Università di Pisa
Dipartimento di Informatica

Technical Report: TR-01-09

# Optimal Semantics and Domains in Logic Program Analysis

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

May 28, 2001

ADDRESS: Corso Italia 40, 56125 Pisa, Italy. TEL: +39 050 2212700 FAX: +39 050 2212726

# Optimal Semantics and Domains in Logic Program Analysis

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

Completeness is important in approximated semantics design by abstract interpretation, ensuring that abstract computations are as precise as possible w.r.t. concrete ones. The authors recently proved that any abstract domain can be systematically refined or simplified in order to provide its closest possible complete domain. The key problem here remains is to find efficient and appropriate representations for objects in complete abstract domains. In this paper we show that these domain trensformations, when applied on abstract interpretations of quantales, can all be explicitly characterized by elegant linear logic-based formulations. We apply our results to logic program analysis, where unification can be interpreted as multiplicative conjunction in a quantale of idempotent substitutions. We characterize the objects in optimal semantics for logic program analysis, viz. the most abstract concrete semantics for any static program analysis property, and the structure of condensing domains in logic program analysis.

Additional Key Words and Phrases: Abstract interpretation, domains, completeness, linear logic, logic program analysis, condensing domains

## 1. INTRODUCTION

It is widely held that the ideal goal of any semantics is to find sound and complete (possibly fully abstract) representations for some properties of concrete (actual) computations. This is not often the case in decidable approximated semantics for program analysis, type inference and automatic program verification, where soundness is always required, while completeness more rarely holds. Abstract interpretation provides a good framework where soundness and completeness of most approximated (abstract) semantics can be studied, independently from its use in static program analysis, comparative semantics and program verification.

Completeness issues in abstract interpretation have been studied since the Cousot

Authors' addresses: R. Giacobazzi, Dipartimento Scientifico e Tecnologico, Università di Verona, Strada Le Grazie, Ca' Vignal 2, 37134 Verona, Italy, e-mail: giaco@sci.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, Corso Italia 40, 56125 Pisa, Italy, e-mail: scozzari@di.unipi.it.
The work of Roberto Giacobazzi and Francesco Ranzato is partly supported by the Italian MURST project: "Automatic program certification by abstract interpretation".

and Cousot seminal paper [Cousot and Cousot 1979b]. The intuition is that a complete abstract interpretation induces an abstract semantics which is as precise as possible relatively to its underlying abstract domains and to the concrete interpretation of reference. The paradigmatic rule of signs is a typical and simple example of an abstract interpretation which is (sound and) complete for integer multiplication but merely sound for integer addition [Mycroft 1993]. Although in static program analysis decidability issues often force to sacrifice completeness for achieving termination and/or efficiency, examples of complete abstract interpretations are common in other fields of application. For instance, several complete abstractions of algebraic polynomial systems have been studied by Cousot and Cousot in [Cousot and Cousot 1997] and many complete abstract interpretations can be found in comparative program semantics [Cousot 1997; Cousot and Cousot 1992; Giacobazzi 1996]. Moreover, being completeness a notion relative to the concrete semantics of reference, complete abstract interpretations which are more concrete than a certain, possibly approximated and decidable, property of interest, yield an absolute upper bound for the precision that one can achieve in computing that property. Thus, complete abstract interpretations may play a useful rôle in static program analysis as well.

These argumentations probably stimulated the recent trend of research on completeness in abstract interpretation [Steffen 1989; Mycroft 1993; Reddy and Kamin 1993; Sekar et al. 1997; Giacobazzi and Ranzato 1997; Giacobazzi et al. 2000]. A key problem in this trend consists in devising systematic and constructive methodologies for transforming abstract domains in such a way that completeness is achieved and the resulting complete abstract domains are as close as possible to the initial (non-complete) ones. This problem has been first raised in a predicate-based approach to abstract interpretation by Mycroft [1993, Section 3.2], who gave a methodology for deriving the most concrete domain, called "canonical abstract interpretation", which is complete and included in a given domain of properties. More recently, we proved [Giacobazzi et al. 2000] that when the concrete semantic function $f$ is continuous, any domain $A$ can always be constructively extended into the most abstract domain which includes $A$ and is complete for $f$ — the so-called *complete shell* of $A$.

In this paper we are concerned with the problem of finding suitable representations of objects in complete abstract domains. This is a key problem in order to use complete abstract domains in semantics and program analysis. A good representation of objects in abstract semantics is essential to automatically (or, at least, quickly) implement abstract domains, e.g. by exploiting the logical properties of the abstract objects (see the use of BDD's in the implementation of classical logic-based abstract domain for groundness analysis [Armstrong et al. 1994]); to study the properties of abstract operations, like for instance their precision; to help the intuition to understand how specific abstract domains work. Logic has been often used to represent objects in semantics [Abramsky and Vickers 1993; Abramsky and Jung 1994] and domains for program analysis [Jensen 1997; Armstrong et al. 1994]. We follow this trend and consider a fragment of propositional linear logic as the basic setting for constructing complete abstract domains. We consider completeness problems arising when concrete semantic binary operations of type $C \times C \longmapsto C$ are assumed to give rise to a *quantales*. Quantales are well-known algebraic structures which turn out to be models of intuitionistic linear logic [Rosenthal 1990; Yetter 1990]. This is particularly important in this context, because quantales naturally model a number of different and new completeness problems arising in abstract interpretation and provide an elegant solution of these problems where complete domain's objects are represented as linear implications, with a clean logical interpretation. In this logical setting, we provide an elegant linear logic-based solution to a number of interesting completeness problems for abstract interpretations. More in detail, a quantale $\langle C_{\leq}, \otimes \rangle$ consists of a complete lattice $C$ together with an operation $\otimes : C \times C \longmapsto C$ which is additive (i.e., preserves lub's) on both arguments. The main feature of quantales is that they 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 $\otimes$ with $a$, gives a result less than or equal to $b$. In other terms, the following *modus ponens* law $a \otimes x \leq b \iff x \leq a \multimap b$ holds. When solving completeness problems in a setting where concrete interpretations are quantales, linear implication allows to elegantly characterize complete domain objects in a variety of situations.

Our first result provides a characterization of the objects in complete abstract domains which are abstractions of a quantale. Denotes by $\mathtt{uco}(C)$ the complete lattice of all abstract domains (modulo isomorphic representation of their objects) abstracting a given domain $C$ (cf. [Cousot and Cousot 1979b]). This lattice is ordered by the relative precision of domains: for each $A, B \in \mathtt{uco}(C)$, $A \sqsubseteq B$ if $A$ is more concrete (more precise) than $B$. Given a quantale $\langle C_{\leq}, \otimes \rangle$ and an abstract domain $A \in \mathtt{uco}(C)$, we characterize the most abstract domain $X \in \mathtt{uco}(C)$ such that $X \sqsubseteq A$ and $X$ is complete for $\otimes$, namely if $\alpha_X : C \longmapsto X$ is the corresponding abstraction map: $\alpha_X(\_ \otimes \_) = \alpha_X(\alpha_X(\_) \otimes \alpha_X(\_))$. We weaken the above standard completeness problem by deriving the most abstract domain $X \in \mathtt{uco}(C)$ such that $X \sqsubseteq A$ and no loss of precision holds by observing in $X$ the result of $\otimes$ when one of its arguments is approximated in $X$, namely if $\alpha_X(\alpha_X(\_) \otimes \alpha_X(\_)) = \alpha_X(\_ \otimes \alpha_X(\_)) = \alpha_X(\alpha_X(\_) \otimes \_)$. We call this problem *disjunctive completeness*. We give solution to these problems, where the objects of the complete domains are fully characterized in logical form as linear implications.

These completeness problems have relevant applications in program analysis. Given a program $P$, we consider a concrete semantics $[\![P]\!]$ characterizing computed answer substitutions in a s-semantics style (cf. [Bossi et al. 1994; Falaschi et al. 1989]). In program analysis, it is often the case that the semantics $[\![P]\!]$ is too concrete for observing a given property $A$, i.e., it contains some redundant information not related to the property $A$. It may be the case that semantics more abstract than $[\![P]\!]$ give exactly the same result, when observing the property $A$. Let us denote by $[\![P]\!]^X$ the approximated (abstract) semantics induced by an abstract domain $X$. When the observations of $A$ on the concrete and abstract semantics yield the same result, i.e., $\alpha_A([\![P]\!]) = \alpha_A([\![P]\!]^X)$, then we say that the domain $X$ is *adequate* for observing the property $A$. We are interested in the most abstract domain $X \in \mathtt{uco}(C)$ which is adequate for observing a property $A$. When it exists, we call it *optimal semantics* for $A$. Note that there is no loss of precision in using the optimal semantics instead of the concrete one, relatively to the property $A$. As usual, unification turns out to be the key operation to take into account in order to build optimal semantics of given properties. Given an abstract domain $A$, we show

how the complete shell of $A$ w.r.t. the operation of unification naturally induces the optimal semantics of $A$. Since sets of idempotent substitutions and unification give rise to a quantale, this leads to a characterization of optimal semantics for program analysis as a standard completeness problems in the quantale of idempotent substitutions and allows us to give an explicit representation of domain's objects in optimal semantics. As a remarkable instance of our construction, we specify the optimal semantics of the groundness property. We then characterize the structure of condensing domains for logic program analysis as a disjunctive completeness problem. A domain is condensing if goal-driven and goal-independent analyses agree. This family of domains is particularly important in logic programming, as they provide optimal conditions for implementing goal-independent analyses with no loss of precision with respect to goal-driven ones. We prove that a domain is condensing if and only if the corresponding abstraction can be removed from any of the arguments of concrete unification, without loss of precision. This corresponds precisely to state that condensation is a disjunctive completeness problem, which admits an elegant solution as implicational domain equation involving linear objects. This is a generalization of an analogous result in [Giacobazzi and Scozzari 1998] characterizing condensing downward closed domains as solutions of domain equations involving intuitionistic implication.

## 2. BASIC NOTIONS

### 2.1 Mathematical notation.

If $S$ and $T$ are sets, then $\wp(S)$ denotes the powerset of $S$, $S \smallsetminus T$ denotes the set-difference between $S$ and $T$, $S \subset T$ denotes strict inclusion, and for a function $f : S \longmapsto T$ 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 $\langle P, \leq\rangle$ denotes a poset $P$ with ordering relation $\leq$, while $\langle P, \leq, \vee, \wedge, \top, \bot\rangle$ denotes a complete lattice $P$, with ordering $\leq$, *lub* $\vee$, *glb* $\wedge$, greatest element (top) $\top$, and least element (bottom) $\bot$. Often, $\leq_P$ will be used to denote the underlying ordering of a poset $P$, and $\vee_P$, $\wedge_P$, $\top_P$ and $\bot_P$ denote the basic operations and elements of a complete lattice. 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 $\downarrow S \stackrel{\text{def}}{=} \{x \in P \mid \exists y \in S.\ x \leq_P y\}$, and for $x \in P$, $\downarrow x$ is a shorthand for $\downarrow \{x\}$. $S \longmapsto T$ denotes the set of all functions from $S$ to $T$. We use the symbol $\sqsubseteq$ to denote pointwise ordering between functions: If $S$ is any set, $P$ a poset, and $f, g : S \longmapsto P$ then $f \sqsubseteq g$ if for all $x \in S$, $f(x) \leq_P g(x)$. Let $C$ and $A$ be complete lattices. Then, $C \stackrel{\text{m}}{\longmapsto} A$, $C \stackrel{\text{c}}{\longmapsto} A$, $C \stackrel{\text{a}}{\longmapsto} A$, and $C \stackrel{\text{coa}}{\longmapsto} A$ denote, respectively, the set of all monotone, (Scott-)continuous, additive, and co-additive functions from $C$ to $A$. Recall that $f \in C \stackrel{\text{c}}{\longmapsto} A$ iff $f$ preserves *lub*'s of (nonempty) chains, and $f : C \longmapsto A$ is (completely) additive if $f$ preserves *lub*'s of all subsets of $C$ (emptyset included). Co-additivity is defined by duality. 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. If $f \in C \stackrel{\text{c}}{\longmapsto} C$ then $\mathit{lfp}(f) = \vee_{i\in\mathbb{N}} f^i(\bot_C)$, where, for any $i \in \mathbb{N}$ and $x \in C$, the $i$-th power of $f$ in $x$ is inductively defined as follows: $f^0(x) = x$; $f^{i+1}(x) = f(f^i(x))$. Dually, if $f$ is co-continuous then $\mathit{gfp}(f) = \wedge_{i\in\mathbb{N}} f^i(\top_C)$. $\{f^i(\bot_C)\}_{i\in\mathbb{N}}$ and $\{f^i(\top_C)\}_{i\in\mathbb{N}}$ are called, respectively, the *upper* and *lower Kleene's iteration sequences of* $f$ (see

[Cousot and Cousot 1979a]).

