Making Abstract Domains Condensing

ROBERTO GIACOBAZZI
Università di Verona, Italy
FRANCESCO RANZATO
Università di Padova, Italy
FRANCESCA SCOZZARI
Università di Pisa, Italy

In this paper we show that reversible analyses of logic languages by abstract interpretation can be performed without loss of precision by systematically refining abstract domains. This is obtained by adding to the abstract domain the minimal amount of concrete semantic information so that this refined abstract domain becomes rich enough to allow goal-driven and goal-independent analyses agree. These domains are known as condensing abstract domains. Essentially, an abstract domain A is condensing when the goal-driven analysis performed on A for a program P and a given query can be retrieved with no loss of precision from the goal-independent analysis on A of P. We show that condensation is an abstract domain property and that the problem of making an abstract domain condensing boils down to the problem of making the corresponding abstract interpretation complete, in a weakened form, with respect to unification. In the case of abstract domains for logic program analysis approximating computed answer substitutions, we provide a clean logical characterization of condensing domains as fragments of propositional linear logic. We apply our methodology to the systematic design of condensing domains for freeness and independence analysis.

Categories and Subject Descriptors: D.3.1 [**Programming Languages**]: Formal Definitions and Theory—*semantics*; D.3.2 [**Programming Languages**]: Language Classifications—*constraint and logic languages*; F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages—*program analysis*

General Terms: Languages, Theory

Additional Key Words and Phrases: Abstract domain, abstract interpretation, completeness, condensation, linear logic, logic program analysis

Authors' addresses: R. Giacobazzi, Dipartimento di Informatica, Università di Verona, Strada Le Grazie 15, Ca' Vignal 2, 37134 Verona, Italy, e-mail: roberto.giacobazzi@univr.it; F. Ranzato, Dipartimento di Matematica Pura ed Applicata, Università di Padova, Via Belzoni 7, 35131 Padova, Italy, e-mail: franz@math.unipd.it; F. Scozzari, Dipartimento di Informatica, Università di Pisa, Via Buonarroti 2, 56127 Pisa, Italy, e-mail: scozzari@di.unipi.it. This work was partially supported by the FIRB Project RBAU018RCZ "Abstract interpretation and model checking for the verification of embedded systems", by the COFIN2000 Project "Abstract Interpretation, type systems and control-flow analysis" and by the COFIN2002 Project "Constraint based verification of reactive systems (CoVer)"

1. INTRODUCTION

Static analyzers for logic languages are often goal-driven (a.k.a. query-directed). This means that the analysis computes an approximation of the semantics of a program for a fixed query with a given initial description. On the other hand, a goal-independent analyzer computes information on a program P for all the possible initial queries for P, and then this whole abstract semantics allows to derive the information of the analysis for a particular query. It is well known that, in general, goal-independent analyses, like those obtainable with bottom-up analyzers [Barbuti et al. 1993; Codish et al. 1994; Marriot et al. 1994], may be less precise than goal-directed ones (cf. [Marriott and Søndergaard 1993, Section 4]). When these two approaches agree, we are in a particularly lucky situation, which can reasonably be called optimal.

The problem. A well known property of concrete operational semantics for logic programs specifies that any refutation for the instance $Q\theta$ of a query Q in a program P yields the same answer substitution as the one that can be obtained by composing θ with the answer substitution computed for Q in P. This technical property, also known as lifting lemma, is fundamental for proving the completeness of SLD-resolution [Apt 1990]. Asking that this property holds for abstract computations too, corresponds to ask whether the analysis of a query can be made independent from its instantiation degree. The general problem of making the analysis independent from the choice of the instantiation of the initial query has been considered by many authors (see e.g. [Codish and Lagoon 2000; Debray 1994; Jacobs and Langen 1992; King and Lu 2002; Langen 1991; Giacobazzi and Scozzari 1998; Marriott and Søndergaard 1993; Schachte 2003]. Langen [1991] first gave a specific solution to this problem by introducing the idea of so-called *condensing procedures*, which capture the essence of the problem: Basically, the approximation of the semantics of each clause in a program is pre-computed (this is called condensed procedure) in such a way that any specific call to a predicate p can be approximated without re-computing a fixpoint, but simply by unifying that call against the condensed procedure for p. However, the process of condensation may loose precision w.r.t. the corresponding goal-directed analysis. Jacobs and Langen [1992, Theorem 3] showed that certain algebraic properties of abstract unification guarantee that this loss of precision does not occur. Abstract domains ensuring condensation without loss of precision are therefore called condensing. The problem of systematically designing condensing abstract domains is still open. Clearly, this is a relevant problem since static analyzers based on condensing abstract domains are both efficient and precise. Moreover, few condensing abstract domains are known, all of them being downward closed domains (i.e., closed by substitution instantiation). Giacobazzi and Scozzari [1998] showed that for any downward closed abstract domain A, A is condensing if and only if A is closed by the so-called *Heyting completion*. This result is not fully satisfactory because it cannot be generalized to arbitrary non-downward closed domains. It is therefore desirable to have a formal theoretical setting where possibly non-downward closed condensing abstract domains can be systematically designed, e.g. for important non-downward closed program properties like variable aliasing and freeness [Jacobs and Langen 1992; Langen 1991].

The main result. In general, condensing is a property of an abstract semantics S with respect to an operation $\otimes$, stating that for any pair of abstract objects a and b, the semantics $S(a \otimes b)$ can be retrieved as $a \otimes S(b)$. This generalizes and exactly captures the above notion of condensation when $\otimes$ is the abstract operation of unification. In particular, this also encompasses the case of the concrete semantics, where condensing for a semantics of computed answer substitutions boils down to the lifting lemma for SLD-resolution. We consider a concrete goal-driven collecting semantics $\mathcal{S}$ for logic programs which for any program P, query Q and initial call set Θ provides the set $\mathcal{S}_{P,Q}(\Theta)$ of computed answer substitutions in P for Q with initial calls in Θ. Of course, the concrete operations involved in the semantic definition of $\mathcal{S}$ are the usual ones: union of sets of substitutions, unification and projection on relevant variables. Moreover, it turns out that the collecting semantics $\mathcal{S}$ involves a linear implication (see below for the definition) of sets of substitutions: We show that this is a natural choice that becomes necessary when lifting the standard semantics for computed answer substitutions to its collecting version managing sets of substitutions. Hence, as usual, for any domain A abstracting the powerset $\wp(\mathtt{Sub})$ of substitutions the concrete collecting semantics induces an abstract semantics $\mathcal{S}^A$ on the domain A obtained by replacing the concrete operations involved in the definition of $\mathcal{S}$ by their corresponding best correct approximations on A. In this way, condensing becomes an abstract domain property: an abstract domain A is condensing when for any program P, query Q and abstract objects $a, b \in A$, $\mathcal{S}^A(a \otimes^A b) = a \otimes^A \mathcal{S}^A(b)$, where $\otimes^A$ is the best correct approximation of the unification $\otimes$ in the abstract domain A.

Our main result shows that it is always possible to make an abstract domain A condensing by minimally refining the domain A, i.e. by adding the least amount of concrete semantic information that makes the resulting domain condensing. We show that this is an instance of a more general problem of making a generic abstract domain A *complete*, in a weakened form, with respect to some concrete semantic operator f. Let us recall that if α_A and γ_A are the abstraction and concretization maps for the abstract domain A and $f^{\mathrm{bca}_A} = \alpha_A \circ f \circ \gamma_A$ is the best correct approximation of f in A, A is complete when $\alpha_A \circ f = f^{\mathrm{bca}_A} \circ \alpha_A$. We showed in [Giacobazzi et al. 2000] that any generic abstract domain A can be made complete with respect to any continuous semantic operator f: this means that A can always be constructively extended to the most abstract domain which includes A and is complete for f — the resulting domain is called the complete shell of A for f. In this paper, we follow this approach to systematically derive condensing abstract domains.

Here, it turns out that a weak form of completeness characterizes the condensing property. In fact, we show that an abstract domain A approximating the powerset $\wp(\mathtt{Sub})$ of concrete substitutions is condensing when there is no loss of precision in observing in A the result of the concrete unification $\otimes$ when just one of its arguments is approximated in A. More in detail, we prove that A is condensing if and only if for any $\Theta, \Phi \subseteq \mathtt{Sub}$, $\alpha_A(\gamma_A(\alpha(\Theta)) \otimes \gamma_A(\alpha_A(\Phi))) = \alpha_A(\Theta \otimes \gamma_A(\alpha_A(\Phi))) = \alpha_A(\gamma_A(\alpha_A(\Theta)) \otimes \Phi)$. This is precisely a weakened form of completeness, thus called *weak-completeness*, for the binary operation $\otimes$ of concrete unification lifted to sets of substitutions. It must be noted that this result holds for abstract domains which satisfy a property of compatibility with the variable projection operation, or, alternatively, which are complete for the projection (therefore this can be achieved by using the results in [Giacobazzi et al. 2000]). Given any abstract domain A, we characterize the most abstract domain X which is both condensing and more concrete than A, which is therefore called the *weak-complete shell* of A for $\otimes$. The crucial point of our result lies in the fact that the concrete domain $\wp(\mathtt{Sub})_{\subseteq}$ endowed with the unification operation $\otimes$ lifted to sets of substitutions gives rise to a *quantale*, i.e. a model of propositional linear logic (see [Rosenthal 1990; Yetter 1990]). In this context, it turns out that the objects of a weak-complete refined abstract domain can be elegantly represented as linear implications, with a clean logical interpretation. More in detail, a quantale $\langle C_{\leq}, \cdot \rangle$ consists of a complete lattice $C_{\leq}$ together with a binary operation $\cdot : C \times C \to C$ which is additive (i.e., preserves arbitrary lub's) on both arguments. As a main feature, quantales support a notion of linear implication between domain's objects: Given $a, b \in C$, there exists a unique greatest object $a \multimap b \in C$ which, when combined by $\cdot$ with a, gives a result which is approximated by b. In other terms, the following *modus ponens* law $a \cdot x \leq b \iff x \leq a \multimap b$ holds. In $\langle \wp(\mathtt{Sub})_{\subseteq}, \otimes \rangle$, $\Theta \multimap \Phi$ represents the set of all the substitutions θ such that $\Theta \otimes \{\theta\} \subseteq \Phi$. Therefore, when refining abstract domains in order to get weak-completeness in a setting where concrete interpretations are quantales of idempotent substitutions, abstract objects are logically characterized as linear implications in this quantale. This generalizes an analogous result given by Giacobazzi and Scozzari [1998], which characterizes condensing and downward closed domains as solutions of domain equations involving intuitionistic implications.

As a relevant example, we apply our methodology for systematically designing a condensing abstract domain which refines a basic domain representing both variable freeness and independence information.

2. BASIC NOTIONS

2.1 Notation

If S and T are sets, then $\wp(S)$ denotes the powerset of S, $S \subseteq_{\mathit{fin}} T$ denotes that S is a finite subset of T, $S \to T$ denotes the set of all functions from S to T, and for a function $f : S \to T$ and $X \subseteq S$, $f(X) \stackrel{\text{def}}{=} \{f(x) \mid x \in X\}$. By $g \circ f$ we denote the composition of the functions f and g, i.e., $g \circ f \stackrel{\text{def}}{=} \lambda x.g(f(x))$. The notation $P_{\leq}$ denotes a poset P with ordering relation $\leq$, while $\langle C, \leq, \vee, \wedge, \top, \bot \rangle$ denotes a complete lattice C, with ordering $\leq$, lub $\vee$, glb $\wedge$, greatest element (top) $\top$, and least element (bottom) $\bot$. Somewhere, $\leq_P$ will be used to denote the underlying ordering of a poset P, and $\vee_C$, $\wedge_C$, $\top_C$ and $\bot_C$ will denote operations and elements of a complete lattice C. Let P be a poset and $S \subseteq P$. Then, $\max(S) \stackrel{\text{def}}{=} \{x \in S \mid \forall y \in S.\ x \leq_P y \Rightarrow x = y\}$ denotes the set of maximal elements of S in P; also, the downward closure of S is defined by $\downarrow S \stackrel{\text{def}}{=} \{x \in P \mid \exists y \in S.\ x \leq_P y\}$. We use the symbol $\sqsubseteq$ to denote pointwise ordering between functions: If S is any set, P a poset, and $f, g : S \to P$ then $f \sqsubseteq g$ if for all $x \in S$, $f(x) \leq_P g(x)$. Let C and D be complete lattices. Then, $C \xrightarrow{\text{m}} D$, $C \xrightarrow{\text{c}} D$ and $C \xrightarrow{\text{a}} D$ denote, respectively, the set of all monotone, (Scott-)continuous and additive functions from C to D. Recall that $f \in C \xrightarrow{\text{c}} D$ if f preserves lub's of (non-empty) chains and

$f : C \xrightarrow{a} D$ is (completely) additive if f preserves lub's of arbitrary subsets of C (empty set included). Co-additivity is dually defined. We denote by $\mathit{lfp}(f)$ and $\mathit{gfp}(f)$, respectively, the least and greatest fixpoint, when they exist, of an operator f on a poset.

2.2 Logic programming

Let $\mathcal{V}$ be an infinite set of variables and `Term` be the set of terms with variables in $\mathcal{V}$. $\mathit{vars}(s)$ denotes the set of variables occurring in any syntactic object s. A syntactic object s is ground if $\mathit{vars}(s) = \varnothing$. We deal here with substitutions defined according to [Jacobs and Langen 1992]. A substitution σ is a mapping from a finite subset of $\mathcal{V}$, its domain $\mathit{dom}(\sigma)$, to `Term`. Note that we do not require substitutions to be idempotent, namely such that $\mathit{dom}(\sigma) \cap \mathit{rng}(\sigma) = \varnothing$, where $\mathit{rng}(\sigma)$ denotes the set of variables in the range of a substitution σ. Given a substitution σ, we will use the notation $\mathit{vars}(\sigma(x))$ even for variables x not in the domain of σ, namely $x \notin \mathit{dom}(\sigma)$, by defining $\mathit{vars}(\sigma(x))$ to be an emptyset. Moreover, for a set of substitutions Θ we define $\mathit{dom}(\Theta) = \cup_{\theta \in \Theta} \mathit{dom}(\theta)$.

This approach to substitutions differs from the standard one where substitutions are mappings from $\mathcal{V}$ to terms which are almost everywhere the identity, since any substitution is explicitly endowed with its domain. In the following, we write a substitution as the set of its bindings, for all the variables in the domain. For instance, $\{x/w, y/w\}$ and $\{x/x, y/w\}$ are substitutions with domain $\{x, y\}$. As usual, substitutions are applied to any syntactic object. We define a preorder on substitutions given by $\sigma \preceq \theta$ iff $\mathit{dom}(\theta) \subseteq \mathit{dom}(\sigma)$ and there exists a substitution δ such that $\forall v \in \mathit{dom}(\theta).\sigma(v) = \delta(\theta(v))$. We denote by $\sim$ the equivalence relation induced by this preorder $\preceq$ and we denote by `Sub` the set of equivalence classes of substitutions w.r.t. to $\sim$. In this way, $\preceq$ becomes a partial order for `Sub`. Thus, the equivalence class of a substitution denotes the sets of its "consistent renamings". For instance, $[\{x/w, y/w\}]_\sim = [\{x/u, y/u\}]_\sim$, $[\{x/x, y/w\}]_\sim = [\{x/u, y/v\}]_\sim$ and $[\{x/w\}]_\sim = [\{x/v\}]_\sim$, while $[\{x/w, y/w\}]_\sim \neq [\{x/u, y/v\}]_\sim$. This simplifies the approach to substitutions, since this makes a variable in the range of a substitution a placeholder.

