of $uco(A)$.*

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

$$\begin{aligned} d \vee_D \gamma(\alpha(d')) &\leqslant \gamma(\alpha(d \vee_D \gamma(\alpha(d')))) \\ &= \gamma(\alpha(d) \vee_A \alpha(\gamma(\alpha(d')))) \\ &= \gamma(\alpha(d) \vee_A \alpha(d')) \\ &= \gamma(\alpha(d \vee_D d')), \end{aligned}$$

which proves the inequality in the hypotheses of Theorem 5.2(ii). □

In the following, we will give a number of additional results for meet- and join-autodependencies. While reduced relative power with respect to the meet operation provides an interesting abstract domain refinement, the same construction relative to the join operation adds no further information to the input abstract domain. In fact, it turns out that $Dep_{\vee}(A)$ is always isomorphic to A.

Proposition 5.4. *If (α, D, A, γ) is a G.i. 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 written 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 for any $a, a' \in A$, $a \leqslant a'$ iff $\lambda x.a \vee x \sqsubseteq \lambda x.a' \vee x$. □

Thus, we will focus on meet-autodependencies only. Let us give a simple example of domain 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

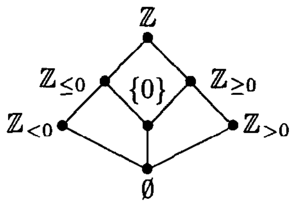


Fig. 3. The abstract domain $Dep_{\cap}(Sign)$.

general result. Thus, $Dep_{\cap}(Sign)$ is precisely the abstract domain in Fig. 3, 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 $\lambda x.\alpha(\{0\} \cap \gamma(x))$, denoted by $\{0\}$.

The next result is an instance of Proposition 3.11, and shows, in the case of meet-autodependencies, the explicit G.i. of A into $Dep_{\wedge}(A)$.

Corollary 5.6. *If D is a cHa and (α, D, A, γ) is a G.i. then $(\bar{\alpha}, Dep_{\wedge}(A), A, \bar{\gamma})$ is a G.i., where, for any $d \in D$ and $a \in A$, $\bar{\alpha}(\lambda x.\alpha(d \wedge \gamma(x))) \overset{\text{def}}{=} \alpha(d)$ and $\bar{\gamma}(a) \overset{\text{def}}{=} \lambda x.a \wedge x$. Moreover, (α, D, A, γ) is the composition of $(\alpha^{\natural}, D, Dep_{\wedge}(A), \gamma^{\natural})$ and $(\bar{\alpha}, Dep_{\wedge}(A), A, \bar{\gamma})$.*

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, $(\bar{\alpha}, Dep_{\wedge}(A), A, \bar{\gamma})$ is a G.i. Compositionality is a straight consequence of Proposition 3.11. □

By Corollary 5.3(i), it turns out that each meet-autodependency defined on a given abstract domain A, is a lower closure on A, i.e. $Dep_{\wedge}(A) \subseteq lco(A)$. In the following, we will characterize precisely the class of G.i.'s (α, D, 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 G.i. (α, D, A, γ) is *separated* if for any $a \in A$, there exists $d \in D$ such that (i) $\alpha(d) = a$; (ii) $\forall x \in A.\ (a \not\leq_A x) \Rightarrow (d \wedge \gamma(x) \leq_D \gamma(\bot_A))$.

For a separated G.i. (α, D, 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 mandatory. First, notice that the above definition implies that α is surjective, and therefore this justifies the requirement that (α, D, A, γ) is a G.i. 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 $a = \bot_A$, by choosing $d = \bot_D$. 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 a. 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 examples help 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 $a \in Sign$, a corresponding separating element $d_a \in \wp(\mathbb{Z})$ can be chosen as follows:

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

It is easy to check for them that 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 next example shows two nonseparated abstract domains.

Example 5.9. The abstract domains $Sign^{\neq 0}$ and $Dep_{\sqcap}(Sign)$ of Examples 4.6 and 5.5, respectively, are not separated in $\wp(\mathbb{Z})$. In fact, consider $\neq 0 \in Sign^{\neq 0}$ and $\mathbb{Z}_{\leqslant 0} \in Dep_{\sqcap}(Sign)$: 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 G.i.'s holds.

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

Proof. Assume that $x \in A \setminus \{\bot_A\}$, 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 $d_x \leqslant_D \gamma(\alpha(d_x)) = \gamma(x)$, the following equalities hold:

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

Thus, $x = \alpha(d_x) \leqslant_A \alpha(\gamma(\bot_A)) = \bot_A$, i.e., $x = \bot_A$, which is a contradiction. □

The following observation will be useful later on and contributes to further clarify the notion of separatedness.

Remark 5.11. The above result implies that if $x \in A \setminus \{\bot_A\}$ is join-reducible, i.e., there exists $X \subseteq A$ such that $x = \vee_A X$ and $x \notin X$, then $\vee_D \gamma(X) <_D \gamma(\vee_A X)$. In other terms, if a G.i. (α, D, A, γ) is separated and A contains join-reducible elements different from the bottom, then (α, D, 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 $Sign^{\neq 0}$ of Example 4.6, abstracting the cHa $\wp(\mathbb{Z})$, is disjunctive and its element $\neq 0$ is join-reducible, we get that $Sign^{\neq 0}$ 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 $\langle \wp(X), \subseteq \rangle$, for some set X, be the concrete domain and $(\alpha, \wp(X), A, \gamma)$ be a G.i. Then, $(\alpha, \wp(X), A, \gamma)$ 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 $x \in A \setminus \{\bot_A\}$. Let us distinguish two cases.

(1) x is join-irreducible. Define $a_x \overset{\text{def}}{=} \vee_A \{z \in A \mid z <_A x\}$. By join-irreducibility of x, it follows that $a_x <_A x$. Now, define $d_x \overset{\text{def}}{=} \gamma(x) \setminus \gamma(a_x)$. Clearly, $\alpha(d_x) \leqslant_A \alpha(\gamma(x)) = x$. If we assume that $\alpha(d_x) <_A x$, we would get $\alpha(d_x) \leqslant_A a_x$, and therefore $d_x \subseteq \gamma(a_x)$, 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 $x \leqslant a_x$, i.e. a contradiction. Thus, $\alpha(d_x) = x$. Consider now $y \in A$ such that $x \nleqslant_A y$. 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 $x \wedge_A y \leqslant_A a_x$, and then $\gamma(x \wedge_A y) \subseteq \gamma(a_x)$. 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 $d_x \overset{\text{def}}{=} \gamma(x) \setminus (\bigcup\{\gamma(z) \mid z <_A x\})$. Note that, by hypothesis, $d_x \neq \emptyset$. Trivially, $\alpha(d_x) \leqslant_A x$. 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 $x \nleqslant_A y$. 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, the following important characterization of separated abstract domains holds.

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

Proof. By duality from what has been recalled in Section 2.2, it turns out that $lco(A)$ is atomic, that is each lower closure different from the least lower closure $\lambda x.\bot_A$ is the *lub* of the atomic lower closures ϕ_a which precede it, where, for $a \in A\setminus\{\bot_A\}$, ϕ_a is defined as follows:

$$\phi_a(x) \overset{\text{def}}{=} \begin{cases} a & \text{if } x \geqslant_A a, \\ \bot_A & \text{otherwise.} \end{cases}$$

($\Rightarrow$) 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 $a \leqslant_A x$ then $\alpha(d_a \wedge \gamma(x)) = \alpha(d_a) = a$, because $d_a \leqslant_D \gamma(a) \leqslant_D \gamma(x)$. The case $a \nleqslant_A x$ follows by the hypothesis of separatedness: $\alpha(d_a \wedge \gamma(x)) \leqslant_A \alpha(\gamma(\bot_A)) = \bot_A$.

($\Leftarrow$) Assume, by contradiction, that (α, D, 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 $x_d \ngeqslant_A \hat{a}$ such that $d \wedge \gamma(x_d) \nleqslant_D \gamma(\bot_A)$. 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 $x_{d_{\hat{a}}} \ngeqslant_A \hat{a}$ such that $d_{\hat{a}} \wedge \gamma(x_{d_{\hat{a}}}) \nleqslant_D \gamma(\bot_A)$. But this is a contradiction, because for any $x \ngeqslant_A \hat{a}$, we have $d_{\hat{a}} \wedge \gamma(x) \leqslant_D \gamma(\alpha(d_{\hat{a}} \wedge \gamma(x))) = \gamma(\phi_{\hat{a}}(x)) = \gamma(\bot_A)$. □

Thus, whenever the concrete domain is a cHa, 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. Such an isomorphism maps any lower closure to its set of fixpoints. Thus, for a separated abstract domain A, by exploiting this isomorphism, the G.i. $(\alpha^\natural, D, Dep_\wedge(A), \gamma^\natural)$ becomes isomorphic to the G.i. $(\tilde{\alpha}, D, \mathcal{M}(A), \tilde{\gamma})$, where the left-adjoint $\tilde{\alpha}$ is defined as $\tilde{\alpha}(d) \overset{\text{def}}{=} \{x \in A \mid \alpha(d \wedge \gamma(x)) = x\}$. Analogously, the G.i. $(\bar{\alpha}, Dep_\wedge(A), A, \bar{\gamma})$ of A in $Dep_\wedge(A)$, given by Corollary 5.6, becomes isomorphic to the G.i. $(\widehat{\alpha}, \mathcal{M}(A), A, \widehat{\gamma})$, where, for any $M \in \mathcal{M}(A)$ and $a \in A$, $\widehat{\alpha}(M) \overset{\text{def}}{=} \vee_A M$ and $\widehat{\gamma}(a) \overset{\text{def}}{=} \downarrow a$. 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 reduced relative power $Dep_\wedge(A)$ corresponds to a powerset-like completion of A, that we will call *dual-Moore-set completion* of A. Of course, such dual-Moore-set completion is in general incomparable 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. Its dual-Moore-set completion $\mathcal{M}(Sign)$ has been characterized in Example 5.5

and it is shown in Fig. 3. By contrast, the disjunctive completion of *Sign* is the domain $Sign^{\neq 0}$ of Example 4.6. Hence, these two abstractions of $\wp(\mathbb{Z})$ are incomparable.