### 2.2 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 1979b]. In the first case, the concrete and abstract domains $C$ and $A$ (both assumed to be complete lattices) are related by a pair of adjoint functions of a GC $(\alpha, C, A, \gamma)$. Also, it is generally assumed that $(\alpha, C, A, \gamma)$ is a Galois insertion (GI), i.e., $\alpha$ is onto or, equivalently, $\gamma$ is 1-1. In the second case instead, an abstract domain is specified as a uco on the concrete domain $C$, i.e., a monotone, idempotent and extensive operator on $C$. These two approaches are equivalent, modulo isomorphic representation of domain's objects. Given a complete lattice $C$, it is well known that the set $\mathtt{uco}(C)$ of all uco's on $C$, endowed with the pointwise ordering $\sqsubseteq$, is a complete lattice $\langle \mathtt{uco}(C), \sqsubseteq, \sqcup, \sqcap, \lambda x.\top_C, id\rangle$ ($id$ denotes the identity function). Let us also recall that each $\rho \in \mathtt{uco}(C)$ is uniquely determined by the set of its fixpoints, which is its image, i.e. $\rho(C) = \{x \in C \mid \rho(x) = x\}$, and that $\rho \sqsubseteq \eta$ iff $\eta(C) \subseteq \rho(C)$ iff $\rho$ is a refinement of $\eta$. Moreover, a subset $X \subseteq C$ is the set of fixpoints of a uco on $C$ iff $X$ is meet-closed, i.e. $X = \mathcal{M}(X) \stackrel{\text{def}}{=} \{\bigwedge_C Y \mid Y \subseteq X\}$ (note that $\top_C = \bigwedge_C \varnothing \in \mathcal{M}(X)$). For any $X \subseteq C$, $\mathcal{M}(X)$ is called the *Moore closure* of $X$ and $X$ is a *generator set* for $\mathcal{M}(X)$. 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, $\langle \mathtt{uco}(C), \sqsubseteq\rangle$ will play the rôle of the lattice of abstract interpretations of $C$ [Cousot and Cousot 1977; Cousot and Cousot 1979b], i.e. the complete lattice of all possible abstract domains of the concrete domain $C$. For an abstract domain $A \in \mathtt{uco}(C)$, $\rho_A \in \mathtt{uco}(C)$ will denote the corresponding uco on $C$, and if $A$ is specified by a GI $(\alpha, C, A, \gamma)$ then $\rho_A \stackrel{\text{def}}{=} \gamma \circ \alpha$. The ordering on $\mathtt{uco}(C)$ 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 $\mathtt{uco}(C)$. Lub and glb on $\mathtt{uco}(C)$ have therefore the following reading as operators on domains. Suppose $\{A_i\}_{i\in I} \subseteq \mathtt{uco}(C)$: (i) $\sqcup_{i\in I} A_i$ is the most concrete among the domains which are abstractions of all the $A_i$'s, i.e. it is their least (w.r.t. $\sqsubseteq$) common abstraction; (ii) $\sqcap_{i\in I} A_i$ is the most abstract among the domains (abstracting $C$) which are more concrete than every $A_i$; this domain is also known as the *reduced product* of all the $A_i$'s.

### 2.3 Quantales and linear logic.

*Quantales* originated as algebraic foundations of the so-called *quantum logic*. They have been successively considered for the lattice-theoretic semantics of Girard's *linear logic* (see [Rosenthal 1990] for an exhaustive treatment of quantales). 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 for a generic binary operation $\otimes$. The idea is to guarantee that, for any two objects $a$ and $b$, there exists a most abstract object $c$ which satisfies $a \otimes c \leq b$. In the following, we restrict our attention to commutative quantales, i.e., quantales where the considered operation $\otimes$ turns out to be commutative. More formally, a (commutative) quantale is an algebra $\langle C_{\leq}, \otimes\rangle$ such that:

—$C_{\leq}$ is a complete lattice, where $\bigvee$ is the join operation;

—$\otimes : C \times C \longmapsto C$ 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 $\{ b_i \}_{i \in I} \subseteq C$.

In other words, a quantale is a complete lattice endowed with a commutative and associative "product" $\otimes$ which distributes over arbitrary l.u.b.'s. Common examples of quantales are complete Boolean algebras, which can be made into a quantale, by choosing as $\otimes$ the meet operation of the lattice $\wedge$. In particular, for any set $A$, the algebra $\langle \wp(A)_{\subseteq}, \bigcap\rangle$ is a quantale. Again, given a commutative and associative operation $\cdot : A \times A \longmapsto A$, an example of quantale is $\langle \wp(A)_{\subseteq}, \otimes\rangle$ where $X \otimes Y = \bigcup \{x \cdot y \mid x \in X, y \in Y\}$ is the lifting of the $\cdot$ operation 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 saying that, for all $a, b, c \in C$, the following property holds.

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

In other words, for all $a \in C$, $\lambda x.a \otimes x$ is a complete join morphism. Hence it has a unique right adjoint $\lambda x.a \multimap x$ which forms a Galois connection. A function $\multimap : C \times C \longmapsto C$ can therefore be 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 $\mathbf{1} \in C$ such that $\mathbf{1} \otimes a = a = a \otimes \mathbf{1}$, for all $a \in C$. Examples of unital, commutative quantales are $\langle \wp(A)_{\subseteq}, \bigcap\rangle$, where the unity is simply $A$.

From a logical point of view, it is well known that quantales turn out to be models of (commutative) linear logic [Yetter 1990; Rosenthal 1990], where the operation $\multimap$ is called *linear implication*. The next proposition summarizes the main properties of linear implications (see [Rosenthal 1990]). It is worth noting that the first property can be thought of as a generalization of the modus ponens law.

PROPOSITION 2.1. *Let $\langle C_{\leq}, \otimes\rangle$ be a unital, commutative quantale with unity* $\mathbf{1}$, $\{ x_i \}_{i \in I} \subseteq C$ *and* $a, b, c \in C$.

| | | | |
|---|---|---|---|
| $(i)$. | $a \otimes (a \multimap c) \leq c$ | $(ii)$. | $a \multimap (b \multimap c) = (b \otimes a) \multimap c$ |
| $(iii)$. | $a \multimap (\bigwedge_{i \in I} x_i) = \bigwedge_{i \in I}(a \multimap x_i)$ | $(iv)$. | $(\bigvee_{i \in I} x_i) \multimap c = \bigwedge_{i \in I}(x_i \multimap c)$ |
| $(v)$. | $a \multimap (b \multimap c) = b \multimap (a \multimap c)$ | $(vi)$. | $1 \multimap a = a$ |
| $(vii)$. | $c \leq_C (c \multimap a) \multimap a$ | $(viii)$. | $((c \multimap a) \multimap a) \multimap a = c \multimap a$ $\square$ |

In particular, from the above properties, it is not hard to check that for all $a \in C$, $\lambda x.(x \multimap a) \multimap a \in \mathtt{uco}(C)$.

## 3. COMPLETENESS IN ABSTRACT INTERPRETATION

One key feature of completeness in abstract interpretation is that this property uniquely depends upon the abstraction function (cf. [Giacobazzi et al. 1998]). Let us consider the simple case of an abstract interpretation specified by an abstract domain $A \in \mathtt{uco}(C)$ and an abstract function $f^\sharp : A \longmapsto A$ approximating a concrete semantic function $f : C \longmapsto C$. Then, $f^\sharp$ is (sound and) complete if $\rho \circ f = f^\sharp \circ \rho$, where $\rho$ is the uco associated with $A$. It is easily seen that if $f^\sharp$ is complete then the best correct approximation of $f$ in $A$, i.e. $\rho \circ f : A \longmapsto A$, is complete as well, and, in this case, $f^\sharp$ indeed coincides with $\rho \circ f$. Thus, given an abstract domain $A \in \mathtt{uco}(C)$, one can define a complete abstract semantic function $f^\sharp : A \longmapsto A$ over $A$ if and only if $\rho \circ f : A \longmapsto A$ is complete. Therefore, an abstract domain $A$ is complete for $f$ iff $\rho \circ f = \rho \circ f \circ \rho$ holds. This simple observation makes completeness an abstract domain property, namely a characteristic of the abstract domain.

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

$$\mathcal{R}_F(\eta) \stackrel{\text{def}}{=} \mathcal{M}\Big( \bigcup_{f \in F, a \in \eta} max(f^{-1}(\downarrow a))\Big)$$

where $f^{-1}(\downarrow a) = \{x \in C \mid f(x) \leq a\}$. The next theorem recalls the main result.

THEOREM [GIACOBAZZI ET AL. 2000]. *Let $F \subseteq C \overset{c}{\longmapsto} C$. A domain $\eta \in \mathtt{uco}(C)$ is complete for $F$ iff $\eta \sqsubseteq \mathcal{R}_F(\eta)$. Moreover, $\mathcal{R}_F$ is co-additive.*

Therefore, the greatest (most abstract) domain which includes $\rho$ and which is complete for $F$ is $gfp(\lambda x.\rho \sqcap \mathcal{R}_F(x))$. This domain is called the *(absolute) complete shell* of $\rho$ for $F$. By a standard result ([Cousot and Cousot 1979b]), complete shells are also fixpoint complete, i.e., , if $\eta$ is the complete shell of $\rho$ for a continuous function $f$, then it holds $\eta(lfp(f)) = lfp(\eta \circ f)$, while the converse, in general, does not hold.

## 4. COMPLETENESS IN LOGICAL FORM

In this section we consider completeness problems on quantales. Let $\langle C_{\leq}, \otimes \rangle$ be a unital, commutative quantale playing the role of concrete interpretation, where $C$ is the concrete semantic domain provided with a semantic operation $\otimes : C \times C \longmapsto C$. In this logical framework, we prove that completeness can be achieved by enhancing domains with suitable objects representing linear implication. We distinguish between two forms of completeness: *absolute completeness* and *disjunctive completeness.*

### 4.1 Absolute completeness

Given an abstraction $\rho \in \mathtt{uco}(C)$, we say that $\rho$ is complete for $\otimes$ when for all concrete objects $x, y \in C$, it holds $\rho(\rho(x) \otimes \rho(y)) = \rho(x \otimes y)$, which is denoted, in a more compact way, by the equation $\rho \circ \otimes \circ \langle \rho, \rho \rangle = \rho \circ \otimes$. Since $\otimes$ is commutative, the above equation boils down to a simpler completeness problem, as shown in the next proposition. This reduction will allows us to define the completeness problem in terms of a suitable set of unary functions. To this aim, let us define, for any $\eta \in \mathtt{uco}(C)$, the set of functions $F_\eta \subseteq C \stackrel{c}{\longmapsto} C$ as follows:

$$F_\eta \stackrel{\text{def}}{=} \{\lambda x.x \otimes y \mid y \in \eta\}.$$

PROPOSITION 4.1. *Let $\langle C_\leq, \otimes \rangle$ be a commutative quantale and $\rho \in \mathtt{uco}(C)$. The following are equivalent.*

—*$\rho$ is complete for $\otimes$, i.e., $\rho \circ \otimes \circ \langle \rho, \rho \rangle = \rho \circ \otimes$*

—*$\rho \circ \otimes \circ \langle \rho, id \rangle = \rho \circ \otimes$*

—*$\rho$ is complete for $F_C$*

PROOF. We first show that the completeness problem $\rho \circ \otimes \circ \langle \rho, \rho \rangle = \rho \circ \otimes$ is equivalent to $\rho \circ \otimes \circ \langle \rho, id \rangle = \rho \circ \otimes$. Assume that $\rho \circ \otimes \circ \langle \rho, \rho \rangle = \rho \circ \otimes$. By monotonicity, $\rho \circ \otimes \leq \rho \circ \otimes \circ \langle \rho, id \rangle \leq \rho \circ \otimes \circ \langle \rho, \rho \rangle = \rho \circ \otimes$, where the last step holds by hypothesis.

Assume now that $\rho \circ \otimes \circ \langle \rho, id \rangle = \rho \circ \otimes$. By monotonicity, $\rho \circ \otimes \leq \rho \circ \otimes \circ \langle \rho, \rho \rangle = \rho \circ \otimes \circ \langle \rho, id \rangle \circ \langle id, \rho \rangle = \rho \circ \otimes \circ \langle id, \rho \rangle$, where the last step holds by hypothesis. By commutativity of $\otimes$, this is equal to $\rho \circ \otimes \circ \langle \rho, id \rangle = \rho \circ \otimes$ by hypothesis.