We denote by ϵ the empty substitution (with empty domain), by ϵ_V, with $V \subseteq_{\mathit{fin}} \mathcal{V}$, the empty substitution with $\mathit{dom}(\epsilon_V) = V$ (that is, $\epsilon_V = \{v/w_v \mid v \in V\}$ where all the w_v are distinct variables not in V). The composition of substitutions is denoted by $\sigma \circ \theta = \lambda x.\sigma(\theta(x))$, where $\mathit{dom}(\sigma \circ \theta) = \mathit{dom}(\sigma) \cup \mathit{dom}(\theta)$. By $s\sigma$ and $\sigma(s)$ we denote the application of σ to any syntactic object s, where we mean that the results of applying a substitution to a syntactic object are considered as equivalence classes w.r.t. $\sim$.

We denote by $\mathit{mgu}(\sigma, \theta)$ the standard most general unifier of two substitutions σ and θ. The most general unifier of $[\sigma]_\sim$ and $[\theta]_\sim$, when it exists, is denoted by $\mathit{mgu}([\sigma]_\sim, [\theta]_\sim) = [\mathit{mgu}(\sigma', \theta')]_\sim$ where $\sigma' \in [\sigma]_\sim$, $\theta' \in [\theta]_\sim$ are such that $\mathit{vars}(\mathit{rng}(\sigma')) \cap \mathit{vars}(\mathit{rng}(\theta')) = \emptyset$. This is a good definition, since it is independent from the choice of the representatives in the equivalence classes. Also, note that, as shown by Palamidessi [1990], if we add a bottom element to `Sub`, it becomes a complete lattice and mgu becomes the meet operation of this complete lattice. In the following, for the sake of brevity, substitutions will denote their corresponding equivalence classes w.r.t. $\sim$.

For any set of variables $V \subseteq \mathcal{V}$ and substitution $\theta \in \texttt{Sub}$, we denote by $\theta_{|V}$ the restriction of the substitution θ to the variables in V, that is $\theta_{|V} = \{v/\theta(v) \mid v \in V \cap dom(\theta)\}$ with $dom(\theta_{|V}) = dom(\theta) \cap V$. Note that this is a good definition, namely it respects the equivalence $\sim$. Thus, in this approach variable projection simply corresponds to restriction: The projection $\pi_V(\Theta)$ for a set of substitutions $\Theta \subseteq \texttt{Sub}$ on a (possibly infinite) set of variables V is defined as $\pi_V(\Theta) \stackrel{\text{def}}{=} \{\theta_{|V} \mid \theta \in \Theta\}$. For instance, $\pi_{\{x,y,z\}}(\{\{x/z, u/f(z), y/w\}\}) = \{\{x/z, y/w\}\}$.

2.3 The lattice of abstract interpretations

In standard Cousot and Cousot's abstract interpretation theory, abstract domains can be equivalently specified either by Galois connections (GCs), i.e., adjunctions, or by (upper) closure operators (uco's) [Cousot and Cousot 1979]. In the first case, concrete and abstract domains C and A are related by a pair of adjoint functions of a GC (α, C, A, γ), where α and γ are the monotone abstraction and concretization maps such that for all $a \in A$ and $c \in C$, $\alpha(c) \leq_A a \;\Leftrightarrow\; c \leq_C \gamma(a)$. It is usually assumed that (α, C, A, γ) is a Galois insertion (GI), i.e., α is onto or, equivalently, γ is 1-1. In the second case, instead, an abstract domain is specified as a closure operator on the concrete domain C, i.e., a monotone, idempotent and extensive operator on C. These two approaches are completely equivalent, modulo isomorphic representations of domain's objects. The closure operator approach has the advantage of being independent from the representation of domain's objects: An abstract domain is a function on the concrete domain of computation. This feature makes it appropriate for reasoning on abstract domains independently from their representation. Given a complete lattice C, it is well known that the set $\mathrm{uco}(C)$ of all uco's on C, endowed with the pointwise ordering $\sqsubseteq$, gives rise to the complete lattice $\langle \mathrm{uco}(C), \sqsubseteq, \sqcup, \sqcap, \lambda x.\top_C, id \rangle$. Let us recall that each $\rho \in \mathrm{uco}(C)$ is uniquely determined by the set of its fixpoints, which is its image, i.e. $\rho(C) = \{x \in C \mid \rho(x) = x\}$, since $\rho = \lambda x. \wedge\{y \in C \mid y \in \rho(C),\ x \leq y\}$. Moreover, a subset $X \subseteq C$ is the set of fixpoints of a uco on C iff X is meet-closed, i.e. $X = \mathcal{M}(X) \stackrel{\text{def}}{=} \{\wedge Y \mid Y \subseteq X\}$ (note that $\top_C = \wedge\varnothing \in \mathcal{M}(X)$). For any $X \subseteq C$, $\mathcal{M}(X)$ is called the Moore-closure of X, and X is a meet generator set for $\mathcal{M}(X)$. Also, $\rho \sqsubseteq \eta$ iff $\eta(C) \subseteq \rho(C)$; in this case, ρ is a so-called refinement of η, and if $\rho \sqsubseteq \eta$ then $\rho \circ \eta = \eta \circ \rho = \eta$. Often, we will identify closures with their sets of fixpoints. This does not give rise to ambiguity, since one can distinguish their use as functions or sets according to the context. In view of the equivalence above, throughout the paper, $\langle \mathrm{uco}(C), \sqsubseteq \rangle$ will play the role of the lattice of abstract interpretations of C [Cousot and Cousot 1977; 1979], i.e. the complete lattice of all the abstract domains of the concrete domain C. When an abstract domain A is specified by a GI (α, C, A, γ), $\rho_A \stackrel{\text{def}}{=} \gamma \circ \alpha \in \mathrm{uco}(C)$ is the corresponding uco on C. Conversely, when $A = \rho(C)$ then $(\rho, C, \rho(C), \lambda x.x)$ is the corresponding GI. The ordering on $\mathrm{uco}(C)$ corresponds to the standard order used to compare abstract domains with regard to their precision: A_1 is more precise than A_2 (i.e., A_1 is more concrete than A_2 or A_2 is more abstract than A_1) iff $A_1 \sqsubseteq A_2$ in $\mathrm{uco}(C)$. Lub and glb on $\mathrm{uco}(C)$ have therefore the following reading as operators on domains. Let $\{A_i\}_{i \in I} \subseteq \mathrm{uco}(C)$: (i) $\sqcup_{i \in I} A_i$ is the most concrete among the domains which are abstractions of all the A_i's; (ii) $\sqcap_{i \in I} A_i$ is the most abstract among the domains which are more concrete than every A_i – this domain is also known as reduced product of all the A_i's.

2.4 Soundness and completeness in abstract interpretation

Let $f : C \xrightarrow{\mathrm{m}} C$ be a concrete semantic function[1] occurring in some semantics definition and let $f^\sharp : A \xrightarrow{\mathrm{m}} A$ be a corresponding abstract function, where $A = \rho(C)$ for some closure operator $\rho \in \mathrm{uco}(C)$. Then, $\langle A, f^\sharp \rangle$ is a sound abstract interpretation — or $f^\sharp$ is a correct approximation of f relatively to A — when $\rho \circ f \sqsubseteq f^\sharp \circ \rho$, or equivalently when $\rho \circ f \circ \rho \sqsubseteq f^\sharp \circ \rho$ [Cousot and Cousot 1977]. $f^{\mathrm{bca}_\rho} \stackrel{\mathrm{def}}{=} \rho \circ f : A \to A$ is called the best correct approximation (bca for short) of f in A. Completeness in abstract interpretation corresponds to require that, in addition to soundness, no loss of precision is introduced by the approximated function $f^\sharp \circ \rho$ on a concrete object c with respect to approximating by ρ the concrete computation $f(c)$, namely, no loss of precision is accumulated in abstract computations by approximating concrete input objects [Cousot and Cousot 1977; 1979]: $\langle A, f^\sharp \rangle$ is complete when $\rho \circ f = f^\sharp \circ \rho$. Giacobazzi et al. [2000] observed that completeness uniquely depends upon the abstraction map, i.e. upon the abstract domain: This means that if $f^\sharp$ is complete then the bca $f^{\mathrm{bca}_A} : A \to A$ of f in A is complete as well, and, in this case, $f^\sharp$ indeed coincides with f^{bca_A}. Thus, for any abstract domain A, one can define a complete abstract semantic operation $f^\sharp : A \to A$ over A if and only if $\rho \circ f : A \to A$ is complete. Hence, an abstract domain $\rho \in \mathrm{uco}(C)$ is defined to be complete for f iff $\rho \circ f = \rho \circ f \circ \rho$ holds. This simple observation makes completeness an abstract domain property, namely an intrinsic characteristic of the abstract domain. It is also worth recalling that, by a well-known result [Cousot and Cousot 1979, Theorem 7.1.0.4], complete abstract domains are "fixpoint complete" as well, i.e., if ρ is complete for f then $\rho(\mathit{lfp}(f)) = \mathit{lfp}(\rho \circ f)$, while the converse, in general, does not hold.

Giacobazzi et al. [2000] gave a constructive characterization of complete abstract domains, under the assumption of dealing with Scott-continuous concrete functions. This result allows to systematically derive complete abstract domains from non-complete ones by minimal refinements. The idea for refining an abstract domain A is to build the greatest (i.e., most abstract) domain in $\mathrm{uco}(C)$ which includes A and is complete for a set F of (continuous) functions, i.e., for each function in F. Given a set of continuous functions $F \subseteq C \xrightarrow{\mathrm{c}} C$, Giacobazzi et al. [2000] define the map $\mathcal{R}_F : \mathrm{uco}(C) \to \mathrm{uco}(C)$ as follows:

$$\mathcal{R}_F(\rho) \stackrel{\mathrm{def}}{=} \mathcal{M}\Big(\bigcup_{f \in F, a \in \rho} \max(\{x \in C \mid f(x) \leq a\})\Big).$$

THEOREM 2.1. [Giacobazzi et al. 2000] *A domain $\rho \in \mathrm{uco}(C)$ is complete for F iff $\rho \sqsubseteq \mathcal{R}_F(\rho)$.*

As a consequence, the most abstract domain which includes ρ and which is complete for F is $\mathit{gfp}(\lambda\eta.\rho \sqcap \mathcal{R}_F(\eta))$. This domain is called the *complete shell* of ρ for F. This result provides a constructive characterization of the complete shell of an abstract domain A as the most abstract domain including A and which is closed by maximal inverse image of any function in F. The complete shell of an abstract domain A with respect to a set of continuous functions F can therefore be equationally

[1]For simplicity, we consider unary functions with the same domain and co-domain, since the extension to the general case is conceptually straightforward.

characterized as the greatest (viz. most abstract) solution of the following domain equation:

$$X = A \sqcap \mathcal{R}_F(X).$$

2.5 Quantales and linear logic

Quantales originated in the algebraic foundations of the so-called quantum logic, and later they have been used as algebraic models of Girard's linear logic [Rosenthal 1990; Yetter 1990]. Informally, quantales can be thought of as a generalization of Boolean algebras, where the *modus ponens* law $a \wedge (a \Rightarrow b) \leq b$ holds relatively to a binary operation $\otimes$ of "conjunction" possibly different from the meet of the underlying algebraic structure. The basic idea in a quantale is to guarantee that, for any two objects a and b, there exists a greatest (i.e., most abstract in abstract interpretation terms) object c such that $a \otimes c \leq b$. In the following, we restrict our attention to commutative quantales, i.e., quantales where the binary operation $\otimes$ is commutative. More formally, a (commutative) quantale is an algebra $\langle C_{\leq}, \otimes \rangle$ such that:

—$\langle C, \leq, \vee, \wedge, \top, \bot \rangle$ is a complete lattice;
—$\otimes : C \times C \to C$ is a commutative and associative operation, i.e., $a \otimes b = b \otimes a$ and $(a \otimes b) \otimes c = a \otimes (b \otimes c)$, for any $a, b, c \in C$;
—$a \otimes (\bigvee_{i \in I} b_i) = \bigvee_{i \in I} (a \otimes b_i)$, for any $a \in C$ and $\{b_i\}_{i \in I} \subseteq C$.

In other words, a quantale is a complete lattice endowed with a commutative and associative "product" $\otimes$ which distributes over arbitrary lub's. Common examples of quantales are complete Boolean algebras, which become quantales by considering as $\otimes$ their meet operation. In particular, for any set S, the algebra $\langle \wp(S)_{\subseteq}, \bigcap \rangle$ is a quantale. Also, given a commutative and associative operation $\cdot : A \times A \to A$, a further basic example of quantale is $\langle \wp(S)_{\subseteq}, \otimes \rangle$, where $X \otimes Y \stackrel{\text{def}}{=} \bigcup \{x \cdot y \mid x \in X, y \in Y\}$ is the lifting of the operation $\cdot$ to sets. The fundamental property of quantales is that, for any $a \in C$, the function $\lambda x. a \otimes x$ has a right adjoint, denoted by $\lambda x. a \multimap x$. This is equivalent to say that one can define a binary operation $\multimap : C \times C \to C$ such that, for all $a, b, c \in C$, the following property holds:

$$a \otimes b \leq c \iff b \leq a \multimap c.$$

This is a straight consequence of the fact that, for all $a \in C$, $\lambda x. a \otimes x$ is additive, and therefore, it has a unique right adjoint $\lambda x. a \multimap x$ giving rise to a GC. As usual for GCs, the right adjoint $\multimap : C \times C \to C$ is defined as follows:

$$a \multimap c \stackrel{\text{def}}{=} \bigvee \{b \in C \mid a \otimes b \leq c\}.$$

A quantale $\langle C_{\leq}, \otimes \rangle$ is called *unital* if there exists an object $\mathbf{1} \in C$, called unit, such that $\mathbf{1} \otimes a = a = a \otimes \mathbf{1}$, for all $a \in C$. $\langle \wp(S)_{\subseteq}, \bigcap \rangle$ is a trivial example of unital, commutative quantale, where S is the unit.

From a logical point of view, it is well known that quantales turn out to be models of (commutative) linear logic [Rosenthal 1990; Yetter 1990], where the linear implication is interpreted as the operation $\multimap$. The next proposition summarizes the basic properties of linear implication (see [Rosenthal 1990]).

Proposition 2.2. *Let $\langle C_{\leq}, \otimes\rangle$ be a unital, commutative quantale with unit* $\mathbf{1}$*,* $\{x_i\}_{i\in I} \subseteq C$ *and* $a, b, c \in C$*.*

(i)	$a \otimes (a \multimap c) \leq c$	(ii)	$a \multimap (b \multimap c) = (b \otimes a) \multimap c$
(iii)	$a \multimap (\bigwedge_{i\in I} x_i) = \bigwedge_{i\in I}(a \multimap x_i)$	(iv)	$(\bigvee_{i\in I} x_i) \multimap c = \bigwedge_{i\in I}(x_i \multimap c)$
(v)	$a \multimap (b \multimap c) = b \multimap (a \multimap c)$	(vi)	$\mathbf{1} \multimap a = a$
(vii)	$\mathbf{1} \multimap \mathbf{1} = \mathbf{1}$	(viii)	$a \otimes (a \multimap a) = a$
(ix)	$c \leq (c \multimap a) \multimap a$	(x)	$((c \multimap a) \multimap a) \multimap a = c \multimap a$
(xi)	*if* $b \leq c$ *then* $a \otimes b \leq a \otimes c$	(xii)	*if* $a \leq b$ *then* $b \multimap c \leq a \multimap c$

In particular, from the above properties, it is easy to check that for all $a \in C$, $\lambda x.(x \multimap a) \multimap a \in \mathrm{uco}(C)$.