Whenever A is separated in D, the lattice of meet-autodependencies $Dep_{\wedge}(A)$ enjoys all the well-known properties of closure operators (see e.g. [50, 64] for a few of them). In particular, the following lattice-theoretic properties of $Dep_{\wedge}(A)$ are inherited from $lco(A)$.

(i) $Dep_{\wedge}(A)$ is atomic[6] (cf. [64]).

(ii) $Dep_{\wedge}(A)$ is distributive iff it is complemented iff it is Boolean iff A is a complete well-ordered chain (cf. [51]).

(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 $Dep_{\wedge}(A)$ is dual-pseudocomplemented (cf. [35]), namely for any $f \in Dep_{\wedge}(A)$, there exists a unique $f^{\star} \in Dep_{\wedge}(A)$ such that $f \sqcup f^{\star} = id$, and, for any $g \in Dep_{\wedge}(A)$, if $f \sqcup g = id$ then $f^{\star} \sqsubseteq g$.

Let us close this section by an 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, $Dep_{\cap}(Sign)$ coincides with $lco(Sign)$.
Consider instead the abstract domain *Parity* in Example 3.9. Since *Parity* is disjunctive and its top is join-reducible, by Remark 5.11, it turns out that $(\alpha_p, \wp(\mathbb{Z}), Parity, \gamma_p)$ is not separated. Thus, by Corollary 5.3(i) and Theorem 5.13, $Dep_{\cap}(Parity) \subset lco(Parity)$. In particular, the lower closure corresponding to the dual-Moore-set $\{\top_p, \bot_p\}$ is not included in $Dep_{\cap}(Parity)$:

Fact 5.16. *Let* $\rho \in lco(Parity)$ *such that* $\rho(Parity) = \{\top_p, \bot_p\}$. $\rho \notin Dep_{\cap}(Parity)$.

Proof. $\rho \in lco(Parity)$ is defined by

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

Assume by contradiction that there exists some $S \subseteq \mathbb{Z}$ such that for any $x \in Parity$, $\alpha_p(S \cap \gamma_p(x)) = \rho(x)$. 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 $\mathbb{Z} =$

[6] As early observed by Nielson [54], 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.

$\gamma_p(ev) \cup \gamma_p(od)$, 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, $Dep_{\sqcap}(Parity) \subset lco(Parity)$. On the other hand, it would not be too difficult to show directly that $Dep_{\sqcap}(Parity) \cong Parity$. □

6. An application to logic program analysis

We present in this section an example of application of the reduced relative power to domains used for abstract interpretation-based program analysis. We show that the well-known abstract domain *Def*, introduced by Dart [25] in the context of groundness analysis for deductive databases, and used by Marriott and Søndergaard [49] for ground-dependency analysis of logic programs, can be systematically derived by considering the meet-autodependencies defined on the more abstract (and much simpler) domain *Gr* representing plain groundness information, introduced by Jones and Søndergaard [45].

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 $\langle \wp^{\downarrow}(Sub), \subseteq \rangle$ (where $\preccurlyeq$ is the pre-order of instantiation over *Sub*), which is a cHa where *lub* and *glb* are, respectively, union and intersection. This concrete domain gives rise to the so-called Clark's interpretations [12] used in the *c*-semantics for logic programs, as described in [28]. The concrete operation of composition on order-ideals of substitutions is given by intersection, namely $\langle \wp^{\downarrow}(Sub), \subseteq, \cap \rangle$ plays the role of the concrete quantale.

The simplest abstract domain representing plain groundness information, relative to a (nonempty) finite set of variables of interest $V \subseteq Var$, is given by $Gr \overset{\text{def}}{=} \langle \wp(V), \supseteq \rangle$. 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 $W \in \wp(V)$,

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

It is worth noting that γ_g actually is closed by instantiation (i.e., if $\theta \in \gamma_g(W)$ and $\theta' \preccurlyeq \theta$ then $\theta' \in \gamma_g(W)$), and therefore, $\gamma_g(W)$ correctly is an order-ideal of substitutions. It is simple to show that $(\alpha_g, \wp^{\downarrow}(Sub), \wp(V), \gamma_g)$ is a G.i. (cf. [45]). For $V = \{x, y\}$, *Gr* is the simple abstract domain depicted in Fig. 4.

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

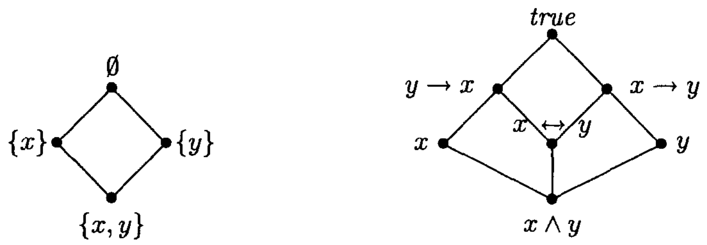


Fig. 4. The abstract domains *Gr*, on the left, and *Def*, on the right, for $V=\{x,y\}$.

interest. Since 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 formulae 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 implicational 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, ordered with respect to $\models$, of all the definite (positive) formulae on V. Analogously, *Con* is the finite lattice 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, respectively, by logical conjunction and disjunction, denoted by $\wedge$ and $\vee$. *Def* is an atomic lattice – the atoms are those formulae f such that $mod(f) = \{V, W\}$, for any $W \subset V$ – where the *lub* is, in general, different from logical disjunction, and can be defined as follows: $f_1 \vee_{Def} f_2 \overset{\text{def}}{=} \wedge\{f \in Def \mid f_1 \vee f_2 \models f\}$. 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 G.i. $(\lambda f.\{x \in V \mid f \models x\}, Def, Gr, \lambda W. \wedge W)$. The abstraction and concretization maps between $\wp(Sub)$ and *Def* are well known: For any $f \in Def$, its concretization is defined as follows:

$$\gamma(f) \overset{\text{def}}{=} \{\sigma \in Sub \mid \forall \sigma' \preccurlyeq \sigma.\ gr(\sigma') \in mod(f)\}.$$

It is possible to show that γ is co-additive and injective, and therefore it gives rise to a G.i. of *Def* into $\wp(Sub)$. This same γ is the concretization map for *Con* and *Pos* as well. In particular, it would not be hard 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

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

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

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

Notice that $\alpha(\emptyset) = \vee_{Def}\, \emptyset = \wedge V$. These two mappings α and γ form a G.i. 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 its instance $\sigma' \preccurlyeq \sigma$ 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, $(\alpha, \wp^{\downarrow}(Sub), Def, \gamma)$ is a G.i. as well.

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

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) $\forall y \in V \backslash W.\ \sigma(x) = z$, 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 $T \in Gr$ such that $W \not\supseteq T$. We have to prove that $\downarrow \sigma \cap \gamma_g(T) \subseteq \gamma_g(V) = \{\theta \in Sub \mid gr(\theta) \supseteq V\}$. Consider any $\delta \in \ \downarrow \sigma \cap \gamma_g(T)$. Then, since $\delta \preccurlyeq \sigma$, for any $x \in W$, $var(\delta(x)) = \emptyset$. Since $W \not\supseteq T$, there exists $x \in T \backslash W$. Then, $var(\delta(x)) = \emptyset$, and therefore, since $\delta \preccurlyeq \sigma$, for any $y \in V \backslash W$, by definition of σ, $var(\delta(y)) = \emptyset$. 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, $(\alpha_g^{\natural}, \wp^{\downarrow}(Sub), Dep_{\cap}(Gr), \gamma_g^{\natural}) \cong (\tilde{\alpha}, \wp^{\downarrow}(Sub), \mathcal{M}(Gr), \tilde{\gamma})$.

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

Proof. Since we deal with G.i.'s, 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

$W = \bigcap\{gr(\theta) \mid \theta \in \gamma(f) \cap \gamma_g(W)\}$. 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 $\theta' \preccurlyeq \theta$. 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 check 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 get that $(\alpha, \wp^{\downarrow}(Sub), Def, \gamma) \cong (\alpha^{\natural}, \wp^{\downarrow}(Sub), Dep_{\cap}(Gr), \gamma^{\natural})$. We have then fully reconstructed the abstract domain *Def* by the systematic operation of reduced relative power. The next result proves that the G.i. of *Gr* into its dual-Moore-set completion $(\widehat{\alpha}, \mathcal{M}(Gr), Gr, \widehat{\gamma})$ is isomorphic to the G.i. of *Gr* into *Def*.

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

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 check 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 $V = \{x, y\}$, *Def* is depicted in Fig. 4.