The completeness problem $\rho \circ \otimes \circ \langle \rho, \rho \rangle = \rho \circ \otimes$ can therefore be reformulated as follows: $\forall x, y \in C$, $\rho(\rho(x) \otimes y) = \rho(x \otimes y)$. By definition of completeness, this is equivalent to say that $\rho$ is complete for the set of unary functions $F_C = \{\lambda x.x \otimes y \mid y \in C\}$. $\square$

COROLLARY 4.2. *Let $\langle C_\leq, \otimes \rangle$ be a commutative quantale and $\rho \in \mathtt{uco}(C)$. The complete shell of $\rho$ for $\otimes$ is $gfp(\lambda x.\rho \sqcap \mathcal{R}_{F_C}(x))$.*

PROOF. Since the $\otimes$ operation is argumentwise additive, each function in $F_C$ is additive as well, and therefore continuous. Hence, we are in the position to directly apply Theorem 3.1, from which it immediately follows that the complete shell of $\rho$ for $\otimes$ is $gfp(\lambda x.\rho \sqcap \mathcal{R}_{F_C}(x))$. $\square$

The previous corollary is a direct consequence of Theorem 3.1, obtained by exploiting the alternative definition of completeness given in Prop. 4.1 in terms of $F_C$. It states that the complete shell of a given domain $\rho$ for $\otimes$ can be constructively built by using the operator $\mathcal{R}_{F_C}$. The aim of this section is to show that this operator, and, more generally, the family of operators $\mathcal{R}_{F_\eta}$, for any $\eta \in \mathtt{uco}(C)$, can all be easily described in terms of sets of linear implications. This logical view will help us in computing and representing the above complete shells. Let us define a domain operator $\stackrel{\curlywedge}{\multimap} : \mathtt{uco}(C) \times \mathtt{uco}(C) \longmapsto \mathtt{uco}(C)$ by lifting linear implication $\multimap$ to abstract domains as follows: For any $A, B \in \mathtt{uco}(C)$:

$$A \stackrel{\curlywedge}{\multimap} B \stackrel{\text{def}}{=} \mathcal{M}(\{a \multimap b \in C \mid a \in A, b \in B\})$$

Hence, $A \stackrel{\curlywedge}{\multimap} B$ is defined to be the most abstract domain in $\mathtt{uco}(C)$ containing all the linear implications from $A$ to $B$.

THEOREM 4.3. *Let $\langle C_\leq, \otimes \rangle$ be a unital, commutative quantale and $\mu, \eta \in \mathtt{uco}(C)$. Then $\mathcal{R}_{F_\mu}(\eta) = \mu \stackrel{\curlywedge}{\multimap} \eta$.*

PROOF. We have to prove that $\mathcal{R}_{F_\mu}(\eta) = \mathcal{M}(\{y \multimap a \mid y \in \mu, a \in \eta\})$.

$$\begin{aligned}
\mathcal{R}_{F_\mu}(\eta) &= \mathcal{M}(\cup_{f \in F_\mu, a \in \eta} max(f^{-1}(\downarrow a))) && [\text{ by definition of } \mathcal{R}_{F_\mu} ]\\
&= \mathcal{M}(\cup_{f \in F_\mu, a \in \eta} max(\{x \in C \mid f(x) \leq a\}))\\
&= \mathcal{M}(\cup_{y \in \mu, a \in \eta} max(\{x \in C \mid x \otimes y \leq a\})) && [\text{ by definition of } F_\mu ]\\
&= \mathcal{M}(\cup_{y \in \mu, a \in \eta} max(\{x \in C \mid y \otimes x \leq a\})) && [\text{ by commutativity of } \otimes ]\\
&= \mathcal{M}(\cup_{y \in \mu, a \in \eta} max(\{x \in C \mid x \leq y \multimap a\})) && [\text{ by definition of } \multimap ]\\
&= \mathcal{M}(\cup_{y \in \mu, a \in \eta} \{ y \multimap a \})\\
&= \mu \overset{\curlywedge}{\multimap} \eta. \quad \square
\end{aligned}$$

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

PROPOSITION 4.4. *For all* $A \in \mathtt{uco}(C)$ *and* $\{B_i\}_{i \in I} \subseteq \mathtt{uco}(C)$*, we have:*

*(1)* $A \overset{\curlywedge}{\multimap} (\bigsqcap_{i \in I} B_i) = \bigsqcap_{i \in I} (A \overset{\curlywedge}{\multimap} B_i)$;
*(2)* $A \overset{\curlywedge}{\multimap} \top = \top$
*(3)* $C \overset{\curlywedge}{\multimap} A \sqsubseteq A$
*(4)* $C \overset{\curlywedge}{\multimap} A = C \overset{\curlywedge}{\multimap} (C \overset{\curlywedge}{\multimap} A)$

PROOF. Points (1) and (2) are immediate.

(3). By Prop. 2.1(vi), for all $a \in A$ it holds $1 \multimap a = a$. Since $1 \in C$, it follows that $a = 1 \multimap a \in C \overset{\curlywedge}{\multimap} A$, and therefore $C \overset{\curlywedge}{\multimap} A \sqsubseteq A$.

(4). By the previous point $C \overset{\curlywedge}{\multimap} A \sqsubseteq A$, and therefore, by monotonicity, we have that $C \overset{\curlywedge}{\multimap} (C \overset{\curlywedge}{\multimap} A) \sqsubseteq C \overset{\curlywedge}{\multimap} A$. For the other direction, consider an element of $\{c \multimap a \in C \mid c \in C, a \in C \overset{\curlywedge}{\multimap} A\}$. By definition, such an element is of the forms: $c \multimap \bigwedge_{i \in I}(d_i \multimap a_i)$, for suitable $c, d_i \in C$, and $a_i \in A$ for all $i \in I$.

$$\begin{aligned}
c \multimap \textstyle\bigwedge_{i \in I}(d_i \multimap a_i) &= \textstyle\bigwedge_{i \in I}(c \multimap (d_i \multimap a_i)) && [\text{ by Prop. 2.1(iii) }]\\
&= \textstyle\bigwedge_{i \in I}(d_i \otimes c) \multimap a_i) && [\text{ by Prop. 2.1(ii) }]
\end{aligned}$$

Since, for all $i \in I$, $d_i \otimes c \in C$ and $a_i \in A$, $(d_i \otimes c) \multimap a_i \in C \overset{\curlywedge}{\multimap} A$. The result follows from monotonicity of the Moore closure constructor. $\square$

It is worth noting that, by Points (1), (3) and (4) above, the function $\lambda x.C \overset{\curlywedge}{\multimap} x : \mathtt{uco}(C) \longmapsto \mathtt{uco}(C)$ is monotone, reductive and idempotent, and therefore it is a lower closure operator on $\mathtt{uco}(C)$.

The following result proves that the greatest (i.e. most abstract) domain which includes $A$ and which is complete for $\otimes$ is the set of linear implications between the concrete domain and the abstract one. This provides a first representation result for the objects of complete abstractions of quantales.

THEOREM 4.5. *Let* $\langle C_{\leq}, \otimes \rangle$ *be a unital, commutative quantale and* $A \in \mathtt{uco}(C)$. *The complete shell of* $A$ *for* $\otimes$ *is* $C \overset{\curlywedge}{\multimap} A$.

PROOF. By Corollary 4.2, the complete shell of $A$ for $\otimes$ is $\mathit{gfp}(\lambda x.A \sqcap \mathcal{R}_{F_C}(x))$ and, by Proposition 4.3, this is $\mathit{gfp}(\lambda x.A \sqcap C \overset{\curlywedge}{\multimap} x)$. Let us now show that $\mathit{gfp}(\lambda x.A \sqcap$

$C \overset{\curlywedge}{\multimap} x) = C \overset{\curlywedge}{\multimap} A$ by computing the Kleene's iteration sequences.

$$
\begin{array}{llll}
(\lambda x. A \sqcap C \overset{\curlywedge}{\multimap} x)(\top) & = A \sqcap C \overset{\curlywedge}{\multimap} \top & = A \sqcap \top = A & [\text{ by Prop. 4.4(2) }] \\
(\lambda x. A \sqcap C \overset{\curlywedge}{\multimap} x)(A) & = A \sqcap C \overset{\curlywedge}{\multimap} A & = C \overset{\curlywedge}{\multimap} A & [\text{ by Prop. 4.4(3) }] \\
(\lambda x. A \sqcap C \overset{\curlywedge}{\multimap} x)(C \overset{\curlywedge}{\multimap} A) & = A \sqcap C \overset{\curlywedge}{\multimap} (C \overset{\curlywedge}{\multimap} A) & = A \sqcap C \overset{\curlywedge}{\multimap} A & [\text{ by Prop. 4.4(4) }] \\
 & & = C \overset{\curlywedge}{\multimap} A & [\text{ by Prop. 4.4(3) }]
\end{array}
$$

Therefore $C \overset{\curlywedge}{\multimap} A$ is the greatest fixpoint of $\lambda x. A \sqcap C \overset{\curlywedge}{\multimap} x$. $\square$

The importance of this result is that, in case of quantales, the fixpoint construction of complete abstract domain converges in two steps. Moreover, the objects of complete domains enjoy a clean logical characterization in terms of linear implication. The following proposition provides an explicit definition of the abstraction function associated with a complete abstract domain in quantales. This is achieved by exploiting the properties of linear implication.

Theorem 4.6. *Let $\langle C_{\leq}, \otimes \rangle$ be a unital, commutative quantale, $A \in \mathtt{uco}(C)$, and let $\rho \in \mathtt{uco}(C)$ be the uco associated with $C \overset{\curlywedge}{\multimap} A$. Then, for all $c \in C$:*

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

Proof. By Prop. 4.4(1), it holds $C \overset{\curlywedge}{\multimap} A = \bigsqcap_{a \in A} C \overset{\curlywedge}{\multimap} \{ \top_C, a \}$. We show that the closure operator $\rho_a$ associated with $C \overset{\curlywedge}{\multimap} \{ \top_C, a \}$ is $\rho_a(c) = (c \multimap a) \multimap a$. Therefore, by definition, $\rho(c) = \bigwedge_{a \in A} \rho_a(c) = \bigwedge_{a \in A} (c \multimap a) \multimap a$. We know that $\rho_a \in \mathtt{uco}(C)$. We first show that $\rho_a$ is complete for $\otimes$. By Prop. 4.1, we have to show that $\rho_a$ is complete for $F_C$, i.e., for any $y \in C$, it holds $\rho_a(\rho_a(x) \otimes y) = \rho_a(x \otimes y)$. We show that $\rho_a(x) \otimes y \leq \rho_a(x \otimes y)$, the other direction follows by monotonicity. By definition of implication, it holds $y \otimes (y \multimap (x \multimap a)) \leq x \multimap a$, 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 that:

$$
\begin{array}{l}
y \otimes (y \multimap (x \multimap a)) \otimes ((x \multimap a) \multimap a) \leq a \\
y \otimes ((x \otimes y) \multimap a) \otimes ((x \multimap a) \multimap a) \leq a \\
y \otimes ((x \multimap a) \multimap a) \leq ((x \otimes y) \multimap a) \multimap a \\
y \otimes \rho_a(x) \leq \rho_a(x \otimes y)
\end{array}
$$

We prove that $\rho_a$ is the least domain complete for $\otimes$ which contain $a$. Suppose, by contrary, that there exists $\alpha \in \mathtt{uco}(C)$ such that $\alpha(a) = a$ and $\alpha$ is strictly more abstract than $\rho$. Therefore, there exists $c \in C$ such that $\alpha(c) > \rho(c)$, that is $\alpha(c) > (c \multimap a) \multimap a$. We show that $\beta$ is not complete for $c \otimes (c \multimap a)$. By hypothesis, $\beta(c) \otimes \beta(c \multimap a) \geq \beta(c) \otimes \rho(c \multimap a) = \beta(c) \otimes ((c \multimap a) \multimap a) \multimap a = \beta(c) \otimes (c \multimap a)$. By definition of $\multimap$, $(c \multimap a) \multimap a$ is the least object $d$ such that $d \otimes (c \multimap a) \leq a$. Since $\beta(c) > (c \multimap a) \multimap a$, it follows that $\beta(c) \otimes (c \multimap a) \not\leq a$, that is $\beta(c) \otimes \beta(c \multimap a) \not\leq a$. But $\beta(c \otimes (c \multimap a)) \leq \beta(a) = a$ and this is a contradiction. $\square$

### 4.2 Disjunctive completeness

In this section we introduce a weaker form of completeness, which is obtained by considering that, in many cases of abstract computations with binary operations, only one of the two arguments may be an abstract object. Following the previous approach for completeness, we want to formalize the intuition that an abstraction $\rho$ can be removed in any computation involving $\otimes$, providing that the computation involves at least one abstract object each time $\otimes$ is applied. For instance, in logic programming this is the case when $\otimes$ is unification and the semantics is made by unifying concrete and abstract objects, which respectively come from program's clauses and the objects in the current interpretation (see [Barbuti et al. 1993; Codish et al. 1994]).

Given an abstract domain $\rho \in \mathtt{uco}(C)$, a *disjunctive completeness problem* is formulated as follows:

$$\text{if either } x \in \rho \text{ or } y \in \rho \text{ then } \rho(\rho(x) \otimes \rho(y)) = \rho(x \otimes y).$$

This is equivalent to requiring that $\rho$ is a solution of the following equalities:

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

which, in turn, is equivalent to say that $\rho \circ \otimes \circ \langle \rho, \rho \rangle = \rho \circ \otimes \circ \langle \rho, id \rangle$ by exploiting the commutativity of $\otimes$.

Given an abstract domain $A \in \mathtt{uco}(C)$, we are interested in characterizing the most abstract domain $\rho \in \mathtt{uco}(C)$ which is more concrete than $A$ and satisfies Equation (1). This domain is called the *disjunctive complete shell* of $A$ (for $\otimes$). Disjunctive completeness problems can be solved by using the same method of absolute completeness on quantales, i.e., by resorting to a recursive abstract domain equation involving linear implication. The next theorem shows a (recursive) characterization of the solutions.