3. COMPLETENESS IN LOGICAL FORM

In this section we provide a linear logic-based characterization of complete abstract interpretations of quantales. Let $\langle C_{\leq}, \otimes\rangle$ be a unital, commutative quantale playing the role of concrete interpretation, that is, C is the concrete domain and $\otimes : C \times C \to C$ is the concrete semantic operation. Let $\rho \in \mathrm{uco}(C)$ be an abstract domain. Recall that ρ is complete for $\otimes$ when for all concrete objects $x, y \in C$, $\rho(\rho(x) \otimes \rho(y)) = \rho(x \otimes y)$. This can be more compactly expressed by the equation $\rho \circ \otimes \circ \langle \rho, \rho\rangle = \rho \circ \otimes$. Given any $\eta \in \mathrm{uco}(C)$, we define the following set of unary additive functions $F_\eta \subseteq C \xrightarrow{\mathrm{a}} C$:

$$F_\eta \overset{\mathrm{def}}{=} \{\lambda x.\ x \otimes y \mid y \in \eta\}.$$

In particular, F_{id} will be also denoted by F_C. It turns out that completeness of ρ for $\otimes$ is equivalent to completeness of ρ for F_C.

Lemma 3.1. *Let $\langle C_{\leq}, \otimes\rangle$ be a commutative quantale and $\rho \in \mathrm{uco}(C)$. The following are equivalent:*

(i) *ρ is complete for $\otimes$;*

(ii) $\rho \circ \otimes \circ \langle \rho, id\rangle = \rho \circ \otimes$*;*

(iii) *ρ is complete for F_C.*

Proof. We first show (i) $\Leftrightarrow$ (ii). Assume that $\rho \circ \otimes \circ \langle \rho, \rho\rangle = \rho \circ \otimes$. Then, by monotonicity and extensivity of ρ, we get $\rho\circ\otimes \sqsubseteq \rho\circ\otimes\circ\langle \rho, id\rangle \sqsubseteq \rho\circ\otimes\circ\langle \rho, \rho\rangle = \rho\circ\otimes$. On the other hand, assume that $\rho \circ \otimes \circ \langle \rho, id\rangle = \rho \circ \otimes$. By monotonicity and extensivity of ρ, $\rho \circ \otimes \sqsubseteq \rho \circ \otimes \circ \langle \rho, \rho\rangle = \rho \circ \otimes \circ \langle \rho, id\rangle \circ \langle id, \rho\rangle = \rho \circ \otimes \circ \langle id, \rho\rangle =$ (by commutativity of $\otimes$) $= \rho \circ \otimes \circ \langle \rho, id\rangle = \rho \circ \otimes$.
Thus, ρ is complete for $\otimes$ iff $\forall x, y \in C$, $\rho(\rho(x)\otimes y) = \rho(x\otimes y)$, and this is equivalent to state that ρ is complete for the set of unary functions $F_C = \{\lambda x.x \otimes y \mid y \in C\}$, which concludes the proof. □

Corollary 3.2. *Let $\langle C_{\leq}, \otimes\rangle$ be a commutative quantale and $\rho \in \mathrm{uco}(C)$. The complete shell of ρ for $\otimes$ is* $\mathit{gfp}(\lambda\eta.\rho \sqcap \mathcal{R}_{F_C}(\eta))$*.*

Proof. By Lemma 3.1, the complete shell of ρ for $\otimes$ coincides with the complete shell of ρ for F_C. Each function in F_C is additive, and therefore continuous. Thus, by Theorem 2.1, the complete shell of ρ for $\otimes$ is $gfp(\lambda\eta.\rho \sqcap \mathcal{R}_{F_C}(\eta))$. $\square$

Thus, the complete shell of any domain ρ for $\otimes$ can be constructively obtained by iterating the operator $\mathcal{R}_{F_C}$. Our aim is to show that this operator and, more generally, the family of operators $\mathcal{R}_{F_\eta}$, for any $\eta \in \mathrm{uco}(C)$, can all be characterized in terms of sets of linear implications. Let us define a binary operator on abstract domains $\overset{\wedge}{\multimap} : \mathrm{uco}(C) \times \mathrm{uco}(C) \to \mathrm{uco}(C)$ by lifting linear implication $\multimap$ to domains as follows: For any $A, B \in \mathrm{uco}(C)$, $A \overset{\wedge}{\multimap} B$ is the most abstract domain containing all the linear implications from A to B:

$$A \overset{\wedge}{\multimap} B \overset{\mathrm{def}}{=} \mathcal{M}(\{a \multimap b \in C \mid a \in A,\, b \in B\}).$$

Theorem 3.3. *Let $\langle C_\leq, \otimes\rangle$ be a unital, commutative quantale. For any $\rho, \eta \in \mathrm{uco}(C)$, $\mathcal{R}_{F_\eta}(\rho) = \eta \overset{\wedge}{\multimap} \rho$.*

Proof.

$$\begin{array}{rr}
\mathcal{R}_{F_\eta}(\rho) = & [\text{ by definition of } \mathcal{R}_{F_\eta}\ ] \\
\mathcal{M}(\cup_{f \in F_\eta, a \in \rho} \max(\{x \in C \mid f(x) \leq a\})) = & [\text{ by definition of } F_\eta\ ] \\
\mathcal{M}(\cup_{y \in \eta, a \in \rho} \max(\{x \in C \mid x \otimes y \leq a\})) = & [\text{ by commutativity of } \otimes\ ] \\
\mathcal{M}(\cup_{y \in \eta, a \in \rho} \max(\{x \in C \mid y \otimes x \leq a\})) = & [\text{ by definition of } \multimap\ ] \\
\mathcal{M}(\cup_{y \in \eta, a \in \rho} \max(\{x \in C \mid x \leq y \multimap a\})) = & \\
\mathcal{M}(\cup_{y \in \eta, a \in \rho} \{y \multimap a\}) = & \\
\eta \overset{\wedge}{\multimap} \rho. \quad \square &
\end{array}$$

The following basic properties of $\overset{\wedge}{\multimap}$ follow directly from the corresponding properties of the linear implication in quantales.

Proposition 3.4. *For all $A \in \mathrm{uco}(C)$ and $\{B_i\}_{i \in I} \subseteq \mathrm{uco}(C)$, we have:*

(i) $A \overset{\wedge}{\multimap} (\bigsqcap_{i \in I} B_i) = \bigsqcap_{i \in I} (A \overset{\wedge}{\multimap} B_i)$;
(ii) $A \overset{\wedge}{\multimap} \top_{\mathrm{uco}(C)} = \top_{\mathrm{uco}(C)}$;
(iii) $\lambda X. A \overset{\wedge}{\multimap} X$ *is monotone;*
(iv) $C \overset{\wedge}{\multimap} A \sqsubseteq A$;
(v) $C \overset{\wedge}{\multimap} A = C \overset{\wedge}{\multimap} (C \overset{\wedge}{\multimap} A)$.

Proof. Points (i), (ii) and (iii) are straightforward.
(iv): By Prop. 2.2 (vi), for all $a \in A$ it holds $\mathbf{1} \multimap a = a$. Since $\mathbf{1} \in C$, it follows that $a = \mathbf{1} \multimap a \in C \overset{\wedge}{\multimap} A$, and therefore $C \overset{\wedge}{\multimap} A \sqsubseteq A$.
(v): By point (iv), $C \overset{\wedge}{\multimap} A \sqsubseteq A$, and therefore, by point (iii), $C \overset{\wedge}{\multimap} (C \overset{\wedge}{\multimap} A) \sqsubseteq C \overset{\wedge}{\multimap} A$. For the other inequality, consider an element in $\{c \multimap a \in C \mid c \in C, a \in C \overset{\wedge}{\multimap} A\}$. By definition, such an element can be written as follows: $c \multimap \bigwedge_{i \in I}(d_i \multimap a_i)$, for suitable $c, d_i \in C$, and $a_i \in A$, for all $i \in I$, where I is a suitable set of indexes. Then,

$$\begin{array}{rll}
c \multimap \bigwedge_{i \in I}(d_i \multimap a_i) & = \bigwedge_{i \in I}(c \multimap (d_i \multimap a_i)) & [\text{ by Prop. 2.2 (iii) }] \\
& = \bigwedge_{i \in I}((d_i \otimes c) \multimap a_i) & [\text{ by Prop. 2.2 (ii) }]
\end{array}$$

Since, for all $i \in I$, $d_i \otimes c \in C$ and $a_i \in A$, $(d_i \otimes c) \multimap a_i \in C \overset{\wedge}{\multimap} A$. Then, by monotonicity of the Moore-closure, we get $C \overset{\wedge}{\multimap} A \sqsubseteq C \overset{\wedge}{\multimap} (C \overset{\wedge}{\multimap} A)$. $\square$

It is worth noting that, by points (iv) and (v) above, the monotone operator $\lambda X. C \overset{\wedge}{\multimap} X : \mathrm{uco}(C) \to \mathrm{uco}(C)$ is reductive and idempotent, and therefore it is a lower closure operator on $\mathrm{uco}(C)$. Also, it is important to note that in general A and $A \overset{\wedge}{\multimap} A$ are incomparable abstract domains.

Example 3.5. Consider the complete lattice of integer intervals `Int` defined as follows [Cousot and Cousot 1977]:

$$\mathtt{Int} = \{[a,b] \mid a,b \in \mathbb{Z}, a \le b\} \cup \{(-\infty, b] \mid b \in \mathbb{Z}\} \cup \{[a,+\infty) \mid a \in \mathbb{Z}\} \cup \{\mathbb{Z}\} \cup \{\varnothing\},$$

ordered by set inclusion. Let $\mathit{Msign} = \{\mathbb{Z}, [-m,-1], [1,m], \varnothing\}$, with $m \ge 1$, be an abstraction of `Int` combining sign and "maxint" information. For finite intervals, let us define $[a,b] \oplus [a',b'] = [a+a', b+b']$, while $\oplus$ is extended to infinite (and empty) intervals in the most natural way (for example, $(-\infty,b] \oplus [a',b'] = (-\infty, b+b']$ and $\varnothing \oplus [a,b] = \varnothing$). It is easy to check that $\langle \mathtt{Int}_{\subseteq}, \oplus \rangle$ is a commutative and unital quantale where $\mathbf{1} = [0,0] = \{0\}$. It turns out that

$$\mathit{Msign} \overset{\wedge}{\multimap} \mathit{Msign} = \{\mathbb{Z}, \{m+1\}, \{-(m+1)\}, \{0\}, \varnothing\}.$$

In fact, we have that $[1,m] \multimap [1,m] = \{0\} = [-m,-1] \multimap [-m,-1]$, $[1,m] \multimap [-m,-1] = \{-(m+1)\}$ and $[-m,-1] \multimap [1,m] = \{m+1\}$. Thus, $\mathit{Msign} \not\sqsubseteq \mathit{Msign} \overset{\wedge}{\multimap} \mathit{Msign}$ and $\mathit{Msign} \overset{\wedge}{\multimap} \mathit{Msign} \not\sqsubseteq \mathit{Msign}$. $\square$

The following result shows that the complete shell of an abstract domain A for $\otimes$ is given by all the linear implications from the concrete domain to A. This provides a first representation result for objects of complete abstractions of quantales.

THEOREM 3.6. *Let $\langle C_{\le}, \otimes \rangle$ be a unital, commutative quantale and $A \in \mathrm{uco}(C)$. The complete shell of A for $\otimes$ is $C \overset{\wedge}{\multimap} A$.*

PROOF. By Corollary 3.2, the complete shell of A for $\otimes$ is $\mathit{gfp}(\lambda X. A \sqcap \mathcal{R}_{F_C}(X))$, and, by Theorem 3.3, this is $\mathit{gfp}(\lambda X. A \sqcap (C \overset{\wedge}{\multimap} X))$. Let us show that $C \overset{\wedge}{\multimap} A = \mathit{gfp}(\lambda X. A \sqcap (C \overset{\wedge}{\multimap} X))$. First, by Proposition 3.4 (iv) and (v), we have that $C \overset{\wedge}{\multimap} A = A \sqcap (C \overset{\wedge}{\multimap} A) = A \sqcap (C \overset{\wedge}{\multimap} (C \overset{\wedge}{\multimap} A))$. Moreover, if X is a fixpoint of $\lambda X. A \sqcap (C \overset{\wedge}{\multimap} X)$ then $X \sqsubseteq A$ and $X \sqsubseteq C \overset{\wedge}{\multimap} X$, and therefore by right monotonicity of $\overset{\wedge}{\multimap}$ (cf. Proposition 3.4 (iii)), $X \sqsubseteq C \overset{\wedge}{\multimap} X \sqsubseteq C \overset{\wedge}{\multimap} A$. Thus, $C \overset{\wedge}{\multimap} A$ actually is the greatest fixpoint of $\lambda X. A \sqcap C \overset{\wedge}{\multimap} X$. $\square$

The relevance of this result stems from the fact that, in the considered case of concrete quantales, the fixpoint construction of the complete shell of an abstract domain converges in two steps, and this provides a clean logical characterization for the objects of the complete shell in terms of linear implications. Furthermore, the following result yields an explicit logical characterization for the abstraction map associated with that complete shell.

THEOREM 3.7. *Let $\langle C_{\le}, \otimes \rangle$ be a unital, commutative quantale and $A \in \mathrm{uco}(C)$. Let $\rho \in \mathrm{uco}(C)$ be the uco associated with $C \overset{\wedge}{\multimap} A$. Then, for all $c \in C$,*

$$\rho(c) = \bigwedge_{a \in A} (c \multimap a) \multimap a.$$

Proof. Since $A = \bigsqcap_{a\in A}\{\top_C, a\}$, by Proposition 3.4 (i), we have that $C \overset{\wedge}{\multimap} A = \bigsqcap_{a\in A} C \overset{\wedge}{\multimap} \{\top_C, a\}$. Let us show that the closure operator $\rho_a \in \mathrm{uco}(C)$ associated with $C \overset{\wedge}{\multimap} \{\top_C, a\}$ is $\rho_a = \lambda c.(c \multimap a) \multimap a$, i.e., by Theorem 3.6, ρ_a is the complete shell of $\{\top_C, a\}$ for $\otimes$. Then, the thesis is a straight consequence, since, by definition, for any $c \in C$, $\rho(c) = \bigwedge_{a\in A} \rho_a(c) = \bigwedge_{a\in A}(c \multimap a) \multimap a$. We first show that ρ_a is complete for $\otimes$. By Lemma 3.1, it is enough to show that for any $x, y \in C$, $\rho_a(\rho_a(x) \otimes y) = \rho_a(x \otimes y)$. We prove that $\rho_a(x) \otimes y \leq \rho_a(x \otimes y)$, since this implies $\rho_a(\rho_a(x) \otimes y) \leq \rho_a(x \otimes y)$ and the other inequality always holds. We have that $y \otimes (y \multimap (x \multimap a)) \leq x \multimap a$, and therefore $y \otimes (y \multimap (x \multimap a)) \otimes ((x \multimap a) \multimap a) \leq (x \multimap a) \otimes ((x \multimap a) \multimap a) \leq a$. As a consequence, we have the following inequalities:

$$\begin{aligned}
&y \otimes (y \multimap (x \multimap a)) \otimes ((x \multimap a) \multimap a) \leq a\\
&y \otimes ((x \otimes y) \multimap a) \otimes ((x \multimap a) \multimap a) \leq a\\
&y \otimes ((x \multimap a) \multimap a) \leq ((x \otimes y) \multimap a) \multimap a\\
&y \otimes \rho_a(x) \leq \rho_a(x \otimes y).
\end{aligned}$$

Thus, ρ_a is complete for $\otimes$. Then, in order to conclude, we prove that ρ_a is the greatest domain complete for $\otimes$ which contains the object a. Suppose, by contradiction, that there exists $\eta \in \mathrm{uco}(C)$ such that $\eta(a) = a$, η is complete for $\otimes$ and $\rho_a \sqsubset \eta$. Therefore, there exists $c \in C$ such that $\eta(c) > \rho_a(c)$, that is $\eta(c) > (c \multimap a) \multimap a$. Then, $\eta(c) \otimes (c \multimap a) \not\leq a$, otherwise we would get $\eta(c) \leq (c \multimap a) \multimap a$, which is a contradiction. As a consequence, $\eta(\eta(c) \otimes \eta(c \multimap a)) \not\leq a$. But, by completeness of η, $\eta(\eta(c) \otimes \eta(c \multimap a)) = \eta(c \otimes (c \multimap a)) \leq \eta(a) = a$, and this is the contradiction which closes the proof. □

A weaker form of completeness can be stated for binary operations in quantales. In this case we require that no loss of precision is accumulated by approximating one argument only.