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: $x \wedge y \rightleftarrows \{\{x, y\}\}$, $x \rightleftarrows \{\{x\}, \{x, y\}\}$, $y \rightleftarrows \{\{y\}, \{x, y\}\}$, $x \leftrightarrow y \rightleftarrows \{\emptyset, \{x, y\}\}$, $x \rightarrow y \rightleftarrows \{\emptyset, \{y\}, \{x, y\}\}$, $y \rightarrow x \rightleftarrows \{\emptyset, \{x\}, \{x, y\}\}$, and $true \rightleftarrows \{\emptyset, \{x\}, \{y\}, \{x, y\}\}$.

It is worth noting that the operation of logical conjunction, i.e., the standard abstract unification used for ground-dependency analysis by *Def* (cf. [49]), can be implemented in *Def* by set-intersection on the corresponding dual-Moore-set completion of *Gr*. Furthermore, *Def* shares the lattice-theoretic properties of lattices 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, $x \rightarrow y \in Def$ 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 [4].

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

Lemma 6.4. $(\alpha, \wp^{\downarrow}(Sub), Def, \gamma)$ *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 $x \to y$. Let us distinguish two cases.

(i) There exists $\theta \in \Theta$ such that $\alpha_{\{\cdot\}}(\theta) = x \to y$. Then, $var(\theta(x)) \supset var(\theta(y)) \neq \emptyset$. Thus, there exists θ' instance of θ such that $y \in gr(\theta')$ and $x \notin gr(\theta')$, and therefore $\theta' \notin \gamma(\bot_{Def})$. Since $y \models x \to y$ and $\theta' \in \Theta \cap \gamma(y)$, 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, $y \in gr(\theta_2)$, 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 $Dep_{\sqcap}(Def) \subset lco(Def)$. On the other hand, Scozzari [60] characterized precisely the meet-autodependencies of Def in $\wp^{\downarrow}(Sub)$, by showing that $Dep_{\sqcap}(Def) \cong Pos$ and that Pos is closed under meet-autodependencies, i.e., $Dep_{\sqcap}(Pos) \cong Pos$.

7. An 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. Let us point out that, although the results of this section are specialized to the case of definite logic programs, they may have a wider validity in the context of systematic design of inductive definitions and transitions systems (e.g. see [8] where positive and negative inductive definitions are interpreted as logic programs).

In the following, a logic program semantics is a pair $\mathscr{X} = \langle C, T \rangle$, where C is a complete lattice (the semantic domain) and $T : Program \mapsto (C \overset{\mathrm{m}}{\longmapsto} C)$. For $P \in Program$, the semantic transformer $T(P)$ is denoted by the more customary T_P. With a slight abuse of notation, a semantics $\langle C, T \rangle$ will be denoted also by $\langle C, T_P \rangle$. The least fixpoint semantics of a program P is then given by $[\![P]\!]^{\mathscr{X}} \overset{\text{def}}{=} lfp(T_P) \in C$. Of course, when T_P is continuous, it turns out that $[\![P]\!]^{\mathscr{X}} = \bigvee_{n<\omega} T_P^n(\bot_C)$. If $\circ$ is a syntactic operator for composing programs, a semantics $\mathscr{X}$ 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. there exists a function $@ : C \times C \mapsto C$ such that, for any $P_1, P_2 \in Program$, $[\![P_1 \circ P_2]\!]^{\mathscr{X}} = [\![P_1]\!]^{\mathscr{X}} @ [\![P_2]\!]^{\mathscr{X}}$. Since logic programs are sets of clauses, the standard and most obvious way of syntactically composing programs is by set union, i.e. $\circ = \cup$.

7.1. The concrete semantics

As Cousot and Cousot showed in [19, 23], program semantics at different levels of abstraction can be derived by abstract interpretation of a more concrete reference semantics. A natural 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 [34], 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: $SLD \overset{\text{def}}{=} \langle State, \{\xrightarrow{c} \mid c \in Clause\}\rangle$, where $State \overset{\text{def}}{=} Atom^* \times Sub$, and transitions $\xrightarrow{c} \subseteq State \times State$, labeled with clauses, are defined by

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

We denote by $\mathbf{T}$ the set of all possible finite traces, i.e.

$$\mathbf{T} \overset{\text{def}}{=} \{s_0 \xrightarrow{c_1} \cdots \xrightarrow{c_n} s_n \mid n \geqslant 1,\ s_0, \ldots, s_n \in State,\ c_1, \ldots, c_n \in Clause\ \forall i \in [1,n].\ s_{i-1} \xrightarrow{c_i} s_i\}.$$

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 $\bar{c} = c_1, \ldots, c_n \in Clause^*$ (hence $n \geqslant 1$), we write $s_0 \xrightarrow{\bar{c}} s_n$ iff there exist $s_1, \ldots, s_{n-1}$ 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 $\pi = s \xrightarrow{\bar{c}} \langle \Lambda, \sigma\rangle$. The set $\mathscr{T}_P^{sld} \subseteq \mathbf{T}$ of program execution traces for a program P is inductively defined by the following rules:

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

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 $\mathscr{T} \subseteq \mathbf{T}$ is AND-compositional if any execution trace starting from any (possibly nonatomic) goal G can be reconstructed from the execution traces in $\mathscr{T}$ starting from the atomic subgoals of G only. It is worth noting that all the well-known fixpoint semantics in [27, 28, 31] are based on denotations which are AND-compositional in an analogous sense. The set $\mathbf{aT}$ of *plain* execution traces for atomic goals is defined as follows:

$$\mathbf{aT} \overset{\text{def}}{=} \{(h \xrightarrow{c_1} \cdots \xrightarrow{c_n} \bar{b})\theta \mid h \in Atom,\ \langle h, \sigma\rangle \xrightarrow{c_1} \cdots \xrightarrow{c_n} \langle \bar{b}, \theta\rangle \in \mathbf{T}\}.$$

Thus, a plain execution trace in $\mathbf{aT}$ is just a different syntactic representation of a corresponding trace in $\mathbf{T}$ starting from an atomic goal. As above, for a (nonempty) sequence of clauses $\bar{c} = c_1, \ldots, c_n \in Clause^*$, for any $h \in Atom$ and $\bar{b} \in Atom^*$, we write $h \xrightarrow{\bar{c}} \bar{b}$ iff there exist $\bar{b}_1, \ldots, \bar{b}_{n-1} \in Atom^*$ 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 $\mathscr{T}_P^{and} \subseteq \mathbf{aT}$ of *AND-compositional plain execution traces* of P for atomic goals is given in [34] as in Fig. 5. Rules (1) and (2) define 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 $c = h \leftarrow b_1, \ldots, b_n$ of P, the second rule specifies how plain execution

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

Fig. 5. Inductive definition of $\mathcal{T}_P^{and}$.

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 $b_1,\dots,b_{k-1}$ of the *body* of c, with the state (goal) produced from a trace for the kth atom b_k of the *body* of c.

Following a standard technique, an operator $\varphi : Program \mapsto (\wp(\mathbf{aT}) \xrightarrow{c} \wp(\mathbf{aT}))$ can be systematically derived from the inductive definition of $\mathcal{T}_P^{and}$ in Fig. 5 as follows (cf. [2]): For any $P \in Program$ and $X \in \wp(\mathbf{aT})$,

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

It turns out that, for any program P, φ_P is continuous on $\langle \wp(\mathbf{aT}), \subseteq\rangle$, and $\mathcal{T}_P^{and} = lfp(\varphi_P)$ (cf. [2]). The following theorem justifies the interest in plain execution traces for atomic goals.

Theorem 7.1 (Giacobazzi [34, Theorem 5.2]). *Let P be a program, $G = b_1,\dots,b_n$ and B be goals and $\sigma, \theta \in Sub$. Then, $\langle G, \sigma\rangle \xrightarrow{\bar{c}} \langle B, \theta\rangle \in \mathcal{T}_P^{sld}$ iff there exist $\langle\langle h_i \xrightarrow{\bar{c}_i} \Lambda\rangle_{i=1}^{k-1}, h_k \xrightarrow{\bar{c}_k} \bar{b}\rangle \ll_G \mathcal{T}_P^{and}$, with $1 \leqslant k \leqslant n$, such that $\delta = mgu(\langle b_1,\dots,b_k\rangle\sigma, \langle h_1,\dots,h_k\rangle)$, $\bar{c} = \bar{c}_1 :: \cdots :: \bar{c}_k$, $B\theta \sim (\bar{b} :: \langle b_{k+1},\dots,b_n\rangle)\sigma\delta$, and $G\theta \sim G\sigma\delta$.*

In the following, $\langle\langle \wp(\mathbf{aT}), \subseteq\rangle, \varphi_P\rangle$ will play the role of 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

Let us consider the following operator of trace-unfolding.