THEOREM 4.7. *Let $\langle C_{\leq}, \otimes \rangle$ be a unital, commutative quantale and $\rho \in \mathtt{uco}(C)$. The following are equivalent.*

— $\rho \circ \otimes \circ \langle \rho, \rho \rangle = \rho \circ \otimes \circ \langle \rho, id \rangle = \rho \circ \otimes \circ \langle id, \rho \rangle$;

— *$\rho$ is complete for $F_\rho = \{\lambda x.x \otimes y \mid y \in \rho\}$;*

— $\rho = \rho \sqcap (\rho \overset{\curlywedge}{\multimap} \rho)$.

PROOF. By commutativity of $\otimes$, solving Eq. (1) is equivalent to solving the equality $\rho \circ \otimes \circ \langle \rho, id \rangle = \rho \circ \otimes \circ \langle \rho, \rho \rangle$. By definition, this means that for all $x \in C$ and $y \in \rho$ it holds $\rho(x \otimes y) = \rho(\rho(x) \otimes y)$, that is to say that $\rho$ is complete for the set of unary functions $F_\rho = \{\lambda x.x \otimes y \mid y \in \rho\}$. By Theorem 3.1, $\rho$ is complete for $F_\rho$ iff $\rho \sqsubseteq \mathcal{R}_{F_\rho}(\rho)$. By Theorem 4.3, this is equivalent to say that $\rho \sqsubseteq \rho \overset{\curlywedge}{\multimap} \rho$, and therefore $\rho = \rho \sqcap \rho \overset{\curlywedge}{\multimap} \rho$. $\square$

COROLLARY 4.8. *Let $\langle C_{\leq}, \otimes \rangle$ be a unital, commutative quantale and $A \in \mathtt{uco}(C)$. The disjunctive complete shell of $A$ (for $\otimes$) is $gfp(\lambda X.A \sqcap X \sqcap (X \overset{\curlywedge}{\multimap} X))$.*

PROOF. Since the operator $\lambda x.x \overset{\curlywedge}{\multimap} x$ is clearly monotone[1], from Theorem 4.7 it follows that the greatest domain which includes $A$ and which is disjunctive complete is $gfp(\lambda x.A \sqcap x \sqcap x \overset{\curlywedge}{\multimap} x)$. $\square$

[1] It is worth noting that, even if $\mathcal{R}_{F_\rho} = \lambda x.\rho \overset{\curlywedge}{\multimap} x$ is co-additive for any $\rho \in \mathtt{uco}(C)$, it does not follow that the operator $\lambda x.x \overset{\curlywedge}{\multimap} x$ is co-additive as well. This is a consequence of the fact that the set of functions $F_\rho$ for which we want to be complete, change at each iteration.

Therefore, the most abstract solution to a disjunctive completeness problem for a domain $A \in \mathtt{uco}(C)$ is exactly the greatest solutions of the following (recursive) abstract domain equation.

$$X = A \sqcap X \sqcap (X \stackrel{\curlywedge}{\multimap} X) \tag{2}$$

## 5. OPTIMAL SEMANTICS IN LOGIC PROGRAM ANALYSIS

In this section, we determine the complete shell of any logic program property w.r.t. a bottom-up semantics characterizing computed answer substitutions, which turns out to be equivalent to the well-known s-semantics [Falaschi et al. 1989]. This complete semantics can be thought of as a logic program semantics which is "optimal" (i.e. neither too concrete nor too abstract) for characterizing a given property of interest. The problem of determining optimal semantics for program analysis was first raised in [Giacobazzi 1996]. We show that optimal semantics can be obtained by solving a completeness problem relatively to the operation of unification on sets of substitutions.

A concrete interpretation is specified by a concrete domain $C$ and a concrete function $f_P : C \longmapsto C$. The semantics $[\![P]\!]$ of a program $P$ is defined as $\mathit{lfp}(f_P)$. An abstract interpretation is given by a corresponding abstract domain $A$ and an abstract function $f_P^A : A \longmapsto A$. The abstract semantics $[\![P]\!]^A$ of a program $P$ is then defined as $\mathit{lfp}(f_P^A)$. Note that, in this framework, we assume that concrete semantics are given as least fixpoint of some semantic operator and, more importantly, that this condition holds for abstract semantics too. This requirement ensures us that, even in the abstract context, abstract semantics can be constructively characterized. Moreover, we assume to deal with abstract functions which are induced by the concrete one, i.e., $f_P^A$ is the best correct approximation of $f_P$ in $A$. As a consequence, whenever the concrete semantic function $f_P$ is a recursive function (i.e., it is effectively given), then the abstract semantic function is recursive too, providing the abstraction function to be effectively defined. This ensures us that optimal semantics can be constructively characterized and, more importantly, that they can be effectively computed.

Given an abstract domain $A$ representing the property we are interested in analyzing, we look for the most abstract domain $B \in \mathtt{uco}(C)$ such that, for any program $P$, the property $A$ is observable in the abstract semantics induced by $B$ without approximation errors, i.e.,

$$\alpha_A([\![P]\!]) = \alpha_A([\![P]\!]^B)$$

or, equivalently, by exploiting the definitions of concrete and abstract semantics, the most abstract solution to the equation:

$$\alpha_A(\mathit{lfp}(f_P)) = \alpha_A(\mathit{lfp}(\alpha_B \circ f_P)) \tag{3}$$

Whenever such a most abstract domain $B$ does exits, we call it *optimal semantics* for $A$ (relatively to $f_P$). Therefore, optimal semantics contain the least amount of information that we need in order to obtain precise abstract computations for the property $A$. We proceed as follows: we first determine the complete shell of any property of idempotent substitutions w.r.t. a bottom-up semantics characterizing computed answer substitutions. The chosen semantics turns out to be equivalent to the well-known s-semantics [Falaschi et al. 1989]. The complete shell is obtained by solving a completeness problem relatively to the operation of unification on the quantale of idempotent substitutions. Then we show that, under nonrestrictive hypotheses, the complete shell constructed in this way is also the most abstract domain for which Equation (3) holds.

## 5.1 Logic programming notation

Let $\mathcal{V}$ be an infinite, recursively enumerable (r.e.) set of variables, $\Sigma$ be a set of function symbols and $\pi$ be a set of predicate symbols, defining a r.e. first-order language $\mathcal{L}$. $\mathtt{Term}$ denotes the set of terms of $\mathcal{L}$. A *substitution* is a finite mapping from $\mathcal{V}$ to terms in $\mathcal{L}$. If $s$ is any syntactic object and $\sigma$ and $\theta$ are substitutions, then $\sigma(s)$ denotes the application of $\sigma$ to $s$, $\sigma \circ \theta$ denotes the standard composition of $\theta$ and $\sigma$ (i.e., $\sigma \circ \theta = \lambda x.\sigma(\theta(x))$), and $vars(s)$ denotes the set of variables occurring in $s$. A term $t$ is *ground* if $vars(t) = \emptyset$. The set of *idempotent* substitutions modulo renaming (i.e., $\theta \sim \sigma$ if and only if there exist $\beta$ and $\delta$ such that $\theta = \sigma \circ \beta$ and $\sigma = \theta \circ \delta$) on $\mathcal{L}$ is denoted by $\mathtt{Sub}$. Both $\mathtt{Sub}$ and $\mathtt{Term}$ are partially ordered by instantiation, denoted by $\preceq$, i.e., $a \preceq b$ iff $\exists \theta \in \mathtt{Sub}.\ a = b\theta$. By adding to $\mathtt{Sub}$ an extra object $\tau$ as least element, one gets a complete lattice $\langle \mathtt{Sub}^{\tau}, \preceq, \vee, \wedge, \epsilon, \tau \rangle$, where $\vee$ is the least general anti-instance, $\wedge$ is the standard unification and $\epsilon$ is the empty substitution (see [Palamidessi 1990; Lassez et al. 1988] for more details). The set of most general atoms is given by $\mathtt{GAtom} \stackrel{\text{def}}{=} \{p(\bar{X}) \mid p \in \pi\}$, where $\bar{X}$ is a tuple of distinct variables. We consider logic programs in normalized form, that is, a generic Horn clause is $p(\bar{X}) :\!- c, q_1(\bar{X}_1), \ldots, q_n(\bar{X}_n)$, where all the tuples of variables are distinct. The constraint $c$ is intended as a (finite) set of term equalities for variable binding (e.g., $X = f(Y), Z = g(Y)$). By abusing notation, we let $c$ denote also the set of most general unifiers which solve it.

## 5.2 $T_P$-completeness

Our basic semantic structure is the unital, commutative quantale $\langle \wp(\mathtt{Sub})_{\subseteq}, \otimes \rangle$, where $\langle \wp(\mathtt{Sub}), \subseteq \rangle$ is a complete lattice, $\otimes : \wp(\mathtt{Sub}) \times \wp(\mathtt{Sub}) \longmapsto \wp(\mathtt{Sub})$ is the obvious lifting of unification $\wedge$ to sets of substitutions, defined as:

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

It is immediate to check that $\langle \wp(\mathtt{Sub})_{\subseteq}, \otimes \rangle$ actually is a unital, commutative quantale, and $\{\epsilon\} \in \wp(\mathtt{Sub})$ is the unit. In the following, we will slightly abuse the notation by applying the operation $\otimes$ also to substitutions.

We define a concrete semantics $T_P$ which is a “simplified” version of the immediate consequences operator $T_P^s$ of s-semantics. The $T_P$ operator is defined by using only the operations of unification $\otimes$ and union of sets of idempotent substitutions and it does not take into account variable projection (the existential quantification over variables). This will simplify the technical development of this section. Note that, the s-semantics operator $T_P^s$ can be recovered from $T_P$ by a simple step of abstract interpretation considering the closure of sets of substitutions over variables projection. More importantly, we prove that the most abstract domain $\rho$ which refines a given domain $A$ and it is complete for the fixpoint of $T_P$, i.e. $\rho(lfp(T_P)) = lfp(\rho \circ T_P)$, turns out to be the the most abstract domain which refines $A$ and it is complete for the fixpoint of $T_P^s$. The following result is essential to prove this statement, showing that, under certain conditions, completeness is independent from the choice of the concrete semantics.

PROPOSITION 5.1. *Let $\alpha \in \mathtt{uco}(C)$ and $f, g : C \longmapsto C$ such that $f \sqsubseteq g \sqsubseteq \alpha \circ f$. Then, $\alpha$ is complete for $f$ iff $\alpha$ is complete for $g$.*

Let $\exists$ denotes the closure under variable renaming of sets of substitutions. Then we have that $T^s_P = \exists \circ T_P$. If $\alpha$ denotes the complete shell of a domain $A$ abstracting computed answer substitutions relatively to $T_P$, because $T_P \sqsubseteq \exists \circ T_P = T^s_P \sqsubseteq \alpha \circ T_P$ holds, by Prop. 5.1, we get that $\alpha$ turns out to be the complete shell of $A$ relatively to $T^s_P$. We will exploit this result for computing the complete shell of abstract domains w.r.t. s-semantics.

We consider a concrete semantic domain $\mathtt{CInt} \stackrel{\text{def}}{=} \mathtt{GAtom} \longmapsto \wp(\mathtt{Sub})$ of functions, as usual called interpretations, which map most general atoms to sets of substitutions. $\mathtt{CInt}$ is trivially a complete lattice, once ordered pointwise. Often, we will find convenient to denote an interpretation $I \in \mathtt{CInt}$ by the set $\{\langle p(\bar{X}), I(p(\bar{X}))\rangle \mid p \in \pi\}$. Then, for any program $P$, the immediate consequences operator $T_P : \mathtt{CInt} \longmapsto \mathtt{CInt}$ is defined as follows, for any $I \in \mathtt{CInt}$:

$$T_P(I)(p(\bar{Y})) \stackrel{\text{def}}{=} \bigcup_{C \lll P} (c \otimes \bigotimes_{i=1..n} I(q_i(\bar{X}_i)) \otimes (\bar{Y} = \bar{X})),$$

where $C = p(\bar{X}) :\!- c, q_1(\bar{X}_1), \ldots, q_n(\bar{X}_n)$. Here, $C \lll P$ denotes that the clause $C$ of $P$ is renamed apart with fresh variables and, as usual, also $I(q_i(\bar{X}_i))$ is renamed apart. As before, $\bar{Y} = \bar{X}$ denotes the set of most general unifiers which solve it.

If $\rho \in \mathtt{uco}(\wp(\mathtt{Sub}))$ is any abstraction on sets of substitutions, a corresponding abstraction on interpretations $\wr\rho\wr \in \mathtt{uco}(\mathtt{CInt})$ which acts accordingly to $\rho$ can be defined as follows, for any $I \in \mathtt{CInt}$:

$$\wr\rho\wr(I) \stackrel{\text{def}}{=} \{\langle p(\bar{X}), \rho(c)\rangle \mid \langle p(\bar{X}), c\rangle \in I\}.$$

Note that $\wr\cdot\wr$ is trivially monotone, i.e., for all $\rho, \eta \in \mathtt{uco}(\wp(\mathtt{Sub}))$, if $\rho \sqsubseteq \eta$ then $\wr\rho\wr \sqsubseteq \wr\eta\wr$.