Definition 3.8. Let $\langle C_{\leq}, \otimes\rangle$ be a quantale. An abstract domain $\rho \in \mathrm{uco}(C)$ is *weak-complete* for $\otimes$ when: if either $x \in \rho$ or $y \in \rho$ then $\rho(\rho(x) \otimes \rho(y)) = \rho(x \otimes y)$.

Weak-completeness is equivalent to require that ρ satisfies the following equalities:

$$\rho \circ \otimes \circ \langle \rho, \rho\rangle = \rho \circ \otimes \circ \langle \rho, id\rangle = \rho \circ \otimes \circ \langle id, \rho\rangle.$$

If in addition $\langle C_{\leq}, \otimes\rangle$ is commutative, this last condition is equivalent to the following single equation:

$$\rho \circ \otimes \circ \langle \rho, \rho\rangle = \rho \circ \otimes \circ \langle \rho, id\rangle. \tag{1}$$

It is worth pointing out that this is actually a weakening of standard completeness, i.e., any ρ complete for $\otimes$ is weak-complete for $\otimes$ as well. The converse clearly does not hold in general. The following result shows that weak-completeness can be equivalently defined in a n-ary formulation.

Lemma 3.9. *Let $\rho \in \mathrm{uco}(C)$. Then ρ is weak-complete iff for all $n > 1$ and $c_1, .., c_n \in C$, $\rho(\rho(c_1) \otimes \ldots \otimes \rho(c_{n-1}) \otimes \rho(c_n)) = \rho(\rho(c_1) \otimes \ldots \otimes \rho(c_{n-1}) \otimes c_n)$.*

Proof. The "if" direction is obvious. On the other hand, the proof is by induction on n.

$(n = 2)$ by definition.
$(n > 2)$ follows by Eq. (1):

$$\begin{array}{rcl}\rho(\rho(c_1) \otimes \rho(c_2) \otimes \ldots \otimes \rho(c_{n-1}) \otimes \rho(c_n)) & = & \text{[by Eq. (1)]} \\ \rho(\rho(c_1) \otimes \rho(\rho(c_2) \otimes \ldots \otimes \rho(c_{n-1}) \otimes \rho(c_n))) & = & \text{[by inductive hypothesis]} \\ \rho(\rho(c_1) \otimes \rho(\rho(c_2) \otimes \ldots \otimes \rho(c_{n-1}) \otimes c_n)) & = & \text{[by Eq. (1)]} \\ \rho(\rho(c_1) \otimes \rho(c_2) \otimes \ldots \otimes \rho(c_{n-1}) \otimes c_n). \;\Box & & \end{array}$$

Our interest in weak-complete abstract domains is related to the typical abstract computation in logic program analysis, where the current abstract store is unified against the abstraction of a program clause. This implies that if the abstract store is iteratively computed, e.g. in the semantics of recursive predicates, then weak-completeness for unification ensures that no loss of precision is accumulated by approximating the input store. For a given abstract domain $A \in \mathrm{uco}(C)$, we are therefore interested in characterizing the most abstract domain $\rho \in \mathrm{uco}(C)$ which is more concrete than A and satisfies Equation (1). This domain, when it exists, is called the *weak-complete shell* of A for $\otimes$. Weak-completeness problems can be solved by exploiting the same technique used for completeness, i.e., by resorting to a recursive abstract domain equation involving linear implication. The next result gives a recursive characterization of the solutions of Equation (1).

THEOREM 3.10. *Let $\langle C_{\leq}, \otimes\rangle$ be a unital, commutative quantale and $\rho \in \mathrm{uco}(C)$. The following are equivalent.*

(i) *ρ is weak-complete;*
(ii) *ρ is complete for $F_\rho = \{\lambda y.x \otimes y \mid x \in \rho\}$;*
(iii) *$\rho = \rho \sqcap (\rho \overset{\wedge}{\multimap} \rho)$.*

PROOF. $\rho \circ \otimes \circ \langle \rho, \rho\rangle = \rho \circ \otimes \circ \langle \rho, id\rangle$ holds iff for all $x \in \rho$ and $y \in C$ it holds $\rho(x \otimes \rho(y)) = \rho(x \otimes y)$, that is to say that ρ is complete for the set of unary functions $F_\rho = \{\lambda y.x \otimes y \mid x \in \rho\}$. By Theorem 2.1, ρ is complete for F_ρ iff $\rho \sqsubseteq \mathcal{R}_{F_\rho}(\rho)$. By Theorem 3.3, this is equivalent to say that $\rho \sqsubseteq \rho \overset{\wedge}{\multimap} \rho$, and therefore $\rho = \rho \sqcap (\rho \overset{\wedge}{\multimap} \rho)$. $\Box$

COROLLARY 3.11. *Let $\langle C_{\leq}, \otimes\rangle$ be a unital, commutative quantale and $A \in \mathrm{uco}(C)$. The weak-complete shell of A for $\otimes$ is $gfp(\lambda X.A \sqcap X \sqcap (X \overset{\wedge}{\multimap} X))$.*

PROOF. Since the operator $\lambda X.X \overset{\wedge}{\multimap} X$ is clearly monotone[2], from Theorem 3.10 it directly follows that the most abstract domain which includes A and is weak-complete for $\otimes$ is given by $gfp(\lambda X.A \sqcap X \sqcap (X \overset{\wedge}{\multimap} X))$. $\Box$

Thus, the weak-complete shell of a domain $A \in \mathrm{uco}(C)$ is exactly the greatest solution in $\mathrm{uco}(C)$ of the following recursive abstract domain equation:

$$X = A \sqcap X \sqcap (X \overset{\wedge}{\multimap} X). \tag{2}$$

As a consequence of Theorem 3.10, we also show that weak-complete closures preserve the structure of unital commutative quantales.

[2]It is worth noting that, even if $\mathcal{R}_{F_\rho} = \lambda\eta.\rho \overset{\wedge}{\multimap} \eta$ is co-additive for any $\rho \in \mathrm{uco}(C)$, this does not imply that the operator $\lambda\eta.\eta \overset{\wedge}{\multimap} \eta$ is co-additive as well. This is a consequence of the fact that the set of functions F_ρ for which we want to be complete, changes at each iteration.

Corollary 3.12. *If $\langle C_{\leq}, \otimes\rangle$ is a unital commutative quantale with unit* $\mathbf{1}$ *and* $\rho \in \mathrm{uco}(C)$ *is weak-complete then* $\langle \rho(C)_{\leq}, \rho\circ\otimes\rangle$ *is a unital commutative quantale where the unit is* $\rho(\mathbf{1})$.

Proof. The abstract operation $\rho\circ\otimes$ is clearly commutative by definition, and associative by weak-completeness. In fact, given $x, y, z \in \rho(C)$ it holds that $\rho(x \otimes \rho(y\otimes z)) = \rho(x\otimes(y\otimes z)) = \rho((x\otimes y)\otimes z) = \rho(\rho(x\otimes y)\otimes z)$. We know that the lub $\vee_\rho$ in the complete lattice $\rho(C)_{\leq}$ is as follows: for $Y \subseteq \rho(C)$, $\vee_\rho Y = \rho(\vee Y)$. Let $x \in \rho(C)$ and $Y \subseteq \rho(C)$: we have therefore to show that $\rho(x\otimes(\vee_\rho Y)) = \vee_\rho\{\rho(x\otimes y) \mid y \in Y\}$. In fact, by weak-completeness of ρ, we have that:

$$\begin{aligned}\rho(x \otimes (\vee_\rho Y)) &= \rho(\rho(x) \otimes \rho(\vee Y))\\ &= \rho(\rho(x) \otimes (\vee Y))\\ &= \rho(\vee_{y\in Y} \rho(x) \otimes y)\\ &= \vee_\rho\{\rho(x \otimes y) \mid y \in Y\}.\end{aligned}$$

It remains to check that $\rho(\mathbf{1})$ is the unit. By monotonicity, since $\mathbf{1} \leq \rho(\mathbf{1})$, it holds that $a = \mathbf{1} \otimes a \leq \rho(\mathbf{1}) \otimes a$. Moreover, since $\mathbf{1} \leq a \multimap a$, it follows that $\rho(\mathbf{1}) \leq \rho(a \multimap a) = a \multimap a$, where the last equality follows from the weak-completeness of ρ. $\square$

4. CHARACTERIZING CONDENSING ABSTRACT DOMAINS

The first attempt to give a formal setting to the notion of condensing procedures introduced by Jacobs and Langen [1992] was made by Marriott and Søndergaard [1993]. In that work, Marriott and Søndergaard [1993] consider abstract domains which are *downward-closed*: $X \in \mathrm{uco}(\wp(\mathtt{Sub})_{\subseteq})$ is downward-closed if for any $\Phi \in X$ and $\phi \in \Phi$, any instance of ϕ belongs to Φ. For a downward-closed domain X, it turns out that the glb of X, namely set intersection, actually plays the role of abstract unification. This allows to simplify the definition of condensing procedures, making condensing a property of the abstract semantics with respect to unification. Let F : *Program* $\rightarrow$ *Query* $\rightarrow \wp(\mathtt{Sub}) \rightarrow \wp(\mathtt{Sub})$ be a query-directed semantics, where $F_{P,Q}(\Phi)$ is the semantics of a query Q w.r.t. the program P for a given set Φ of initial substitutions for Q. A domain $X \in \mathrm{uco}(\wp(\mathtt{Sub}))$ is called *condensing* in [Marriott and Søndergaard 1993] for the query-directed semantics F if for any program P, query Q, and $\phi, \phi' \in X$, the following equation holds:

$$F^X_{P,Q}(\phi \wedge \phi') = \phi \wedge F^X_{P,Q}(\phi')$$

where $F^X_{P,Q} : X \rightarrow X$ is the abstract semantics induced by the abstraction X from the concrete semantics $F_{P,Q} : \wp(\mathtt{Sub}) \rightarrow \wp(\mathtt{Sub})$. As usual, the induced abstract semantics $F^X_{P,Q}$ is defined simply by replacing each basic operation on $\wp(\mathtt{Sub})$ involved in the definition of $F_{P,Q}$, namely union of sets of substitutions, unification and variable projection, with their corresponding best correct approximations on X. More recently, Giacobazzi and Scozzari [1998] gave a characterization of downward-closed condensing abstract domains as so-called Heyting-closed abstract domains. Complete Heyting algebras are particular quantales where the linear implication coincides precisely with the intuitionistic implication, i.e. the quantale multiplication is the meet operation. If $\wp^{\downarrow}(\mathtt{Sub})$ denotes the set of downward closed sets of substitutions then it is well known that $\langle \wp^{\downarrow}(\mathtt{Sub})_{\subseteq}, \cap\rangle$ is a quantale which is a complete Heyting algebra. Thus, the construction in [Giacobazzi and Scozzari 1998] is not applicable to generic non-downward closed abstractions of $\wp(\texttt{Sub})_\subseteq$.

Both [Marriott and Søndergaard 1993] and [Giacobazzi and Scozzari 1998] constructions can be generalized to arbitrary abstractions of a complete lattice. The following definition generalizes the notion of condensing semantics to an arbitrary semantic operator on any complete lattice.

Definition 4.1. Let $f : C \to C$ be a function on a complete lattice C and let $\otimes : C \times C \to C$ be a binary commutative and associative operation. We say that f is *condensing* for $\otimes$ if for all $a, b \in C$, $f(a \otimes b) = a \otimes f(b)$.

The following characterization of condensing semantics on unital quantales is important although easy.

Proposition 4.2. *Let $\langle C, \otimes\rangle$ be a unital commutative quantale with unit $\mathbf{1}$ and let $f : C \longrightarrow C$. Then, f is condensing for $\otimes$ if and only if for any $a \in C$, $f(a) = a \otimes f(\mathbf{1})$.*

Proof. One implication is obvious. Now assume that for any $a \in C$, it holds that $f(a) = a \otimes f(\mathbf{1})$. Let $b \in C$. Then, $f(a \otimes b) = (a \otimes b) \otimes f(\mathbf{1}) = a \otimes (b \otimes f(\mathbf{1})) = a \otimes f(b)$. $\square$

It is then natural to apply this simple idea to logic program semantics defined on the unital, commutative quantale $\langle \wp(\texttt{Sub})_\subseteq, \otimes\rangle$, where $\langle \wp(\texttt{Sub}), \subseteq\rangle$ is a complete lattice and $\otimes : \wp(\texttt{Sub}) \times \wp(\texttt{Sub}) \to \wp(\texttt{Sub})$ is the lifting of unification to sets of substitutions, namely:

$$X \otimes Y \overset{\text{def}}{=} \{mgu(\theta, \delta) \mid \theta \in X,\ \delta \in Y,\ \theta \text{ and } \delta \text{ unify}\}.$$

As observed in Section 2.5, $\langle \wp(\texttt{Sub})_\subseteq, \otimes\rangle$ is a unital, commutative quantale, where $\mathbf{1} = \{\epsilon\} \in \wp(\texttt{Sub})$ is the unit. Thus, the above simple result states that for a logic program semantics f on $\wp(\texttt{Sub})$, f is condensing for unification $\otimes$ if and only if the semantics $f(\Phi)$ from a given initial store $\Phi \in \wp(\texttt{Sub})$ can be fully reconstructed by unification from Φ and $f(\mathbf{1})$, namely the semantics from the most general store $\{\epsilon\}$.

In the following, we will slightly abuse the notation by applying the operation $\otimes$ also to single substitutions: in particular, when we will use the notation $\theta \otimes \delta$ we will mean that θ and δ unify.