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

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

Thus, $X \nabla Y$ denotes the result of unfolding any trace in X by using traces 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., $\forall Y \in \wp(\mathbf{aT}).\ \emptyset \nabla Y = \emptyset$).*

Proof. We omit the proof of associativity for ∇, because it follows from a similar result proved in [26]. Moreover, by definition, it is immediate that ∇ is monotone, left-additive, right-continuous, and $\emptyset$ is a left-annihilator for ∇. $\square$

Thus, $\langle \wp(\mathbf{aT}), \subseteq, \nabla \rangle$ turns out to be a semi-quantale and, by Theorem 3.6, if $(\alpha_1, \wp(\mathbf{aT}), D_1, \gamma_1)$ and $(\alpha_2, \wp(\mathbf{aT}), D_2, \gamma_2)$ are G.c.'s, then $(\alpha, \wp(\mathbf{aT}), D_1 \overset{\nabla}{\longmapsto} D_2, \gamma)$ is a G.i., 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 G.c. $(\alpha, \wp(\mathbf{aT}), A, \gamma)$ actually induces an abstract semantics $\langle A, \alpha \circ \varphi_P \circ \gamma \rangle$, where $\alpha \circ \varphi_P \circ \gamma : A \overset{\text{m}}{\longmapsto} A$ is the best correct approximation of φ_P in A, which is correct[7] 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 [34], 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 s-semantics of Falaschi et al. [28], characterizing the observable notion of computed answer substitution.

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

$$\alpha_{\mathcal{S}}(I) \overset{\text{def}}{=} \{ h \in Atom \mid h \xrightarrow{\bar{c}} \Lambda \in I \}.$$

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

Observe that $\alpha_{\mathscr{S}}$ is additive and onto, and therefore it defines a G.i. $(\alpha_{\mathscr{S}}, \wp(\mathbf{aT}), \wp(Atom), \gamma_{\mathscr{S}})$. It is also easy to check that the corresponding concretization map $\gamma_{\mathscr{S}}$ is additive. As shown in [34, Example 5.4], this G.i. induces an abstract semantics $\mathscr{S} \stackrel{\text{def}}{=} \langle \wp(Atom), \alpha_{\mathscr{S}} \circ \varphi_P \circ \gamma_{\mathscr{S}} \rangle$ 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 $T_P^s : \wp(Atom) \xmapsto{c} \wp(Atom)$ defined as follows [28]:

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

It turns out that the best correct approximation $\alpha_{\mathscr{S}} \circ \varphi_P \circ \gamma_{\mathscr{S}}$ of φ_P in $\wp(Atom)$ and T_P^s coincide (cf. [34, Example 5.4]). For the sake of completeness, we include this proof. Actually, we demonstrate a stronger relation between T_P^s and φ_P, namely $\alpha_{\mathscr{S}} \circ \varphi_P = T_P^s \circ \alpha_{\mathscr{S}}$, known as completeness [21, 38].

Proposition 7.4. *For all $P \in Program$, $\alpha_{\mathscr{S}} \circ \varphi_P = T_P^s \circ \alpha_{\mathscr{S}}$.*

Proof. Let $I \in \wp(\mathbf{aT})$. By definition, $h \in \alpha_{\mathscr{S}}(\varphi_P(I))$ iff either (1) $h \leftarrow \in P$, or (2) there exist $c = h' \leftarrow b_1, \ldots, b_n \in P$ and $\langle a_i \xrightarrow{\bar{c}_i} \Lambda \rangle_{i=1}^n \ll_c I$, such that $\theta = mgu(\langle b_1, \ldots, b_n \rangle, \langle a_1, \ldots, a_n \rangle)$ and $h \equiv h'\theta$. In both cases, it is clear that $h \in T_P^s(\alpha_{\mathscr{S}}(I))$. On the other hand, $h\theta \in T_P^s(\alpha_{\mathscr{S}}(I))$ iff either (3) $h\theta \leftarrow \in P$, or (4) there exist $c = h \leftarrow b_1, \ldots, b_n \in P$ and $\langle a_1, \ldots, a_n \rangle \ll_c \alpha_{\mathscr{S}}(I)$ such that $\theta = mgu(\langle b_1, \ldots, b_n \rangle, \langle a_1, \ldots, a_n \rangle)$. If (3) holds, then $h\theta \in \alpha_{\mathscr{S}}(\varphi_P(I))$. If (4) holds, then there exist $a_1 \xrightarrow{\bar{c}_1} \Lambda, \ldots, a_n \xrightarrow{\bar{c}_n} \Lambda \in I$, and therefore $(h \xrightarrow{c} \langle b_1, \ldots, b_n \rangle \xrightarrow{\bar{c}_1} \cdots \xrightarrow{\bar{c}_n} \Lambda)\theta \in \varphi_P(I)$. Hence, $h\theta \in \alpha_{\mathscr{S}}(\varphi_P(I))$, which concludes the proof. □

In a similar way, it would not be too hard to show (see [34, Proposition 5.3]) that the *Herbrand abstraction* $\alpha_{\mathscr{H}} \stackrel{\text{def}}{=} \lambda I.ground(\alpha_{\mathscr{S}}(I)) : \langle \wp(\mathbf{aT}), \subseteq \rangle \mapsto \langle \wp(ground(Atom)), \subseteq \rangle$ gives rise to an abstract semantics $\mathscr{H}$ equivalent to the well-known van Emden and Kowalski ground Herbrand semantics [27].

7.4. *Autodependencies of s-semantics*

In the following, we denote simply by $Dep_{\nabla}(\mathscr{S})$ the domain of autodependencies induced by the G.i. $(\alpha_{\mathscr{S}}, \wp(\mathbf{aT}), \wp(Atom), \gamma_{\mathscr{S}})$ w.r.t. the concrete semi-quantale $\langle \wp(\mathbf{aT}), \subseteq, \nabla \rangle$, while the corresponding G.i. is denoted by $(\alpha, \wp(\mathbf{aT}), Dep_{\nabla}(\mathscr{S}), \gamma)$. Therefore, for a trace-interpretation $I \in \wp(\mathbf{aT})$, $\alpha(I) = \lambda x.\alpha_{\mathscr{S}}(I \nabla \gamma_{\mathscr{S}}(x))$ is a generic autodependency in $Dep_{\nabla}(\mathscr{S})$. As observed in Section 5, $\lambda x.\alpha_{\mathscr{S}}(I \nabla \gamma_{\mathscr{S}}(x))$ is the best correct approximation of the concrete operator $\lambda x.I \nabla x : \wp(\mathbf{aT}) \mapsto \wp(\mathbf{aT})$. As observed in [26], it turns out that any such operator $\lambda x.I \nabla x$ 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: Thus, $\alpha_* : \langle \wp(\mathbf{aT}), \subseteq\rangle \mapsto \langle \wp(\mathit{Clause}), \subseteq\rangle$ is defined as

$$\alpha_*(I) \overset{\text{def}}{=} \{h \leftarrow \bar{b} \in \mathit{Clause} \mid h \xrightarrow{\bar{c}} \bar{b} \in I\}.$$

It is straightforward to check that α_* is additive and onto, and therefore it gives rise to a G.i. $(\alpha_*, \wp(\mathbf{aT}), \wp(\mathit{Clause}), \gamma_*)$. As demonstrated by Comini et al. [15] in a similar context, one could show that this latter G.i. induces an abstract semantics which is equivalent to the compositional semantics of resultants of Gabbrielli et al. [32].

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

Proof. First, notice that for any $P \subseteq \mathit{Clause}$ and $X \subseteq \mathit{Atom}$, $T^s_P(X) = \bigcup_{c \in P} T^s_{\{c\}}(X)$. Thus, since ∇ is left-additive, and $\alpha_{\mathscr{S}}$ and α_* are additive, it is sufficient to prove that for any $\pi \in \mathbf{aT}$ and $X \subseteq \mathit{Atom}$, $\alpha_{\mathscr{S}}(\{\pi\} \nabla \gamma_{\mathscr{S}}(X)) = T^s_{\alpha_*(\{\pi\})}(X)$. Let $\pi = h \xrightarrow{\bar{c}} \langle b_1, \dots, b_n\rangle$.

($\subseteq$) Assume that $h' \in \alpha_{\mathscr{S}}(\{\pi\} \nabla \gamma_{\mathscr{S}}(X))$. Therefore, there exist $\langle a_i \xrightarrow{\bar{c}_i} \Lambda\rangle_{i=1}^n \ll_\pi \gamma_{\mathscr{S}}(X)$ such that $\theta = mgu(\langle b_1, \dots, b_n\rangle, \langle a_1, \dots, a_n\rangle)$ and $h' = h\theta$. Hence, $\langle a_1, \dots, a_n\rangle \ll_\pi X$. Thus, by definition, $h' \in T^s_{\alpha_*(\{\pi\})}(X)$.

($\supseteq$) Assume that $h' \in T^s_{\alpha_*(\{\pi\})}(X)$. This means that there exist $\langle a_1, \dots, a_n\rangle \ll_\pi X$ such that $\theta = mgu(\langle b_1, \dots, b_n\rangle, \langle a_1, \dots, a_n\rangle)$ and $h' = h\theta$. Consequently, there exist $\langle a_i \xrightarrow{\bar{c}_i} \Lambda\rangle_{i=1}^n \ll_\pi \gamma_{\mathscr{S}}(X)$. Thus, $(h \xrightarrow{\bar{c}} \langle b_1, \dots, b_n\rangle \xrightarrow{\bar{c}_1} \cdots \xrightarrow{\bar{c}_n} \Lambda)\theta \in \{\pi\} \nabla \gamma_{\mathscr{S}}(X)$, and hence, $h\theta = h' \in \alpha_{\mathscr{S}}(\{\pi\} \nabla \gamma_{\mathscr{S}}(X))$. $\square$

As a consequence, we get that $\mathit{Dep}_\nabla(\mathscr{S}) = \{T^s_P : \wp(\mathit{Atom}) \mapsto \wp(\mathit{Atom}) \mid P \in \mathit{Program}\}$, and for any program P, $T^s_P = \lambda X.\alpha_{\mathscr{S}}(\gamma_*(P) \nabla \gamma_{\mathscr{S}}(X)) = \alpha(\gamma_*(P))$. Thus, intuitively, if $I \in \wp(\mathbf{aT})$ then the intermediate states of any trace $\pi \in I$ are not relevant in $\lambda X.\alpha_{\mathscr{S}}(I \nabla \gamma_{\mathscr{S}}(X))$, as formalized by the next lemma.