Given a property of interest $\rho \in \mathtt{uco}(\wp(\mathtt{Sub}))$, our goal is to find the most abstract domain which contains $\rho$ (more precisely, $\wr\rho\wr$) and is complete for any $T_P$ operator. Our strategy consists in characterizing the complete shell of $\rho$ for the basic operations involved in the definition of $T_P$, and then in showing that, under reasonable hypotheses, this domain turns out to be the right one. Since every abstract domain is trivially always complete for l.u.b.'s, it turns out that union of sets of substitutions is troubleless, and therefore it is enough to concentrate on unification $\otimes$. Indeed, the following result shows that completeness for $\otimes$ implies completeness for any $T_P$.

THEOREM 5.2. *Let $\rho \in \mathtt{uco}(\wp(\mathtt{Sub}))$. If $\rho$ is complete for $\otimes$ then $\wr\rho\wr$ is complete for any $T_P$.*

PROOF. We have to show that for each $I \in \mathtt{GAtom} \longmapsto \wp(\mathtt{Sub})$ it holds:

$$\{\langle p(\bar{X}), \rho(c)\rangle \mid \langle p(\bar{X}), c\rangle \in T_P(I)\} = \{\langle p(\bar{X}), \rho(c)\rangle \mid \langle p(\bar{X}), c\rangle \in T_P(\wr\rho\wr(I))\}.$$

Let $I \in \mathtt{GAtom} \longmapsto \wp(\mathtt{Sub})$. Then, for all $C = p(\bar{X}) :\!- c, q_1(\bar{X}_1), \ldots, q_n(\bar{X}_n) \lll P$

and $\langle q_i(\bar{X}_i), c_i\rangle \in I$, for $i = 1..n$, the following equalities hold.

$$\begin{array}{lll} & \rho(\bigcup_{C \lll P}(c \otimes \bigotimes_{i=1..n} \rho(c_i) \otimes (\bar{Y} = \bar{X}))) & \\ = & \rho(\bigcup_{C \lll P} \rho(c \otimes \bigotimes_{i=1..n} \rho(c_i) \otimes (\bar{Y} = \bar{X}))) & [\text{ since } \rho \text{ is complete for } \bigcup\ ] \\ = & \rho(\bigcup_{C \lll P}(\rho(c) \otimes \rho(\bigotimes_{i=1..n} \rho(c_i) \otimes \rho(\bar{Y} = \bar{X})))) & [\text{ since } \rho \text{ is complete for } \otimes\ ] \\ = & \rho(\bigcup_{C \lll P} \rho(c \otimes \bigotimes_{i=1..n} c_i \otimes (\bar{Y} = \bar{X}))) & [\text{ since } \rho \text{ is complete for } \otimes\ ] \\ = & \rho(\bigcup_{C \lll P}(c \otimes \bigotimes_{i=1..n} c_i \otimes (\bar{Y} = \bar{X}))). & [\text{ since } \rho \text{ is complete for } \bigcup\ ] \end{array}$$

□

By Theorem 4.5, given $\rho \in \texttt{uco}(\wp(\texttt{Sub}))$, the complete shell of $\rho$ for the quantale operation $\otimes$ exists and it is the domain $\wp(\texttt{Sub}) \overset{\curlywedge}{\multimap} \rho$ of linear implications from $\wp(\texttt{Sub})$ to $\rho$. The domain $\wp(\texttt{Sub}) \overset{\curlywedge}{\multimap} \rho$ turns out to be the right one for characterizing optimal semantics, whenever the abstract domain $\rho$ satisfies the following weak decidability property.

Definition 5.3. An abstract domain $\rho \in \texttt{uco}(\wp(\texttt{Sub}))$ is *decidable* if there exists $X \subseteq \rho$ such that $\mathcal{M}(X) = \rho$ and any $S \in X$ is a r.e. set.

It should be clear that decidability is a reasonable requirement for most abstract domains used in program analysis. In fact, for such a decidable abstraction (of sets) of substitutions, an effective procedure for checking whether a substitution belongs to (is approximated by) an abstract object is available. Moreover, the above notion of decidable abstract domain is much weaker than the standard notion of decidable property. While the latter one is a problem of computability (i.e., the whole abstract domain is r.e. itself), the first one is only a matter of representability (i.e., we can represent each abstract object by a r.e. set). It is worth noting that, most of the abstract domains used in logic program analysis (e.g., the basic domains for groundness and sharing properties) are decidable, while the corresponding abstract properties are not. Moreover, given a nondecidable domain $\rho$, in order to obtain a decidable domain it suffices to consider the abstract domain $\mathcal{M}(\rho \cap \{S \in \wp(\texttt{Sub}) \mid S \text{ is r.e.}\})$.

As announced, the following key result shows that decidable domains always admit optimal semantics, and, more importantly, that the optimal semantics does coincide with the complete shell for the operation of unification $\otimes$. In other words, the most abstract domain $B$ which satisfies Equation (3) for any program $P$, is exactly the most abstract domain which is complete for $\otimes$.

Theorem 5.4. *Let $\rho \in \texttt{uco}(\wp(\texttt{Sub}))$ be a decidable abstract domain. The optimal semantics for $\wr\rho\wr$ is $\wr\wp(\texttt{Sub}) \overset{\curlywedge}{\multimap} \rho\wr$.*

Proof. By Theorem 4.5, $\wp(\texttt{Sub}) \overset{\curlywedge}{\multimap} \rho \sqsubseteq \rho$ and is complete for $\otimes$. Therefore, by Theorem 5.2 we have that $\wr\wp(\texttt{Sub}) \overset{\curlywedge}{\multimap} \rho\wr$ is complete for any $T_P$, and therefore fixpoint complete, i.e., :

$$\wr\wp(\texttt{Sub}) \overset{\curlywedge}{\multimap} \rho\wr(\mathit{lfp}(T_P)) = \wr\wp(\texttt{Sub}) \overset{\curlywedge}{\multimap} \rho\wr(\mathit{lfp}(\wr\wp(\texttt{Sub}) \overset{\curlywedge}{\multimap} \rho\wr \circ T_P)).$$

Since $\wp(\texttt{Sub}) \overset{\curlywedge}{\multimap} \rho \sqsubseteq \rho$, by applying $\wr\rho\wr$ on both sides, it holds:

$$\wr\rho\wr(\mathit{lfp}(T_P)) = \wr\rho\wr(\mathit{lfp}(\wr\wp(\texttt{Sub}) \overset{\curlywedge}{\multimap} \rho\wr \circ T_P)).$$

We now show that $\wr\wp(\texttt{Sub}) \overset{\curlywedge}{\multimap} \rho\wr$ is the most abstract domain with this property, for any program $P$. By definition of $\overset{\curlywedge}{\multimap}$, a generator set for $\wp(\texttt{Sub}) \overset{\curlywedge}{\multimap} \rho$ is clearly

$\{\Theta \multimap a \mid \Theta \in \wp(\texttt{Sub}), a \in \rho\}$. Moreover, by Prop. 2.1(iv), it holds:

$$\Theta \multimap a = (\bigcup_{\theta \in \Theta} \{\,\theta\,\}) \multimap a = \bigcap_{\theta \in \Theta} (\{\,\theta\,\} \multimap a).$$

Therefore, a smaller generator set for $\wp(\texttt{Sub}) \overset{\curlywedge}{\multimap} \rho$ is simply $\{\{\,\theta\,\} \multimap a \mid \theta \in \texttt{Sub}, a \in \rho\}$. Since $\rho$ is a decidable domain, there exists $X \subseteq \rho$ which generates $\rho$ and thus $\{\{\,\theta\,\} \multimap a \mid \theta \in \texttt{Sub}, a \in X\}$ is still a generator set, $\multimap$ being right co-additivity by Prop. 2.1(iii).

We will show that, given any generator $\{\,\theta\,\} \multimap a$, there exist two programs $P_1$ and $P_2$ whose semantics satisfy:

$$[\![P_1]\!](p(\bar{V})) = \{\,\theta\,\} \multimap a$$

$$[\![P_2]\!](q(\bar{V})) = \{\,\theta\,\}$$

We can assume that no predicate is shared between $P_1$ and $P_2$.

We show that for any domain $\beta$ which satisfies $\wr\rho\wr([\![P]\!]) = \wr\rho\wr([\![P]\!]^{\beta})$ for any program $P$, it holds $\beta([\![P_1]\!])(p(\bar{V})) = \{\,\theta\,\} \multimap a$. Suppose, by contradiction, that there exists $\beta \in \texttt{uco}(\texttt{GAtom} \longmapsto \wp(\texttt{Sub}))$ such that $\wr\rho\wr([\![P]\!]) = \wr\rho\wr([\![P]\!]^{\beta})$ and $\beta([\![P_1]\!])(p(\bar{V})) \neq \{\,\theta\,\} \multimap a$. Since $[\![P_1]\!](p(\bar{V}) \leq \beta([\![P_1]\!])(p(\bar{V}))$, then it holds $\beta([\![P_1]\!])(p(\bar{V})) > \{\,\theta\,\} \multimap a$. We then consider a program $P_3$ defined as follows:

$$\{\,\mathtt{r(X)} :- \mathtt{p(X)}, \mathtt{q(X)}.\,\} \cup \mathtt{P_1} \cup \mathtt{P_2}$$

where we assume that the predicate $r$ does not appear neither in $P_1$ nor in $P_2$. Note that the semantics of $P_3$ satisfies

$$[\![P_3]\!](r(\bar{V})) = (\{\,\theta\,\} \multimap a) \otimes \{\,\theta\,\} \subseteq a.$$

Since $\{\,\theta\,\} \multimap a$ is the most abstract object which, when combined by $\otimes$ with $\{\,\theta\,\}$, is approximated by $a$, it follows that the semantics of $P_3$ satisfies $\beta([\![P_3]\!])(r(\bar{V})) > a$. Since $a \in \rho$, then $\beta$ is not complete. This proves that $\wr\wp(\texttt{Sub}) \overset{\curlywedge}{\multimap} \rho\wr$ is the most abstract domain $\beta$ which satisfies $\wr\rho\wr([\![P]\!]) = \wr\rho\wr([\![P]\!]^{\beta})$, and therefore it is the optimal semantics of $\wr\rho\wr$.

Now we have to show that $P_1$ and $P_2$ as above do exist. $P_2$ is trivially $\{\,\mathtt{q}(\bar{\mathtt{V}}\theta).\,\}$, where $\bar{V} = \{v \in \mathcal{V} \mid \theta(v) \neq v\}$. In order to show the existence of $P_1$ we note that, under the assumption of a r.e. term language, the set:

$$\{\,\theta\,\} \multimap a = \{\delta \in \texttt{Sub} \mid \text{if } \delta \wedge \theta \neq \tau \text{ then } \theta \wedge \delta \in a\}$$

is r.e.. This is because Sub is r.e., unification is recursive, and testing for membership in an abstract object is semidecidable by the hypothesis that $\rho$ is decidable ($a$ is chosen between the r.e. elements of $\rho$). Therefore, by Church's Thesis, there exists a program $P$ whose semantics is $\{\,\theta\,\} \multimap a$, for some predicate $p$. $\square$

The previous theorem states that, for any decidable domain $\rho \in \texttt{uco}(\wp(\texttt{Sub}))$, $\wr\wp(\texttt{Sub}) \overset{\curlywedge}{\multimap} \rho\wr$ is the most abstract domain whose abstract semantics is fixpoint complete when observing the property $\rho$, for any program $P$. As already announced, when the s-semantics $T^s_P$ is adequate for observing the property $\rho$ we are interested in, that is $\wr\rho\wr(\mathit{lfp}(T_P)) = \wr\rho\wr(\mathit{lfp}(T^s_P))$, it turns out that $\wr\wp(\texttt{Sub}) \overset{\curlywedge}{\multimap} \rho\wr$ is also the optimal semantics for $\rho$ relatively to the immediate consequences operator $T^s_P$ of s-semantics. Note that, the standard s-semantics operator $T^s_P$ can be obtained by $T_P$ by projecting the nonrelevant variables, i.e., $T^s_P = \exists \circ T_P$ where

$$\exists(I) \stackrel{\text{def}}{=} \{\langle p(\bar{X}), \exists_{-X}(c)\rangle \mid \langle p(\bar{X}), c\rangle \in I\}$$

and $\exists_{-X}(c)$ is existential quantification (on substitutions) of all variables except those in $X$. Moreover, $\wr \wp(\mathtt{Sub}) \stackrel{\curlywedge}{\multimap} \rho \wr$ clearly induces a semantic transformer less precise than $T^s_P$, i.e., $T^s_P \sqsubseteq \wr \wp(\mathtt{Sub}) \stackrel{\curlywedge}{\multimap} \rho \wr \circ T_P$. Hence, by Theorem 5.4, we get the following desired consequence.