4.1 Concrete query-directed semantics

We consider a core logic programming language computing substitutions. We assume programs as (finite) sets of procedure declarations, where each procedure is declared by exactly one clause of the form $p(\bar{x}) \leftarrow A$. This assumption simplifies technically our approach: With each predicate name p a single clause is allowed in a program. The non-deterministic choice in the definition of a predicate is encoded by allowing disjunction (denoted by $\bigvee$) in clause-bodies. The following grammar defines the syntax of programs P and agents (i.e., clause-bodies) A, where $\theta \in \texttt{Sub}$.

$$\begin{array}{rrcl} \textit{Program} & P & ::= & \varnothing \mid p(\bar{x}) \leftarrow A \mid P.P \\ \textit{Agent} & A & ::= & \theta \mid p(\bar{x}) \mid A \otimes A \mid \bigvee_{i=1}^{n} A_i \end{array}$$

The standard semantics of our language is defined by the query-directed function $S_A : \texttt{Sub} \to \wp(\texttt{Sub})$, where A is any agent, presented as a set of recursive equations inductively defined on program's syntax specifying the forward SLD operational semantics. Let P be a program and $\sigma \in \texttt{Sub}$ (in order to simplify the notation, here and in the following we omit the subscript P denoting the program):

$$\begin{array}{lll} S_\theta(\sigma) & = \theta \otimes \sigma & \\ S_{A_1 \otimes A_2}(\sigma) & = S_{A_1}(\sigma) \otimes S_{A_2}(\sigma) & \\ S_{\bigvee_{i=1}^n A_i}(\sigma) & = \bigcup_{i=1}^n S_{A_i}(\sigma) & \\ S_{p(\bar{x})}(\sigma) & = \pi_V(S_A(\sigma)) & \begin{array}{l}\text{where } p(\bar{x}) \leftarrow A \lll_V P \\ \text{and } V = \bar{x} \cup dom(\sigma).\end{array} \end{array} \quad (3)$$

In this definition $\lll_V$ selects the renamed clause $p(\bar{x}) \leftarrow A$ from P where variables in $vars(A) \setminus \bar{x}$ are renamed apart from $V = \bar{x} \cup dom(\sigma)$. It is clear that condensation for this semantics S w.r.t. unification $\otimes$ corresponds to ask that the well known lifting lemma for SLD resolution holds [Apt 1990, Lemma 3.19]. This is clearly the case being our semantics an equational presentation of the standard SLD operational semantics of logic programs.

As usual in abstract interpretation [Cousot and Cousot 1977], the abstract objects in an abstract domain represent sets of substitutions. This means that the concrete standard semantics (3) needs to be lifted to sets of substitutions, namely to the so-called collecting semantics. It turns out that this is not a straightforward extension of the above standard semantics (3), since we obviously need that the concrete collecting semantics, as well as the standard one, is condensing for unification. The forward collecting semantics is defined by the following query-directed function $\mathcal{S}_A : \wp(\texttt{Sub}) \to \wp(\texttt{Sub})$ which is inductively defined on the syntax of A for any $\Phi \in \wp(\texttt{Sub})$ as follows (P denotes the program):

$$\begin{array}{lll} \mathcal{S}_\theta(\Phi) & = \theta \otimes \Phi & \\ \mathcal{S}_{A_1 \otimes A_2}(\Phi) & = \mathcal{S}_{A_1}(\Phi) \otimes \mathcal{S}_{A_2}(\Phi \multimap \Phi) & \\ \mathcal{S}_{\bigvee_{i=1}^n A_i}(\Phi) & = \bigcup_{i=1}^n \mathcal{S}_{A_i}(\Phi) & \\ \mathcal{S}_{p(\bar{x})}(\Phi) & = \pi_V(\mathcal{S}_A(\Phi)) & \begin{array}{l}\text{where } p(\bar{x}) \leftarrow A \lll_V P \\ \text{and } V = \bar{x} \cup dom(\Phi).\end{array} \end{array} \quad (4)$$

The forward concrete semantics of a logic program P for the query $p(\bar{x})$ and initial call set Φ is therefore $\mathcal{S}_{p(\bar{x})}(\Phi)$. The key point in the above collecting semantics definition is the linear object $\Phi \multimap \Phi$ in the second equation. This construction is motivated by the fact that the simple lifting of the second equation of the standard semantics to the equation $\mathcal{S}_{A_1 \otimes A_2}(\Phi) = \mathcal{S}_{A_1}(\Phi) \otimes \mathcal{S}_{A_2}(\Phi)$ would lead us to a non-condensing semantics w.r.t. unification. In fact, consider for instance the following trivial program $\{p(\bar{x}) \leftarrow \theta_1 \otimes \theta_2\}$. In this case, for any $\Phi \in \wp(\texttt{Sub})$ we would have:

$$\begin{array}{rl} \mathcal{S}_{p(\bar{x})}(\Phi) & = \mathcal{S}_{\theta_1}(\Phi) \otimes \mathcal{S}_{\theta_2}(\Phi) \\ & = (\theta_1 \otimes \Phi) \otimes (\theta_2 \otimes \Phi) \\ & = \theta_1 \otimes \theta_2 \otimes \Phi \otimes \Phi \end{array}$$

Clearly, if Φ contains more than one substitution then $\theta_1 \otimes \theta_2 \otimes \Phi \otimes \Phi$ may well contain substitutions which are not computed by SLD resolution, namely those substitutions coming from the combination of unifications in $\Phi \otimes \Phi$. This would make the collecting semantics noncondensing: $\mathcal{S}_{p(\bar{x})}(\Phi) \neq \Phi \otimes \mathcal{S}_{p(\bar{x})}(\{\epsilon\}) = \theta_1 \otimes \theta_2 \otimes \Phi$. A number of semantic definitions might solve this problem, e.g. by defining $\mathcal{S}_{A_1 \otimes A_2}(\Phi) = \mathcal{S}_{A_1}(\Phi) \otimes \mathcal{S}_{A_2}(\{\epsilon\})$, because $\Phi \otimes \{\epsilon\} = \Phi$. However, the largest set

$X \in \wp(\mathtt{Sub})$ such that $\Phi \otimes X = \Phi$ is, by definition of linear implication, exactly $\Phi \multimap \Phi$. The resulting definition (4) is an intensional collecting and condensing semantics for computed answer substitutions (this is simple to check and we leave it to the reader) presented in a recursive equational form.

As a simple example, consider $P = p(x) \leftarrow \{x/0\} \vee (\{x/f(k), y/k\} \otimes p(y))$. Then,

$$\begin{aligned}
\mathcal{S}_{p(x)}(\{\epsilon\}) &= \pi_x(\mathcal{S}_{\{x/0\}\vee(\{x/f(k),y/k\}\otimes p(y))}(\{\epsilon\})) \\
&= \pi_x(\{x/0\} \cup \mathcal{S}_{\{x/f(k),y/k\}\otimes p(y)}(\{\epsilon\})) \\
&= \pi_x(\{x/0\} \cup (\mathcal{S}_{\{x/f(k),y/k\}}(\{\epsilon\}) \otimes \mathcal{S}_{p(y)}(\{\epsilon\}))) \\
&= \pi_x(\{x/0\} \cup (\{x/f(k), y/k\} \otimes \mathcal{S}_{p(y)}(\{\epsilon\}))).
\end{aligned}$$

Obviously, the most abstract solution for $\mathcal{S}_{p(x)}(\{\epsilon\})$ is $\{\{x/f^i(0)\} \in \mathtt{Sub} \mid i \geq 0\}$.

4.2 Induced abstract semantics

Following a standard scheme (as e.g. in [Marriott and Søndergaard 1993; Marriot et al. 1994]), for any abstract domain $\rho \in \mathrm{uco}(\wp(\mathtt{Sub}))$, the above collecting semantics definition $\mathcal{S}$ induces a corresponding abstract semantics $\mathcal{S}^\rho$ on ρ obtained from $\mathcal{S}$ by replacing the concrete operations involved in the definition of $\mathcal{S}$, namely set-union, unification $\otimes$, projection π and linear implication $\multimap$, with their corresponding best correct approximations on ρ.

Thus, for any agent A and abstract set of substitutions $\Phi \in \rho$, the induced abstract semantics $\mathcal{S}^\rho_A : \rho \to \rho$ is inductively defined as follows:

$$\begin{aligned}
\mathcal{S}^\rho_\theta(\Phi) &= \rho(\theta \otimes \Phi) \\
\mathcal{S}^\rho_{A_1\otimes A_2}(\Phi) &= \rho(\mathcal{S}^\rho_{A_1}(\Phi) \otimes \mathcal{S}^\rho_{A_2}(\rho(\Phi \multimap \Phi))) \\
\mathcal{S}^\rho_{\bigvee_{i=1}^n A_i}(\Phi) &= \rho(\textstyle\bigcup_{i=1}^n \mathcal{S}^\rho_{A_i}(\Phi)) \\
\mathcal{S}^\rho_{p(\bar{x})}(\Phi) &= \rho(\pi_V(\mathcal{S}^\rho_A(\Phi))) \qquad \text{where } p(\bar{x}) \leftarrow A \lll_V P \text{ and } V = \bar{x} \cup \mathit{dom}(\Phi).
\end{aligned}$$

It is important to remark that, in each equation above, $\mathcal{S}^\rho$ actually is recursively defined as composition of the bca's induced by the abstraction ρ of the basic concrete operations $\bigcup$, $\otimes$, π and $\multimap$. The abstract query-directed semantics of a query $p(\bar{x})$ in a program P with abstract initial call $\Phi \in \rho$ is therefore given by $\mathcal{S}^\rho_{p(\bar{x})}(\Phi)$.

4.3 Condensing abstract domains

Since the induced abstract semantics $\mathcal{S}^\rho$ depends on the abstract domain ρ only, condensation for the abstract semantics $\mathcal{S}^\rho$ becomes an abstract domain property.

Definition 4.3. An abstract domain $\rho \in \mathrm{uco}(\wp(\mathtt{Sub}))$ is condensing when, for any program P and predicate $p(\bar{x})$, the induced abstract semantics $\mathcal{S}^\rho_{p(\bar{x})}$ is condensing w.r.t. $\otimes^{\mathrm{bca}_\rho} = \rho \circ \otimes$, i.e., for all $\Theta, \Phi \in \rho$, $\mathcal{S}^\rho_{p(\bar{x})}(\rho(\Theta \otimes \Phi)) = \rho(\Theta \otimes \mathcal{S}^\rho_{p(\bar{x})}(\Phi))$.

The following result is then a straight consequence of Corollary 3.12 and Proposition 4.2.

Proposition 4.4. *Let $\rho \in \mathrm{uco}(\wp(Sub))$ be weak-complete and let $\mathbf{1}^\rho = \rho(\{\epsilon\})$. Then, ρ is condensing iff for any $\Psi \in \rho$, $\mathcal{S}^\rho_{p(\bar{x})}(\Psi) = \rho(\Psi \otimes \mathcal{S}^\rho_{p(\bar{x})}(\mathbf{1}^\rho))$.*

This means that the abstract semantics $\mathcal{S}^\rho_{p(\bar{x})}(\Psi)$ for the predicate $p(\bar{x})$ with initial abstract store Ψ can be equivalently obtained by unifying the substitutions in

Ψ with the semantics $\mathcal{S}^{\rho}_{p(\bar{x})}(\rho(\{\epsilon\}))$ of the same predicate $p(\bar{x})$ in the most general abstract store $\rho(\{\epsilon\})$. This corresponds to the typical way an analysis of a query in an initial state is derived from a goal-independent condensing analysis: the computations which do not satisfy the given initial state are filtered out by unification (cf. [Barbuti et al. 1993]).

Example 4.5. Two variables $x, y \in \mathcal{V}$ are *independent* for a substitution θ when $vars(\theta(x)) \cap vars(\theta(y)) = \varnothing$. Let I_{xy} be the set of substitutions θ such that x and y are independent for θ:

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

Let us consider a finite set of variables of interest $VI \subseteq_{\mathit{fin}} \mathcal{V}$, which are the relevant variables. The basic abstract domain PSh_{VI} for detecting pair-sharing, that is pairs of variables which may share a common variable, is given by the most abstract domain which contains all the objects I_{xy}, for any $x, y \in VI$, with $x \neq y$:

$$\mathsf{PSh}_{VI} \overset{\text{def}}{=} \mathcal{M}(\{I_{xy} \mid x, y \in VI, x \neq y\}).$$

Thus, the closure $\rho \in \mathrm{uco}(\wp(\mathtt{Sub}))$ corresponding to the domain PSh_{VI} is defined as follows: For all $\Theta \in \wp(\mathtt{Sub})$,

$$\rho(\Theta) \overset{\text{def}}{=} \bigcap \{I_{xy} \mid x, y \in VI, x \neq y, \forall \theta \in \Theta.\ vars(\theta(x)) \cap vars(\theta(y)) = \varnothing\}.$$

Let P be the following trivial program:

$$p(X, Y) \leftarrow \{X/a\} \vee \{Y/a\}$$

where a is any ground term. For $VI = \{X, Y\}$ we have that $\mathsf{PSh}_{VI} = \{\top, I_{XY}\}$, where $\top$ stands for $\mathtt{Sub}$. In this simple case, note that the bca $\rho \circ \otimes$ on PSh_{VI} of unification is trivially defined as follows: For all $A, B \in \mathsf{PSh}_{VI}$, $\rho(A \otimes B) = \top$: this is a consequence of the fact that $\{X/W, Z/W\} \otimes \{Y/W, Z/W\} = \{X/W, Y/W, Z/W\}$ which does not belong to I_{XY} while both $\{X/W, Z/W\}$ and $\{Y/W, Z/W\}$ do. If we compute $\mathcal{S}^{\rho}_{p(X,Y)}$ with initial query $\Phi = \top$ we obtain:

$$\begin{aligned}
\mathcal{S}^{\rho}_{p(X,Y)}(\top) &= \rho(\pi_{\mathcal{V}}(\mathcal{S}^{\rho}_{\{X/a\} \vee \{Y/a\}}(\top))) \\
&= \rho(\pi_{\mathcal{V}}(\rho(\mathcal{S}^{\rho}_{\{X/a\}}(\top) \cup \mathcal{S}^{\rho}_{\{Y/a\}}(\top)))) \\
&= \rho(\pi_{\mathcal{V}}(\rho(\rho(\{X/a\} \otimes \top) \cup \rho(\{Y/a\} \otimes \top)))) \\
&= \rho(\pi_{\mathcal{V}}(\rho(I_{XY} \cup I_{XY}))) \\
&= \rho(\pi_{\mathcal{V}}(\rho(I_{XY}))) \\
&= I_{XY}.
\end{aligned}$$

Therefore, since $\rho(I_{XY} \otimes \top) = \top$, we have that:

$$\mathcal{S}^{\rho}_{p(X,Y)}(\rho(I_{XY} \otimes \top)) = \mathcal{S}^{\rho}_{p(X,Y)}(\top) = I_{XY}$$

$$\rho(I_{XY} \otimes \mathcal{S}^{\rho}_{p(X,Y)}(\top)) = \rho(I_{XY} \otimes I_{XY}) = \top.$$

As a consequence, the domain ρ is not condensing. Note that the variable projection does not act, i.e. it acts as the identity, in this example. □

4.4 Condensing domains in logical form

Basically, the abstract semantics of a predicate p in a program P is obtained by iterating the abstract unification of a concrete substitution θ in the body of some clause in P against the result of the previous computation, which is the current abstract substitution. Indeed, the fixpoint of this iterated procedure gives the semantics of the predicate p. When an abstract domain is condensing, i.e. the corresponding induced abstract semantics is condensing for the abstract unification, the abstract semantics of a specific query $p(\bar{x})\theta$ can be obtained with no loss of precision by abstract unification from the abstract semantics of $p(\bar{x})$, i.e., from the semantics of the most general query. This means that it is possible to propagate the information carried on by the semantics of a most general query back to the semantics of any query Q, without recomputing the semantics of that query Q, but simply by a unification operation. This corresponds to ask that a lifting-lemma like property holds for abstract computations as well as it holds for concrete ones. It should be clear that completeness for unification is sufficient to ensure condensation, since all the intermediate abstractions can be removed from the fixpoint computation of the abstract semantics of each predicate (see also [Schachte 2003]). However, a weaker form of completeness is enough, by exploiting the observation that one of the two arguments of unification is always an abstract object. In fact, it turns out that still no loss of precision is accumulated in the abstract computation when at least one argument of unification is an abstract object.