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

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

Let us now introduce the following abstraction function from programs to autodependencies $\alpha_{cl} : \langle \wp(\mathit{Clause}), \subseteq\rangle \mapsto \mathit{Dep}_\nabla(\mathscr{S})$, such that, for any set of clauses (viz. program) $P \subseteq \mathit{Clause}$,

$$\alpha_{cl}(P) \overset{\text{def}}{=} \lambda X.\alpha_{\mathscr{S}}(\gamma_*(P) \nabla \gamma_{\mathscr{S}}(X)).$$

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

Proof. Firstly, note that γ_* is additive. Hence, the additivity of α_{cl} follows by left-additivity of ∇ (cf. Proposition 7.3) and additivity of $\alpha_{\mathscr{S}}$. Consider now $\lambda X.\alpha_{\mathscr{S}}(I\nabla\gamma_{\mathscr{S}}(X)) \in Dep_{\nabla}(\mathscr{S})$, for some $I \in \wp(\mathbf{aT})$. Then, by considering $\alpha_*(I) \subseteq Clause$, by Lemma 7.6 we get $\alpha_{cl}(\alpha_*(I)) = \lambda X.\alpha_{\mathscr{S}}(I\nabla\gamma_{\mathscr{S}}(X))$, therefore proving that α_{cl} is onto. □

Thus, the mapping α_{cl} induces a G.i. $(\alpha_{cl}, \wp(Clause), Dep_{\nabla}(\mathscr{S}), \gamma_{cl})$. Summing up, we have split the G.i. relating $Dep_{\nabla}(\mathscr{S})$ to $\wp(\mathbf{aT})$ into the composition of the G.i.'s $(\alpha_*, \wp(\mathbf{aT}), \wp(Clause), \gamma_*)$ and $(\alpha_{cl}, \wp(Clause), Dep_{\nabla}(\mathscr{S}), \gamma_{cl})$.

Proposition 7.8. *$(\alpha, \wp(\mathbf{aT}), Dep_{\nabla}(\mathscr{S}), \gamma)$ is the composition of $(\alpha_*, \wp(\mathbf{aT}), \wp(Clause), \gamma_*)$ and $(\alpha_{cl}, \wp(Clause), Dep_{\nabla}(\mathscr{S}), \gamma_{cl})$.*

Proof. By Lemma 7.6, for all $I \in \wp(\mathbf{aT})$, we have that $\alpha_{cl}(\alpha_*(I)) = \lambda X.\alpha_{\mathscr{S}}(\gamma_*(\alpha_*(I))\nabla\gamma_{\mathscr{S}}(X)) = \lambda X.\alpha_{\mathscr{S}}(I\nabla\gamma_{\mathscr{S}}(X)) = \alpha(I)$, and this suffices to conclude the proof. □

It is important to remark that since both γ_* and γ_{cl} are additive functions, by Proposition 7.8, we have that γ is additive as well, i.e., $Dep_{\nabla}(\mathscr{S})$ is a disjunctive abstraction of $\wp(\mathbf{aT})$.

The next result characterizes the equivalence between programs induced by the above abstraction map α_{cl}: It turns out that two programs are equivalent for α_{cl} iff their s-semantics immediate consequences operators coincide.

Theorem 7.9. *For any $P, Q \subseteq Clause$, $\alpha_{cl}(P) = \alpha_{cl}(Q)$ iff $T_P^s = T_Q^s$.*

Proof. $(\Rightarrow)$ If $\alpha_{cl}(P) = \alpha_{cl}(Q)$ then $\lambda X.\alpha_{\mathscr{S}}(\gamma_*(P)\nabla\gamma_{\mathscr{S}}(X)) = \lambda X.\alpha_{\mathscr{S}}(\gamma_*(Q)\nabla\gamma_{\mathscr{S}}(X))$. Thus, by Lemma 7.5, and since $\alpha_* \circ \gamma_* = id$, we get $T_P^s = T_Q^s$.

$(\Leftarrow)$ Since α_* is surjective, there exist $I, J \in \wp(\mathbf{aT})$ such that $\alpha_*(I) = P$ and $\alpha_*(J) = Q$. Thus, by Lemmata 7.5 and 7.6, $\alpha_{cl}(\alpha_*(I)) = \alpha_{cl}(\alpha_*(J))$, and therefore, $\alpha_{cl}(P) = \alpha_{cl}(Q)$. □

Let us denote by $\mathscr{S} \mapsto \mathscr{S} \overset{\text{def}}{=} \langle Dep_{\nabla}(\mathscr{S}), T_P^{\mathscr{S}\mapsto\mathscr{S}} \rangle$ the semantics induced by $(\alpha, \wp(\mathbf{aT}), Dep_{\nabla}(\mathscr{S}), \gamma)$, where $T_P^{\mathscr{S}\mapsto\mathscr{S}} \overset{\text{def}}{=} \alpha \circ \varphi_P \circ \gamma$. Since γ is additive, it turns out that the semantic transformer $T_P^{\mathscr{S}\mapsto\mathscr{S}}$ is continuous on $Dep_{\nabla}(\mathscr{S})$. Thus, the abstract semantics induced by $Dep_{\nabla}(\mathscr{S})$ is $[\![P]\!]^{\mathscr{S}\mapsto\mathscr{S}} \overset{\text{def}}{=} lfp(T_P^{\mathscr{S}\mapsto\mathscr{S}}) = \bigsqcup_{n<\omega}(T_P^{\mathscr{S}\mapsto\mathscr{S}})^n(\bot_{Dep_{\nabla}(\mathscr{S})})$ (recall that the *lub* $\sqcup$ of $Dep_{\nabla}(\mathscr{S})$ is defined pointwise).

In some of the following proofs, for the sake of simplicity, 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 equivalent notions, γ_* coincides with $\gamma_{\mathscr{S}}$ when restricted to unit clauses, and α_* coincides with $\alpha_{\mathscr{S}}$ when restricted to sets of successful traces. In order to characterize the semantics $[\![\cdot]\!]^{\mathscr{S}\mapsto\mathscr{S}}$, we will need the following lemma.

Lemma 7.10. *For any $I \in \wp(\mathbf{aT})$, $T_P^{\mathscr{S}\mapsto\mathscr{S}}(\alpha(I)) = \alpha(\varphi_P(I))$.*

Proof. We prove that for any $I \subseteq \mathbf{aT}$ and $X \subseteq Atom$, $\alpha(\varphi_P(\gamma(\alpha(I))))(X) = \alpha(\varphi_P(I))(X)$.

($\supseteq$) 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 $T^s_{\alpha_*(\varphi_P(\gamma(\alpha(I))))}(X) \subseteq T^s_{\alpha_*(\varphi_P(I))}(X)$. Hence, let $h \in T^s_{\alpha_*(\varphi_P(\gamma(\alpha(I))))}(X)$. Then, by definition, there exist $c' = h' \leftarrow b_1, \ldots, b_n \in \alpha_*(\varphi_P(\gamma(\alpha(I))))$ (with $n \geqslant 0$) and $\langle b'_1, \ldots, b'_n \rangle \ll_{c'} X$ such that $\theta = mgu(\langle b_1, \ldots, b_n \rangle, \langle b'_1, \ldots, b'_n \rangle)$ and $h = h'\theta$. Therefore, there exists $\bar{c}$ such that $\pi = h' \xrightarrow{\bar{c}} b_1, \ldots, b_n \in \varphi_P(\gamma(\alpha(I)))$. By definition of φ_P, we can have the following two cases.

(i) If $c' = h' \leftarrow b_1, \ldots, b_n \in P$ then $h' \xrightarrow{c'} b_1, \ldots, b_n \in \varphi_P(I)$, and therefore $h \in T^s_{\alpha_*(\varphi_P(I))}(X)$.

(ii) Otherwise if $h' \xrightarrow{\bar{c}} b_1, \ldots, b_n \in P \nabla \gamma(\alpha(I))$ then $h \in \alpha_*((P \nabla \gamma(\alpha(I))) \nabla \gamma_*(X))$. By associativity of ∇ (cf. Proposition 7.3), $h \in \alpha_*(P \nabla (\gamma(\alpha(I)) \nabla \gamma_*(X)))$, and since $\gamma_*(X) = \gamma_{\mathscr{S}}(X)$, we get $h \in \alpha_*(P \nabla (\gamma(\alpha(I)) \nabla \gamma_{\mathscr{S}}(X)))$. Thus, we may have the following two cases.

(a) If h is a unit-clause in P (i.e., $h \leftarrow \; \in P$), then h is a unit-clause in $\alpha_*(\varphi_P(I))$, and therefore $h \in T^s_{\alpha_*(\varphi_P(I))}(X)$.