Corollary 5.5. *Let $\rho \in \mathtt{uco}(\wp(\mathtt{Sub}))$ be a decidable abstract domain, such that $\wr\rho\wr(\mathit{lfp}(T_P)) = \wr\rho\wr(\mathit{lfp}(T^s_P))$, for any $P$. Then, $\wr \wp(\mathtt{Sub}) \stackrel{\curlywedge}{\multimap} \rho \wr$ is the optimal semantics for $\wr\rho\wr$ relatively to any $T^s_P$.*

Proof. We have to show that $\wr \wp(\mathtt{Sub}) \stackrel{\curlywedge}{\multimap} \rho \wr$ is the most abstract domain $\alpha$ which satisfies $\wr\rho\wr(\mathit{lfp}(T^s_P)) = \wr\rho\wr(\mathit{lfp}(\alpha \circ T^s_P))$. We first show that $\wr \wp(\mathtt{Sub}) \stackrel{\curlywedge}{\multimap} \rho \wr$ enjoys this property. By hypothesis, $\wr\rho\wr(\mathit{lfp}(T_P)) = \wr\rho\wr(\mathit{lfp}(\exists \circ T_P))$. Since $\wr \wp(\mathtt{Sub}) \stackrel{\curlywedge}{\multimap} \rho \wr$ is the most abstract domain $\beta$ which satisfies $\wr\rho\wr(\mathit{lfp}(T_P)) = \wr\rho\wr(\mathit{lfp}(\beta \circ T_P))$ and $\exists \in \mathtt{uco}(\mathtt{GAtom} \longmapsto \wp(\mathtt{Sub}))$, it follows that $\exists \sqsubseteq \wr \wp(\mathtt{Sub}) \stackrel{\curlywedge}{\multimap} \rho \wr$. Thus we have that:

$$\begin{array}{ll}
\quad \wr\rho\wr(\mathit{lfp}(\wr \wp(\mathtt{Sub}) \stackrel{\curlywedge}{\multimap} \rho \wr \circ T^s_P)) & \\
= \wr\rho\wr(\mathit{lfp}(\wr \wp(\mathtt{Sub}) \stackrel{\curlywedge}{\multimap} \rho \wr \circ \exists \circ T_P)) & [\text{ by definition of } T^s_P\ ] \\
= \wr\rho\wr(\mathit{lfp}(\wr \wp(\mathtt{Sub}) \stackrel{\curlywedge}{\multimap} \rho \wr \circ T_P)) & [\text{ since } \exists \sqsubseteq \wr \wp(\mathtt{Sub}) \stackrel{\curlywedge}{\multimap} \rho \wr\ ] \\
= \wr\rho\wr(\mathit{lfp}(T_P)) & [\text{ by definition of optimal semantics }] \\
= \wr\rho\wr(\mathit{lfp}(T^s_P)) & [\text{ by hypothesis. }]
\end{array}$$

To prove that $\wr \wp(\mathtt{Sub}) \stackrel{\curlywedge}{\multimap} \rho \wr$ is the least abstraction with this property, we show that whenever $\alpha$ satisfies $\wr\rho\wr(\mathit{lfp}(T^s_P)) = \wr\rho\wr(\mathit{lfp}(\alpha \circ T^s_P))$, then it satisfies $\wr\rho\wr(\mathit{lfp}(T_P)) = \wr\rho\wr(\mathit{lfp}(\alpha \circ T_P))$ also. Assume $\wr\rho\wr(\mathit{lfp}(T^s_P)) = \wr\rho\wr(\mathit{lfp}(\alpha \circ T^s_P))$. Then we have that $\wr\rho\wr(\mathit{lfp}(\alpha \circ T_P)) \leq \wr\rho\wr(\mathit{lfp}(\alpha \circ T^s_P)) = \wr\rho\wr(\mathit{lfp}(T^s_P)) = \wr\rho\wr(\mathit{lfp}(T_P))$. The thesis then follows from the fact that $\wr \wp(\mathtt{Sub}) \stackrel{\curlywedge}{\multimap} \rho \wr$ is, by definition of optimal semantics, the most abstract domain with this property. □

### 5.3 Optimal semantics for groundness

Groundness analysis is arguably one of the most important analysis for logic-based programming languages. Groundness analysis aims to statically detect whether variables will be bound to ground terms in successful derivations. This allows optimizing compilers to significantly speed up unification. By instantiating the results above, we are able to characterize the most abstract refinement of the basic domain representing plain groundness information, which is complete for the fixpoint of any immediate consequences operator of s-semantics. The resulting semantics can therefore be interpreted as the "optimal" semantics for groundness analysis.

If $V \subseteq \mathcal{V}$ is a finite set of variables of interest, the simplest abstract domain for representing plain groundness information of variables in $V$ is $\mathcal{G}_V \stackrel{\text{def}}{=} \langle \wp(V), \supseteq \rangle$, as first put forward in [Jones and Søndergaard 1987]. The intuition is that each abstract object $W \in \mathcal{G}_V$ represents the set of substitutions which ground every variable in $W$. $\mathcal{G}_V$ is related to the concrete domain $\wp(\mathtt{Sub})$ by the following concretization map, for each $W \in \mathcal{G}$.

$$\gamma_{\mathcal{G}_V}(W) \stackrel{\text{def}}{=} \{\theta \in \mathtt{Sub} \mid \forall v \in W.\ \mathit{vars}(\theta(v)) = \varnothing\}.$$

As usual, we shall abuse the notation by denoting with $\mathcal{G}_V \in \mathtt{uco}(\wp(\mathtt{Sub}))$ the corresponding isomorphic image $\gamma_{\mathcal{G}_V}(\mathcal{G}_V)$ in $\wp(\mathtt{Sub})$. Note that, the choice of an abstract domain $\mathcal{G}_V$ depends on the number of variables used during the analysis, and therefore on the specific program to analyze. A program independent abstract domain $\mathcal{G} \in \mathtt{uco}(\mathtt{CInt})$ for representing plain groundness information can therefore be defined as follows:

$$\mathcal{G}(I) \stackrel{\text{def}}{=} \{\langle p(\bar{X}), \mathcal{G}_{\bar{X}}(c)\rangle \mid \langle p(\bar{X}), c\rangle \in I\}.$$

It is easy to check that $\mathcal{G}$ is actually a closure on the concrete semantic domain $\mathtt{CInt}$. Furthermore, $\mathcal{G}$ is clearly a decidable domain and $T_P^s$ is adequate for observing groundness, i.e., $\mathcal{G}(\mathit{lfp}(T_P)) = \mathcal{G}(\mathit{lfp}(T_P^s))$ ([Giacobazzi 1996]). Thus, as an easy consequence of Corollary 5.5, we can prove the following result, where $V \subset_f \mathcal{V}$ means that $V$ is a finite subset of $\mathcal{V}$.

Theorem 5.6. *The optimal semantics for $\mathcal{G}$ relatively to the s-semantics is* $\bigsqcap_{V \subset_f \mathcal{V}} \wr \wp(\mathtt{Sub}) \overset{\curlywedge}{\multimap} \mathcal{G}_V \int$.

Proof. For each finite $V \subset_f \mathcal{V}$, the optimal semantics for $\wr \mathcal{G}_V \int$ relatively to $T_P^s$ is $\wr \wp(\mathtt{Sub}) \overset{\curlywedge}{\multimap} \mathcal{G}_V \int$. Therefore, the optimal semantics for $\wr \mathcal{G}_V \int$ for each $V \in \mathcal{V}$ must contain $\bigsqcap_{V \subset_f \mathcal{V}} \wr \wp(\mathtt{Sub}) \overset{\curlywedge}{\multimap} \mathcal{G}_V \int$. The thesis follows since reduced product of complete domains is still complete. □

Example 5.7. Consider the language $\mathcal{L} = \{ a/0, f/1 \} \cup \mathcal{V} \cup \pi$, where $\pi = \{ p/1 \}$ and $\mathcal{V} = \{ X_i \}_{i \in \mathbb{N}}$. Then, let us consider the following program $P$ built over $\mathcal{L}$:

$$\begin{aligned} &p(X_0) :- X_0 = a. \\ &p(X_0) :- X_0 = f(X_1). \end{aligned}$$

Recall that the s-semantics $[\![P]\!]^s$ of a program $P$ is obtained from $T_P$ as $\mathit{lfp}(\exists \circ T_P)$. Moreover, the Clark semantics $[\![P]\!]^c$ (also known as c-semantics [Falaschi et al. 1989]) of a program $P$ characterizes the correct answer substitutions for $P$, and can be obtained from the s-semantics $[\![P]\!]^s$ by the closure under instantiation $\lambda X. \downarrow X$. Thus, in our notation, for any predicate symbol $p$, $[\![P]\!]^c(p(\bar{X})) = \downarrow([\![P]\!]^s(p(\bar{X})))$. It is then an easy task to check that, for any variable $X_i \in \mathcal{V}$,

$$\begin{aligned} &[\![P]\!]^s(p(X_i)) = \{ \{ X_i \leftarrow a \} \} \cup \{ \{ X_i \leftarrow f(X_j) \} \mid X_j \in \mathcal{V}, X_i \neq X_j \} \\ &[\![P]\!]^c(p(X_i)) = \{ \sigma \in \mathtt{Sub} \mid \forall X_j \in \mathcal{V}. \sigma(X_i) \neq X_j \}. \end{aligned}$$

In this case, the most abstract domain which includes $\mathcal{G}$ and it is complete for the fixpoint of the s-semantics (which, by Theorem 5.6, is given by $\bigsqcap_{V \subset_f \mathcal{V}} \wr \wp(\mathtt{Sub}) \overset{\curlywedge}{\multimap} \mathcal{G}_V \int$), when applied to s-semantics $[\![P]\!]^s$, differs from both s-semantics and c-semantics, and acts as follows:

$$(\bigsqcap_{V \subset_f \mathcal{V}} \wr \wp(\mathtt{Sub}) \overset{\curlywedge}{\multimap} \mathcal{G}_V \int)([\![P]\!]^s) = \lambda\, p(X_i).\mathtt{Sub} \tag{4}$$

Before proving this equality, let us informally justify it. Intuitively, a complete semantics including groundness information should be able to observe both groundness and failures: groundness, of course, because it is as concrete as the domain of observation; failures because, being complete for unification, it is able to discriminate between successful and failed computations. Moreover, if the set of computed answers, for a given predicate $p(X)$, includes all possible ground substitutions and some nonground substitutions, then any goal $p(t)$ can never fail (since the semantics contains at least one ground answer for the goal) and, at the same time, the set of computed answers for the most general atom $p(X)$ is not ground. Thus, the most abstract semantics including groundness information does not need to distinguish between such a program and one whose computed answers are all possible substitutions, since both such programs have the same (negative) result for groundness analysis and do not fail for any goal.

In order to prove Eq. (4), we refer to the characterization of optimal semantics given in Theorem 4.6 and compute:

$$\Big(\bigsqcap_{V \subset_f \mathcal{V}} \wr \wp(\mathtt{Sub}) \overset{\curlywedge}{\multimap} \mathcal{G}_V \wr\Big)(\llbracket P \rrbracket^s) = \bigcap_{V \subset_f \mathcal{V}} \bigcap_{g \in \mathcal{G}_V} (\llbracket P \rrbracket^s(p(X_i)) \multimap g) \multimap g.$$

We first show that, for all $V \subset_f \mathcal{V}$ and $g \in \mathcal{G}_V$, $\llbracket P \rrbracket^s(p(X_i)) \multimap g = g$. It suffices to show that $\llbracket P \rrbracket^s(p(X_i)) \multimap \varnothing = \varnothing$, and, for any $v \in \mathcal{V}$, $\llbracket P \rrbracket^s(p(X_i)) \multimap \{ v \} = \{ v \}$. By definition, $\delta \in \llbracket P \rrbracket^s(p(X_i)) \multimap \varnothing$ if and only if for all $\theta \in \llbracket P \rrbracket^s(p(X_i))$ it holds $\delta \wedge \theta = \tau$. But note that, given any substitution $\delta$, there always exists a $\sigma \in \llbracket P \rrbracket^s(p(X_i))$ which unifies with $\delta$, and therefore $\llbracket P \rrbracket^s(p(X_i)) \multimap \varnothing = \varnothing$. In the second case, for each $v \in \mathcal{V}$ we have that:

$$\llbracket P \rrbracket^s(p(X_i)) \multimap \{ v \} = \{\delta \in \mathtt{Sub} \mid \{ \delta \} \otimes \llbracket P \rrbracket^s(p(X_i)) \subseteq \{ v \}\}.$$