In the following we prove that any weak-complete abstract domain A, namely a solution of the recursive domain equation $X = X \sqcap (X \overset{\wedge}{\multimap} X)$ (cf. Theorem 3.10), is condensing and, under additional nonrestrictive hypotheses, condensing abstract domains are all and only the weak-complete abstract domains. This result shows a precise connection between completeness in abstract interpretation and the property of being condensing and, more importantly, it gives computational relevance in static program analysis to the notion of weak-completeness. We say that an abstract domain $\rho \in \text{uco}(\wp(\texttt{Sub}))$ is *compatible* with π when the abstract projection distributes over the abstract unification, according to the following definition.

Definition 4.6. An abstract domain $\rho \in \text{uco}(\wp(\texttt{Sub}))$ is *compatible* with π if for all $\Theta \in \wp(\texttt{Sub})$ and $W \subseteq \mathcal{V}$, if $\pi_W(\Theta) = \Theta$ then for all $a \in \rho$ it holds that $\rho(\pi_W(\rho(\Theta \otimes a))) = \rho(\Theta \otimes \rho(\pi_W(a)))$.

Compatibility with π allows us to get rid of projection when reasoning about condensation. Observe that, when ρ is the identity closure, the above definition boils down to a standard property of projection, namely that for any $\Theta, \Phi \in \wp(\texttt{Sub})$, $\pi_W(\Theta \otimes \Phi) = \Theta \otimes \pi_W(\Phi)$ whenever $\pi_W(\Theta) = \Theta$. Alternatively, one might assume that ρ is complete for any π_W (where $W \subseteq \mathcal{V}$), and therefore this completeness property can be achieved by exploiting Theorem 2.1 [Giacobazzi et al. 2000].

THEOREM 4.7. *If $\rho \in \text{uco}(\wp(\texttt{Sub}))$ is weak-complete and compatible with π then ρ is condensing.*

PROOF. For any given program P and agent A, according to Proposition 4.2, it is sufficient to prove that for any $\Psi \in \rho$, $\mathcal{S}^{\rho}_{A}(\Psi) = \rho(\Psi \otimes \mathcal{S}^{\rho}_{A}(\mathbf{1}^{\rho}))$, where $\mathbf{1}^{\rho} = \rho(\{\epsilon\})$. This is proved by induction on the structure of A.

—Base case $\mathcal{S}^\rho_\theta$. Since ρ is weak-complete we have that $\mathcal{S}^\rho_\theta(\Psi) = \rho(\theta \otimes \Psi) = \rho(\rho(\Psi) \otimes (\theta \otimes \mathbf{1}^\rho)) = \rho(\Psi \otimes \rho(\theta \otimes \mathbf{1}^\rho)) = \rho(\Psi \otimes \mathcal{S}^\rho_\theta(\mathbf{1}^\rho))$.

—Inductive case $\mathcal{S}^\rho_{A_1 \otimes A_2}$. By weak-completeness, Lemma 3.9 and inductive hypothesis we have that

$$\begin{array}{rcl}
\mathcal{S}^\rho_{A_1\otimes A_2}(\Psi) & = & \\
\rho(\mathcal{S}^\rho_{A_1}(\Psi) \otimes \mathcal{S}^\rho_{A_2}(\rho(\Psi \multimap \Psi))) & = & [\text{ by Th. 3.10, } \Psi \multimap \Psi \in \rho \text{ }] \\
\rho(\mathcal{S}^\rho_{A_1}(\Psi) \otimes \mathcal{S}^\rho_{A_2}(\Psi \multimap \Psi)) & = & [\text{ by inductive hypothesis }] \\
\rho(\rho(\Psi \otimes \mathcal{S}^\rho_{A_1}(\mathbf{1}^\rho)) \otimes \rho((\Psi \multimap \Psi) \otimes \mathcal{S}^\rho_{A_2}(\mathbf{1}^\rho))) & = & [\text{ by weak-completeness }] \\
\rho(\Psi \otimes \mathcal{S}^\rho_{A_1}(\mathbf{1}^\rho) \otimes \rho((\Psi \multimap \Psi) \otimes \mathcal{S}^\rho_{A_2}(\mathbf{1}^\rho))) & = & [\text{ since } \Psi \in \rho \text{ }] \\
\rho(\rho(\Psi) \otimes \mathcal{S}^\rho_{A_1}(\mathbf{1}^\rho) \otimes \rho((\Psi \multimap \Psi) \otimes \mathcal{S}^\rho_{A_2}(\mathbf{1}^\rho))) & = & [\text{ by weak-completeness }] \\
\rho(\rho(\Psi) \otimes \rho(\mathcal{S}^\rho_{A_1}(\mathbf{1}^\rho) \otimes \rho((\Psi \multimap \Psi) \otimes \mathcal{S}^\rho_{A_2}(\mathbf{1}^\rho))) & = & [\text{ since } \mathcal{S}^\rho_{A_1}(\mathbf{1}^\rho) \in \rho \text{ }] \\
\rho(\rho(\Psi) \otimes \rho(\rho(\mathcal{S}^\rho_{A_1}(\mathbf{1}^\rho)) \otimes \rho((\Psi \multimap \Psi) \otimes \mathcal{S}^\rho_{A_2}(\mathbf{1}^\rho))) & = & [\text{ by weak-completeness }] \\
\rho(\rho(\Psi) \otimes \rho(\rho(\mathcal{S}^\rho_{A_1}(\mathbf{1}^\rho)) \otimes (\Psi \multimap \Psi) \otimes \mathcal{S}^\rho_{A_2}(\mathbf{1}^\rho))) & = & [\text{ by weak-completeness }] \\
\rho(\rho(\Psi) \otimes \rho(\mathcal{S}^\rho_{A_1}(\mathbf{1}^\rho)) \otimes (\Psi \multimap \Psi) \otimes \mathcal{S}^\rho_{A_2}(\mathbf{1}^\rho)) & = & [\text{ since } \Psi, \mathcal{S}^\rho_{A_1}(\mathbf{1}^\rho) \in \rho \text{ }] \\
\rho(\Psi \otimes \mathcal{S}^\rho_{A_1}(\mathbf{1}^\rho) \otimes (\Psi \multimap \Psi) \otimes \mathcal{S}^\rho_{A_2}(\mathbf{1}^\rho)) & = & [\text{ since } \Psi \otimes (\Psi \multimap \Psi) = \Psi \text{ }] \\
\rho(\Psi \otimes \mathcal{S}^\rho_{A_1}(\mathbf{1}^\rho) \otimes \mathcal{S}^\rho_{A_2}(\mathbf{1}^\rho)) & = & [\text{ since } \mathbf{1}^\rho \multimap \mathbf{1}^\rho = \mathbf{1}^\rho \text{ }] \\
\rho(\Psi \otimes \mathcal{S}^\rho_{A_1}(\mathbf{1}^\rho) \otimes \mathcal{S}^\rho_{A_2}(\mathbf{1}^\rho \multimap \mathbf{1}^\rho)) & = & [\text{ since } \Psi \in \rho \text{ }] \\
\rho(\rho(\Psi) \otimes \mathcal{S}^\rho_{A_1}(\mathbf{1}^\rho) \otimes \mathcal{S}^\rho_{A_2}(\mathbf{1}^\rho \multimap \mathbf{1}^\rho)) & = & [\text{ by weak-completeness }] \\
\rho(\rho(\Psi) \otimes \rho(\mathcal{S}^\rho_{A_1}(\mathbf{1}^\rho) \otimes \mathcal{S}^\rho_{A_2}(\mathbf{1}^\rho \multimap \mathbf{1}^\rho))) & = & [\text{ by definition }] \\
\rho(\rho(\Psi) \otimes \mathcal{S}^\rho_{A_1\otimes A_2}(\mathbf{1}^\rho)) & = & [\text{ since } \Psi \in \rho \text{ }] \\
\rho(\Psi \otimes \mathcal{S}^\rho_{A_1\otimes A_2}(\mathbf{1}^\rho)). & &
\end{array}$$

—Inductive case $\mathcal{S}^\rho_{\bigvee_{i=1}^n A_i}$. Recall that $\otimes$ is additive and that $\rho(\bigcup_{i\in I} \rho(X_i)) = \rho(\bigcup_{i\in I} X_i)$. Therefore, by inductive hypothesis and because ρ is weak-complete we have that

$$\begin{array}{rcl}
\mathcal{S}^\rho_{\bigvee_{i=1}^n A_i}(\Psi) & = & \rho(\bigcup_{i=1}^n \mathcal{S}^\rho_{A_i}(\Psi)) \\
& = & \rho(\bigcup_{i=1}^n \rho(\Psi \otimes \mathcal{S}^\rho_{A_i}(\mathbf{1}^\rho))) \\
& = & \rho(\bigcup_{i=1}^n \Psi \otimes \mathcal{S}^\rho_{A_i}(\mathbf{1}^\rho)) \\
& = & \rho(\Psi \otimes \bigcup_{i=1}^n \mathcal{S}^\rho_{A_i}(\mathbf{1}^\rho)) \\
& = & \rho(\Psi \otimes \bigcup_{i=1}^n \rho(\mathcal{S}^\rho_{A_i}(\mathbf{1}^\rho))) \\
& = & \rho(\Psi \otimes \rho(\bigcup_{i=1}^n \rho(\mathcal{S}^\rho_{A_i}(\mathbf{1}^\rho)))) \\
& = & \rho(\Psi \otimes \rho(\bigcup_{i=1}^n \mathcal{S}^\rho_{A_i}(\mathbf{1}^\rho))) \\
& = & \rho(\Psi \otimes \mathcal{S}^\rho_{\bigvee_{i=1}^n A_i}(\mathbf{1}^\rho)).
\end{array}$$

—Inductive case $\mathcal{S}^\rho_{p(\bar{x})}$. Let $p(\bar{x}) \leftarrow A \lll_V P$ where $V = \bar{x} \cup \mathit{dom}(\Psi)$. Note that $\pi_V(\Psi) = \Psi$ since $\mathit{dom}(\Psi) \subseteq V$. Thus, by inductive hypothesis, weak-completeness and compatibility woth π we have that

$$\begin{array}{rcl}
\mathcal{S}^\rho_{p(\bar{x})}(\Psi) & = & \rho(\pi_V(\mathcal{S}^\rho_A(\Psi))) \\
& = & \rho(\pi_V(\rho(\Psi \otimes \mathcal{S}^\rho_A(\mathbf{1}^\rho)))) \\
& = & \rho(\Psi \otimes \rho(\pi_V(\mathcal{S}^\rho_A(\mathbf{1}^\rho)))) \\
& = & \rho(\Psi \otimes \mathcal{S}^\rho_{p(\bar{x})}(\mathbf{1}^\rho)). \quad \square
\end{array}$$

The converse of Theorem 4.7 states that condensing domains are weak-complete under some additional hypotheses. Let us call an abstract domain $\rho \in \mathrm{uco}(\wp(\mathtt{Sub}))$ *finitely generated* when for any $\Phi, \Psi \in \rho$ there exists $\Theta \subseteq_{\mathit{fin}} \Phi \multimap \Psi$ such that $\rho(\Theta) = \rho(\Phi \multimap \Psi)$.

THEOREM 4.8. *Let $\rho \in \mathrm{uco}(\wp(\mathtt{Sub}))$ be compatible with π and finitely generated. If ρ is condensing then ρ is weak-complete.*

PROOF. Let ρ be condensing. In order to prove that ρ is weak-complete, by Theorem 3.10, we prove that $\rho \sqsubseteq \rho \overset{\wedge}{\multimap} \rho$, i.e., that for any $\Phi, \Psi \in \rho$, $\rho(\Phi \multimap \Psi) = \Phi \multimap \Psi$. Suppose, by contradiction, that there exist some $\Phi, \Psi \in \rho$ such that $\Phi \multimap \Psi \subsetneq \rho(\Phi \multimap \Psi)$. We distinguish two cases. Let us first assume that $\Phi \otimes \mathbf{1}^\rho \neq \Phi$, where $\mathbf{1}^\rho = \rho(\{\epsilon\})$. Since $\{\epsilon\} \subseteq \mathbf{1}^\rho$ it follows that $\Phi \otimes \mathbf{1}^\rho \supsetneq \Phi$. Therefore, if we consider the trivial program $P = \{p \leftarrow \epsilon\}$ we have that $\mathcal{S}^\rho_p(\Phi) = \rho(\pi_{dom(\Phi)}\rho(\epsilon \otimes \Phi)) = \rho(\pi_{dom(\Phi)}\Phi) = \rho(\Phi) = \Phi$. On the other hand, $\mathcal{S}^\rho_p(\mathbf{1}^\rho) = \rho(\pi_{dom(\mathbf{1}^\rho)}\rho(\epsilon \otimes \mathbf{1}^\rho)) = \rho(\pi_{dom(\mathbf{1}^\rho)}\mathbf{1}^\rho) = \rho(\mathbf{1}^\rho) = \mathbf{1}^\rho$, and thus $\rho(\Phi \otimes \mathcal{S}^\rho_p(\mathbf{1}^\rho)) = \rho(\Phi \otimes \mathbf{1}^\rho) \supseteq \Phi \otimes \mathbf{1}^\rho \supsetneq \Phi = \mathcal{S}^\rho_p(\Phi)$, a contradiction, since ρ is condensing.
Now assume that $\Phi \otimes \mathbf{1}^\rho = \Phi$. Let us observe that $\Phi \otimes \rho(\Phi \multimap \Psi) \not\subseteq \Psi$, otherwise we would have the contradiction $\rho(\Phi \multimap \Psi) \subseteq \Phi \multimap \Psi$, and therefore $\rho(\Phi \otimes \rho(\Phi \multimap \Psi)) \not\subseteq \Psi$. By hypothesis there exists $\Theta \subseteq_{\mathit{fin}} \Phi \multimap \Psi$ such that $\rho(\Theta) = \rho(\Phi \multimap \Psi)$. Let $\bar{x} = \cup_{\theta \in \Theta} dom(\theta)$ (this is a finite set of variables, since Θ is finite; the order does not matter) and consider the program $P = \{p(\bar{x}) \leftarrow \bigvee_{\theta \in \Theta} \theta\}$. Let $V = \bar{x} \cup dom(\mathbf{1}^\rho)$ and $W = \bar{x} \cup dom(\Phi)$. Then, we have that:

$$\begin{array}{rcl}
\rho(\Phi \otimes \mathcal{S}^\rho_{p(\bar{x})}(\mathbf{1}^\rho)) & = & [\text{ by definition }] \\
\rho(\Phi \otimes \rho(\pi_V \mathcal{S}^\rho_{\bigvee_{\theta \in \Theta}\theta}(\mathbf{1}^\rho))) & = & [\text{ by definition }] \\
\rho(\Phi \otimes \rho(\pi_V \rho(\bigcup_{\theta \in \Theta} \mathcal{S}^\rho_\theta(\mathbf{1}^\rho)))) & = & [\text{ by definition }] \\
\rho(\Phi \otimes \rho(\pi_V \rho(\bigcup_{\theta \in \Theta} \rho(\theta \otimes \mathbf{1}^\rho)))) & = & [\text{ since } \rho \text{ is a closure }] \\
\rho(\Phi \otimes \rho(\pi_V \rho(\bigcup_{\theta \in \Theta} \theta \otimes \mathbf{1}^\rho))) & = & \\
\rho(\Phi \otimes \rho(\pi_V (\rho(\Theta \otimes \mathbf{1}^\rho)))) & \supseteq & [\text{ by monotonicity of } \pi \text{ }] \\
\rho(\Phi \otimes \rho(\pi_V (\Theta \otimes \mathbf{1}^\rho)))) & \supseteq & [\text{ since } \Theta \otimes \mathbf{1}^\rho \supseteq \Theta \text{ }] \\
\rho(\Phi \otimes \rho(\pi_V \Theta)) & = & [\text{ since } \pi_V \Theta = \Theta \text{ }] \\
\rho(\Phi \otimes \rho(\Theta)) & = & [\text{ since } \rho(\Theta) = \rho(\Phi \multimap \Psi) \text{ }] \\
\rho(\Phi \otimes \rho(\Phi \multimap \Psi)) & \not\subseteq & \Psi.
\end{array}$$