(b) Otherwise, there exist $c'' = h'' \leftarrow r_1, \ldots, r_m \in P$ and $\langle r'_1, \ldots, r'_m \rangle \ll_{c''} \alpha_*(\gamma(\alpha(I)) \nabla \gamma_{\mathscr{S}}(X))$ such that $\sigma = mgu(\langle r_1, \ldots, r_m \rangle, \langle r'_1, \ldots, r'_m \rangle)$ and $h = h''\sigma$. Therefore, we also have that $\langle r'_1, \ldots, r'_m \rangle \ll_{c''} \alpha_{\mathscr{S}}(\gamma(\alpha(I)) \nabla \gamma_{\mathscr{S}}(X))$. Since $\alpha(\gamma(\alpha(I))) = \alpha(I)$, $\alpha_{\mathscr{S}}(\gamma(\alpha(I)) \nabla \gamma_{\mathscr{S}}(X)) = \alpha_{\mathscr{S}}(I \nabla \gamma_{\mathscr{S}}(X))$. So, $\langle r'_1, \ldots, r'_m \rangle \ll_{c''} \alpha_{\mathscr{S}}(I \nabla \gamma_{\mathscr{S}}(X))$, and therefore we have that $h \in \alpha_{\mathscr{S}}(P \nabla (I \nabla \gamma_{\mathscr{S}}(X)))$. By associativity of ∇, we have that $h \in \alpha_{\mathscr{S}}((P \nabla I) \nabla \gamma_{\mathscr{S}}(X))$. Since $P \nabla I \subseteq \varphi_P(I)$, by monotonicity of ∇ and $\alpha_{\mathscr{S}}$, $\alpha_{\mathscr{S}}((P \nabla I) \nabla \gamma_{\mathscr{S}}(X)) \subseteq \alpha_{\mathscr{S}}(\varphi_P(I) \nabla \gamma_{\mathscr{S}}(X))$, and therefore $h \in \alpha_{\mathscr{S}}(\varphi_P(I) \nabla \gamma_{\mathscr{S}}(X))$. Thus, by Lemma 7.5, $h \in T^s_{\alpha_*(\varphi_P(I))}(X)$. □

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

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

We are then able to characterize the semantics $[\![\cdot]\!]^{\mathscr{S}\mapsto\mathscr{S}}$ as follows.

Theorem 7.11. *For any $P \in Program$, $[\![P]\!]^{\mathscr{S}\mapsto\mathscr{S}} = T^s_{\alpha_*(\mathscr{T}_P^{and})} = \lambda X . \alpha_{\mathscr{S}}(\mathscr{T}_P^{and} \nabla \gamma_{\mathscr{S}}(X))$.*

Proof. Let us consider the following equalities:

$$\begin{aligned}
[\![P]\!]^{\mathscr{S}\mapsto\mathscr{S}} &= lfp(T_P^{\mathscr{S}\mapsto\mathscr{S}})\\
&= \bigsqcup_{n<\omega}(T_P^{\mathscr{S}\mapsto\mathscr{S}})^n(\bot_{Dep_\nabla(\mathscr{S})})\\
&= \bigsqcup_{n<\omega}(T_P^{\mathscr{S}\mapsto\mathscr{S}})^n(\alpha(\emptyset))\\
\text{(by Lemma 7.10)} &= \bigsqcup_{n<\omega}\alpha(\varphi_P^n(\emptyset))\\
&= \alpha\left(\bigcup_{n<\omega}\varphi_P^n(\emptyset)\right)\\
&= \alpha(lfp(\varphi_P)).
\end{aligned}$$

Since $lfp(\varphi_P)=\mathscr{T}_P^{and}$, we get $[\![P]\!]^{\mathscr{S}\mapsto\mathscr{S}}=\lambda X.\alpha_{\mathscr{S}}(\mathscr{T}_P^{and}\nabla\gamma_{\mathscr{S}}(X))$, and therefore, by Lemma 7.5, $[\![P]\!]^{\mathscr{S}\mapsto\mathscr{S}}=T^s_{\alpha_*(\mathscr{T}_P^{and})}$. □

As announced above, the semantics $[\![\cdot]\!]^{\mathscr{S}\mapsto\mathscr{S}}$ turns out to be compositional w.r.t. set-union of program modules. We prove such theorem of compositionality by exploiting some results taken from Bossi et al. [10]. We use $[\![\cdot]\!]^{\#}:\wp(Clause)\mapsto\wp(Clause)$ to denote the compositional (w.r.t. set-union of programs) semantics of Bossi et al. [10], with all predicates considered as open. Hence, for every set of clauses $P\subseteq Clause$, the semantics $[\![P]\!]^{\#}$ is still a set of clauses. Let us remark that the results of [10] are given for a domain of so-called Ω-denotations which is an abstraction of $\wp(Clause)$, since a syntactic equivalence (cf. [10, Definition 3.2]) over clauses is considered. In order to avoid tedious technical details of little interest, we do not take into account this further level of abstraction, although this could be done with full rigour.

Lemma 7.12. *For any $P,Q\subseteq Clause$, $\alpha_{cl}(P)=\alpha_{cl}(Q)\Rightarrow T^s_{[\![P]\!]^{\#}}=T^s_{[\![Q]\!]^{\#}}$.*

Proof. By a straightforward inductive argument, it is easy to prove that if $T_P^s=T_Q^s$ then, for any $n\in\mathbb{N}$, $(T^s_{P\cup Id})^n=(T^s_{Q\cup Id})^n$, where $Id\overset{\text{def}}{=}\{p(x_1,\dots,x_n)\leftarrow p(x_1,\dots,x_n)\mid p\in\Pi\}$.[8] By [10, Definition 4.9, Theorem 4.14], for any program R, $[\![R]\!]^{\#}=\bigcup_{n\in\mathbb{N}}R_n$, where $R_1\overset{\text{def}}{=}R$ and $R_{n+1}\overset{\text{def}}{=}R_n\nabla(R\cup Id)$, and, as a consequence of [10, Lemma 4.12 (2)], for any $n\in\mathbb{N}$, $T^s_{R_{n+1}}=T^s_R\circ(T^s_{R\cup Id})^n$. If $\alpha_{cl}(P)=\alpha_{cl}(Q)$ then, by Theorem 7.9, $T_P^s=T_Q^s$, and therefore, for any $n\in\mathbb{N}$, $T^s_{P_{n+1}}=T^s_{Q_{n+1}}$. Thus, $T^s_{\bigcup_{n\in\mathbb{N}}P_n}=T^s_{\bigcup_{n\in\mathbb{N}}Q_n}$, i.e. $T^s_{[\![P]\!]^{\#}}=T^s_{[\![Q]\!]^{\#}}$. □

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

Proof. Let P and Q be two programs. It is proved in [34, Example 5.4] that $\alpha_*(\mathscr{T}_P^{and})=[\![P]\!]^{\#}$ (this fact will be recalled in more detail below). Hence, by Lemma 7.6 and Theorem 7.11, $\alpha_{cl}([\![P]\!]^{\#})=\lambda X.\alpha_{\mathscr{S}}(\gamma_*(\alpha_*(\mathscr{T}_P^{and}))\nabla\gamma_{\mathscr{S}}(X))=\lambda X.\alpha_{\mathscr{S}}(\mathscr{T}_P^{and}\nabla\gamma_{\mathscr{S}}(X))=[\![P]\!]^{\mathscr{S}\mapsto\mathscr{S}}$. Moreover, we know from [10, Theorem 2.13] that $[\![P\cup Q]\!]^{\#}=[\![[\![P]\!]^{\#}\cup[\![Q]\!]^{\#}]\!]^{\#}$.

[8] If n is the arity of $p\in\Pi$ then $x_1,\dots,x_n$ are distinct variables.

Thus, $[\![P \cup Q]\!]^{\mathscr{S}\mapsto\mathscr{S}} = \alpha_{cl}([\![[\![P]\!]^{\mathscr{B}} \cup [\![Q]\!]^{\mathscr{B}}]\!]^{\mathscr{B}}) = [\![[\![P]\!]^{\mathscr{B}} \cup [\![Q]\!]^{\mathscr{B}}]\!]^{\mathscr{S}\mapsto\mathscr{S}}$. Hence, it remains to demonstrate that $[\![[\![P]\!]^{\mathscr{B}} \cup [\![Q]\!]^{\mathscr{B}}]\!]^{\mathscr{S}\mapsto\mathscr{S}} = [\![\gamma_{cl}(\alpha_{cl}([\![P]\!]^{\mathscr{B}})) \cup \gamma_{cl}(\alpha_{cl}([\![Q]\!]^{\mathscr{B}}))]\!]^{\mathscr{S}\mapsto\mathscr{S}}$. We observed after Proposition 7.8 that γ_{cl} is additive, and therefore, it suffices to show that $[\![[\![P]\!]^{\mathscr{B}} \cup [\![Q]\!]^{\mathscr{B}}]\!]^{\mathscr{S}\mapsto\mathscr{S}} = [\![\gamma_{cl}(\alpha_{cl}([\![P]\!]^{\mathscr{B}} \cup [\![Q]\!]^{\mathscr{B}}))]\!]^{\mathscr{S}\mapsto\mathscr{S}}$. Since $\alpha_{cl}(\gamma_{cl}(\alpha_{cl}([\![P]\!]^{\mathscr{B}} \cup [\![Q]\!]^{\mathscr{B}}))) = \alpha_{cl}([\![P]\!]^{\mathscr{B}} \cup [\![Q]\!]^{\mathscr{B}})$, by Lemma 7.12, we get that

$$T^{s}_{[\![\gamma_{cl}(\alpha_{cl}([\![P]\!]^{\mathscr{B}} \cup [\![Q]\!]^{\mathscr{B}}))]\!]^{\mathscr{B}}} = T^{s}_{[\![[\![P]\!]^{\mathscr{B}} \cup [\![Q]\!]^{\mathscr{B}}]\!]^{\mathscr{B}}} = T^{s}_{[\![P]\!]^{\mathscr{B}} \cup [\![Q]\!]^{\mathscr{B}}}. \qquad (*)$$