It clearly holds $\llbracket P \rrbracket^s(p(X_i)) \multimap \{ v \} \supseteq \{ v \}$, since $\{ v \}$ is closed by instantiation. On the other hand, given any $\delta \notin \{ v \}$, if $Z$ is a fresh variable not in $\delta$, it holds $\delta \wedge \{ X_i \leftarrow f(Z) \} \notin \{ v \}$. Therefore $\{ \delta \} \otimes \llbracket P \rrbracket^s(p(X_i)) \not\subseteq \{ v \}$ and any substitution $\delta \in \llbracket P \rrbracket^s(p(X_i)) \multimap \{ v \}$ belongs to $\{ v \}$. Thus, for any finite set of variables $V$, it holds $\llbracket P \rrbracket^s(p(X_i)) \multimap V = V$. Hence, by Theorem 4.6, we have that:

$$\begin{aligned}
&(\textstyle\bigsqcap_{V \subset_f \mathcal{V}} \wr \wp(\mathtt{Sub}) \overset{\curlywedge}{\multimap} \mathcal{G}_V \wr)(\llbracket P \rrbracket^s)(p(X_i)) \\
=\ & \textstyle\bigwedge_{V \subset_f \mathcal{V}} \bigwedge_{g \in \mathcal{G}_V} (\llbracket P \rrbracket^s(p(X_i)) \multimap g) \multimap g \\
=\ & \textstyle\bigwedge_{V \subset_f \mathcal{V}} \bigwedge_{g \in \mathcal{G}_V} g \multimap g \\
=\ & \mathtt{Sub}.
\end{aligned}$$

It is also worth noting that if we consider the equivalence relation on programs induced by the semantics $(\bigsqcap_{V \subset_f \mathcal{V}} \wr \wp(\mathtt{Sub}) \overset{\curlywedge}{\multimap} \mathcal{G}_V \wr)(\llbracket P \rrbracket^s)$, the program $P$ is in the same equivalence class as the program $\{ p(X_0). \}$. $\square$

## 6. CHARACTERIZING CONDENSING DOMAINS

It is well known that, in general, goal-independent analyses of logic programs, like those obtainable with generic bottom-up/top-down analyzers [Barbuti et al. 1993; Bruynooghe 1991; Codish et al. 1994], may be less precise than goal-directed ones (cf. [Marriott and Søndergaard 1993]). This is because goal-independent analysis can be thought of as an analysis for all possible initial states, while in goal-directed analysis, a given query with a fixed initial state is executed in the abstract domain. In this latter case the result may be refined by knowing in advance the initial calls. On the contrary, it may also happen that goal-independent analyses are more precise than goal-directed ones, for instance when failure properties are observed. Abstract domains where goal-independent and goal-directed analyses have the same precision are called *condensing* domains.

The notion of condensation has been first introduced by [Langen 1990] in the context of dataflow analysis of logic programs. The original definition of condensing domain for $X \in \texttt{uco}(C)$ can be given for a generic abstract semantic function $f^\sharp : X \longmapsto X$, with respect to a binary operation $\otimes^\sharp : X \times X \longmapsto X$. A domain $X \in \texttt{uco}(C)$ is condensing if for any $x, y \in X$ we have that

$$x \otimes^\sharp f^\sharp(y) = f^\sharp(x \otimes^\sharp y).$$

[Marriott and Søndergaard 1993] proposed a stronger notion of condensing, where the abstract function $f^\sharp$ and the abstract operation $\otimes^\sharp$ are supposed to be induced from a concrete semantic function $f : C \longmapsto C$ and a concrete operation $\otimes : C \times C \longmapsto C$, i.e., they consider the best correct approximations of the corresponding concrete operations. This is what makes it possible to consider condensing an abstract domain property. We follow [Marriott and Søndergaard 1993] and consider the following definition of condensing.

Definition 6.1. An abstract domain $\rho \in \texttt{uco}(C)$ is *condensing* for $f : C \longmapsto C$ w.r.t. $\otimes : C \times C \longmapsto C$ if for any $x, y \in \rho$, we have that

$$\rho(x \otimes f^\rho(y)) = f^\rho(\rho(x \otimes y))$$

where $f^\rho : \rho \longmapsto \rho$ is the abstract semantics induced by $f$ on the domain $\rho$.

The above property of condensation is particularly relevant in analysis of constraint logic programs, where $\otimes$ plays the rôle of unification (conjunction) of constraints [Langen 1990; Marriott and Søndergaard 1993]. More formally, if *true* represents the top (weakest) constraint, representing any initial store, then in any condensing abstract domain, for any constraint $c \in \rho$, we have that $\rho(c \otimes f^\rho(\rho(true))) = f^\rho(c)$. The idea here is that computing the semantics of a query with initial constraint $c$, i.e., $f^\rho(c)$, is the same as unifying $c$ with the result of the semantics of the same query in the most general environment *true*, which is $f^\rho(\rho(true))$. This encodes the typical way we derive the analysis of a query in an initial state from a goal-independent analysis: we filter out, by unification, those computations which do not satisfy the given initial state (cf. [Barbuti et al. 1993]). Unfortunately, for most domains, this information cannot be derived with the same precision as that obtainable in goal-directed analysis. However, as observed in [Langen 1990] and [Jacobs and Langen 1992], under the hypothesis that the abstract domain is condensing, no loss of precision is accumulated in deriving information about specific calls from goal-independent analyses.

In this section we show that, the abstract domains which are solutions of the recursive domain equation $\rho = \rho \overset{\curlywedge}{\multimap} \rho$, all satisfy the relevant property of being *condensing* (see [Langen 1990] and [Marriott and Søndergaard 1993]), and under additional nonrestrictive hypotheses, condensing domains are all and only the solutions of $\rho = \rho \overset{\curlywedge}{\multimap} \rho$. This property shows a surprising link between completeness in abstract interpretation and condensing domains and, more importantly, gives computational relevance in static program analysis to the notion of disjunctive completeness.

Consider a simple constraint logic programming (*clp*) language (cf. [Jaffar and Maher 1994]). We consider here a *ccp*-like presentation of (*clp*: processes interact with each other by monotonically refining a common store, which is represented by a set of constraints [Saraswat et al. 1991]. For simplicity, we do not consider here any synchronization mechanism, like blocking-ask; therefore processes are only allowed to *tell* constraints to a global store. Because of this restriction, our language boils down to a standard (nondeterministic) *clp* language. The syntax of programs is as follows, where $P$ are programs and $A$ are (*clp*-agents:

$$\begin{array}{lcl} P & ::= & \varepsilon \mid p(\bar{x}) \leftarrow A \mid P.P \\ A & ::= & c \mid A \wedge A \mid \sum_{i=1}^{n} A_i \mid p(\bar{x}) \end{array}$$

We abuse by considering programs as (finite) sets of predicate or process declarations, and we assume that each process can be declared in at most one clause of the form $p(\bar{x}) \leftarrow A$. Also, we assume here that Store is a poset of constraints (i.e., a constraint system), ordered by logical implication, including equational constraints. The semantic domain is $\langle \wp(\texttt{Store}), \otimes \rangle$, which is assumed to be a unital, commutative quantale, with unity *true*. Moreover, we assume that $\otimes$ is idempotent, i.e., $c \otimes c = c$, for all $c \in \wp(\texttt{Store})$. These are weak enough conditions to include most constraint systems developed in the literature (e.g., see [Saraswat et al. 1991]). The (forward) semantics $[\![\langle p(\bar{x}), s\rangle]\!]_P$ of a process $p$, executed with initial store $s \in \wp(\texttt{Store})$ in a program $P$, is defined as usual as the least fixpoint of the following recursive function $\mathcal{S}^s_{P,p(\bar{x})} : \wp(\texttt{Store}) \longmapsto \wp(\texttt{Store})^2$, for any $\sigma \in \wp(\texttt{Store})$.

$$\begin{array}{lcl} \mathcal{S}_c(\sigma) & = & s \otimes c \otimes \sigma \\ \mathcal{S}_{A_1 \wedge A_2}(\sigma) & = & \mathcal{S}_{A_1}(\sigma) \otimes \mathcal{S}_{A_2}(\sigma) \\ \mathcal{S}_{\sum_{i=1}^{n} A_i}(\sigma) & = & \bigvee_{i=1}^{n} \mathcal{S}_{A_i}(\sigma) \\ \mathcal{S}_{p(\bar{x})}(\sigma) & = & \mathcal{S}_A(\sigma \otimes \{\, \bar{x} = \bar{y} \,\}) \ \text{ where } C = p(\bar{y}) \leftarrow A \lll P. \end{array}$$

Note that parameter passing is performed, as usual in logic programming, by telling an equation to the global store, which makes formal and actual parameters matching. In this definition $\lll$ selects a clause from $P$ whose variables are renamed apart from the rest of the computation. As before, we let a constraint $c$ denote the set of most general unifiers which solve it.

The following result states that it is always possible to systematically design a condensing domain for a given basic property of interest $A \in \texttt{uco}(\wp(\texttt{Store}))$, by solving the recursive domain equation $\rho = A \sqcap \rho \overset{\curlywedge}{\multimap} \rho$. In order to prove this statement, we consider the following abstract semantics of processes, induced by the concrete one. An abstract semantics is associated with any abstract domain $\rho \in \texttt{uco}(\wp(\texttt{Store}))$, such that for any $\sigma \in \rho$ and initial state $s \in \rho$ we have that

$$\begin{array}{lcl} \mathcal{S}^{\rho}_c(\sigma) & = & \rho(s \otimes c \otimes \sigma) \\ \mathcal{S}^{\rho}_{A_1 \wedge A_2}(\sigma) & = & \rho(\mathcal{S}^{\rho}_{A_1}(\sigma) \otimes \mathcal{S}^{\rho}_{A_2}(\sigma)) \\ \mathcal{S}^{\rho}_{\sum_{i=1}^{n} A_i}(\sigma) & = & \rho(\bigvee_{i=1}^{n} \mathcal{S}^{\rho}_{A_i}(\sigma)) \\ \mathcal{S}^{\rho}_{p(\bar{x})}(\sigma) & = & \mathcal{S}^{\rho}_A(\rho(\sigma \otimes \{\, \bar{x} = \bar{y} \,\})) \ \text{ if } p(\bar{y}) \leftarrow A \lll P. \end{array}$$

For any process $p(\bar{x})$, abstract initial store $s \in \rho$, and program $P$, we define $[\![\langle p(\bar{x}), s\rangle]\!]^{\rho}_P = \mathit{lfp}(\mathcal{S}^{s,\rho}_{P,p(\bar{x})})$. Note that, in each equation above, $\mathcal{S}^{\rho}$ is recursively defined as the best correct approximation of $\mathcal{S}$ in $\rho$. The next example shows an abstract domain which is not condensing, where the concrete domain $\wp(\texttt{Store})$ and $\otimes$ are, respectively, sets of substitutions and standard unification.

[2]When the program $P$ and the initial state $s$ are clear from the context, we simply write $\mathcal{S}_{p(\bar{x})}$.

EXAMPLE 6.2. Consider the abstract domain $\mathcal{G}_V$ for groundness analysis of logic programs presented in Section 5.3, for $V = \{x, y\}$. Recall that the concrete domain is $\wp(\mathtt{Sub})$, the operation $\otimes$ is standard unification, and the unity *true* is $\{\epsilon\}$, as defined in Section 5.2. We show that the domain $\mathcal{G}_V$ is not condensing. Consider the program $P$ defined as follows:

$$\mathtt{p(X,Y)} :- \mathtt{X} = \mathtt{Y}.$$

The semantics of the query $p(X,Y)$ with initial state *true* is $[\![\langle p(X,Y), true\rangle]\!]_P^{\mathcal{G}_V} = \mathcal{G}_V(\{X = Y\}) = \mathtt{Sub}$, and therefore $\mathcal{G}_V(\mathcal{G}_V(\{X = a\}) \otimes [\![\langle p(X,Y), true\rangle]\!]_P^{\mathcal{G}_V} = \mathcal{G}_V(\{X = a\}) = \{X\}$, which means that the first argument of $p$ is definitely ground. On the other hand, $[\![\langle p(X,Y), \mathcal{G}_V(\{X = a\})\rangle]\!]_P^{\mathcal{G}_V} = \mathcal{G}_V(\mathcal{G}_V(\{X = a\}) \otimes \{X = Y\}) = \{X, Y\}$, which states that both arguments are definitely ground. This shows that the domain $\mathcal{G}_V$ is not condensing. □

Before proving the main theorem, let us show the following simple property which will be frequently used in this section.

PROPOSITION 6.3. *Let $\langle C_{\leq}, \otimes\rangle$ be a unital, commutative quantale, $\rho \in \mathtt{uco}(C)$ and $c_1, .., c_n \in C$ for $n < \omega$. If $\rho$ satisfies Eq. (1), then for all $i$, with $1 \leq i \leq n$, it holds:*

$$\rho(\rho(c_1) \otimes \ldots \rho(c_i) \cdots \otimes \rho(c_n)) = \rho(\rho(c_1) \otimes \ldots c_i \cdots \otimes \rho(c_n))$$

PROOF. Straightforward from Eq. (1). □

THEOREM 6.4. *Let $P$ be a program, $\rho \in \mathtt{uco}(\wp(\mathtt{Store}))$, and $p(\bar{x})$ be a process. If $\rho = \rho \sqcap \rho \overset{\curlywedge}{\multimap} \rho$ then $\rho$ is condensing for $\lambda s \in \rho.\ [\![\langle p(\bar{x}), s\rangle]\!]_P^\rho$.*

PROOF. Let $\rho$ be a solution of the recursive domain equation $\rho = \rho \sqcap \rho \overset{\curlywedge}{\multimap} \rho$. Let $\psi, \varphi \in \rho$. Since $\langle \wp(\mathtt{Store})_{\subseteq}, \otimes\rangle$ is a unital, commutative quantale, it is sufficient to prove, by induction on the recursive calls to the function $\mathcal{S}^\rho$ in the definition of $[\![\langle p(\bar{x}), \psi \otimes \varphi\rangle]\!]_P^\rho$, that $\mathcal{S}^\rho$ is condensing, for any program $P$ and initial state $s \in \wp(\mathtt{Store})$, i.e., $\mathcal{S}^\rho(\rho(\psi \otimes \varphi)) = \rho(\psi \otimes \mathcal{S}^\rho(\varphi))$.