Moreover we also have that:

$$\begin{array}{rcl}
\mathcal{S}^\rho_{p(\bar{x})}(\rho(\Phi \otimes \mathbf{1}^\rho)) & = & [\text{ since } \Phi \otimes \mathbf{1}^\rho = \Phi \in \rho \text{ }] \\
\mathcal{S}^\rho_{p(\bar{x})}(\Phi) & = & [\text{ by definition }] \\
\rho(\pi_W \mathcal{S}^\rho_{\bigvee_{\theta \in \Theta}\theta}(\Phi)) & = & [\text{ by definition }] \\
\rho(\pi_W \rho(\bigcup_{\theta \in \Theta} \mathcal{S}^\rho_\theta(\Phi))) & = & [\text{ by definition }] \\
\rho(\pi_W \rho(\bigcup_{\theta \in \Theta} \rho(\theta \otimes \Phi))) & = & [\text{ by properties of } \rho \text{ }] \\
\rho(\pi_W \rho(\bigcup_{\theta \in \Theta} \theta \otimes \Phi)) & = & \\
\rho(\pi_W \rho(\Theta \otimes \Phi)) & = & [\text{ by compatibility with } \pi \text{ }] \\
\rho(\Theta \otimes \rho(\pi_W(\Phi))) & = & [\text{ since } W = dom(\Phi) \cup dom(\Theta) \text{ }] \\
\rho(\Theta \otimes \Phi) & = & \\
\rho(\Phi \otimes \Theta) & \subseteq & [\text{ since } \Theta \subseteq \Phi \multimap \Psi \text{ }] \\
\rho(\Phi \otimes (\Phi \multimap \Psi)) & \subseteq & [\text{ since } \Phi \otimes (\Phi \multimap \Psi) \subseteq \Phi \text{ }] \\
\rho(\Psi) & = & [\text{ since } \Psi \in \rho \text{ }] \\
\Psi. & &
\end{array}$$

Therefore, we would have that $\mathcal{S}^\rho_{p(\bar{x})}(\Phi) \neq \rho(\Phi \otimes \mathcal{S}^\rho_{p(\bar{x})}(\mathbf{1}^\rho))$, that is ρ would not be condensing, which is a contradiction. $\square$

The above hypothesis of being finitely generated for an abstract domain is not too restrictive — it is satisfied by most domains used in logic program analysis. For an abstract domain ρ, this condition states that, for any $x, y \in \rho$, the abstraction in ρ of any implicational object $x \multimap y$ can be always obtained as abstraction in ρ of a finite subset of $x \multimap y$. This allows us, in Theorem 4.8, to construct, for any such implicational object $x \multimap y$, a (finite) program whose abstract semantics is given by $\rho(x \multimap y)$. Thus, the key point in order to prove the converse of Theorem 4.7 lies in the fact that any implicational object in $\rho \overset{\wedge}{\multimap} \rho$ has to be the semantics of some well-defined program.

To conclude, the following characterization of condensing domains is therefore an immediate consequence of Theorems 3.10, 4.7 and 4.8.

Corollary 4.9. *Let $\rho \in \mathrm{uco}(\wp(\mathtt{Sub}))$ be compatible with π and finitely generated. Then, ρ is condensing if and only if $\rho = \rho \sqcap (\rho \overset{\wedge}{\multimap} \rho)$.*

5. FREENESS AND INDEPENDENCE ANALYSIS

We say that a variable x is free in a substitution θ if $\theta(x)$ is a variable whenever $x \in dom(\theta)$. As already said, two variables x and y are independent in a substitution θ if $vars(\theta(x)) \cap vars(\theta(y)) = \varnothing$, whenever $x, y \in dom(\theta)$. Note that these notions of freeness and independence are well-defined, namely if $\theta \sim \sigma$ then x is free in θ iff x is free in σ, and x, y are independent in θ iff x, y are independent in σ. We are interested in a compound property, namely that a given variable is free and independent from a fixed set of variables. Given $V \subseteq \mathcal{V}$, we denote by x_V the set of substitutions θ such that the variable x is free in θ and independent from all the variables in V:

$$x_V \overset{\text{def}}{=} \{\theta \in \mathtt{Sub} \mid dom(\theta) \subseteq V,\ x \in dom(\theta) \Rightarrow \theta(x) \in \mathcal{V} \setminus (\cup_{v \in dom(\theta) \smallsetminus \{x\}} vars(\theta(v)))\}.$$

Observe that for any x_V, we have that $\epsilon \in x_V$. For any finite set of variables $V \subseteq_{\mathit{fin}} \mathcal{V}$, let $\mathcal{F}_V \overset{\text{def}}{=} \mathcal{M}(\{x_V \mid x \in V\})$ be the abstract domain for observing this compound freeness and independence property for the variables in V and let $\mathcal{F} \overset{\text{def}}{=} \sqcap_{V \subseteq_{\mathit{fin}} \mathcal{V}} \mathcal{F}_V$ be the domain for observing the property for any program variable, which is defined as reduced product of all the $\mathcal{F}_V$'s. For instance, $\mathcal{F}_{\{x,y\}} = \{\top, x_{\{x,y\}}, y_{\{x,y\}}, x_{\{x,y\}} \cap y_{\{x,y\}}\}$, where $\top$ stands for $\mathtt{Sub}$. Note that any $a \in \mathcal{F}_V$ can be written as $a = \cap_{v \in W} v_V$, for some (possibly empty) subset $W \subseteq V$. Let us denote by ρ_V the closure operator associated to $\mathcal{F}_V$. In the following proof we will exploit the fact that the best correct approximation of unification on the domain $\mathcal{F}_V$ is as follows: Given any $W_1, W_2 \subseteq V$, we have that $\cap_{v \in W_1} v_V \otimes \cap_{v \in W_2} v_V = \cap_{v \in W_1 \cap W_2} v_V$.

Our goal is to compute the weak-complete shell of the variable-independent domain $\mathcal{F}$. We start by computing the weak-complete shell of $\mathcal{F}_V$, for any $V \subseteq_{\mathit{fin}} \mathcal{V}$.

Lemma 5.1. *Let $V \subseteq_{\mathit{fin}} \mathcal{V}$. Then, $\mathcal{F}_V \sqcap (\mathcal{F}_V \overset{\wedge}{\multimap} \mathcal{F}_V) = \mathcal{F}_V \cup \{\varnothing\}$.*

Proof. We know by Proposition 2.2 that $a \multimap \bigwedge_{i \in I} b_i = \bigwedge_{i \in I}(a \multimap b_i)$. Moreover, observe that $a \multimap \top = \top$. Thus, we only need to compute the implications from elements of $\mathcal{F}_V$ to x_V for some $x \in V$. We distinguish the following cases.

—$\top \multimap x_V = \{\theta \in \mathtt{Sub} \mid \forall \delta \in \top.\ \theta \otimes \delta \in x_V\} = \{\theta \in \mathtt{Sub} \mid\ \downarrow \theta \subseteq x_V\} = \varnothing$, since, for any θ, there always exists an instance of θ such that $\theta(x)$ is ground, and therefore $\downarrow \theta \not\subseteq x_V$.

—$(\bigcap_{v\in V\setminus\{x\}} v_V) \multimap x_V = \{\theta \in \texttt{Sub} \mid \forall\delta \in \bigcap_{v\in V\setminus\{x\}} v_V.\ \theta\otimes\delta \in x_V\} = \varnothing$. In fact, let θ be any substitution and consider $\delta = \{x/t\}$ where t is any ground instance of $\theta(x)$ if $x \in \mathit{dom}(\theta)$, otherwise t is any ground term. Observe that $\delta \in \bigcap_{v\in V\setminus\{x\}} v_V$. Moreover, θ and δ obviously unify but $\theta\otimes\delta \notin x_V$, and therefore $\theta \notin (\bigcap_{v\in V\setminus\{x\}} v_V) \multimap x_V$.

—$x_V \multimap x_V = x_V$. Since $\{\epsilon\}$ is the unit and $\{\epsilon\} \subseteq x_V$, we have that $x_V \multimap x_V \subseteq x_V$. To show the other direction, from $\rho_V(x_V \otimes x_V) = x_V$, it follows that $x_V \subseteq x_V \multimap x_V$ simply by definition of linear implication $\multimap$.

—$(\bigcap_{v\in V} v_V) \multimap x_V = x_V$. As in the previous case, since $\{\epsilon\} \subseteq \bigcap_{v\in V} v_V$, we have that $\bigcap_{v\in V} v_V \multimap x_V \subseteq x_V$. On the other hand, from $\rho_V((\bigcap_{v\in V} v_V)\otimes x_V) = x_V$, it follows that $x_V \subseteq (\bigcap_{v\in V} v_V) \multimap x_V$.

Now, let $W_1, W_2 \subseteq V$ such that $x \notin W_1$ and $x \in W_2$. Since $\bigcap_{v\in W_1} v_V \supseteq \bigcap_{v\in V\smallsetminus\{x\}} v_V$, by Proposition 2.2 (xii), $(\bigcap_{v\in W_1} v_V) \multimap x_V \subseteq (\bigcap_{v\in V\smallsetminus\{x\}} v_V) \multimap x_V =$ (as shown above) $= \varnothing$. Moreover, since $x_V \supseteq \bigcap_{v\in W_2} v_V \supseteq \bigcap_{v\in V} v_V$, it follows from Proposition 2.2 (xii) and what has been shown above that $x_V = x_V \multimap x_V \subseteq (\bigcap_{v\in W_2} v_V) \multimap x_V \subseteq (\bigcap_{v\in V} v_V) \multimap x_V = x_V$. Thus, summing up, we have that:

—$\top \multimap x_V = \varnothing$,

—$(\bigcap_{v\in W_1} v_V) \multimap x_V = \varnothing$, if $x \notin W_1$,

—$(\bigcap_{v\in W_2} v_V) \multimap x_V = x_V$, if $x \in W_2$.

Hence, it turns out that $\mathcal{F}_V \sqcap (\mathcal{F}_V \overset{\wedge}{\multimap} \mathcal{F}_V) = \mathcal{M}(\mathcal{F}_V \cup \{\varnothing\}) = \mathcal{F}_V \cup \{\varnothing\}$. □

The next result shows that $\mathcal{F}_V \sqcap (\mathcal{F}_V \overset{\wedge}{\multimap} \mathcal{F}_V)$ is closed for $\overset{\wedge}{\multimap}$.

Lemma 5.2. *Let $V \subseteq_{\mathit{fin}} \mathcal{V}$ and $\mathcal{F}_V^{\varnothing} \stackrel{\text{def}}{=} \mathcal{F}_V \cup \{\varnothing\}$. Then, $\mathcal{F}_V^{\varnothing} \overset{\wedge}{\multimap} \mathcal{F}_V^{\varnothing} = \mathcal{F}_V^{\varnothing}$, namely $\mathcal{F}_V^{\varnothing}$ is the weak-complete shell of $\mathcal{F}_V$.*

Proof. By Proposition 2.2 (iii) and Lemma 5.1 we only need to show that for each $A \in \mathcal{F}_V$, $A \multimap \varnothing$, $\varnothing \multimap A$ and $\varnothing \multimap \varnothing$ all belong to $\mathcal{F}_V^{\varnothing}$. Let us examine these cases.

—$\varnothing \multimap A = \varnothing \multimap \varnothing = \top$, simply by definition of linear implication $\multimap$.

—$A \multimap \varnothing = \varnothing$. We know that $\epsilon \in A$. Therefore, if $\theta \in A \multimap \varnothing$ then we would have the contradiction $\theta \otimes \epsilon = \theta \in \varnothing$.

Thus, we have that $\mathcal{F}_V^{\varnothing} \overset{\wedge}{\multimap} \mathcal{F}_V^{\varnothing} = \mathcal{F}_V^{\varnothing}$. □

Given $W \subseteq \mathcal{V}$ we will abuse the notation and write $\varnothing_W = \{\theta \in \texttt{Sub} \mid \mathit{dom}(\theta) \subseteq W\}$ and we also write x_W when $x \notin W$ for $\varnothing_W$. Let us also use the following notation: $\mathcal{F}^{\varnothing} \stackrel{\text{def}}{=} \sqcap_{V\subseteq_{\mathit{fin}}\mathcal{V}} \mathcal{F}_V^{\varnothing}$.

Theorem 5.3. *$\mathcal{F}^{\varnothing} = \cup_{V\subseteq_{\mathit{fin}}\mathcal{V}} \mathcal{F}_V^{\varnothing} \cup \{\varnothing_W \mid W \subseteq_{\mathit{fin}} \mathcal{V}\}$. Moreover, $\mathcal{F}^{\varnothing} \overset{\wedge}{\multimap} \mathcal{F}^{\varnothing} = \mathcal{F}^{\varnothing}$, namely $\mathcal{F}^{\varnothing}$ is the weak-complete shell of $\mathcal{F}$.*

Proof. First note that, given $x_A \in \mathcal{F}_A, y_B \in \mathcal{F}_B$ then $x_{A\cap B} \subseteq x_A$, simply by definition of x_A. Thus $x_A \cap y_B \supseteq x_{A\cap B} \cap y_{A\cap B}$. Moreover, for any $\theta \in x_A \cap y_B$, we have that $\mathit{dom}(\theta) \subseteq A$ and $\mathit{dom}(\theta) \subseteq B$, thus $\mathit{dom}(\theta) \subseteq A \cap B$. Since $\theta \in x_A$, it follows that $\theta \in x_{A\cap B}$. Thus, $x_A \cap y_B \subseteq x_{A\cap B} \cap y_{A\cap B}$, from which we have that $x_A \cap y_B = x_{A\cap B} \cap y_{A\cap B}$. Hence, more in general, $\cap_{i\in I} x^i_{A_i} = \cap_{i\in I} x^i_{\cap_{j\in I} A_j}$ and therefore $\cap_{i\in I} x^i_{A_i} \in \mathcal{F}_{\cap_{i\in I} A_i}$. This shows that $\mathcal{F}^{\varnothing} = \cup_{V \subseteq_{\mathit{fin}} \mathcal{V}} \mathcal{F}^{\varnothing}_V \cup \{\varnothing_W \mid W \subseteq_{\mathit{fin}} \mathcal{V}\}$.

We now show that $\mathcal{F}^{\varnothing} \overset{\wedge}{\multimap} \mathcal{F}^{\varnothing} = \mathcal{F}^{\varnothing}$. Following the pattern of the previous proofs, it is enough to consider $\cap_{v\in A} v_V \in \mathcal{F}_V$ and $y_W \in \mathcal{F}_W$ for some $V, W \subseteq_{\mathit{fin}} \mathcal{V}$ and $A \subseteq V$, and to show that $\cap_{v\in A} v_V \multimap y_W \in \mathcal{F}^{\varnothing}$. We know that $\epsilon \in \cap_{v\in A} v_V$, hence $\cap_{v\in A} v_V \multimap y_W \subseteq y_W$. Moreover, note that any $\theta \in \cap_{v\in A} v_V \multimap y_W$ must be such that $dom(\theta) \subseteq W$, otherwise we would have that $\theta \otimes \epsilon \notin y_W$. We distinguish the following three cases.
(1) If $\cap_{v\in A} v_V \subseteq y_W$ then $\cap_{v\in A} v_V \multimap y_W \supseteq y_W \multimap y_W = y_W$, from which $\cap_{v\in A} v_V \multimap y_W = y_W \in \mathcal{F}_W$.
(2) If $y \notin V$ then it follows that $y_W \otimes \cap_{v\in A} v_V \subseteq y_W$, and therefore again we have that $\cap_{v\in A} v_V \multimap y_W = y_W$
(3) Finally assume that both (1) and (2) do not hold, i.e., $\cap_{v\in A} v_V \not\subseteq y_W$ and $y \in V$. Since any $\theta \in \cap_{v\in A} v_V \multimap y_W$ is such that $dom(\theta) \subseteq W$, then $y \notin A$. It follows that $\delta = \{y/t\} \in \cap_{v\in A} v_V$ for any ground term t. Thus $\cap_{v\in A} v_V \multimap y_W = \varnothing \in \mathcal{F}^{\varnothing}$. $\square$

Corollary 5.4. *$\mathcal{F}$ is the most abstract domain which is condensing and includes $\mathcal{F}_V$.*

Proof. By the above lemmata and Corollary 4.9, it remains to show that $\mathcal{F}$ is finitely generated and compatible with π.