By Theorem 7.11, we have that for any program R,

$$[\![R]\!]^{\mathscr{S}\mapsto\mathscr{S}} = T^{s}_{\alpha_*(\mathscr{T}^{and}_{R})} = T^{s}_{[\![R]\!]^{\mathscr{B}}},$$

and therefore, by $(*)$, this concludes the proof. □

As the attentive reader should have guessed, it turns out that our compositional semantics $[\![\cdot]\!]^{\mathscr{S}\mapsto\mathscr{S}}$ is an abstract interpretation of Bossi et al.'s [10] compositional semantics $[\![\cdot]\!]^{\mathscr{B}}$. In fact, Giacobazzi [34, Example 5.4] shows that $(\alpha_*, \wp(\mathbf{aT}), \wp(Clause), \gamma_*)$ induces exactly (up to the further syntactic level of abstraction cited above) Bossi et al.'s semantics: If, for any program P, $T^{\mathscr{B}}_P$ denotes Bossi et al.'s [10, Definition 4.3] immediate consequences operator such that $[\![P]\!]^{\mathscr{B}} = lfp(T^{\mathscr{B}}_P)$, we have that the completeness relation $\alpha_* \circ \varphi_P = T^{\mathscr{B}}_P \circ \alpha_*$ holds, and therefore, $[\![P]\!]^{\mathscr{B}} = lfp(\alpha_* \circ \varphi_P \circ \gamma_*) = \alpha_*(lfp(\varphi_P))$ (the last equality follows by completeness, cf. [21, 38]). Thus, since by Proposition 7.8, $T^{\mathscr{S}\mapsto\mathscr{S}}_P = \alpha \circ \varphi_P \circ \gamma = \alpha_{cl} \circ \alpha_* \circ \varphi_P \circ \gamma_* \circ \gamma_{cl}$, we get that $T^{\mathscr{S}\mapsto\mathscr{S}}_P = \alpha_{cl} \circ T^{\mathscr{B}}_P \circ \gamma_{cl}$. Actually, it turns out that we deal with a strict abstraction, as the following example shows.

Example 7.14. Let us consider the clauses $c_1 = p(x) \leftarrow q(x)$ and $c_2 = p(x) \leftarrow q(x), q(x)$ (these two clauses are not equivalent for the Bossi et al.'s [10, Definition 3.2] syntactic relation mentioned above, as observed in [10, Example 3.4]). It is immediate to check that $\mathscr{T}^{and}_{\{c_1\}} = \{p(x) \xrightarrow{c_1} q(x)\}$ and $\mathscr{T}^{and}_{\{c_2\}} = \{p(x) \xrightarrow{c_2} q(x), q(x)\}$, and therefore, for $i = 1, 2$, $\alpha_*(\mathscr{T}^{and}_{\{c_i\}}) = \{c_i\}$. Hence, for $i = 1, 2$, $[\![\{c_i\}]\!]^{\mathscr{B}} = \alpha_*(\mathscr{T}^{and}_{\{c_i\}}) = \{c_i\}$. On the other hand, observe that $T^{s}_{\{c_1\}} = T^{s}_{\{c_2\}}$. Thus, by Theorem 7.11, $[\![\{c_1\}]\!]^{\mathscr{S}\mapsto\mathscr{S}} = T^{s}_{\alpha_*(\mathscr{T}^{and}_{\{c_1\}})}$ $= T^{s}_{\{c_1\}} = T^{s}_{\{c_2\}} = T^{s}_{\alpha_*(\mathscr{T}^{and}_{\{c_2\}})} = [\![\{c_2\}]\!]^{\mathscr{S}\mapsto\mathscr{S}}$.

The above compositional semantics $[\![\cdot]\!]^{\mathscr{S}\mapsto\mathscr{S}}$ is just a good example of how semantics can be systematically defined by means of suitable operators for abstract domain combination. An appealing feature of this approach is that, being based on arbitrary G.c.'s, it is independent from any specific choice for representing semantic denotations. In fact, one 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 for the standard s-semantics approach to logic programming (cf. [9]). For instance, likewise to $Dep_{\nabla}(\mathscr{S})$, one could consider $Dep_{\nabla}(\mathscr{H})$ as a variant of the compositional semantics in [33, 48], or, analogously, one could combine by reduced relative power the semantics for call patterns, partial answers, etc. Moreover, a semantic domain of functions like $Dep_{\nabla}(\mathscr{S})$ can be further approximated in different ways (e.g. by using binary relations, pointwise abstraction, etc.) as suggested in [19, 23, 24]. It should be therefore clear that the reduced relative power operation may play an important role also for designing suitable semantics to be used as concrete semantics for program analysis.

8. Related work

Cousot and Cousot firstly introduced a *reduced cardinal power* operation on abstract domains [21, 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 more 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 benefit of the full generality of our theory in the 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 even commutative).

We have investigated the relationship between Nielson's *tensor product* [54, 55] and reduced relative power operations. 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. [44]) of using the powerset of the cartesian product (see, e.g., [54, 55] 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 [54, p. 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 operation, 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 the reduced relative power operation, and it is, in general, incomparable.

Bagnara [5] has developed a meta-language for abstract domain manipulation based on a subset of a concurrent constraint (*cc*) language, where a domain is required to be a particular sort 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. [59]). 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 considered in our application in logic program semantics design in Section 7.

Recently, Giacobazzi and Scozzari [40] gave an elegant logical interpretation of the reduced relative power, showing that whenever the concrete domain is a complete Heyting algebra, the reduced relative power of two abstract domains D_1 and D_2 coincides with the domain of all the intuitionistic implications (i.e. relative pseudocomplements) from D_1 to D_2. This topic needs to be deepened, since we believe that Giacobazzi and Scozzari's results constitute a first step towards a logic-based interpretation of abstract domain refinement operators.

As far as applications are concerned, Dart [25] introduced the domain of formulae *Def* describing ground-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 [49] 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 fine example of a typical and well-known abstract 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 [11] also have introduced a compositional semantics based on functions as program denotations. The main difference is that in their approach 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 $Dep_{\nabla}(\mathscr{S})$ 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 used in [10].

Acknowledgements

We wish to thank Gilberto Filé, Francesca Scozzari and Harald Søndergaard for many stimulating discussions. This work was mostly done in early 1995, while the authors were both 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.

References

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

[2] P. Aczel, An introduction to inductive definitions, in: J. Barwise (Ed.), Handbook of Mathematical Logic, North-Holland, Amsterdam, 1977, pp. 739–782.

[3] K.R. Apt, Introduction to logic programming, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, vol. B: Formal Models and Semantics, Elsevier, Amsterdam and MIT Press, Cambridge, MA, 1990, pp. 495–574.

[4] T. Armstrong, K. Marriott, P. Schachte, H. Søndergaard, Two classes of Boolean functions for dependency analysis, Sci. Comput. Program. 31(1) (1998) 3–45.

[5] R. Bagnara, A hierarchy of constraint systems for data-flow analysis of constraint logic-based languages, Sci. Comput. Program. 30(1–2) (1998) 119–155.

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

[7] G. Birkhoff, Lattice Theory, AMS Colloquium Publications, vol. XXV, 3rd ed., AMS, Providence, RI, 1967.

[8] R. Bol, J.F. Groote, The meaning of negative premises in transition system specification, J. ACM 43(5) (1996) 863–914.

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

[10] A. Bossi, M. Gabbrielli, G. Levi, M.C. Meo, A compositional semantics for logic programs, Theoret. Comput. Sci. 122(1–2) (1994) 3–47.

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

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

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

[14] M. Comini, G. Levi, An algebraic theory of observables, in: M. Bruynooghe (Ed.), Proc. Internat. Logic Programming Symp. (ILPS '94), MIT Press, Cambridge, MA, 1994, pp. 172–186.

[15] M. Comini, G. Levi, M.C. Meo, Compositionality of SLD-derivations and their abstractions, in: J. Lloyd (Ed.), Proc. Internat. Logic Programming Symp. (ILPS '95), MIT Press, Cambridge, MA, 1995.

[16] A. Cortesi, G. Filé, R. Giacobazzi, C. Palamidessi, F. Ranzato, Complementation in abstract interpretation, ACM Trans. Program. Lang. Systems 19(1) (1997) 7–47.

[17] A. Cortesi, B. Le Charlier, 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), ACM Press, New York, 1994, pp. 227–239.

[18] P. Cousot, Méthodes Itératives de Construction et d'Approximation de Points Fixes d'Opérateurs Monotones sur un Treillis, Analyse Sémantique des Programmes, Ph.D. Thesis, Université Scientifique et Médicale de Grenoble, Grenoble, France, 1978.

[19] P. Cousot, Constructive design of a hierarchy of semantics of a transition system by abstract interpretation (Invited Paper), in: S. Brookes, M. Mislove (Eds.), Proc. 13th Internat. Symp. on Mathematical Foundations of Programming Semantics (MFPS '97), vol. 6, Electronic Notes in Theoretical Computer Science. Elsevier, Amsterdam, 1997.