—Consider $\mathcal{S}_c^\rho$. By Proposition 6.3 we have that

$$\begin{aligned} \mathcal{S}_c^\rho(\rho(\psi \otimes \varphi)) &= \rho(s \otimes c \otimes \rho(\psi \otimes \varphi)) \\ &= \rho(s \otimes c \otimes \psi \otimes \varphi) \\ &= \rho(\psi \otimes c \otimes s \otimes \varphi) \\ &= \rho(\psi \otimes \rho(c \otimes s \otimes \varphi)) \\ &= \rho(\psi \otimes \mathcal{S}_c^\rho(\rho(\varphi))) \end{aligned}$$

—Consider $\mathcal{S}_{A_1 \wedge A_2}^\rho$. By Proposition 6.3 and inductive hypothesis we have that

$$\begin{aligned} \mathcal{S}_{A_1 \wedge A_2}^\rho(\rho(\psi \otimes \varphi)) &= \rho(\mathcal{S}_{A_1}^\rho(\rho(\psi \otimes \varphi)) \otimes \mathcal{S}_{A_2}^\rho(\rho(\psi \otimes \varphi))) \\ &= \rho(\psi \otimes \mathcal{S}_{A_1}^\rho(\varphi) \otimes \psi \otimes \mathcal{S}_{A_2}^\rho(\varphi)) \\ &= \rho(\rho(\psi) \otimes \rho(\mathcal{S}_{A_1}^\rho(\varphi) \otimes \mathcal{S}_{A_2}^\rho(\varphi))) \\ &= \rho(\rho(\psi) \otimes \mathcal{S}_{A_1 \wedge A_2}^\rho(\varphi)). \end{aligned}$$

—Consider $\mathcal{S}_{\sum_{i=1}^n A_i}^\rho$. Recall that, by definition of quantale, the operation $\otimes$ is additive. Therefore, by inductive hypothesis and Proposition 6.3 we have that

$$\begin{aligned}\mathcal{S}^{\rho}_{\sum_{i=1}^{n} A_i}(\psi \otimes \varphi) &= \rho(\textstyle\bigvee_{i=1}^{n} \mathcal{S}^{\rho}_{A_i}(\psi \otimes \varphi)) \\ &= \rho(\textstyle\bigvee_{i=1}^{n} \psi \otimes \mathcal{S}^{\rho}_{A_i}(\varphi)) \\ &= \rho(\psi \otimes \textstyle\bigvee_{i=1}^{n} \mathcal{S}^{\rho}_{A_i}(\varphi)) \\ &= \rho(\psi \otimes \rho(\textstyle\bigvee_{i=1}^{n} \mathcal{S}^{\rho}_{A_i}(\varphi))) \\ &= \rho(\psi \otimes \mathcal{S}^{\rho}_{\sum_{i=1}^{n} A_i}(\varphi)).\end{aligned}$$

—Consider $\mathcal{S}^{\rho}_{p(\bar{x})}$. The thesis follows immediately by definition and inductive hypothesis. □

The key point in the previous theorem is that the operation $\otimes$ is additive, i.e., it distributes over arbitrary l.u.b.'s. Thus, whenever the abstract domain contains all linear implications between any two abstract objects, no loss of precision is accumulated by distributing the abstract unification over nondeterministic computations, which are modeled by abstract disjunction.

In order to prove the converse of Theorem 6.4, i.e., that condensing domains are solutions of Eq. (1), we need some additional hypotheses. The next result, together with Theorem 6.4, provides a constructive characterization of condensing domains which are at least as precise as a given domain $A$, as all and only those domains which are solutions of a recursive domain equation of the form $X = A \sqcap X \sqcap X \overset{\curlywedge}{\multimap} X$.

THEOREM 6.5. *Let $P$ be a program, $p(\bar{x})$ be a process, $\rho \in \mathtt{uco}(\wp(\mathtt{Store}))$ and let $X$ be a generator for $\rho$, i.e., $\mathcal{M}(X) = \rho$. Assume that for any $x, y \in X$ there exist $\{c_1, \ldots c_n\} \subseteq \mathtt{Store}$, $n < \omega$, such that the following conditions hold.*

—$\bigvee_{i=1}^{n} c_i \leq x \multimap y$

—$\rho(\bigvee_{i=1}^{n} c_i) = \rho(x \multimap y)$

*If $\rho$ is condensing for $\lambda s \in \rho.\, [\![\langle p(\bar{x}), s\rangle]\!]^{\rho}_{P}$ then $\rho = \rho \sqcap \rho \overset{\curlywedge}{\multimap} \rho$.*

PROOF. Let $\rho$ be condensing. We have to prove that $\rho \sqsubseteq \rho \overset{\curlywedge}{\multimap} \rho$, i.e., that for any $a, b \in \rho$ $\rho(a \multimap b) = a \multimap b$. Since $X$ is a generator set for $\rho$, it suffices to show that for any $a, b \in X$ $\rho(a \multimap b) = a \multimap b$. Suppose, by contradiction, that there exist $a, b \in X$ such that $\rho(a \multimap b) > a \multimap b$. Note that, by definition, $a \multimap b$ is the most abstract object $c$ which satisfies $a \otimes c \leq b$. Therefore, $a \otimes \rho(a \multimap b) \not\leq b$. By hypothesis there exists $\{c_1, \ldots c_n\} \subseteq \mathtt{Store}$, $n < \omega$ such that the above conditions hold. Consider the program $P = \{\, p :\!-\ \sum_{i=1}^{n} c_i \,\}$. Then, by hypothesis, we have that

$$\begin{aligned}\rho(a \otimes [\![\langle p, true\rangle]\!]^{\rho}_{P}) &= \rho(a \otimes \rho(\textstyle\bigvee_{i=1}^{n} \rho(true \otimes c_i))) \\ &= \rho(a \otimes \rho(\textstyle\bigvee_{i=1}^{n} \rho(c_i))) \\ &= \rho(a \otimes \rho(\textstyle\bigvee_{i=1}^{n} c_i)) \\ &= \rho(a \otimes \rho(a \multimap b)) \\ &\not\leq b \\ &\geq \rho(a \otimes (a \multimap b)) \\ &\geq \rho(a \otimes \textstyle\bigvee_{i=1}^{n} c_i) \\ &= \rho(\textstyle\bigvee_{i=1}^{n} a \otimes c_i) \\ &= \rho(\textstyle\bigvee_{i=1}^{n} \rho(a \otimes c_i)) \\ &= [\![\langle p, a\rangle]\!]^{\rho}_{P}.\end{aligned}$$

Therefore $\rho$ would not be condensing, which is a contradiction. □

The hypothesis in Theorem 6.5 is not restrictive for most well-known domains in logic program analysis. Conditions (1) and (2) say that any implicational object $x \multimap y$ always approximates a finite disjunction of basic stores. This allows us, in Theorem 6.5, to construct, for any implicational object $x \multimap y$, a (finite) program having that object in the abstract semantics. The idea is that, in order to prove the converse of Theorem 6.4, any implicational object has to be the semantics of some well-defined program. The following characterization of condensing is therefore immediate by Theorems 6.4 and 6.5.

COROLLARY 6.6. *Under the hypothesis of Theorem 6.5, $\rho$ is condensing for the semantic function $\lambda s \in \rho.\, [\![\langle p(\bar{x}), s\rangle]\!]^{\rho}_{P}$ if and only if $\rho = \rho \sqcap \rho \overset{\curlywedge}{\multimap} \rho$.*

## REFERENCES


ABRAMSKY, S. AND JUNG, A. 1994. Domain theory. In *Handbook of Logic in Computer Science*, S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, Eds. Vol. 3. Clarendon Press, Oxford, U.K., 1–168.

ABRAMSKY, S. AND VICKERS, S. 1993. Quantales, observational logic and process semantics. *Math. Struct. Comput. Sci. 3,* 2, 161–228.

ARMSTRONG, T., MARRIOTT, K., SCHACHTE, P., AND SØNDERGAARD, H. 1994. Boolean functions for dependency analysis: Algebraic properties and efficient representation. In *Proceedings of the 1st International Static Analysis Symposium* (*SAS '94*), B. Le Charlier, Ed. Lecture Notes in Computer Science, vol. 864. Springer-Verlag, Berlin, 266–280.

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.

BOSSI, A., GABBRIELLI, M., G.LEVI, AND MARTELLI, M. 1994. The s-semantics approach: theory and applications. *J. Logic Programm. 19-20*, 149–197.

BRUYNOOGHE, M. 1991. A practical framework for the abstract interpretation of logic programs. *J. Logic Programm. 10*, 91–124.

CODISH, M., DAMS, D., AND YARDENI, E. 1994. Bottom-up abstract interpretation of logic programs. *Theor. Comput. Sci. 124,* 1, 93–126.

COUSOT, P. 1997. Types as abstract interpretations (Invited Paper). In *Conference Record of the 24th ACM Symp. on Principles of Programming Languages* (*POPL '97*). ACM Press, New York, 316–331.

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. 1979a. Constructive versions of Tarski's fixed point theorems. *Pacific Journal of Mathematics 82,* 1, 43–57.

COUSOT, P. AND COUSOT, R. 1979b. 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.

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

COUSOT, P. AND COUSOT, R. 1997. Abstract interpretation of algebraic polynomial systems. In *Proc. 6th AMAST Conf.* Lecture Notes in Computer Science, vol. 1349. Springer-Verlag, Berlin, 138–154.

FALASCHI, M., LEVI, G., MARTELLI, M., AND PALAMIDESSI, C. 1989. Declarative modeling of the operational behavior of logic languages. *Theoretical Computer Science 69,* 3, 289–318.

GIACOBAZZI, R. 1996. "Optimal" collecting semantics for analysis in a hierarchy of logic program semantics. In *Proc. of the 13th Int'l Symposium on Theoretical Aspects of Computer Science*

*(STACS '96)*, C. Puech and R. Reischuk, Eds. Lecture Notes in Computer Science, vol. 1046. Springer-Verlag, Berlin, 503–514.

Giacobazzi, R. and Ranzato, F. 1997. Completeness in abstract interpretation: A domain perspective. In *Proceedings of the 6th International Conference on Algebraic Methodology and Software Technology* (*AMAST '97*), M. Johnson, Ed. Lecture Notes in Computer Science, vol. 1349. Springer-Verlag, Berlin, 231–245.

Giacobazzi, R., Ranzato, F., and Scozzari, F. 1998. Complete abstract interpretations made constructive. In *Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science* (*MFCS '98*), J. Gruska and J. Zlatuska, Eds. Lecture Notes in Computer Science, vol. 1450. Springer-Verlag, Berlin, 366–377.

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

Giacobazzi, R. and Scozzari, F. 1998. A logical model for relational abstract domains. *ACM Transactions on Programming Languages and Systems 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.

Jaffar, J. and Maher, M. J. 1994. Constraint logic programming: A survey. *J. Logic Programm. 19-20*, 503–581.

Jensen, T. 1997. Disjunctive program analysis for algebraic data types. *ACM Trans. Program. Lang. Syst. 19,* 5, 751–803.

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.

Langen, A. 1990. Static analysis for independent And-parallelism in logic programs. Ph.D. thesis, Univ. of Southern California, Los Angeles, Calif.

Lassez, J.-L., Maher, M. J., and Marriott, K. 1988. Unification revisited. In *Foundations of Deductive Databases and Logic Programming*, J. Minker, Ed. Morgan Kaufmann, Los Altos, 587–625.

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

Mycroft, A. 1993. Completeness and predicate-based abstract interpretation. In *Proceedings of the ACM Symposium on Partial Evaluation and Program Manipulation* (*PEPM '93*). ACM Press, New York, 179–185.

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.

Reddy, U. and Kamin, S. 1993. On the power of abstract interpretation. *Computer Languages 19,* 2, 79–89.

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

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

Sekar, R., Mishra, P., and Ramakrishnan, I. 1997. On the power and limitation of strictness analysis. *J. ACM 44,* 3, 505–525.

Steffen, B. 1989. Optimal data flow analysis via observational equivalence. In *Proceedings of the 14th International Symposium on Mathematical Foundations of Computer Science* (*MFCS '89*). Lecture Notes in Computer Science, vol. 379. Springer-Verlag, Berlin, 492–502.

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