By Theorem 5.3 we need to show that, for any $a, b \in \rho_V$ there exists $\Theta \subseteq_{\mathit{fin}} \mathtt{Sub}$ such that $\Theta \subseteq a \multimap b$ and $\rho(\Theta) = \rho(a \multimap b)$. By Lemma 5.1, we only need to show that for any $A \subseteq V$ there exists $\Theta \subseteq_{\mathit{fin}} \mathtt{Sub}$ such that $\Theta \subseteq \bigcap_{v\in A} v_V$ and $\rho(\Theta) = \bigcap_{v\in A} v_V$. We can choose $\Theta = \{\{v/a \mid v \in V \setminus A\} \cup \{v/w_v \mid v \in A\}\}$ where a is any ground term and all the w_v are distinct variables. Moreover, given $\varnothing_V$ we can choose $\Theta = \{\{v/a \mid v \in V\}\}$.

We now show that ρ is compatible with π. As usual, in the following let $\top$ denotes $\mathtt{Sub}$. Let $W \subseteq \mathcal{V}$, $a \in \rho$ and $\Theta \subseteq \mathtt{Sub}$ such that $\pi_W(\Theta) = \Theta$. We assume $a \notin \varnothing$ (otherwise the thesis follows trivially). If $|dom(\Theta)| = |\mathbb{N}|$, then by definition of ρ we have that $\rho(\Theta) = \top$. Since ρ is weak-complete, it follows that $\rho(\pi_W(\rho(\Theta \otimes a))) = \rho(\pi_W(\rho(\rho(\Theta) \otimes a))) = \rho(\pi_W(\rho(\top \otimes a))) = \top$ since $a \neq \varnothing$. On the other hand, $\rho(\Theta \otimes \rho(\pi_W(a))) = \rho(\rho(\Theta) \otimes \rho(\pi_W(a))) = \rho(\top \otimes \rho(\pi_W(a))) = \top$ since $a \neq \varnothing$.

Now assume that $dom(\Theta) \subseteq_{\mathit{fin}} \mathcal{V}$. By definition of ρ it follows that $dom(\rho(\Theta)) = dom(\Theta)$ since $\Theta \subseteq \varnothing_{dom(\Theta)} \in \mathcal{F}$ and $dom(\varnothing_{dom(\Theta)}) = dom(\Theta)$. Since $\pi_W(\Theta) = \Theta$, it follows that $W \supseteq dom(\Theta)$, and therefore $\pi_W(\rho(\Theta)) = \rho(\Theta)$, from which $\rho(\pi_W(\rho(\Theta))) = \rho(\Theta)$.

Since ρ is weak-complete, it is sufficient to show that, for all $a, b \in \rho$ and such that $\rho(\pi_W(a)) = a$ it holds that $\rho(\pi_W(\rho(a \otimes b))) = \rho(a \otimes \rho(\pi_W(b)))$. Let $W \subseteq \mathcal{V}$ and $A, B \subseteq V$. First note that $\rho(\pi_W(\bigcap_{v\in A} v_V)) = \rho(\pi_{W\cap V}(\bigcap_{v\in A} v_V))$ since $dom(\bigcap_{v\in A} v_V) \subseteq V$. Thus, $\rho(\pi_{W\cap V}(\bigcap_{v\in A} v_V)) = \bigcap_{v\in A\cap W\cap V} v_{W\cap V}$ by definition of π. Since $A \subseteq V$ then $\rho(\pi_W(\bigcap_{v\in A} v_V)) = \bigcap_{v\in A\cap W} v_{W\cap V}$. Moreover, note that $\rho(\pi_W(\top)) = \top$ if and only if $W \not\subseteq_{\mathit{fin}} \mathcal{V}$, that is $|W| = |\mathbb{N}|$. In the following we assume that $|W| = |\mathbb{N}|$ (otherwise the thesis is trivially satisfied since $\pi_{\varnothing}(\Theta) = \{\epsilon\}$).

We distinguish the following cases:

—$\rho(\pi_W(\rho(\top \otimes \varnothing))) = \rho(\pi_W(\varnothing)) = \varnothing$ and $\rho(\top \otimes \rho(\pi_W(\varnothing))) = \rho(\top \otimes \varnothing) = \varnothing$.

—Assume $\rho(\pi_W(\top)) = \top$. If $C \neq \varnothing$ then $\epsilon \in C$ and $\rho(\pi_W(\top \otimes C)) = \rho(\pi_W(\top)) = \top$ since $W \neq \varnothing$ by hypothesis.
Moreover, $\rho(\top \otimes \rho(\pi_W(C))) = \top$ since $\rho(\pi_W(C)) \neq \varnothing$ being $C \neq \varnothing$ and $W \neq \varnothing$.

—Assume that $\rho(\pi_W(\cap_{v\in A} v_V)) = \cap_{v\in A} v_V$. Since it holds that $\rho(\pi_W(\cap_{v\in A} v_V)) = \cap_{v\in A\cap W} v_{V\cap W}$ it follows that $V \subseteq W$. If $V = \varnothing$ then $\cap_{v\in A} v_V = \{\epsilon\}$ and there is nothing to prove. Otherwise, $\rho(\pi_W(\top)) = \top$, since $W \neq \varnothing$. Therefore $\rho(\pi_W(\rho(\cap_{v\in A} v_V \otimes \top))) = \rho(\pi_W(\top)) = \top$. Moreover, $\rho(\cap_{v\in A} v_V \otimes \rho(\pi_W(\top))) = \rho(\cap_{v\in A} v_V \otimes \top) = \top$.

—Assume that $\rho(\pi_W(\cap_{v\in A} v_V)) = \cap_{v\in A} v_V$. Then, $\rho(\pi_W(\rho(\cap_{v\in A} v_V \otimes \cap_{v\in B} v_V))) = \rho(\pi_W(\cap_{v\in A\cap B} v_V)) = \cap_{v\in A\cap B\cap W} v_{V\cap W}$. Also, $\rho(\cap_{v\in A} v_V \otimes \rho(\pi_W(\cap_{v\in B} v_V))) = \rho(\cap_{v\in A} v_V \otimes \cap_{v\in B\cap W} v_{V\cap W}) = \cap_{v\in A\cap B\cap W} v_{V\cap W}$. □

The following simple example shows how the abstract domain $\mathcal{F}^{\varnothing}$ can be used to detect redundant arguments in a predicate definition.

Example 5.5. Consider the following program:

$$p(x,u,v) \leftarrow \{x/0\} \vee (\{x/f(w), y/w\} \otimes p(y,u,v)).$$

Let $A \equiv \{x/0\} \vee (\{x/f(w), y/w\} \otimes p(y,u,v))$. Note that both the second and third arguments of p are redundant. Let ρ be the closure associated to $\mathcal{F}^{\varnothing}$. The abstract semantics $\mathcal{S}^{\rho}_{p(x,u,v)}$ for the query $\rho(\{\epsilon\}) = \{\epsilon\} = \varnothing_{\varnothing}$ is given by the following recursive equation:

$$\begin{aligned}
\mathcal{S}^{\rho}_{p(x,u,v)}(\{\epsilon\}) &= \rho(\pi_{\{x,u,v\}}\mathcal{S}^{\rho}_{A}(\{\epsilon\})) \\
&\qquad \text{where } A \equiv \{x/0\} \vee (\{x/f(w), y/w\} \otimes p(y,u,v)) \\
&= \rho(\pi_{\{x,u,v\}}(\rho(\mathcal{S}^{\rho}_{\{x/0\}}(\{\epsilon\}) \cup \mathcal{S}^{\rho}_{\{x/f(w),y/w\}\otimes p(y,u,v)}(\{\epsilon\})))) \\
&= \rho(\pi_{\{x,u,v\}}(\rho(\rho(\{x/0\} \otimes \{\epsilon\}) \\
&\quad \cup \rho(\mathcal{S}^{\rho}_{\{x/f(w),y/w\}}(\{\epsilon\}) \otimes \mathcal{S}^{\rho}_{p(y,u,v)}(\{\epsilon\})))))) \\
&= \rho(\pi_{\{x,u,v\}}(\rho(\varnothing_{\{x\}} \\
&\quad \cup \rho(\rho(\{x/f(w), y/w\}) \otimes \mathcal{S}^{\rho}_{p(y,u,v)}(\{\epsilon\}))))) \\
&= \rho(\pi_{\{x,u,v\}}\rho(\varnothing_{\{x\}} \cup \rho(\varnothing_{\{x,y\}} \otimes \mathcal{S}^{\rho}_{p(y,u,v)}(\{\epsilon\}))))
\end{aligned}$$

The most abstract solution for $\mathcal{S}^{\rho}_{p(x,u,v)}(\{\epsilon\})$ of this recursive equation is the object $\varnothing_{\{x\}}$, telling that in any computed answer substitution σ for $p(x,u,v)$ in P it turns out that $dom(\sigma) \subseteq \{x\}$. As an example of condensation, let us now consider the substitution $\theta = \{x/w, u/w\}$ and compute $\mathcal{S}^{\rho}_{p(x,u,v)}(\rho(\{\theta\}))$. Since $\mathcal{F}^{\varnothing}$ is condensing $\rho(\rho(\{\theta\}) \otimes \{\epsilon\}) = \rho(\{\theta\}) = \varnothing_{\{x,u\}}$, we have that:

$$\mathcal{S}^{\rho}_{p(x,u,v)}(\rho(\{\theta\})) = \rho(\rho(\{\theta\}) \otimes \mathcal{S}^{\rho}_{p(x,u,v)}(\{\epsilon\})) = \rho(\varnothing_{\{x,u\}} \otimes \varnothing_{\{x\}}) = \varnothing_{\{x,u\}},$$

and this shows that the variable v is still free and independent from all the other variables. □

6. CONCLUSION

We have shown a strong link between completeness in quantale-like structures and condensation of abstract domains for logic program analysis, by providing a characterization of condensing domains as models of a fragment of propositional linear logic. The relationship between completeness and reversible logic program analysis has gained much attention in the last few years. As observed by King and Lu [2002], the possibility of reusing code in logic programming is often related to the problem of figuring out how to query a program, and backward analysis allows to automatically derive the possible modalities in which predicates must be called. As shown by King and Lu [2002], this property needs condensing abstract domains. By this observation and from our characterization of condensing abstract domains in logical form, it seems possible to characterize reversible abstract interpretations in a pure domain-theoretic form. There are still many open questions along this line of research. It is for instance a challenge to design condensing abstract domains for aliasing. The Corollary 4.9 provides necessary and sufficient conditions to systematically design these domains, but the construction of non-downward closed condensing abstract domains, although clarified and made systematic, is still quite difficult due to the complex structure of the quantale of the powerset of substitutions and unification. For instance, it is an important and challenging open question to characterize the weak-complete shell of *Sharing*, a well known abstract domain devoted to the static analysis of variable aliasing [Jacobs and Langen 1992; Langen 1991]. Our results can be also used to prove that already known domains are condensing. Scozzari [2002] proved that the domain *Pos* for groundness analysis [Armstrong et al. 1998] is the most abstract solution of the abstract domain equation $X = \mathit{Ground} \sqcap (X \to X)$, where *Ground* is the basic domain of plain groundness by Jones and Søndergaard [1987]. Thus, by Corollary 4.9, this provides an alternative proof of the known fact that *Pos* is condensing [Marriott and Søndergaard 1993].

REFERENCES

Apt, K. 1990. Logic Programming. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, J. van Leeuwen, Ed. Elsevier.

Armstrong, T., Marriott, K., Schachte, P., and Søndergaard, H. 1998. Two classes of Boolean functions for dependency analysis. *Sci. Comput. Program 31,* 1, 3–45.

Barbuti, R., Giacobazzi, R., and Levi, G. 1993. A general framework for semantics-based bottom-up abstract interpretation of logic programs. *ACM Trans. Program. Lang. Syst. 15,* 1, 133–181.

Codish, M., Dams, D., and Yardeni, E. 1994. Bottom-up abstract interpretation of logic programs. *Theor. Comput. Sci. 124,* 1, 93–126.

Codish, M. and Lagoon, V. 2000. Type dependencies for logic programs using ACI-unification. *Theor. Comput. Sci. 238*, 131–159.

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

Cousot, P. and Cousot, R. 1979. Systematic design of program analysis frameworks. In *Conference Record of the 6th ACM Symposium on Principles of Programming Languages* (*POPL '79*). ACM Press, New York, 269–282.

Debray, S. K. 1994. Formal bases for dataflow analysis of logic programs. In *Advances in Logic Programming Theory*, G. Levi, Ed. Oxford University Press, New York, NY.

Giacobazzi, R., Ranzato, F., and Scozzari, F. 2000. Making abstract interpretations complete. *J. ACM 47,* 2, 361–416.

Giacobazzi, R. and Scozzari, F. 1998. A logical model for relational abstract domains. *ACM Trans. Program. Lang. Syst. 20,* 5, 1067–1109.

Jacobs, D. and Langen, A. 1992. Static analysis of logic programs for independent AND-parallelism. *J. Logic Program. 13,* 2-3, 154–165.

Jones, N. D. and Søndergaard, H. 1987. A semantics-based framework for the abstract interpretation of Prolog. In *Abstract Interpretation of Declarative Languages*, S. Abramsky and C. Hankin, Eds. Ellis Horwood Ltd, Chichester, UK, 123–142.

King, A. and Lu, L. 2002. A Backward Analysis for Constraint Logic Programs. *Theory and Practice of Logic Programming 2,* 4, 517–547.

Langen, A. 1991. Static analysis for independent And-parallelism in logic programs. Ph.D. thesis, Univ. of Southern California, Los Angeles, Calif.

Marriot, K., Søndergaard, H., and Jones, N. D. 1994. Denotational abstract interpretation of logic programs. *ACM Trans. Program. Lang. Syst. 16,* 3, 607–648.

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

Palamidessi, C. 1990. Algebraic properties of idempotent substitutions. In *Proc. of the 17th International Colloquium on Automata, Languages and Programming*, M. S. Paterson, Ed. Lecture Notes in Computer Science, vol. 443. Springer-Verlag, Berlin, 386–399.

Rosenthal, K. 1990. *Quantales and their Applications.* Longman Scientific & Technical, Harlow, Essex, U.K.

Schachte, P. 2003. Precise goal-independent abstract interpretation of constraint logic programs. *Theor. Comput. Sci. 293,*, 557–577.

Scozzari, F. 2002. Logical optimality of groundness analysis. *Theor. Comput. Sci. 277,* 1-2, 149–184.

Yetter, D. 1990. Quantales and (noncommutative) linear logic. *J. Symbolic Logic 55,* 1, 41–64.

Received April 2002; revised March 2003 and June 2003; accepted June 2003