[20] P. Cousot, R. Cousot, Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints, in: Conf. Record of the 4th ACM Symp. on Principles of Programming Languages (POPL '77), ACM Press, New York, 1977, pp. 238–252.

[21] P. Cousot, R. Cousot, Systematic design of program analysis frameworks, in: Conf. Record of the 6th ACM Symp. on Principles of Programming Languages (POPL '79), ACM Press, New York, 1979, pp. 269–282.

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

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

[24] P. Cousot, R. Cousot, Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and PER analysis of functional languages) (Invited Paper), Proc. IEEE Internat. Conf. on Computer Languages (ICCL '94), IEEE Computer Society Press, Los Alamitos, CA, 1994, pp. 95–112.
[25] P. Dart, On derived dependencies and connected databases, J. Logic Program. 11(2) (1991) 163–188.
[26] F. Denis, J.P. Delahaye, Unfolding, procedural and fixpoint semantics of logic programs, in: C. Choffrut, M. Jantzen (Eds.), Proc. 8th Internat. Symp. on Theoretical Aspects of Computer Science (STACS '91), vol. 480, Lecture Notes in Computer Science, Springer, Berlin, 1991, pp. 511–522.
[27] M.H. Van Emden, R.A. Kowalski, The semantics of predicate logic as a programming language, J. ACM 23(4) (1976) 733–742.
[28] M. Falaschi, G. Levi, M. Martelli, C. Palamidessi, Declarative modeling of the operational behavior of logic languages, Theoret. Comput. Sci. 69(3) (1989) 289–318.
[29] G. Filé, R. Giacobazzi, F. Ranzato, A unifying view of abstract domain design, ACM Comput. Surv. 28(2) (1996) 333–336.
[30] G. Filé F. Ranzato, The powerset operator on abstract interpretations, Theoret. Comput. Sci. (1998), to appear.
[31] M. Gabbrielli, G. Levi, M.C. Meo, Observable behaviors and equivalences of logic programs, Inform. Comput. 122(1) (1995) 1–29.
[32] M. Gabbrielli, G. Levi, M.C. Meo, Resultant semantics for Prolog, J. Logic Comput. 6(4) (1996) 491–521.
[33] H. Gaifman, E. Shapiro, Fully abstract compositional semantics for logic programs, Conf. Record of the 16th ACM Symp. on Principles of Programming Languages (POPL '89), ACM Press, New York, 1989, pp. 134–142.
[34] R. Giacobazzi, "Optimal" collecting semantics for analysis in a hierarchy of logic program semantics, in: C. Puech, R. Reischuk (Eds.), Proc. 13th Internat. Symp. on Theoretical Aspects of Computer Science (STACS '96), vol. 1046, Lecture Notes in Computer Science, Springer, Berlin, 1996, pp. 503–514.
[35] R. Giacobazzi, C. Palamidessi, F. Ranzato, Weak relative pseudo-complements of closure operators, Algebra Universalis 36(3) (1996) 405–412.
[36] R. Giacobazzi, F. Ranzato, Functional dependencies and Moore-set completions of abstract interpretations and semantics, in: J. Lloyd (Ed.), Proc. 1995 Internat. Logic Programming Symp. (ILPS '95), MIT Press, Cambridge, MA, 1995, pp. 321–335.
[37] R. Giacobazzi, F. Ranzato, Refining and compressing abstract domains, in: P. Degano, R. Gorrieri, A. Marchetti-Spaccamela (Eds.), Proc. 24th Internat. Colloq. on Automata, Languages and Programming (ICALP '97), vol. 1256, Lecture Notes in Computer Science, Springer, Berlin, 1997, pp. 771–781.
[38] R. Giacobazzi, F. Ranzato, Completeness in abstract interpretation: a domain perspective, in: M. Johnson (Ed.), Proc. 6th Internat. Conf. on Algebraic Methodology and Software Technology (AMAST'97), vol. 1349, Lecture Notes in Computer Science, Springer, Berlin, 1997, pp. 231–245.
[39] R. Giacobazzi, F. Ranzato, Optimal domains for disjunctive abstract interpretation, Sci. Comput. Program. 32 (1–3) (1998) 177–210.
[40] R. Giacobazzi, and F. Scozzari, A logical model for relational abstract domains, ACM Trans. Program. Lang. Syst. (1998), to appear.
[41] G. Grätzer. General Lattice Theory, Birkhäuser Verlag, Basel, 1978.
[42] T.P. Jensen, Disjunctive strictness analysis, Proc. 7th IEEE Symp. on Logic in Computer Science (LICS '92), IEEE Computer Society Press, Los Alamitos, CA, 1992, pp. 174–185.
[43] T.P. Jensen, Disjunctive program analysis for algebraic data types, ACM Trans. Program. Lang. Systems 19(5) (1997) 751–803.
[44] N.D. Jones, S.S. Muchnick, Complexity of flow analysis, inductive assertion synthesis and a language due to Dijkstra, in: S.S. Muchnick, N.D. Jones (Eds.), Program Flow Analysis: Theory and Applications, Prentice-Hall, Englewood Cliffs, NJ, 1981, pp. 380–393.
[45] N.D. Jones, H. Søndergaard, A semantics-based framework for the abstract interpretation of Prolog, in: S. Abramsky, C. Hankin (Eds.), Abstract Interpretation of Declarative Languages, Ellis Horwood Ltd., Chichester, UK, 1987, pp. 123–142.
[46] R.S. Kemp, G.R. Ringwood, Reynolds and Heyting models of definite clauses, Tech. Report 575, Comput. Science Dept., Queen Mary and Westfield College, London, 1991.
[47] J.L. Lassez, M.J. Maher, K. Marriott, Unification revisited, in: J. Minker (Ed.), Foundations of Deductive Databases and Logic Programming, Morgan Kaufmann, Los Altos, CA. 1988, pp. 587–625.

[48] M.J. Maher, Equivalences of logic programs, in: J. Minker (Ed.), Foundations of Deductive Databases and Logic Programming, Morgan Kaufmann, Los Altos, CA, 1988, pp. 627–658.
[49] K. Marriott, H. Søndergaard, Precise and efficient groundness analysis for logic programs, ACM Lett. Program. Lang. Systems 2(1–4) 1993, pp. 181–196.
[50] J. Morgado, Some results on the closure operators of partially ordered sets, Portug. Math. 19(2) (1960) 101–139.
[51] J. Morgado, Note on complemented closure operators of complete lattices, Portug. Math. 21(3) (1962) 135–142.
[52] C.J. Mulvey, &, Suppl. Rend. Circ. Mat. Palermo 12 (1986) 99–104.
[53] K. Muthukumar, M. Hermenegildo, Combined determination of sharing and freeness of program variables through abstract interpretation, in: K. Furukawa (Ed.), Proc. 8th Internat. Conf. on Logic Programming (ICLP'91), MIT Press, Cambridge, MA, 1991, pp. 49–63.
[54] F. Nielson, Abstract interpretation using domain theory, Ph.D. Thesis, CST-31-84, Department of Computer Science, University of Edinburgh, Edinburgh, Scotland, 1984.
[55] F. Nielson, Tensor products generalize the relational data flow analysis method, in: M. Arató, I. Kátai, L. Varga (Eds.), Proc. 4th Hungarian Computer Science Conf., 1985, pp. 211–225.
[56] F. Nielson, Two-level semantics and abstract interpretation, Theoret. Comput. Sci. 69 (1989) 117–242.
[57] F. Nielson, H. Riis Nielson, The tensor product in Wadler's analysis of lists, Sci. Comput. Program. 22 (1994) 327–354.
[58] K.I. Rosenthal, Quantales and their Applications, Pitman Research Notes in Mathematics Series, Longman Scientific & Technical, Essex, UK, 1990.
[59] V. Saraswat, V.A. Rinard, P. Panangaden, Semantic foundations of concurrent constraint programming, in: Conf. Record of the 18th ACM Symp. on Principles of Programming Languages (POPL'91), ACM Press, New York, 1991, pp. 333–353.
[60] F. Scozzari, Logical optimality of groundness analysis, in: P. Van Hentenryck (Ed.), Proc. 4th Internat. Static Analysis Symp. (SAS '97), vol. 1302, Lecture Notes in Computer Science, Springer, Berlin, 1997, pp. 83–97. Extended version to appear in Theoret. Comput. Sci.
[61] Z. Shmuely, The structure of Galois connections, Pacific J. Math. 54(2) (1974) 209–225.
[62] H. Søndergaard, Immediate fixpoints and their use in groundness analysis, in: V. Chandru, V. Vinay (Eds.), Proc. 16th Conf. on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'96), vol. 1180, Lecture Notes in Computer Science, Springer, Berlin, 1996, pp. 359–370.
[63] R. Sundararajan, J. Conery, An abstract interpretation scheme for groundness, freeness, and sharing analysis of logic programs, in: R. Shyamasundar (Ed.), Proc. 12th Conf. on Foundations of Software Technology and Theoretical Computer Science (FST&TCS '92), vol. 652, Lecture Notes in Computer Science, Springer, Berlin, 1992, pp. 203–216.
[64] M. Ward, The closure operators of a lattice, Ann. of Math. 43(2) (1942) 191–196.